Cesar Smaniotto Júnior

SoNDA: Um software para apoio à análise qualitativa de postagens de redes sociais

Florianópolis - SC

2017/1

Cesar Smaniotto Júnior

SoNDA: Um software para apoio à análise qualitativa de postagens de redes sociais

Trabalho de conclusão de curso apresentado
como parte dos requisitos para obtenção do
grau de Bacharel em Ciências da Computação

Universidade Federal de Santa Catarina

Departamento de Informática e Estatística

Curso de Ciências da Computação

Orientador: Vinicius Faria Culmant Ramos

Coorientador: Maurício Floriano Galimberti

Florianópolis - SC

2017/1

Cesar Smaniotto Júnior

SoNDA: Um software para apoio à análise qualitativa de postagens de redes sociais

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação

---

Vinicius Faria Culmant Ramos, D.Sc.
Orientador

Maurício Floriano Galimberti, Dr.
Coorientador

Patricia Vilain, Dra.
Membro da banca

Andrea Brandão Lapa, Dra.
Membro da banca

Florianópolis - SC
2017/1

Agradecimentos

Aos meus pais, pela educação que me foi dada, pelo apoio incondicional durante os meus estudos e pela compreensão pelos vários fins de semanas que passei longe de casa durante a graduação.

Aos meus amigos, tanto os de infância quanto os que conheci durante os anos da graduação, e ao meu irmão pela parceria e companheirismo.

Ao Professor Vinicius, pela amizade, confiança e orientação deste TCC.

À Professora Andrea, pela oportunidade de ingressar no grupo de pesquisa COMUNIC e desenvolver este trabalho. Agradeço aos colegas do núcleo de pesquisa pela convivência e por contribuírem com dicas e sugestões para o trabalho.

À CAPES pelo apoio financeiro para a realização deste trabalho, através do edital Obeduc pelo projeto "Rede de Políticas Públicas e Educação" coordenado pela Professora Tamara Egler do IPPUR/UFRJ.

Resumo

Com a intenção de observar a ação política nas redes sociais, o grupo de pesquisa COMUNIC do CED/UFSC elaborou uma metodologia de análise qualitativa de postagens de redes sociais. Esta metodologia especifica uma série de passos para filtrar um conjunto potencialmente grande de postagens extraídos das redes sociais, até obter algumas instâncias de interesse, onde é efetuada análise de conteúdo nos diálogos. O presente trabalho consiste no desenvolvimento de uma aplicação, denominada SoNDA (Social Network Data Analysis), que oferece suporte às três etapas de análise previstas pela metodologia. O trabalho não se restringe apenas na implementação do software, mas também visa contribuir com a metodologia criada pelo COMUNIC. Foi proposto e realizado experimentos com um método alternativo para a categorização das postagens, através de técnicas de mineração de textos. As postagens foram categorizadas utilizando algoritmos de classificação multirrótulo, que permitem associar mais de uma categoria a uma postagem.

Palavras-chave: software para análise de dados; análise de redes sociais; mineração de textos; classificação multirrótulo.

Abstract

With the intention of observing the political action in social networks, Comunic research group developed a methodology of qualitative analysis of social networking posts. This methodology specifies a series of steps to filter a potentially large set of posts extracted from social networks, to get some instances of interest, where it's made content analysis in the dialogues. This work is the development of an application, called SoNDA (Social Network Data Analysis), which supports the three stages of analysis provided by the methodology. The work is not restricted only in the implementation of the software itself, but also aims to contribute to the methodology created by COMUNIC. It was proposed and carried out experiments with an alternative method for the categorization of the posts, through techniques of text mining. Posts were categorized using multi-label classification algorithms, which allow you to associate more than one category with a post.

Keywords: data analysis software; social network analysis; text mining; multi-label classification.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
COMUNIC	Grupo de Pesquisa em Mídia-Educação e Comunicação Educacional
CRISP-DM	Cross Industry Standard Process for Data Mining
CSS	Cascading Style Sheets
CSV	Comma-separated values
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
KDD	Knowledge Discovery in Databases
MIT	Massachusetts Institute of Technology
NoSQL	Not Only SQL
PHP	PHP Hypertext Preprocessor
REST	Representational State Transfer
SEMMA	Sample, Explore, Modify, Model, and Assess
SMO	Sequential minimal optimization
SoNDA	Social Network Data Analysis
SQL	Structured Query Language
TF-IDF	Term frequency–inverse document frequency
UFSC	Universidade Federal de Santa Catarina
XML	eXtensible Markup Language

Sumário

1 Introdução

A web 2.0 estabeleceu uma série de novos conceitos, além de ter promovido o usuário como gerador de conteúdo, ao invés de um mero consumidor da informação fornecida pelos sites. Esses avanços contribuíram para a evolução dos serviços de redes sociais online. Para este trabalho, foi escolhida a seguinte definição para sites de redes sociais:

> Serviços baseados na web que permitem que indivíduos (1) construam um perfil público ou semi-público dentro de um sistema limitado, (2) articulem uma lista de outros usuários com quem eles compartilham uma conexão, e (3) visualizem e atravessem sua lista de conexões e aquelas feitos por outros dentro do sistema. A natureza e nomenclatura dessas conexões podem variar de site para site. (ELLISON et al., 2007, p. 211)

Segundo Boyd (2010), o perfil representa um indivíduo numa rede social e serve como local de interação com os demais integrantes da rede. As interações geralmente acontecem na forma de troca de mensagens privadas entre si, como um chat, e envio de mensagens para o perfil virtual do usuário, chamadas postagens/publicações. Quanto ao caráter público ou semipúblico de um perfil, isso é uma restrição que o usuário pode impor a respeito de quais informações expostas por ele estão disponíveis para os demais visualizarem, e de que forma os outros membros da rede virtual podem interagir com ele. A rede de conexões de um usuário é, usualmente, constituída por pessoas que fazem parte do seu círculo social e indivíduos que compartilham interesses em comum.

Serviços de *microblogging* normalmente têm as mesmas características de redes sociais citadas acima, porém costumam impor restrição ao tamanho do texto que o usuário publica. Publicações em microblog são, tipicamente, expressões sucintas do estado do usuário em relação a algum tema, podendo este ser qualquer mensagem relacionada ao usuário, como, por exemplo, algo do seu cotidiano, uma notícia, um evento ou outros interesses.

Java et al. (2007) diferenciam os *blogs* tradicionais de *microblogs* sob dois aspectos: primeiro, por incentivar a postagem de textos curtos, os *microblogs* requerem menos tempo do usuário para a geração de conteúdo; o outro fator que os difere é a frequência de atualização. Enquanto usuários de *blogs* costumam atualizar a sua página em um intervalo de dias, um usuário de *microblog* pode publicar várias atualizações por dia em seu perfil. Tornando, portanto, os serviços de *microblogging* uma ferramenta de comunicação extremamente ágil.

Uma das ferramentas de microblogging mais populares atualmente é o Twitter. O ranking Alexa[1] indica que é o oitavo site mais acessado no mundo. O Twitter permite os

[1] http://www.alexa.com/topsites. Acessado em 08 de outubro de 2016.

usuários enviarem mensagens de no máximo 140 caracteres, conhecidos como “tweets”. Segundo dados divulgados pela própria empresa, o Twitter possui cerca de 313 milhões de usuários ativos mensalmente[2]. Não há dados atualizados sobre o número médio de tweets publicados por mês, mas a última vez que a empresa divulgou essa estatística foi em novembro de 2013, como informa Oreskovic (2015). Naquela altura, cerca de 500 milhões de tweets eram publicados por mês em média, e a tendência é que este número tenha crescido.

Ampofo et al. (2015) apontam que a imensidão de conteúdo gerada por usuários de plataformas de redes sociais aliado ao surgimento de ferramentas e técnicas para armazenamento dos dados em tempo real e análise automatizada foram fundamentais para que o potencial dessas informações fossem exploradas por diversas áreas, como da inteligência de negócio, da área da saúde e das ciências sociais.

Ainda de acordo com Ampofo et al. (2015), as empresas exploram o conteúdo das mídias sociais com o objetivo de obter ganho comercial. Exemplos deste tipo de aplicação são os trabalhos de Mishne, Glance et al. (2006) e de Asur e Huberman (2010) onde foram utilizados análise de conteúdo e de sentimentos em dados de mídias sociais para predizer o faturamento de filmes com bilheteria. No campo das ciências sociais, Papacharissi e Oliveira (2011) aplicaram análise de conteúdo automatizada e análise de discurso em posts do Twitter relacionados aos protestos no Egito que levaram a renúncia do presidente Hosni Mubarak, a fim de identificar e estudar as características das notícias reportadas.

Também da área de ciências sociais e humanas, o COMUNIC (Grupo de Pesquisa em Mídia-Educação e Comunicação Educacional), do Centro de Ciências da Educação da Universidade Federal de Santa Catarina (UFSC) elaborou uma metodologia para investigar a ação política promovida nas redes sociais por grupos ativistas. O desafio dos pesquisadores do COMUNIC é aplicar este desenho de pesquisa tendo como entrada um conjunto de posts de redes sociais, coletados em momentos de grande mobilização social e analisá-los qualitativamente, para identificar fatores e circunstâncias relevantes nos espaços sociais virtuais da internet para a formação crítica.

O presente trabalho visa dar suporte tecnológico aos pesquisadores, através da criação de um software que dê suporte às atividades especificadas na metodologia. Uma vez que o volume de dados extraído de redes sociais tende a ser grande, é necessário um software para auxiliar os pesquisadores em rotinas comuns do estudo, como na organização dos dados da pesquisa, filtragens e aquisição de novos dados a partir dos existentes. Este trabalho não se limita ao desenvolvimento do software que solucione o problema do COMUNIC, mas também visa aprimorar métodos de filtragem utilizados pela metodologia, propondo e realizando experimentos com um outro método para categorização das postagens, empregando técnicas de mineração de texto.

[2] https://about.twitter.com/company. Acessado em 08 de outubro de 2016.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolvimento de uma aplicação web capaz de carregar e processar postagens de sites de redes sociais, possibilitando a análise qualitativa seguindo os passos descritos na metodologia de análise de redes sociais criada pelo COMUNIC/UFSC.

1.1.2 Objetivos específicos

- Análise comparativa das ferramentas de mercado para análise de dados qualitativos, com foco no suporte a análise de dados de redes sociais.
- Pesquisar e definir um processo de desenvolvimento de *software* que se adeque a uma equipe composta por pesquisadores de diversas áreas e aos aspectos deste projeto.
- Implementar um processo de descoberta de conhecimento em textos, com o objetivo de categorizar automaticamente posts de redes sociais em mais de uma categoria.

1.2 Organização do documento

O Capítulo 2 aborda conceitos relacionados ao problema da análise de dados de redes sociais que o SoNDA resolve, bem como apresenta a base teórica a respeito dos experimentos com mineração de texto realizados no trabalho. O Capítulo 3 apresenta o comparativo das principais funcionalidades do SoNDA entre outras soluções de *softwares* já consolidadas no mercado.

O Capítulo 4 aborda o processo de desenvolvimento do SoNDA, sua arquitetura e as tecnologias utilizadas no sistema desenvolvido. O Capítulo 5 trata da implementação do sistema, abordando de que forma os requisitos foram implementados. No Capítulo 6, são detalhados os experimentos realizados com mineração de texto para otimizar uma das etapas da metodologia do COMUNIC. Por fim, no Capítulo 7 são apresentadas as conclusões e sugestões de trabalhos futuros.

2 Fundamentação Teórica

Nesta seção são apresentadas algumas definições importantes para a compreensão do restante do trabalho. Os três primeiros conceitos expostos estão fortemente relacionados ao sistema desenvolvido, uma vez que a análise de dados qualitativos provenientes de redes sociais é o propósito do SoNDA e o mesmo situa-se no segmento de software CAQDAS (*Computer assisted qualitative data analysis software*). As demais seções fornecem a base teórica para os experimentos com mineração de textos.

2.1 Análise de dados qualitativos

São considerados dados qualitativos, segundo Taylor e Gibbs (2010), informações não-numéricas, comumente em formato não-estruturado como textos, imagens, áudios e vídeos. A fonte mais comum de dados qualitativos envolve o que pessoas fizeram ou falaram. Alguns exemplos são: transcrições de entrevistas, documentos (jornais, revistas, livros, páginas *web*), fotografias e filmes. Taylor e Gibbs (2010) conceituam análise de dados qualitativos como o conjunto de processos e procedimentos aplicados a dados qualitativos de modo a obter alguma explicação, compreensão ou interpretação de fenômenos e pessoas investigadas.

De acordo com Taylor e Gibbs (2010), as duas atividades mais comuns no processo de análise de dados qualitativos são a escrita e a identificação de temas. Enquanto a escrita está presente em quase todas as abordagens de análise, a identificação de temas pode não ser uma exigência, porém está presente na grande maioria das metodologias de análise qualitativa.

A escrita envolve escrever sobre os dados coletados e o que são encontrados nos mesmos. Geralmente são registradas ideias analíticas, porém em muitos casos, são sumários ou resumos dos dados. A identificação de temas ocorre através da codificação. Esta tarefa consiste na busca por trechos de texto ou outro tipo de dado que pertence a uma categoria ou ideia temática e a atribuição de um código que a identifique. A aplicação de códigos aos fragmentos de dados permite ao pesquisador recuperar e examinar todos os itens codificados com alguma ideia temática, possibilitando-o efetuar comparações e identificar padrões.

2.2 Computer Assisted Qualitative Data Analysis Software (CAQDAS)

CAQDAS (acrônimo de *Computer assisted qualitative data analysis software*, em português *Software* de apoio à análise de dados qualitativos) são, segundo Silver e Lewins (2014), programas de computador que dispõem de ferramentas que auxiliam a análise qualitativa em textos, áudios, vídeos, imagens e outros tipos de dados não-estruturados. Cada *software* possui suas particularidades e sofisticação de recursos, porém, algumas características são comuns a todos eles. Silver e Lewins (2014) enumeram as funcionalidades mais comuns encontradas em ferramentas CAQDAS:

- Disponibilizar uma estrutura para centralizar e gerenciar as fontes de dados de um projeto de pesquisa.
- Permitir a inserção de anotações e anexá-las a fragmentos de dados, para registrar lembretes, comentários ou conclusões obtidas.
- Fornecer mecanismos de pesquisa no texto, como busca por palavras, frases ou coleção de palavras.
- Permitir conectar trechos de dados, indicando algum processo ou relação entre os itens ligados.
- Criação de esquemas de codificação: definição dos códigos, que geralmente representam temas, instâncias e de sua estrutura, que pode ser hierárquica ou não-hierárquica.
- Codificação: atribuição de um ou mais códigos a segmentos de dados.
- Recuperação dos segmentos codificados, de modo que permita o pesquisador verificar o que e o quanto foi codificado até o momento, além de identificar padrões, relações e possíveis inconsistências nos itens codificados.
- Prover mecanismos para questionar a base de dados, onde o pesquisador possa verificar suas hipóteses, buscar por padrões e tirar conclusões sobre os dados analisados.
- Permitir a geração de relatórios, a fim de obter uma cópia do estado da pesquisa. Pode ser disponibilizado em formato de impressão ou para download, que viabiliza trabalhar com os dados em outros *softwares*, como sistemas de planilhas eletrônicas.

Reis, Costa e Souza (2016) relatam que, em 2015, já existiam mais de 40 *softwares* para apoio a análise de dados qualitativos. Entre os sistemas gratuitos e/ou de código aberto, estão entre os de maior relevância: Aquad, Cassandre, CATMA - Computer Aided Textual Markup & Analysis, CAT - Coding Analysis Toolkit, Compendium, ELAN, FreeQDA,

LibreQDA, QDA Miner Lite (versão gratuita do QDA Miner com menos recursos), RQDA, TAMS Analyzer e o Transana (para áudio e vídeo).

Ainda de acordo com Reis, Costa e Souza (2016), os sistemas proprietários possuem atualmente a maior fatia de mercado no que se refere a *softwares* para análise de dados qualitativos. Entre os que necessitam de instalação, destacam-se ATLAs.ti, Coding Analysis Toolkit (CAT), ConnectedText, Dedoose, HyperRESEARCH, MAXQDA, NVivo, QDA Miner, Qiqqa, Quirkos, Saturate, XSight. Entre as aplicações *web*, destacam-se: Dedoose, o LibreQDA, o QCAmap e o webQDA. No Capítulo 3, alguns destes pacotes de *software* são detalhados e comparados.

2.3 Dados de redes sociais online

Segundo França et al. (2014), os dois principais métodos para extração de grandes volumes de dados de redes sociais são através de APIs disponibilizadas pelas plataformas de redes sociais e *crawlers* desenvolvidos pelos usuários. Uma API (*Application Programming Interface*, em português Interface de Programação de Aplicações) no contexto de aplicações *web*, refere-se ao conjunto de funcionalidades disponibilizados por um sistema *web*, cuja interação com os mesmos se dá através de requisições e respostas HTTP , tipicamente no formato XML (*eXtensible Markup Language*) ou JSON (*JavaScript Object Notation*). *Crawlers* são robôs que navegam sistematicamente por páginas da *web* com algum propósito, neste caso, percorrem as páginas com o objetivo de extrair o seu conteúdo.

De acordo com França et al. (2014), normalmente as APIs que recuperam dados de redes sociais se enquadram em duas categorias: as que permitem pesquisar e coletar dados publicados no passado (respeitando um limite de tempo) e as que se baseiam no conceito de *streaming*, onde a coleta é feita em tempo real. Benevenuto, Almeida e Silva (2011) destacam que a grande vantagem do uso de APIs, é que os dados são retornados em um formato estruturado, como XML ou JSON. A Figura 1 mostra um exemplo no formato JSON da resposta de uma requisição à API do Twitter (https://api.twitter.com/1.1/search/tweets.json?q=passe%20livre), que efetua a pesquisa por tweets contendo o termo “passe livre“. Vários atributos presentes no resultado original foram omitidos por simplificação.

Em comparação com as APIs, os *crawlers* possuem algumas desvantagens. Primeiro, é necessário conhecer a estrutura interna da página em que os dados serão capturados, a fim de programar o *crawler*. Além disso, este mecanismo pode exigir que sejam percorridas muitas páginas de um website para obter a mesma quantidade de informação que seria obtida com uma ou poucas requisições para uma API. Sendo assim, a utilização de *crawlers* é encorajada em situações onde é necessário coletar dados em que não há API disponível ou para contornar limites de requisições impostas pelas mesmas. O seção 5.9 trata de um

Figura 1 – Resposta simplificada da API do Twitter


```
{
"created_at":"Mon Mar 27 10:24:01 +0000 2017",
"id":846306805680892416O,
"text":"Exemplo tweet com termo passe livre",
"user": {
    "id":20280595,
    "name":"Cesar Smaniotto Jr",
    "screen_name":"tcc_cesar",
    "location":"Florianópolis",
    "description":"Descrição do perfil",
    "followers_count":997,
    "friends_count":233,
    "listed_count":28,
    "created_at":"Fri Feb 06 23:03:36 +0000 2009",
    "favourites_count":1349,
    "time_zone":"Brasilia"
    },
"is_quote_status":false,
"retweet_count":0,
"favorite_count":2,
"favorited":false,
"retweeted":false,
"lang":"pt"
}
```


Fonte: Produzido pelo autor

caso onde a implementação de um crawler se mostra adequada e eficaz para resolver o problema de coletar dados de plataformas de redes sociais.

Existem também, aplicações que abstraem as APIs de redes sociais e simplificam a tarefa de extração de dados, como é o caso do TAGS[1] e do Analytics for Twitter[2].

2.4 Metodologia de análise de redes sociais do COMUNIC

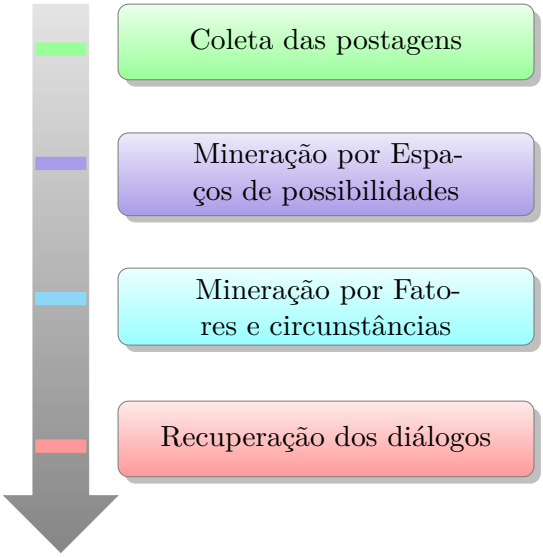
Com o propósito de investigar a ação política motivada pelo ativismo nas redes sociais, o núcleo UFSC do COMUNIC, formado por pesquisadores da área de ciências sociais e humanas, elaborou uma metodologia de análise qualitativa de postagens de redes sociais. A aplicação deste estudo pretende identificar fatores e circunstâncias relevantes para a formação crítica nas redes sociais. Segundo os proponentes, o objetivo da pesquisa

[1] https://tags.hawksey.info/
[2] https://www.microsoft.com/en-us/download/details.aspx?id=26213

é orientar professores e educadores em suas práticas, dentro e fora da escola. O entregável final resultante da aplicação desta metodologia servirá para a elaboração de guias para a formação de professores, com o referencial de fatores e circunstâncias para a formação crítica de sujeitos na cultura digital.

Esta seção apresenta uma visão geral da metodologia criada pelo COMUNIC, onde será introduzida, brevemente, cada etapa e alguns conceitos definidos pelos autores. Os detalhes podem ser verificados no trabalho de Lapa et al. (2015).

Figura 2 – Etapas da metodologia de análise de redes sociais do COMUNIC



Fonte: Produzido pelo autor

A Figura 2 apresenta o diagrama com as etapas da metodologia. Enquanto a primeira etapa consiste na coleta dos dados iniciais da pesquisa, as demais são fases de análise, onde ao final de cada uma destas fases, o conjunto de dados de interesse é reduzido, de modo que após a última etapa de mineração permaneçam apenas os posts mais relevantes para pesquisa, onde serão submetidos a análise de conteúdo nos diálogos. Os itens a seguir detalham as atividades de cada etapa.

- Etapa 1: consiste na coleta de postagens de redes sociais por um determinado período de tempo. São obtidos posts que contenham palavras-chaves previamente escolhidas e relacionadas ao tema da análise. Além da mensagem em si, outras informações como: nome de usuário, data de publicação e geolocalização (quando disponível), são capturadas. Após a aquisição dos dados, o processo de análise inicia. As próximas três etapas visam obter resultados cada vez mais qualitativos ao término do processo.

- Etapa 2: nesta fase é realizado o primeiro filtro no conjunto de dados, conjunto este que é chamado de “dataset” pelos proponentes. Os posts são separados em

categorias especificadas pelos pesquisadores, que a metodologia define como Espaços de Possibilidades. Segundo os autores, Espaços de Possibilidades são momentos com potencial para revelar processos que devem ser observados de acordo com os propósitos da pesquisa. Esta etapa se divide em duas atividades. A primeira é a identificação das categorias. A partir da leitura de amostras extraídas do conjunto de dados, os pesquisadores formulam as categorias. São obtidas três amostras, do início, meio e fim do dataset, ordenado por data de publicação crescente. Cada categoria é composta pela descrição do seu significado e por uma biblioteca de termos e palavras-chaves. No segundo estágio, todo o dataset é filtrado com base na coleção de palavras-chave de cada espaço de possibilidade. Esta filtragem no conjunto de dados é feita verificando a ocorrência de um termo ou palavra de um Espaço de Possibilidade no texto da postagem. É importante ressaltar que as categorias não são mutuamente exclusivas, portanto, um post pode pertencer a nenhuma ou várias categorias. A metodologia inicialmente propõe esse método de ocorrência de termo no texto para realizar a mineração por espaços de possibilidade. O presente trabalho introduz um segundo método para a filtragem por espaços de possibilidades, não previsto na metodologia, que é através da aplicação de técnicas avançadas de mineração de textos.

- Etapa 3: neste estágio, o conjunto de dados ainda é volumoso e necessita de mais um processo de filtragem até atingir uma quantidade manipuláveis de posts para proceder a análise de diálogos. A metodologia define esta etapa como mineração por Fatores e Circunstâncias, que são um segundo conjunto de categorias, definidos anteriormente através de revisão bibliográfica, que descrevem os processos buscados dentro dos espaços de possibilidade. Cada fator e circunstância é composto pela sua descrição e pelas métricas, que servem como guia para o analista identificar se um post dentro dos espaços possibilidades pertence ou não a um dado fator e circunstância. Diferente da mineração por espaços de possibilidade, onde a categorização é feita de forma automática, nesta etapa o processo de categorização é feito manualmente pelo analista.

- Etapa 4: definida pelos autores como fase de compilação dos diálogos. Nesta fase da análise, o conjunto de dados foi reduzido de modo a permanecer apenas os posts de maior interesse para a pesquisa. A partir de posts potenciais identificados e anotados na fase anterior, são recuperados todo o rastro de diálogo no qual um dado post faz parte. O rastro de conversação é composto pela troca de mensagens encadeadas entre um grupo de usuários. No caso do Twitter, a conversação é caracterizada pelos *replies* (respostas) a um tweet, enquanto no Facebook, são os comentários de uma postagem. Com os diálogos recuperados, o analista pode efetuar análise de conteúdo e buscar respostas para suas investigações.

2.5 Mineração de textos

A Mineração de textos, também conhecida como Mineração de Dados Textuais ou Descoberta de Conhecimento em Textos, é um termo que refere-se ao processo de extração de padrões ou conhecimento não-trivial, interessante e previamente desconhecido de documentos de texto (Tan et al. (1999)). É nítido que a mineração de textos está fortemente relacionada com a mineração de dados, cujo processo também visa extrair padrões ou conhecimento relevante em bancos de dados.

Soares (2013) enfatiza que a principal diferença entre os dois processos é que a Mineração de Textos é um processo que extrai conhecimento de bases de dados textuais, ou seja, documentos em linguagem natural que possuem pouca ou nenhuma estrutura. Desta forma, a fase de pré-processamento destina-se a identificação e extração de características representativas do texto, para que seja possível representá-lo de maneira estruturada. O autor ainda pontua que, embora haja distinção na estrutura dos dados de entrada no processo de Mineração de Dados e de Texto, ambos utilizam técnicas de aprendizado de máquina semelhantes.

De acordo com Miner (2012), a mineração de texto é dividida nas seguintes sete áreas de atuação: Pesquisa e recuperação da informação, Clusterização de documentos, Classificação de documentos, Web Mining, Extração da informação (IE), Processamento de linguagem natural (NLP) e Extração de conceitos. Dentre as áreas citadas, a que melhor atende aos propósitos deste trabalho é a de classificação de documentos, uma vez que o sistema pretende classificar posts de redes sociais em categorias de Espaços de Possibilidade, com base em um conjunto de instâncias rotuladas.

2.5.1 Classificação de textos

A classificação de texto é definida por Miner (2012) como um processo que atribui uma classificação a um documento de texto a partir de um conjunto predefinido de rótulos de classe. De acordo com Soares (2013), a abordagem mais comum para a classificação de texto é através de técnicas de Aprendizagem de Máquina, onde um processo indutivo cria um modelo de classificação com base nas características de alguns exemplos pré-rotulados, para classificar os itens onde a classe é desconhecida.

Soares (2013) cita dois tipos de classificadores: monocategórico (ou monorrótulo) e multicategórico (ou multirrótulo). O classificador monocategórico atribui exatamente uma classe a um dado documento. Entretanto, o classificador multicategórico, é capaz de atribuir mais de uma classe a um documento. A categorização de tweets feita neste trabalho, utiliza classificadores multirrótulo, pois, conforme especificado na metodologia apresentada anteriormente, uma postagem pode pertencer a mais de uma categoria.

2.5.2 Métodos para classificação multirrótulo

Os métodos para classificação multirrótulo podem ser agrupados em duas categorias, de acordo com Tsoumakas e Katakis (2006).

- Métodos baseados na transformação do problema: decompõem o problema de classificação multirrótulo em um ou mais problemas de classificação monorrótulo e aplicam algoritmos de classificação monorrótulo para cada subproblema.
- Métodos baseados na adaptação do algoritmo: extendem algoritmos de mineração monorrótulo para lidar diretamente com a classificação multirrótulo.

Os dados da Tabela 1 e o conjunto de classes $C = \{D, IS, COO, SC\}$ serão usados como exemplo para compreensão de alguns métodos de classificação multirrótulo.

Tabela 1 – Exemplos de instâncias com múltiplos rótulos

Instância	Conjunto de classes
1	{D,IS}
2	{SC}
3	{D,IS,COO}
4	{D,COO}

Fonte: Produzido pelo autor

O *Binary Relevance* (Relevância Binária) é uma das técnicas mais populares de transformação do problema, que cria um classificador binário monorrótulo para cada rótulo de classe do conjunto de dados de treino. Com estes classificadores binários, cada classe assume dois valores, positivo ou negativo. Uma nova instância classificada por este método, é dada pela união dos rótulos classificados como positivo por cada classificador binário. Sorower (2010) comenta que embora esta abordagem seja utilizada em muitas aplicações, a técnica é criticada por não levar em consideração a dependência entre as classes. A partir da Tabela 1, o método *Binary Relevance* produziria e treinaria os conjuntos de dados mostrados na Tabela 2.

O *Label Powerset* (Conjunto Potência) considera cada subconjunto de rótulos dos dados de treinamento como uma nova classe. Portanto, este método transforma um problema de classificação multirrótulo em um único problema de classificação monorrótulo, onde uma nova instância é classificada em uma dessas novas classes, que na verdade representa um conjunto de classes. O método Label Powerset aplicado aos dados da Tabela 1 geraria a combinação de classes da Tabela 3, onde k indica o tamanho de cada subconjunto de classes. A Tabela 4 apresenta a transformação aplicada no conjunto de dados da Tabela 1 pelo *Label Powerset*.

Tabela 2 – Conjunto de dados produzidos pelo método Binary Relevance

Inst	Rótulo
1	D
2	¬D
3	D
4	D

Inst	Rótulo
1	¬SC
2	SC
3	¬SC
4	¬SC

Inst	Rótulo
1	IS
2	¬IS
3	IS
4	¬IS

Inst	Rótulo
1	¬COO
2	¬COO
3	COO
4	COO

Fonte: Produzido pelo autor

Tabela 3 – Combinação de classes produzida pelo método Label Powerset

$k = 1$	$k = 2$	$k = 3$	$k = 4$
{D}	{D,IS} {D,COO} {D,SC}	{D,IS,COO} {D,IS,SC} {D,COO,SC}	{D,IS,COO,SC}
{IS}	{IS,COO} {IS,SC}	{IS,COO,SC}	
{COO}	{COO,SC}		
{SC}			

Fonte: Produzido pelo autor

Tabela 4 – Transformação no conjunto de dados pelo método Label Powerset

Instância	Conjunto de classes
1	D,IS
2	SC
3	D,IS,COO
4	D,COO

Fonte: Produzido pelo autor

Embora o *Label Powerset* considere a dependência entre as classes, este método possui duas desvantagens, de acordo com Sorower (2010). A complexidade computacional é limitada superiormente por $min(n, 2^{|C|})$, onde n é o número de instâncias do conjunto de dados e $\mid C \mid$ a quantidade de classes existentes no respectivo conjunto de instâncias (antes da transformação). Na prática, a complexidade seria menor que 2^k, mas para valores de n e $\mid C \mid$ grandes, isto pode ser um problema. Outra questão envolvendo esta técnica, é o alto número de classes geradas que poderiam estar associadas a poucos, ou até mesmo nenhuma instância do conjunto de dados, levando ao problema do desbalanceamento entre classes.

2.5.3 Avaliação de classificadores multirrótulo

De acordo com Sorower (2010), para problemas envolvendo classificação monorrótulo, há uma série de métricas que podem ser empregadas para avaliar o modelo gerado, como a acurácia, precisão, *recall* e *f-measure*. Entretanto, na classificação multirrótulo as predições para uma instância são um conjunto de rótulos, e a predição pode estar totalmente incorreta, parcialmente correta (com diversos níveis de corretude) ou totalmente incorreta. Nenhuma das métricas usadas em classificadores monorrótulo são capazes de capturar estes comportamentos na sua fórmula original. Desse modo, diferentes medidas para avaliação de classificadores multirrótulo foram criadas ou adaptadas de métricas para classificadores monorrótulo.

Tsoumakas, Katakis e Vlahavas (2009) classificam estas métricas em dois grupos: medidas baseadas em bipartições e medidas baseadas em *rankings*. Há duas abordagens para medidas baseadas em bipartições: baseadas em exemplo, que avaliam as bipartições considerando todas as instâncias do conjunto de testes e, baseadas em rótulo, onde o processo de avaliação é decomposto e cada rótulo é avaliado individualmente. As medidas baseadas em *ranking* avaliam o *ranking* dos rótulos com respeito ao conjunto de dados multirrótulo.

Para a definição dessas medidas, serão usadas as mesmas convenções de Tsoumakas, Katakis e Vlahavas (2009). Considere um conjunto de testes com instância com múltiplos rótulos (x_i, Y_i), $i = 1 \ldots m$, onde $Y_i \subseteq L$ é o conjunto de rótulos verdadeiros e $L = \{\lambda_j : j = 1 \ldots q\}$ é o conjunto de todos os rótulos. Dada a instância x_i, o conjunto de rótulos preditos por um classificador multirrótulo é denotado por Z_i. O *ranking* predito para um rótulo λ é denotado por $r_i(\lambda)$. O rótulo mais relevante recebe o maior *ranking*, enquanto o menos relevante recebe o menor *ranking*. A seguir serão apresentadas as medidas utilizadas para avaliar os classificadores multirrótulo deste trabalho.

- Acurácia: é uma medida baseada em exemplos, onde a acurácia de cada instância predita é definida como a proporção de rótulos corretamente preditos em relação ao número total (preditos e real) de rótulos para essa instância. (Sorower (2010)). A acurácia geral é dada pela média entre as acurácias de cada instância.

$$Acurácia = \frac{1}{m} \sum_{i=1}^{m} \frac{| Y_i \cap Z_i |}{| Y_i \cup Z_i |} \tag{2.1}$$

- *Hamming Loss*: métrica baseada em exemplos, que avalia quantas vezes o rótulo de uma instância é classificado incorretamente, ou seja, quando o rótulo não pertence à instância e é predito ou quando o rótulo pertencente à instância não é predito. Quanto menor o valor de *Hamming Loss*, melhor o desempenho do classificador. O

símbolo Δ denota a diferença simétrica entre o conjunto de rótulos da instância e conjunto de rótulos preditos.

$$HammingLoss = \frac{1}{m}\sum_{i=1}^{m}\frac{Y_i \Delta Z_i}{\mid L \mid} \tag{2.2}$$

- *One-error*: métrica baseada em *ranking*, que avalia quantas vezes o rótulo com maior *ranking* não está presente no conjunto de rótulos da instância. Quanto mais próximo de zero o valor da métrica, melhor a performance do classificador.

$$One - error = \frac{1}{m}\sum_{i=1}^{m}\delta(argmin_{\lambda \in L} r_i(\lambda)) \tag{2.3}$$

onde $\delta(\lambda) = 1$ se $\lambda \notin Y_i$, 0 caso contrário

2.6 Processo de descoberta de conhecimento em textos

Segundo Miner (2012), para o processo de mineração de dados existem diversas metodologias que são relativamente maduras e comumente utilizadas, como o CRISP-DM (Cross Industry Standard Process for Data Mining), KDD (Knowledge Discovery in Databases) e SEMMA (Sample, Explore, Modify, Model, and Assess). Porém, para a mineração de textos não há um modelo de processo consagrado.

Miner (2012) e Silva, Ferneda e Prado (2007) propuseram um processo para mineração de textos baseado na metodologia CRISP-DM. Aranha, Vellasco e Passos (2007) seguiram a tendência de outros trabalhos da área e definiram um processo para mineração de texto que consiste de quatro etapas: coleta, pré-processamento, mineração e análise. A principal diferença para as abordagens mencionadas acima, inspiradas no CRISP-DM, é que estas incluem uma etapa inicial de compreensão do problema e dos objetivos da mineração e uma última etapa de consolidação do conhecimento extraído. Dentre estas três opções, foi escolhida a abordagem para mineração de textos definida por Aranha, Vellasco e Passos (2007) para ser detalhada nesta seção e aplicada no trabalho. Esta opção se deve ao fato de ser um processo mais enxuto. As outras duas abordagens preveem algumas etapas que são dispensáveis no contexto deste trabalho. Por exemplo, a etapa de integração dos resultados acaba se tornando dispensável, tendo em vista que, não faz parte do escopo do trabalho integrar os resultados da mineração ao sistema. Foram realizados apenas experimentos envolvendo a classificação de posts no contexto da metodologia do COMUNIC. A Figura 3 apresenta um fluxograma das etapas do processo de descoberta de conhecimento em textos de Aranha, Vellasco e Passos (2007), a seguir, cada uma será brevemente abordada.

Figura 3 – Processo de mineração de texto

Fonte: Adaptado de Aranha, Vellasco e Passos (2007)

1. Coleta dos dados: consiste em formar a base de documentos textuais na qual será efetuada a mineração. Esta base pode ser estática, quando o conjunto de documentos não muda, ou dinâmica, onde robôs autônomos adicionam novos documentos. Os principais desafios da etapa de coleta são descobrir onde os documentos estão armazenados e como obter documentos relevantes ao domínio de conhecimento. As três principais fontes para busca por documentos são: nos diretórios de pastas do disco rígido, em tabelas de diferentes bancos de dados, e na web. A coleta de dados na web comumente é realizada por *crawlers*.

2. Pré-processamento: Consiste na aplicação de um conjunto de transformações numa coleção de textos, com o objetivo de melhorar sua qualidade e convertê-los para um formato estruturado, que permita a indexação ou a execução de algoritmos de mineração de dados.

   Muitas das transformações durante o pré-processamento envolvem técnicas de Processamento de Linguagem Natural. Algumas rotinas de Processamento de Linguagem Natural são a tokenização, a remoção de *stopwords*, aplicação de técnicas de *stemming* e classificação gramatical. Para realizar a descoberta de conhecimento em textos neste trabalho, foram utilizadas as técnicas de tokenização e *stopwords*, as quais são descritas a seguir.

   a) Tokenização: é o processo de dividir o texto em grupos de caracteres, conhecidos como tokens. Geralmente, cada token corresponde a uma palavra do texto, mas pode ser também mais de uma palavra, um símbolo ou um caractere de pontuação. Na abordagem mais comum, espaços em branco e caracteres de pontuação são utilizados para delimitar um token. Os caracteres delimitadores fazem parte da lista de tokens de um texto, porém os espaços em branco são descartados.

   b) Remoção de palavras não-discriminantes (stopwords): de acordo com Soares (2013) o objetivo desta etapa é reduzir a alta dimensionalidade dos dados textuais eliminando palavras com baixo poder discriminatório, conhecidas como stopwords. Estes termos com baixo poder de relevância costumam ser pronomes, artigos, conjunções e preposições. Segundo Miner (2012), esta remoção é possível

sem perda de informação pois na grande maioria dos algoritmos e tarefas de mineração de texto, essas palavras pouco impactam no resultado final do algoritmo.

3. Indexação: refere-se às técnicas aplicadas para armazenar e recuperar com eficiência o conteúdo dos textos.

4. Mineração: esta etapa envolve escolher quais algoritmos de mineração de dados serão aplicados. A decisão está relacionada ao objetivo do processo de mineração, que determina a abordagem de aprendizagem de máquina que deve ser aplicada ao problema.

5. Análise: após a fase de mineração de dados, a etapa de análise avalia e interpreta os resultados obtidos. De acordo com Soares (2013), são empregadas métricas de avaliação de desempenho para analisar os resultados de um processo de mineração de textos.

3 Revisão de softwares para análise qualitativa

Neste capítulo, é apresentado um comparativo entre alguns *softwares* do tipo CAQDAS disponíveis no mercado. A comparação não se resume a verificar as funcionalidades que cada um tem, mas, também, apresentar aquelas que estão implementadas no SoNDA e suas características, que são fundamentais para viabilizar a análise de dados de redes sociais. Assim como o SoNDA, os demais *softwares* analisados oferecem a maioria dos recursos citados na seção 2.2, como permitir a criação e atribuição de códigos e a recuperação dos segmentos codificados, e, portanto, não serão avaliadas neste trabalho.

O comparativo foi feito com base na versão mais recente e com maior número de recursos de cada *software*, executando em ambientes Windows ou Linux. Foram avaliados o NVivo 11 na versão Plus, Atlas.ti 7, Dedoose 7.1.3, MAXQDA 12 na versão Pro e webQDA 3, apontados por Reis, Costa e Souza (2016) como as alternativas de software para análise de dados qualitativos que mais se destacam, pela sua popularidade, potencialidades ou suporte ao trabalho em equipe. Cabe ressaltar que, todos os programas citados são *softwares* proprietários e para os fins deste trabalho, foi utilizada a versão de avaliação de cada um. As características a serem comparadas são as seguintes:

- Extração de dados de redes sociais: esta é uma característica que seria desejável, pois dispensaria o uso de outras ferramentas para realizar a coleta dos dados.

- Importação de dados de redes sociais via arquivo, como uma planilha do Excel ou no formato CSV (Comma-separated values). O CSV é um formato de representação de uma tabela em arquivo texto, onde as colunas são separadas por vírgula. Indiferente se a aplicação dá ou não suporte à extração, é necessário que ela permita a importação de dados de redes sociais obtidos previamente utilizando outras ferramentas que solucionam esse problema, como o TAGS[1].

- Codificação automática de texto: possuir esta abordagem de codificação é imprescindível, pois geralmente ao realizar análise sob dados de redes sociais, o conjunto de dados de entrada é muito grande, inviabilizando a execução manual desta tarefa. A codificação automática compreende desde técnicas simples como codificar itens pela ocorrência de frases e palavras-chave até métodos mais avançados, que aplicam mineração de texto.

[1] https://tags.hawksey.info/

- Permitir trabalho colaborativo simultâneo e em tempo real: esta funcionalidade é fundamental para que uma equipe de pesquisadores possa compartilhar o mesmo projeto e cada um poder trabalhar paralelamente.

As seções a seguir apresentam uma visão geral sobre cada sistema, além de relatar se cada um dos quatro aspectos avaliados estão presentes no sistema, e, caso estejam, detalhar como é o seu funcionamento. O comparativo avaliando aspectos mais gerais de cada *software*, pode ser conferido em Reis, Costa e Souza (2016).

3.1 NVivo

O NVivo é um software desenvolvido pela QSR International[2] disponível para Windows e Mac. O NVivo possui três versões distintas para Windows. A versão que contém os recursos mais avançados é a Pro, sendo esta a que será detalhada. O NVivo Pro suporta uma ampla variedade de tipos de arquivo como textos, imagens, vídeos, áudios, dados de redes sociais como Twitter e Facebook, além de possibilitar a análise de redes.

O NVivo permite a extração de conteúdo de páginas *web* e redes sociais como Facebook, Twitter e LinkedIn. Para esta tarefa, é necessária a instalação de uma extensão no navegador, o NCapture. Aqui, será aprofundado como o processo é feito para coletar dados do Twitter, o passo-a-passo para as demais fontes podem ser conferidos na documentação do NCapture[3]. Os dados coletados no Twitter podem ser armazenados no NVivo de duas formas: como um PDF da página contendo os tweets ou como um conjunto de dados, onde é possível ordená-lo, filtrá-lo e codificá-lo. O NCapture oferece diversas opções de coleta, como extração de tweets de uma lista de usuários, busca por *hashtag* ou palavras-chave, ou por uma lista de tweets marcados como favorito. A extensão porém, não possui a opção de recuperar automaticamente todo o diálogo no qual um determinado tweet se encontra. Além da importação direta, o NVivo também disponibiliza a opção de importar um arquivo de texto no formato CSV contendo postagens de redes sociais e armazená-lo como um conjunto de dados, permitindo a manipular individualmente cada post.

Em relação à codificação automática, o NVivo oferece várias possibilidades. É possível efetuar a busca por frases e palavras-chave pesquisando cada termo individualmente ou combinando múltiplos termos usando operadores booleanos. Além disso, é possível especificar um termo de busca com caracteres-curinga asterisco (*) e interrogação (?). O símbolo * quando inserido entre uma palavra, faz a correspondência e substituição por zero ou múltiplos caracteres. Exemplo: manifesta* inclui nos resultados, manifesta, manifestação, manifestar. Já o símbolo ? faz a correspondência e substituição por exatamente um

[2] http://www.qsrinternational.com/
[3] http://help-ncapture.qsrinternational.com/desktop/welcome/welcome.htm

caractere. Exemplo: manifesta? inclui nos resultados manifestar, mas não inclui manifesta e manifestação.

Além da busca e codificação por termos e expressões, a ferramenta também permite codificar automaticamente documentos de textos ou conjuntos de dados em temas e sentimentos. Esta tarefa não requer nenhum conhecimento de mineração de dados, análise de texto, métodos estatísticos e suas terminologias por parte do usuário.

A codificação por temas ocorre da seguinte forma: as fontes de dados selecionadas são submetidas a um processo que detecta locuções substantivas (por exemplo, "terminal de ônibus") e conta suas ocorrências. Os temas são agrupados hierarquicamente, e os resultados são apresentados como um código para o tópico central e sub-códigos para cada tema específico dentro desta estrutura hierárquica. O NVivo apresenta os temas mais relevantes identificados e fica a cargo do usuário escolher quais destes códigos ele deseja aplicar.

Na codificação por sentimentos, um sistema de pontuação é utilizado. Cada palavra que possui sentimento detém uma pontuação predefinida. Um registro ou fragmento de texto é codificado para um conjunto de cinco códigos de sentimentos: muito negativo, negativo, neutro, positivo e muito positivo. Esse processo analisa cada palavra individualmente, sem levar em consideração o contexto da frase ou texto a que pertence. Os códigos que serão aplicados em um registro ou fragmento vão depender da presença de palavras associadas a um dos cinco nós citados. Portanto, um item pode ser codificado pelo sistema como positivo e negativo simultaneamente. Porém, o usuário pode realizar uma consulta pelos itens codificados e remover os códigos em casos que julgar necessário. Cabe ressaltar que no NVivo não é possível codificar as fontes de dados automaticamente a partir de códigos definidos pelo pesquisador.

Para realizar análises em tempo-real, em conjunto com outros pesquisadores, é necessário adquirir uma licença extra, que só está disponível para Windows.

3.2 Atlas.ti

O Atlas.ti[4] é um *software* para análise de dados qualitativos mantido pela ATLAS.ti Scientific Software Development, com versões para Windows, MacOS e para dispositivos móveis, como iPad e Android. Assim como outras aplicações similares, o Atlas.ti permite analisar uma ampla variedade de arquivos multimídia, entretanto como diferencial, o Atlas.ti suporta dados geoespaciais, a partir da integração com o Google Earth.

O Atlas.ti não possui a funcionalidade de coleta direta de dados do Twitter e outras redes sociais. No entanto, permite a importação de arquivos no formato CSV, porém não

[4] http://atlasti.com/

de maneira genérica como uma tabela qualquer. Arquivos CSV importados são tratados pela aplicação como respostas de um questionário de pesquisa. Cada linha da tabela é armazenada como um documento único, conhecido como *case* (caso) que representa cada resposta às perguntas do questionário. Desta forma, quando um conjunto de dados do Twitter é importado, ele é armazenado como um *case* onde em vez de perguntas e respostas, o *case* possui o nome das colunas do arquivo e os respectivos valores para cada tweet.

O Atlas.ti possui três ferramentas para pesquisa por texto que auxiliam no processo de codificação automática. O método mais simples busca por palavras ou frases com exatidão. Há também a busca por categoria, onde uma série de termos são agrupados e pesquisados simultaneamente. Além disso, é possível pesquisar por texto utilizando expressões regulares, o que torna a busca muito mais flexível e poderosa do que apenas a especificação de termos com caracteres-curinga. O Atlas.ti possui apenas um subconjunto de metacaracteres, quantificadores, classes de caracteres e outros itens da sintaxe do padrão Perl, porém suficiente para aumentar o poder de busca em relação aos outros métodos.

A codificação automática pode ser feita com ou sem o controle do analista. Na codificação totalmente automática, a cada termo de busca encontrado o software atribui o código. Na codificação semi-automática, o analista deve confirmar a inclusão do código no trecho apontado pela aplicação.

O Atlas.ti viabiliza o trabalho colaborativo, porém de forma assíncrona. Desta forma, cada pesquisador trabalha em seu projeto, e posteriormente cada projeto individual dos membros da equipe são mesclados em um único projeto.

3.3 Dedoose

O Dedoose[5] é um sistema *web* para apoio à análise qualitativa desenvolvido pela UCLA (Universidade da Califórnia em Los Angeles). Assim como outros *softwares* desta categoria, o Dedoose suporta métodos mistos de pesquisa além de possibilitar a análise dos principais formatos de texto, áudio, imagem e vídeo.

Este sistema não suporta a captação de dados diretamente das redes sociais. Entretanto, o sistema permite a importação de uma planilha do Excel contendo os dados previamente coletados. Dados importados a partir de planilhas não são tratados genericamente, a aplicação trata como se fossem perguntas e respostas de um questionário. Por exemplo, uma planilha contendo tweets com os atributos separados por colunas, o Dedoose trata o nome dos atributos no cabeçalho da planilha como perguntas e o respectivo valor do atributo em cada linha como a resposta.

O Dedoose não oferece nenhum método de codificação automática, seja por pesquisa

[5] http://www.dedoose.com/

por termos e frases ou por técnicas mais avançadas. Sendo assim, o pesquisador deve atribuir os códigos manualmente.

Um dos diferenciais deste *software* é a possibilidade de compartilhar o projeto entre vários analistas e os mesmos trabalharem simultaneamente em tempo real. O Dedoose possui duas abordagens para o controle de acesso a um projeto. Na mais simples, todos os usuários tem o mesmo nível de autoridade e podem acessar todos os arquivos e funcionalidades do sistema. Há também esquemas de permissões mais complexos, onde os usuários pertencem a determinados grupos que delimitam seus privilégios de acesso.

3.4 MAXQDA

O MAXQDA[6] é um software distribuído pela VERBI para análise de dados qualitativos e métodos mistos de investigação. A aplicação está disponível para os sistemas operacionais Windows e Mac. O MAXQDA viabiliza a análise de uma ampla variedade de dados não-estruturados, como textos, arquivos multimídia e respostas de questionários.

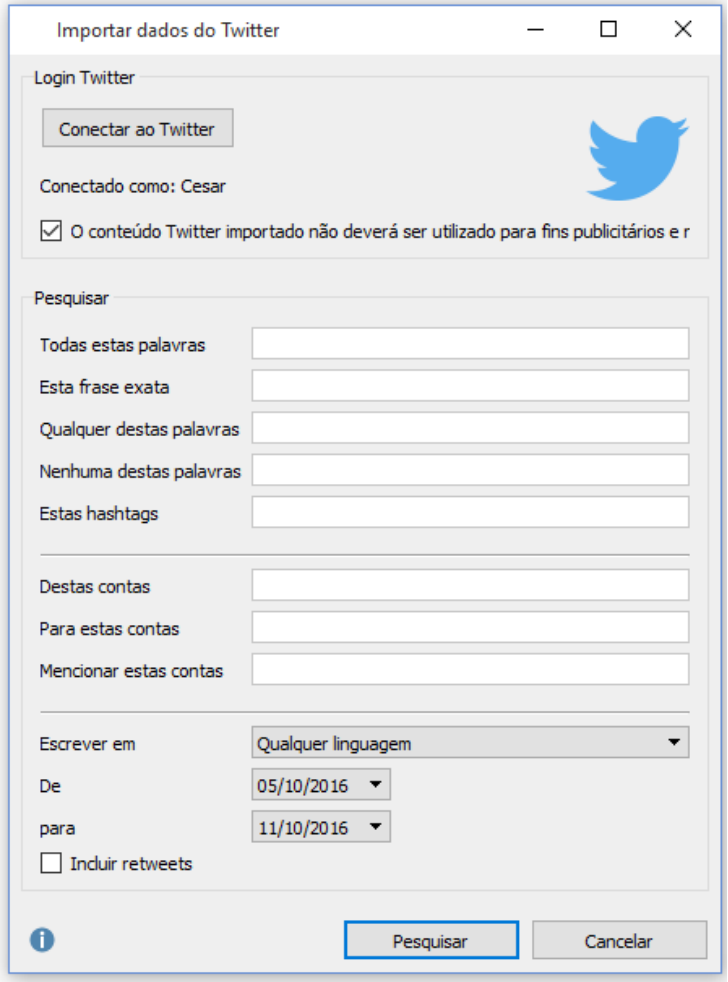
O MAXQDA possibilita a extração de dados de redes sociais, porém apenas do Twitter. Além disso, não é necessária a instalação de extensões adicionais, como é o caso do NVivo. Basta o usuário fornecer as credenciais de sua conta no Twitter e autorizar a aplicação a ler a *timeline* e ter acesso à lista de amigos. Conceder essas permissões são necessárias para a realização da coleta dos tweets. O MAXQDA oferece diversos tipos de filtros para realizar a busca, como mostra a Figura 4. Cada operação de importação permite extrair no máximo 10000 tweets. Os tweets são armazenados como um conjunto de dados e exibidos num formato tabular, o que facilita sua manipulação. Cabe ressaltar que assim como no NVivo, no MAXQDA não existe a possibilidade de a partir de um tweet recuperar o diálogo no qual ele fez parte.

Imediatamente após a importação ou posteriormente, o MAXQDA fornece a opção de codificar automaticamente os tweets em termos do autor ou *hashtag*. A aplicação também permite a importação de dados do Twitter e outros serviços de redes sociais a partir de um arquivo do Excel. Porém, desta forma não é possível codificar automaticamente a partir do autor e hashtags da mensagem. Cada linha da tabela é importada para o *software* como um documento e armazenado em um grupo de documentos.

Quanto à codificação automática em geral, o MAXQDA permite efetuar pesquisa por frases e palavras-chave, buscando por um único termo ou por múltiplos termos e combinando com operadores booleanos. No MAXQDA também é possível a utilização de curingas para especificar os termos de pesquisa. Dois deles são os mesmos do NVivo (os símbolos * e ?) e operam da mesma maneira. Os outros dois curingas permitem especificar o sufixo ou prefixo da palavra a ser pesquisada. Por exemplo, o termo <(pass)

[6] http://www.maxqda.com/

Figura 4 – Importação de tweets no MAXQDA



Fonte: Produzido pelo autor

consegue recuperar documentos contendo palavras com pass no prefixo, como passe, passo e passagem. Já uma consulta com o termo (eiro)> recupera documentos contendo palavras com o sufixo eiro, como passageiro, bagageiro e banqueiro.

O MAXQDA não permite múltiplos usuários trabalharem no mesmo projeto simultaneamente e em tempo real. Uma das formas de superar essa limitação é dividir o trabalho em vários projetos, onde cada pesquisador pode trabalhar em paralelo e ao final todos eles serem unificados.

3.5 webQDA

O webQDA[7] é uma aplicação *web* para análise de dados qualitativos desenvolvido numa parceria entre a empresa Micro IO e a Universidade de Aveiro, Portugal. Possui

[7] http://www.webqda.net

estrutura e propósito similar a outras aplicações do gênero, porém um dos seus diferenciais é a viabilização do trabalho colaborativo simultâneo e em tempo real.

Este sistema não realiza extração de dados de redes sociais. Além disso, sequer permite a importação de um arquivo do Excel ou de texto CSV contendo os posts, de modo que o sistema armazene como um conjunto de dados. A única forma de trabalhar com as postagens neste *software* é através da importação de um arquivo texto nos formatos aceitos, por exemplo docx ou txt. Entretanto, na importação de arquivo texto nestes formatos o sistema considera todas as postagens como um único corpo de texto, o que dificulta a manipulação individual de cada postagem.

No webQDA, todo o processo de codificação é feito de forma manual, não há mecanismos para automatizar essa tarefa. O sistema dispõe de busca por frases e palavras-chave, de modo semelhante ao NVivo, porém não permite atribuir um código para os resultados das buscas.

Como destacado anteriormente, o webQDA possibilita que múltiplos usuários trabalhem simultaneamente no mesmo projeto. O gestor de um projeto no webQDA pode convidar a sua equipe e atribuir aos membros dois papéis: colaborador ou convidado. O colaborador tem a permissão de incluir e modificar os arquivos de um projeto, enquanto o colaborador pode apenas visualizar os dados de um projeto.

3.6 Resumo

A Tabela 5 apresenta sucintamente o comparativo realizado ao longo do capítulo com a inclusão do SoNDA. No Capítulo 5 são apresentados os detalhes a respeito dos aspectos aqui avaliados no SoNDA. As células marcadas com o símbolo ✓ indicam que a respectiva funcionalidade está coberta por completo ou parcialmente pela aplicação. Os itens assinalados com o símbolo ✗ apontam que o referido *software* não possui a funcionalidade.

Entre os *softwares* comparados neste capítulo, o que possui o maior número de características em comum com o SoNDA é o NVivo. A principal diferença entre ambos, com relação as funcionalidades aqui comparadas, é que o SoNDA é capaz de recuperar o diálogo a partir de uma postagem, enquanto o NVivo não oferece esta opção.

Tabela 5 – Comparação resumida entre o SoNDA, NVivo, MAXQDA, Atlas.ti, Dedoose e webQDA

	SoNDA	**NVivo**	**MAXQDA**	**Atlas.ti**	**Dedoose**	**webQDA**
Extração de dados de redes sociais	✓	✓	✓	✗	✗	✗
Importação de dados de redes sociais via arquivo	✓	✓	✓	✓	✓	✗
Codificação automática de texto	✓	✓	✓	✓	✗	✗
Permitir trabalho colaborativo simultâneo e em tempo real	✓	✓	✗	✗	✓	✓

Fonte: Produzido pelo autor

4 Projeto do Sistema

O SoNDA é um software para análise de posts de redes sociais, cujas funcionalidades foram projetadas para dar suporte às atividades descritas na metodologia criada pelo COMUNIC. Uma das principais razões de sua existência é o fato de outros softwares similares do mercado, comparados no Capítulo 3, não satisfazerem as necessidades dos pesquisadores para apoiar seu estudo aplicando a metodologia. Tanto é, que todas as funcionalidades do SoNDA estão fortemente relacionadas à alguma etapa da metodologia, como será discutido mais adiante neste capítulo.

No comparativo entre as principais soluções de software para análise qualitativa descritos no Capítulo 3, a mais próxima de corresponder às atividades da metodologia é o NVivo. O único aspecto não coberto por este software é a capacidade de recuperar todas as trocas comunicativas na qual um dado post está situado. Pois bem, o NVivo, assim como outros softwares comerciais analisados, não possibilitam a implementação de plug-ins e extensões que acrescentam funcionalidades à solução já existente, o que poderia resolver esse problema. Uma outra alternativa, seria recuperar o diálogo no qual um post está inserido, de forma manual ou automática, usando um software específico, importar estes dados para o NVivo, e criar um link entre o post e o seu diálogo. Desta maneira, o software semi-automatizaria as atividades previstas na metodologia. O principal objetivo do uso de um software para apoiar as pesquisas aplicando a metodologia do COMUNIC é, automatizar completamente tarefas que não dependam da interpretação do pesquisador. Este é um ponto importante, pois seria inconveniente para o usuário executar manualmente tarefas que poderiam ser realizadas automaticamente após poucos cliques em uma interface gráfica, sobretudo em um problema como o apresentado, onde o volume de dados a serem manipulados e analisados é abundante.

Podemos citar duas grandes diferenças do SoNDA para outras ferramentas do tipo CAQDAS. A primeira, é que a estrutura teórica é rígida, isto é, dependente do modelo de investigação do COMUNIC. Esta decisão levou em consideração que, em um primeiro momento, outros modelos de pesquisa não seriam utilizados no SoNDA. Em outros softwares do segmento, a estrutura é flexível, isto é, não impõe ao pesquisador seguir um determinado modelo de pesquisa para utilizar a ferramenta. Outra diferença está nos tipos de dados para análise suportados. Enquanto o SoNDA permite importar apenas posts de redes sociais, outros softwares de mercado permitem trabalhar com uma variedade de tipos de dados, como texto, áudios e vídeos em diversos formatos.

Em termos funcionais, o SoNDA é similar aos softwares analisados no Capítulo 3. Em todos eles, o pesquisador está apto a incluir os dados de entrada para análise, criar

categorias, associar trechos dos dados de entrada a categorias, filtrar e questionar os dados, com o objetivo de responder às questões de sua pesquisa.

4.1 Escopo do sistema

O sistema desenvolvido neste trabalho apoia três etapas da metodologia descrita na seção 2.4, são elas: etapas 2, 3 e 4. Portanto, a primeira etapa, conhecida como etapa de extração dos dados das redes sociais, não é contemplada no software, pois consideramos que há diversas ferramentas que atendem essa necessidade como, por exemplo, o TAGS[1]. Assim, o SoNDA foca na carga, pré-processamento e análise dos dados. Sendo assim, no contexto do SoNDA, a etapa 1 é denominada "configuração da pesquisa", enquanto as demais permanecem a mesma nomenclatura definida anteriormente.

Apesar do SoNDA pertencer ao segmento de aplicações CAQDAS, ele não dispõe de algumas características e funcionalidades comumente encontradas em outros *softwares* deste tipo, como por exemplo: possibilidade de criar estruturas hierárquicas de códigos e anexação de comentários e lembretes aos dados. O escopo do trabalho abrange apenas os requisitos necessários para o apoio à metodologia do COMUNIC.

Uma outra limitação do SoNDA é em relação aos serviços de redes sociais em que dá suporte. Atualmente, só é possível analisar dados extraídos do Twitter. Existe uma grande diferença no formato dos dados extraídos nas mais diversas redes sociais e, como a metodologia foi elaborada para trabalhar sobre os dados do Twitter, focamos apenas nesta plataforma.

4.2 Processo de desenvolvimento do SoNDA

O processo de desenvolvimento do sistema foi baseado em uma metodologia ágil, uma vez que, no início do projeto, identificou-se que os requisitos do sistema poderiam mudar de prioridade, ou mesmo sofrerem alterações com frequência, e que necessitava-se entregar novos recursos e melhorias no software aos pesquisadores com certa regularidade. De acordo com Sommerville (2011), métodos ágeis são a abordagem de processo de desenvolvimento mais indicada para projetos de *software* com os aspectos citados.

Para o planejamento e gerência do projeto do software, foi adotada a metodologia Scrum (Schwaber e Beedle (2002)), com algumas adaptações. Essa escolha se deu pelo Scrum ser uma abordagem para gerenciamento de projeto enxuta e que prevê a entrega de novas funcionalidades com frequência, assim como todos os métodos ágeis. A seguir, são expostos os papéis fundamentais que a metodologia Scrum define e quais membros do projeto exerceram cada um.

[1] https://tags.hawksey.info/

- *Product Owner*: desempenhado pela líder do grupo de pesquisa COMUNIC, uma vez que é a pessoa que detém conhecimento técnico a respeito da análise de dados de redes sociais e parte mais interessada, cabia a ela definir e priorizar os requisitos do sistema.

- *Scrum Master*: papel assumido pelo orientador do projeto, sendo o elo entre os membros do grupo de pesquisa e o desenvolvedor. Coube ao Scrum Master, solucionar dúvidas do desenvolvedor, bem como, refinar as demandas dos *product owners*.

- Desenvolvedor: desempenhado pelo autor do projeto, responsável por documentar e implementar as funcionalidades requeridas e priorizadas pelos *product owners*.

O desenvolvimento se deu em ciclos iterativo de quatro semanas, conhecidos no Scrum como Sprint. No início de cada ciclo, uma reunião envolvendo as partes envolvidas era realizada tendo dois objetivos: (1) levantar novos requisitos ou melhorias ao sistema em produção, que são incluídos ao *Product backlog* e (2) construir o *Sprint backlog*, isto é, a lista de requisitos priorizada a ser implementados no ciclo atual.

Com o trabalho da iteração priorizado, cabia ao desenvolvedor analisar e modelar os requisitos e implementá-los. As tarefas eram registradas no gerenciador de tarefas Producteev para que o *Scrum Master* pudesse acompanhar o progresso da *sprint*. Em cada uma das oito sprints realizadas, o processo de desenvolvimento foi constituído por três atividades:

- Análise e modelagem dos requisitos: os requisitos priorizados na sprint eram modelados e adicionados ao documento de casos de uso.

- Implementação: esta é a etapa da iteração que consumia maior tempo, onde as funcionalidades são implementadas e testadas. Foram realizados testes unitários para verificar a corretude dos métodos de cada classe.

- Entrega: as funcionalidades e melhorias implementadas durante o ciclo eram integradas ao sistema em produção. O encerramento do ciclo era marcado por uma reunião de *feedback*, onde o desenvolvedor realizava a apresentação, através de demonstrações, das funcionalidades implementadas durante a *sprint*.

4.3 Arquitetura do sistema

O ponto de partida para o desenvolvimento foi a definição da arquitetura do sistema e das tecnologias empregadas na implementação. Na primeira elicitação de requisitos do sistema, foi estabelecida a restrição de que o *software* deveria ser executado a partir de um navegador web. Esta decisão de projeto impactou na escolha do modelo da arquitetura,

onde foi feita a opção por um sistema cliente-servidor. Sommerville (2011) explica que neste modelo, o usuário interage com uma aplicação executando em seu computador, por exemplo, um navegador web e comunica-se com um programa em execução em um computador remoto para adquirir os recursos fornecidos por ele, como páginas da web e imagens.

O SoNDA é constituído dessas duas camadas: cliente e servidor, onde cada uma pode ser tratada como um subsistema. Na aplicação servidor, estão implementadas a lógica da aplicação, a persistência dos dados e a interface para recuperar e modificar os dados. A aplicação cliente é responsável pela interação com o usuário, que envolve o carregamento das páginas, a validação inicial de entradas de formulários e a comunicação com o servidor para atender os objetivos do usuário.

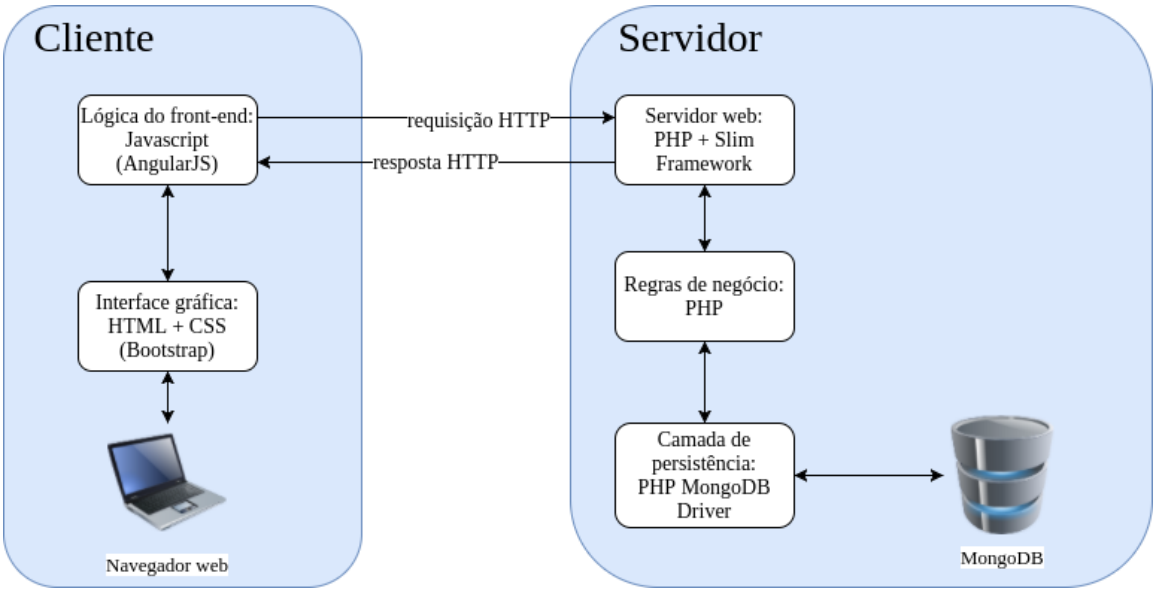
A comunicação entre os subsistemas é efetuada através de requisições HTTP a uma API REST. REST é acrônimo para *Representational State Transfer* (em português, Transferência de Estado Representacional). Para Sommerville (2011), é um estilo arquitetural baseado na transferência de representação de recursos do servidor para o cliente, utilizando o protocolo HTTP para a comunicação entre as partes.

A Figura 5 ilustra os componentes da arquitetura do SoNDA e as respectivas tecnologias empregadas na implementação de cada um. A aplicação servidor foi escrita com a linguagem PHP . O *framework* Slim foi utilizado para criação da API, que acessa a camada de persistência para atender as requisições feitas pelos usuários. Para persistência dos dados, optou-se pelo MongoDB. Essa escolha se deve ao fato dos dados importados para o sistema não obedecerem uma estrutura rígida, o sistema é flexível quanto a nomes e quantidade de atributos de cada arquivo importado. O MongoDB atende este propósito, pois não impõe uma estrutura fixa para armazenar os registros.

A aplicação cliente é uma *single-page application* (aplicação de página única) implementada na linguagem Javascript e o *framework* AngularJS. Aplicações web deste gênero garantem uma experiência de uso similar às aplicações *desktop*, pois a mesma não é recarregada após uma ação do usuário. Para atender as solicitações dos usuários, a aplicação cliente efetua requisições à API REST implementada na aplicação servidor. Por fim, para o *layout* do sistema foi optado por um tema com design minimalista. De modo que o grupo de pesquisa não contava com um designer para criação do visual da aplicação, foi feita a opção por utilizar um template gratuito e de código aberto. Dentre as opções existentes, foi escolhido o tema construído com o *framework* Bootstrap SB Admin[2] para ser modificado e adaptado às necessidades do SoNDA.

[2] https://startbootstrap.com/template-overviews/sb-admin/

Figura 5 – Arquitetura do SoNDA



Fonte: Produzido pelo autor

4.3.1 Tecnologias utilizadas

4.3.1.1 Slim Framework

O Slim[3] é um *micro framework* PHP projetado para auxiliar o desenvolvimento de aplicações web e APIs de modo simples e robusto. Basicamente, o Slim recebe uma requisição HTTP, invoca uma rotina de retorno para tratá-la, e envia uma resposta HTTP. O Slim é uma ferramenta de código aberto e está sob licença MIT[4].

4.3.1.2 MongoDB

O MongoDB é um sistema de gerenciamento de banco de dados orientado a documentos. É classificado como banco de dados NoSQL, isto é, a estrutura de armazenamento dos registros difere do esquema de tabelas e relações dos bancos de dados relacionais.

O MongoDB trabalha com os conceitos de coleção e documento. Uma coleção é um agrupamento de documentos do MongoDB, é o equivalente a uma tabela de um banco de dados relacional. Os documentos basicamente são um conjunto de pares chave-valor. Uma das principais características do MongoDB é que os documentos não possuem estrutura fixa, ou seja, documentos de uma coleção podem ter conjuntos de campos distintos e os valores dos campos em comum dos documentos de uma coleção podem ser de tipos de dados diferentes, até mesmo outros documentos.

[3] http://www.slimframework.com/
[4] Maiores detalhes sobre a licença MIT: https://opensource.org/licenses/MIT

A flexibilidade em relação ao esquema de um documento foi um fator decisivo na escolha do MongoDB para este projeto. Se optássemos pela rigidez de um banco de dados relacional, teríamos de lidar com relacionamentos complexos entre as entidades do sistema, o que exigiriam muitas operações de *join* (junção) entre as tabelas para recuperar todos os dados necessários. Com o MongoDB, o relacionamento entre documentos é realizado de um modo bem mais simplificado.

Além disso, levamos em consideração que o MongoDB possui alta performance para escritas, o que é importante uma vez que estaremos lidando com grandes volumes de inserções na base de dados e também pela fácil escalabilidade. Escalabilidade não é um requisito deste projeto, porém preferimos escolher uma ferramenta de persistência que possibilite fazer isso de modo simplificado.

O MongoDB é livre e de código aberto, com licença de utilização que é uma combinação da GNU Affero General Public License e da Apache License[5]

4.3.1.3 AngularJS

O AngularJS é um *framework* Javascript de código aberto, mantido pelo Google, que auxilia na construção de *single-page applications*. O Angular estende a sintaxe HTML tradicional através de *tags* específicas, chamadas de diretivas, que adicionam comportamento dinâmico à página.

Uma das principais características do *framework* é a ligação bidirecional dos dados (*two-way data binding*), o AngularJS monitora a aplicação e ao detectar alteração nos dados na camada de visão, sincroniza automaticamente com o modelo. Do mesmo modo, quando os dados são modificados no modelo, essas alterações são refletidas na camada de visão. O AngularJS está sob licença MIT.

4.3.2 Bootstrap

O Bootstrap é um *framework* HTML, CSS e Javascript de código aberto que auxilia no desenvolvimento de páginas *web* responsivas. Isso significa, que o layout da página se adapta automaticamente às dimensões do dispositivo utilizado para navegação.

O Bootstrap possui uma ampla variedade de componentes customizados como formulários, botões, fontes e tabelas. Um dos principais recursos do Bootstrap é o sistema de *grids*, que simplifica a estruturação do *layout* da página, além de manter a responsividade. O Bootstrap está sob licença MIT.

[5] Maiores detalhes sobre a licença de uso do MongoDB: https://www.mongodb.com/legal/terms-of-use.

4.4 Requisitos funcionais

Os requisitos funcionais de um sistema descrevem o que ele deve ser capaz de fazer. Os requisitos elicitados e implementados nos ciclos de desenvolvimento são expostos a seguir, agrupados pelas etapas da metodologia do COMUNIC que o SoNDA apoia, indicando quais funcionalidades são necessárias para atender as demandas de uma determinada fase.

- Etapa 1: Configuração da pesquisa
    1. O sistema deve permitir o cadastro de projetos.
    2. O sistema deve possibilitar a inclusão de membros ao projeto.
    3. O sistema deve permitir a importação de arquivo texto no formato CSV contendo tweets a um projeto.
- Etapa 2: Mineração por Espaços de possibilidades
    1. O sistema deve permitir a extração de amostras dos datasets importados no projeto.
    2. O sistema deve permitir o cadastro de categorias do tipo Espaços de possibilidades.
    3. O sistema deve ser capaz de filtrar os datasets por categorias de Espaços de possibilidades.
- Etapa 3: Mineração por Fatores e circunstâncias
    1. O sistema deve permitir o cadastro de categorias do tipo Fatores e circunstâncias.
    2. O sistema deve prover mecanismo para o pesquisador associar uma categoria do tipo Fator e circunstância a um tweet.
- Etapa 4: Recuperação dos diálogos
    1. O sistema deve fornecer mecanismos para o pesquisador interrogar os datasets codificados em Fatores e circunstâncias
    2. O sistema deve ser capaz de a partir de um tweet, recuperar o diálogo onde o mesmo está inserido.

4.5 Requisitos não-funcionais

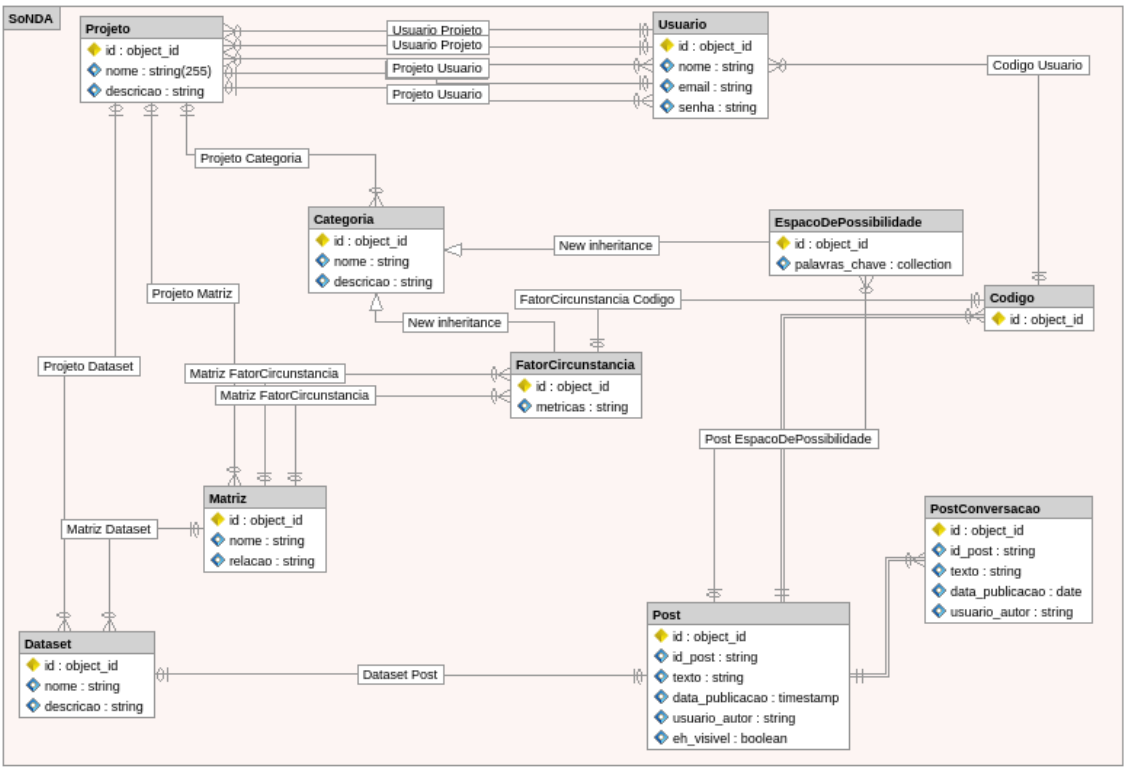
Requisitos não-funcionais são restrições impostas aos serviços ou funcionalidades oferecidas pelo sistema. No SoNDA, estas restrições estão relacionados ao acesso às funcionalidades e dados do sistema.

1. Somente usuários registrados e identificados poderão ter acesso ao sistema.
2. Usuários só poderão ter acesso a projetos que criou e aos que foi convidado a colaborar.

4.6 Esquema do banco de dados

A seção 4.3 justificou a escolha do MongoDB como sistema de gerenciamento de banco de dados (SGBD) para este projeto. Agora, veremos na prática como as características do MongoDB contribuem para que o SoNDA tenha uma boa performance para filtrar um grande volume de posts de redes sociais. A modelagem do banco de dados do SoNDA é apresentada na Figura 6.

Figura 6 – Modelagem do banco de dados da aplicação



Fonte: Produzido pelo autor

Como mencionado na subseção 4.3.1.2, o MongoDB é um banco de dados não relacional e livre de esquema. Ou seja, os documentos de uma mesma coleção não são forçados a possuírem os mesmos campos com seus respectivos tipos de dados. Na Figura 7, podemos verificar essa flexibilidade existente no MongoDB, ambos os documentos representam um Post importado no SoNDA e pertencem a coleção *posts* no banco de dados. Note que há diferença em relação aos atributos existentes em cada um. Outro ponto a se destacar é que a sintaxe de um documento no MongoDB é bastante similar a de um objeto JSON.

Figura 7 – Entidade Post representado como documentos do MongoDB

```
{
        "_id" : ObjectId("576472cc31b38863458b456f"),
        "id_post" : "553315797247217664",
        "texto" : "Exemplo de post inserido no banco de dados",
        "autor" : "tcc_cesar",
        "data_publicacao" : ISODate("2017-06-19T23:05:19Z"),
        "idDataset" : "576472b331b38865458b4568",
        "eh_visivel" : true,
        "espacos_de_possibilidade" : [
                ObjectId("576472cc31b38863458b456a"),
                ObjectId("576472cc31b38863458b456e")
        ],
        "codigos" : [
                {
                        "fator_e_circunstancia" :
                        ObjectId("576472cc31338863458b456f"),
                        "codificado_por" : [
                                ObjectId("576472cc34b38863458b456a"),
                                ObjectId("5764345c31b38863458b456a")
                        ]
                },
                {
                        "fator_e_circunstancia" :
                        ObjectId("576472cc31b38863458b4564"),
                        "codificado_por" : [
                                ObjectId("576472cc34b38863458b456a"),
                                ObjectId("5764545c31b38863458b456a")
                        ]
                }
        ]
}

{
        "_id" : ObjectId("576472cc31b38863458b451f"),
        "id_post" : "553315797247217664",
        "texto" : "Outro exemplo de post inserido no banco de dados",
        "autor" : "tcc_cesar",
        "data_publicacao" : ISODate("2017-06-19T23:05:25Z"),
        "idDataset" : "576472b331b38865458b4568",
        "eh_visivel" : true,
}
```

Fonte: Produzido pelo autor

Em situações onde necessitamos relacionar documentos, podemos seguir duas

abordagens: referenciando o identificador de outro documento ou incorporando documentos. Referenciar outro documento pelo identificador é similar a referenciar uma outra tabela por sua chave-estrangeira em bancos de dados relacionais. A incorporação de documentos consiste em inserir um documento por completo em outro.

No primeiro documento da Figura 7, o atributo *espacos_de_possibilidade* referencia uma lista de documentos que representam um Espaço de Possibilidade, ou seja, indica em quais categorias o documento Post pertence. Enquanto o atributo *codigos* contém uma lista de documentos do tipo Código, que representam os códigos atribuídos a um dado Post. Como os documentos da coleção *espacos_de_possibilidades* são referenciados por outras coleções, optamos por apenas armazenar seu identificador no documento post. Incorporar o documento neste caso, não seria prático, uma vez que não há integridade referencial no MongoDB e ficaria a cargo da aplicação atualizar todas as cópias de um documento deste tipo quando o mesmo fosse atualizado.

Enquanto que para armazenar os códigos de um Post, foi decidido incorporar documentos Código ao documento Post. Isso possibilita por exemplo, consultar apenas o documento Post para recuperar os posts codificados em um dado fator e circunstância e por um dado usuário. Se esse problema fosse modelado pensando em banco de dados relacional, seria necessário juntar mais de uma tabela para satisfazer essa consulta, o que degradaria a performance de buscas como essa.

5 Implementação das funcionalidades

As seções a seguir relatam de que forma as funcionalidades do sistema foram implementadas, principalmente do ponto de vista do usuário. São apresentados também os detalhes de implementação de requisitos mais complexos. Funcionalidades triviais, como a inserção de um novo projeto, terão seus detalhes de implementação omitidos.

5.1 Projeto

Um projeto reúne as informações referentes a uma pesquisa, isto é, os datasets importados e as categorias e matrizes cadastradas, além dos pesquisadores participantes e seus respectivos papéis. A criação do projeto é o primeiro passo no fluxo de trabalho utilizando o SoNDA. Para incluir um novo projeto, basta indicar o nome para identificá-lo.

Quanto ao compartilhamento de um projeto por vários usuários, o SoNDA segue a filosofia do *software* webQDA. Possibilitar que múltiplos pesquisadores trabalhem simultaneamente no mesmo projeto, permite integrar os membros de um projeto que raramente tem contato presencial, além de paralelizar o trabalho, uma vez que um pesquisador não depende que outro termine sua parte, a entregue em um arquivo para que finalmente consiga começar a sua contribuição à pequisa.

Na plataforma SoNDA, os integrantes de um projeto possuem papéis que restringem o acesso a determinadas ações. Em um projeto, o usuário pode assumir um dos três perfis existentes:

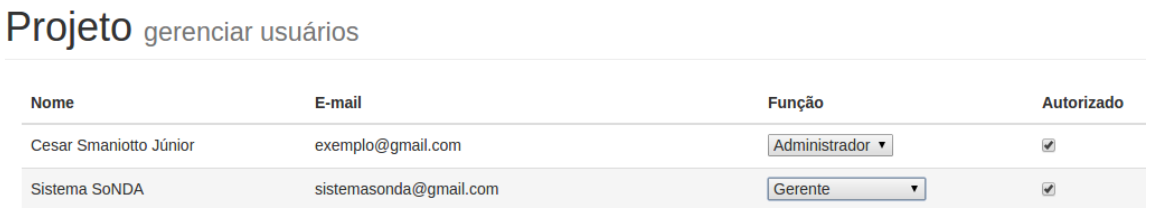
- Administrador: perfil assumido pelo usuário que criou o projeto, tem acesso para manipular livremente todos os itens de um projeto. Cabe ao administrador, convidar e excluir membros de um projeto, além de alterar o papel de um membro em um projeto.

- Gerente: usuários com este perfil tem permissão para importar novos datasets, criar categorias e matrizes, além de modificar aquelas inicialmente incluídas por outros membros. Com relação ao administrador, a única diferença é que o gerente não tem privilégio de gerenciar os usuários integrantes de um projeto.

- Pesquisador: é o usuário com menor nível de privilégio, não possui permissão para incluir novos datasets, porém pode cadastrar novas categorias e matrizes e modificar apenas os itens que criou. As consultas e filtragens nos datasets são habilitadas a todos os perfis de acesso.

Para convidar outros usuários a integrar um projeto, estes devem estar previamente cadastrados no sistema. O administrador do projeto deve inserir o e-mail dos membros que deseja convidar, para habilitar o acesso. A Figura 8 apresenta a área do sistema onde o administrador do projeto pode gerenciar os membros do projeto, podendo modificar seu perfil de acesso na opção "Função" e liberar ou revogar o acesso dos mesmos na opção "Autorizado".

Figura 8 – Tela do sistema para gerência de usuários de um projeto



Fonte: Produzido pelo autor

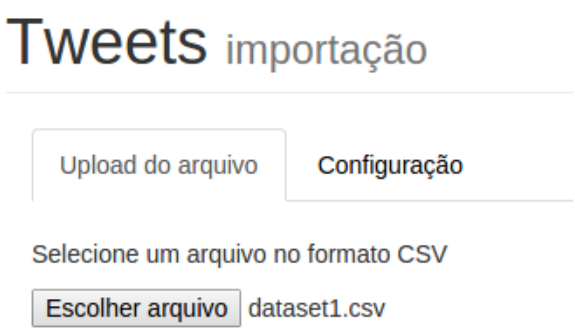
5.2 Dataset

Um dataset é uma coleção de tweets, coletados durante um período de tempo e carregado para o sistema. É um conjunto de dados bruto, que será submetido aos métodos de filtragem existentes no sistema, com o objetivo de obter um subconjunto contendo as postagens mais relevantes, que merecem um estudo mais aprofundado dos pesquisadores.

A importação dos tweets pelo usuário ocorre em dois passos. Primeiramente, o usuário indica o arquivo a ser importado pelo sistema. Após o *upload*, é necessário informar alguns parâmetros para que o arquivo possa ser corretamente lido, armazenado e indexado pelo sistema. A Figura 9 exibe a tela do sistema para seleção do arquivo e a Figura 10 a tela para definição das configurações do arquivo. O usuário deve informar quais os caracteres delimitadores de texto e de campo do arquivo. De modo que a aplicação é flexível quanto ao arquivo importado, ou seja, não exige um formato fixo de campos e atributos, o usuário deve informar qual coluna da tabela corresponde a alguns atributos do tweet. Isto é necessário para realizar corretamente o pré-processamento e indexação dos dados.

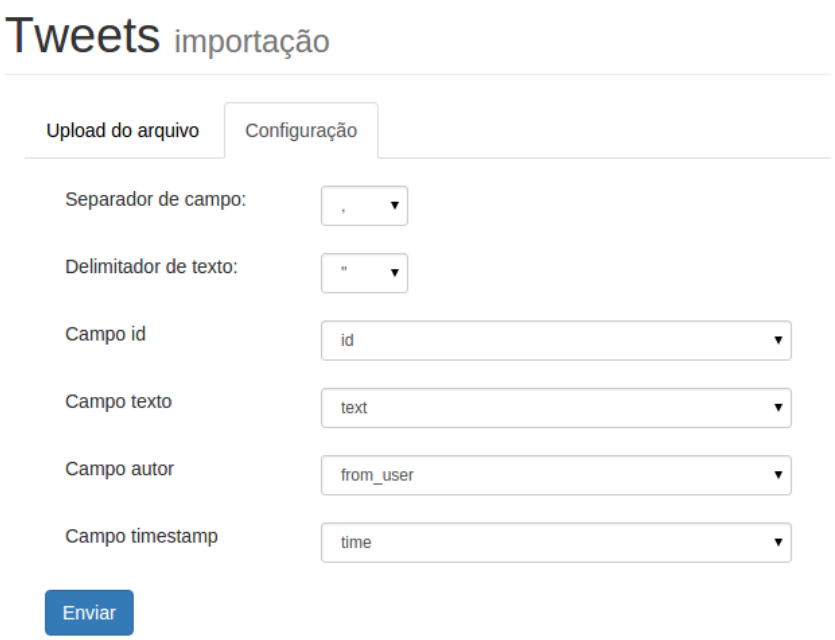
A Tabela 6 mostra um exemplo de uma tabela contendo tweets que são passíveis de serem importados pelo sistema. Comumente, um tweet é composto por diversos atributos, a nomenclatura e a quantidade de campos dependem da ferramenta usada para extração. Para simplificar, vamos lidar apenas com atributos obrigatórios, isto é, aqueles que o usuário deve informar a coluna correspondente na importação. No exemplo estão especificados como: *id, text, from_user e created_at*. O campo *id* representa o identificador único do tweet, enquanto *text* obviamente é a mensagem do tweet. O atributo *from_user* indica o

Figura 9 – Tela do sistema para seleção do arquivo com o dataset



Fonte: Produzido pelo autor

Figura 10 – Tela do sistema para informar os parâmetros da importação do arquivo com o dataset



Fonte: Produzido pelo autor

nome do usuário que publicou o tweet, e *created_at* indica a data e hora da publicação, em formato de representação definido pelo Twitter.

5.3 Extração de fragmentos do dataset

A extração de fragmentos permite ao pesquisador obter alguns trechos do dataset para proceder a leitura e identificar os principais temas que compõem os Espaços de Possibilidade.

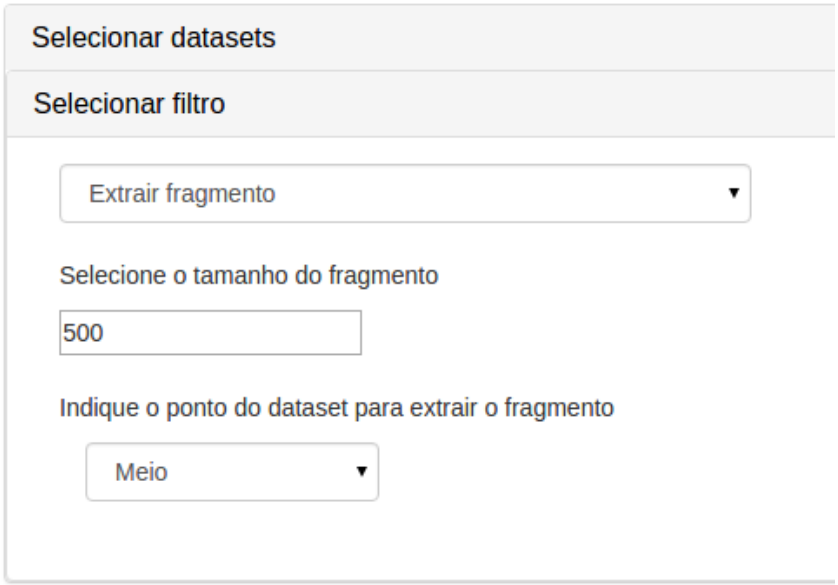
Tabela 6 – Exemplo de conjunto de tweets compatível com o módulo de importação do SoNDA

id	text	from_user	created_at
1	sou a favor das manifestações pelo passe livre	u_ficticio1	Wed Feb 08 13:53:54 +0000 2017
2	sou contra as manifestações pelo passe livre	u_ficticio2	Fri Feb 10 20:10:03 +0000 2017
3	sou neutro quanto as manifestações pelo passe livre	u_ficticio3	Sat Feb 11 08:15:34 +0000 2017

Fonte: Produzido pelo autor

A obtenção desse subconjunto é feita da seguinte forma: o pesquisador determina o tamanho da amostra e o ponto do dataset de onde será retirada, podendo ser do início, meio ou fim. O recorte é realizado com o dataset ordenado por data de publicação crescente. A Figura 11 exibe a tela do sistema onde as amostras são extraídas. Seja n o tamanho do fragmento e a cardinalidade de um dataset D, denotada por $|D|$. O fragmento do início recupera os primeiros n tweets do dataset. Já o fragmento do meio, extrai os tweets ordenados entre $|D|/2 - n/2$ e $|D|/2 + n/2$. O fragmento do fim, obtém os n últimos tweets do dataset.

Figura 11 – Tela do sistema para extração de fragmentos do dataset

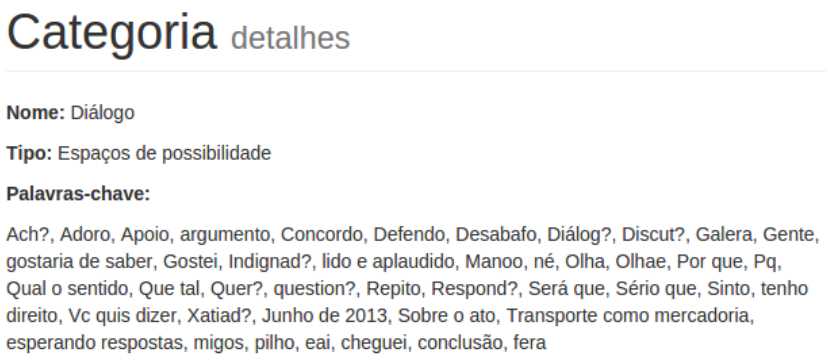


Fonte: Produzido pelo autor

5.4 Espaços de possibilidades

Espaços de possibilidades são categorias analíticas definidas pelos pesquisadores a partir da análise de recortes dos datasets. É composta pelo nome do processo social investigado, a descrição do seu significado e os termos e palavras-chave relacionados. A partir da biblioteca de termos associada ao espaço de possibilidade, é possível fazer o primeiro refinamento nos datasets do projeto.

Figura 12 – Descrição do Espaço de Possibilidade Diálogo



Fonte: Produzido pelo autor

A Figura 12 apresenta a especificação da categoria Diálogo. A biblioteca de palavras-chave pode ser composta por palavras, frases e termos com caracteres curingas. Caracteres curingas são utilizados para realizar a correspondência de padrões e substituir caracteres desconhecidos por um ou mais caracteres.

No SoNDA, o caractere-curinga interrogação (?) realiza a correspondência e substituição de um único caractere desconhecido. Por exemplo, o termo “Indignad?” casa com as palavras Indignado e Indignada. Enquanto o curinga asterisco (*) faz o casamento e substituição com qualquer número de caracteres desconhecidos. Por exemplo, a expressão “Indigna*” casa com as palavras Indignado e Indignação.

5.5 Mineração por Espaços de Possibilidade

Existem dois métodos para a mineração por Espaços de Possibilidade. O primeiro, é especificado pela metodologia e também é o mais simples, onde as palavras-chave associadas a um Espaço de possibilidades são utilizadas para filtrar o dataset. O segundo e mais complexo, é através de técnicas de mineração de texto. Este foi desenvolvido de maneira experimental, ou seja, não está disponível para uso dos pesquisadores até esta versão do SoNDA. O Capítulo 6 é dedicado para descrever esse método e os experimentos realizados.

5.5.0.1 Pesquisa por palavras-chave

Cada categoria de Espaço de Possibilidade possui uma biblioteca de termos e palavras-chave. Estas são utilizados para filtrar o dataset. Caso um tweet contenha algum termo dessa biblioteca, ele é incluído nos resultados.

Figura 13 – Resultados da filtragem por Diálogo através do conjunto de palavras-chave

Fonte: Produzido pelo autor

A Figura 13 exibe a tela do sistema com a apresentação dos resultados da filtragem por Diálogo. A exibição dos resultados das demais consultas, filtragens e cruzamentos de tweets obedecem esse formato tabular e com paginação.

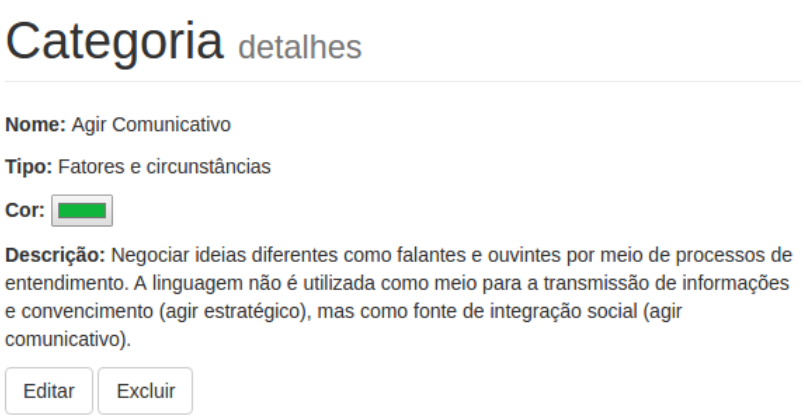
5.6 Fatores e circunstâncias

Fatores e circunstâncias são categorias analíticas formuladas a partir de revisão bibliográfica, cujos tweets pertencentes às mesmas, serão buscados nos datasets filtrados por Espaços de possibilidades. A Figura 14 apresenta a especificação do Fator e Circunstância "Agir Comunicativo". Uma categoria deste tipo é formada pelo nome, a descrição contendo as métricas, que atuam como guia para o pesquisador no processo de codificação das postagens e possui uma cor associada, para facilitar a identificação da categoria durante o processo de codificação.

5.7 Codificação por Fatores e Circunstâncias

O processo de codificação por Fatores e Circunstâncias compreende o terceiro estágio da metodologia, onde os pesquisadores codificam os tweets segundo as métricas

Figura 14 – Descrição do Fator e Circunstância Agir Comunicativo



Fonte: Produzido pelo autor

especificadas para cada uma destas categorias. De modo que o SoNDA permite que múltiplos usuários compartilhem o mesmo projeto e manipulem as mesmas fontes de dados, a codificação de cada usuário é feita de modo independente. Desta forma, cada usuário codifica os tweets sem saber se eles já foram codificados por outra pessoa, garantindo que as atribuições feitas por outros membros não influencie no processo de codificação do usuário.

Figura 15 – Filtragem por Diálogo com após a inclusão de códigos

Fonte: Produzido pelo autor

O ambiente para codificação é o mesmo onde são apresentados os resultados da Mineração por Espaços de Possibilidades, como mostra a Figura 13. A Figura 15 apresenta a mesma interface com algumas instâncias codificadas. A codificação é feita através das

ações de arrastar e soltar (*drag-and-drop*) uma das categorias de Fatores e Circunstâncias listadas até o tweet onde deseja-se codificar.

5.8 Questionamento dos datasets

De acordo com Souza, Costa e Moreira (2011), o ato de questionar a base de dados refere-se ao ato de formular perguntas que guiem a descoberta de padrões, inferências e conclusões sobre os dados analisados. Este módulo atua como um segundo filtro sob os datasets, cujos questionamentos realizados pelo analista com relação ao tweets codificados por Fatores e Circunstâncias têm como objetivo identificar os tweets mais relevantes para sua pesquisa, de modo a recuperar o seu rastro de conversação no próximo passo. O SoNDA oferece dois mecanismos para interrogar os datasets, que são através da criação de matrizes e execução de consultas.

5.8.1 Matrizes

As matrizes são uma das formas de realizar o cruzamento entre as postagens codificadas em Fatores e Circunstâncias. Em uma matriz é necessário indicar um nome para identificá-la, os datasets em que os tweets serão pesquisados, as categorias de Fatores e Circunstâncias que irão compor as linhas e colunas e a relação a ser aplicada para o cruzamento entre as linhas e colunas, podendo ser E (interseção), OU (união) e NÃO (exclusão).

Figura 16 – Exemplo de matriz de relação entre tweets

Fonte: Produzido pelo autor

A Figura 16 exibe a tela do sistema com o exemplo de uma matriz. Esta matriz possui apenas uma linha (Mundo Comum) e duas colunas (Espaço de Aparência e Pluralidade). A relação aplicada entre cada linha e cada coluna é E (intersecção). A utilização de matrizes permite uma grande flexibilidade no processo de questionamento. Com esta matriz, é possível responder a seguinte pergunta: "quais tweets foram codificados simultaneamente nos Fatores e Circunstâncias Mundo Comum e Espaço de Aparência, e Mundo Comum e

Pluralidade?". As células da tabela contém o número de tweets existentes em cada relação, e ao clicar na célula, é possível listar os tweets que pertencem a relação, apresentados de modo semelhante aos da Figura 15.

5.8.2 Consultas

As consultas ajudam a responder questionamentos frequentes dos pesquisadores que não são possíveis de serem satisfeitos com a construção de matrizes. Nesse caso, os questionamentos são predefinidos, isto é, não possuem a mesma flexibilidade das matrizes, onde o pesquisador é livre para formulá-las. As consultas já estão implementadas no sistema, então basta o pesquisador fornecer os parâmetros necessários. A seguir, a descrição de cada consulta existente no SoNDA.

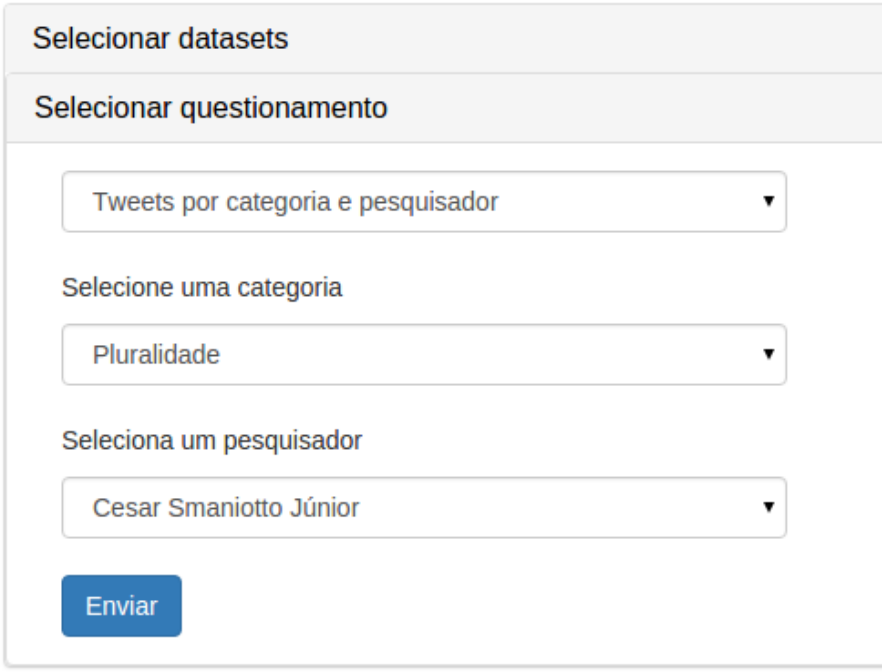
- Tweets por Fatores e Circunstâncias (FC) e Pesquisador: Dado um FC e um pesquisador, o sistema recupera todos os tweets codificados pelo pesquisador em um determinado FC. Por esta consulta, é possível descobrir quais tweets o usuário Cesar codificou no Fator e Circunstância "Pluralidade". A Figura 17 exibe a tela de sistema com esta consulta.
- Tweets por Fatores e Circunstância e quantidade de atribuições: Dado um FC, o sistema recupera todos os tweets que foram codificados por um FC por um número mínimo de pesquisadores. Esta consulta permite descobrir por exemplo, quais tweets foram codificados por pelo menos três pesquisadores diferentes como sendo do FC Pluralidade.
- Tweets por quantidade de Fatores e Circunstâncias distintos: O sistema recupera todos os tweets que foram codificados com um determinado número mínimo de FCs diferentes.

5.9 Recuperação dos diálogos

A recuperação de um diálogo consiste em resgatar as trocas de mensagens encadeadas entre um grupo de usuários, obtidas a partir de um tweet potencial, presente no resultado de um questionamento feito aos datasets do projeto. A recuperação e análise dos diálogos são atividades previstas na etapa final da metodologia do COMUNIC. O Twitter possui uma API pública[1] que disponibiliza diversas interfaces para acessar seus dados, como por exemplo, a recuperação dos tweets mais recentes de um usuário. Porém, a API não disponibiliza um método que, dado um tweet, recupere toda a sua *thread* de conversação, o que obriga a adotar outra abordagem para resolver este problema.

[1] https://dev.twitter.com/rest/public

Figura 17 – Consulta de tweets por Fator e Circunstância e Pesquisador

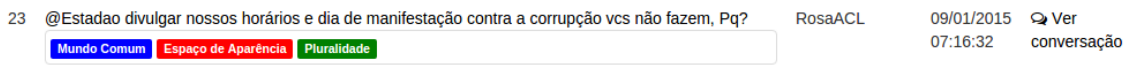


Fonte: Produzido pelo autor

Deste modo, foi implementado um *web crawler* para cumprir este requisito. A partir de um tweet, independente da posição em que apareça em um diálogo, o *crawler* coleta as mensagens anteriores e posteriores ao tweet em questão. Esse processo consiste em sucessivas varreduras nas páginas *web* do Twitter, onde o conteúdo de determinadas *tags* HTML são extraídos. Ao término dessa tarefa, os itens são salvos no banco de dados da aplicação e apresentados ao usuário.

A Figura 18 mostra um tweet presente nos resultados gerados pelo módulo de questionamento. Ao clicar em "ver conversação", o sistema recupera o diálogo armazenado no banco de dados, caso a extração tiver sido previamente feita. Caso contrário, o *crawler* tenta recuperar e armazenar os tweets envolvidos no diálogo e em caso de êxito, o sistema apresenta os resultados, como mostra a Figura 19. O bloco com a borda esquerda na cor azul identifica o tweet usado para a recuperar as demais interações.

Figura 18 – Tweet alvo para a recuperação do diálogo



Fonte: Produzido pelo autor

Essa estratégia para recuperação dos diálogos possui limitações e em pelo menos

três situações identificadas, não é possível obter o diálogo: quando o autor da postagem utilizada para recuperar as outras interações apagou ou desativou sua conta ou então, alterou a privacidade do perfil de público para privado, e quando a postagem em questão tiver sido excluída. Nestes casos, o sistema informa o erro ao usuário.

Figura 19 – Diálogo recuperado pelo sistema a partir de um tweet

Fonte: Produzido pelo autor

5.10 Detalhes de implementação

Esta seção visa apresentar os detalhes de implementação de alguns requisitos mais complexas, cujo funcionamento para o usuário foi especificado nas seções anteriores.

5.10.1 Mineração por Espaços de Possibilidades por palavras-chave

Em termos de implementação, para satisfazer a pesquisa por termos e palavras-chave, é necessário fazer uma varredura no dataset verificando se algum dos termos do conjunto de palavras-chave está presente num tweet e salvar no próprio tweet essa informação. As palavras-chave são transformadas em expressões regulares e cada tweet de um dataset é submetido às expressões regulares criadas. Se houver pelo menos uma correspondência entre o tweet e uma das expressões regulares, então o mesmo é incluído ao respectivo Espaço de Possibilidade o qual a palavra-chave pertence. O código-fonte na Figura 20 descreve o método implementado no sistema que realiza essa atualização. Essa verificação é realizada em três situações: (1) quando uma categoria do tipo Espaço de Possibilidade é cadastrada, (2) quando uma categoria já existente possui sua biblioteca

de palavras-chave alterada e (3) quando um novo dataset é adicionado ao projeto e já existem Espaços de Possibilidade cadastrados.

Figura 20 – Método que checa e atualiza os tweets de um dataset por um dado Esp. de Possibilidade

```php
<?php
public function atualizarEspacosPossibilidade($esp, $idDataset)
{
    $mongoRegex = array();
    foreach ($esp->getPalavrasChaveAsRegex() as $kwAsRegex) {
        $mongoRegex[] = new \MongoRegex($kwAsRegex);
    }
    $this->collection->update(array(
        '$and' => array(
            array(
                'idDataset' => array(
                    '$in' => new MongoId($idDataset)
                )
            ),
            array(
                'texto' => array(
                    '$in' => $mongoRegex
                )
            )
        )
    ), array(
        '$addToSet' => array(
            'espacos_de_possibilidades' => $esp->getId()
        )
    ));
}
```

Fonte: Produzido pelo autor

Este processo é caro computacionalmente, uma vez que não é possível otimizar com índices uma pesquisa que utiliza expressões regulares, o que obriga a consultar o dataset inteiro em busca das correspondências das expressões regulares. Após esse processo, o próprio post detém a informação se ele é categorizado em um dado Espaço de Possibilidade e recuperar essa informação e retorná-la ao usuário é trivial. A Figura 21 apresenta o método implementado no SoNDA para realizar a busca de tweets por Espaços de Possibilidades.

5.10.2 Questionamento dos datasets através de matrizes

O método público *buscarPorRelacaoEntreCodigos* especificado na Figura 22 seleciona o método privado adequado para recuperar os tweets de um ou mais datasets que

Figura 21 – Método que recupera os tweets pertencentes a um Espaço de Possibilidade

```php
<?php

public function buscarPorEspacosPossibilidade($idDataset, $idEspPossibilidade)
{
    $cursor = $this->collection->find(array(
        '$and' => array(
            array(
                'idDataset' => array(
                    '$in' => $idDataset
                )
            ),
            array(
                'espacos_de_possibilidades' => array(
                    '$in' => $idEspPossibilidade
                )
            )
        )
    ));
    return $this->getTweets($cursor, $options);
}
```

Fonte: Produzido pelo autor

pertencem à relação entre uma linha e uma coluna da matriz.

Figura 22 – Método público que invoca o método privado capaz de buscar os tweets de uma relação

```php
<?php

public function buscarPorRelacaoEntreCodigos($elemMatriz, $options)
{

    $relacao = $elemMatriz->getMatriz()->getRelacao();

    switch ($relacao) {
        case 'NOT':
            return $this->buscarNotEntreCodigos($elemMatriz, $options);
            break;
        default:
            return $this->buscarAndOrEntreCodigos($elemMatriz, $options);
            break;
    }
}
```

Figura 23 – Método privado que recupera os tweets pertencentes a relação AND ou OR

```php
<?php
private function buscarAndOrEntreCodigos($elemMatriz, $options){
    $relacao        = $elemMatriz->getMatriz()->getRelacao();
    $relacaoMongoDB = '';
    switch ($relacao) {
        case 'AND':
            $relacaoMongoDB = '$and';
            break;
        case 'OR':
            $relacaoMongoDB = '$or';
            break;
    }
    $cursor = $this->collection->find(array(
        '$and' => array(
            array(
                'idDataset' => array(
                    '$in' => $elemMatriz->getMatriz()->getIdsDatasets()
                )
            ),
            array(
                $relacaoMongoDB => array(
                    array(
                        "codigos" => array(
                            '$elemMatch' => array(
                                "fator_e_circunstancia" =>
                                new \MongoId($elemMatriz->getIdLinha())
                            )
                        )
                    ),
                    array(
                        "codigos" => array(
                            '$elemMatch' => array(
                                "fator_e_circunstancia" =>
                                new \MongoId($elemMatriz->getIdColuna())
                            )
                        )
                    )
                )
            )
        )
    ));
    return $this->getTweets($cursor, $options);}
```

O método *buscarAndOrEntreCodigos* descrito na Figura 23 retorna os tweets pertencentes a uma relação AND ou OR. No caso da relação AND, são incluídos nos resultados

os tweets codificados simultaneamente com os Fatores e Circunstâncias especificados no elemento da matriz passado por parâmetro e dos datasets também passado por parâmetro. Para a relação OR, o tweet deve ter sido codificado em pelo menos uma das categorias do elemento da matriz.

Já o método *buscarNotEntreCodigos* especificado na Figura 24 busca pelos tweets em um ou mais datasets que pertencem a relação NOT entre uma linha e uma coluna da matriz. Para ser incluso nos resultados, o tweet deve ter sido codificado pelo Fator e Circunstância da linha da matriz e não ter sido atribuído ao Fator e Circunstância da coluna da matriz.

Figura 24 – Método privado que recupera os tweets pertencentes a relação NOT

```php
<?php
private function buscarNotEntreCodigos($elemMatriz, $options)
{
    $cursor = $this->collection->find(array(
        '$and' => array(
            array(
                'idDataset' => array(
                    '$in' => $elemMatriz->getMatriz()->getIdsDatasets()
                )
            ),
            array(
                "codigos" => array(
                    '$elemMatch' => array(
                        "fator_e_circunstancia" =>
                        new \MongoId($elemMatriz->getIdLinha())
                    )
                )
            ),
            array(
                "codigos" => array(
                    '$not' => array(
                        '$elemMatch' => array(
                            "fator_e_circunstancia" =>
                            new \MongoId($elemMatriz->getIdColuna())
                        )
                    )
                )
            )
        )
    ));
    return $this->getTweets($cursor, $options);
}
```

6 Experimentos com Mineração de Textos

Este capítulo trata da abordagem dada ao processo de descoberta de conhecimento em textos, descrito na seção 2.6 para realizar os experimentos com mineração de texto, com o objetivo de otimizar a execução da etapa de mineração por Espaços de Possibilidades da metodologia do COMUNIC.

A metodologia do COMUNIC, inicialmente, prevê a pesquisa por termos e palavras-chave nos tweets como método de categorização dos posts na segunda etapa. Porém, durante o progresso da aplicação da metodologia em um estudo, observou-se que o processo para consolidação do dicionário de palavras-chave de cada Espaço de Possibilidades, como o mostrado na Figura 12, é um processo demorado que exige muitos refinamentos. Os pesquisadores precisavam reler os trechos dos datasets, seja para obter mais palavras-chave ou para verificar se algum termo incluído anteriormente é realmente significativo. Todo esse retrabalho onera os pesquisadores e acaba se tornando um gargalo da metodologia.

Além disso, Mitra e Acharya (2005) citam dois problemas envolvendo a abordagem de categorização adotada para a etapa 2, que acaba não levando em consideração o significado semântico das palavras usadas na busca. Um dos problemas são os sinônimos, onde em um documento pode não existir um determinado termo, porém ser altamente relacionado ao termo em questão, já que na linguagem natural duas palavras diferentes podem ter o mesmo significado. Também é possível que uma palavra seja polissêmica, isto é, possua mais de um significado. Os experimentos com técnicas de mineração de textos serão utilizadas para diminuir o impacto dos problemas citados.

É importante destacar que na versão atual do SoNDA, este método alternativo não está disponível para uso, o trabalho contempla apenas a especificação e realização de experimentos.

6.1 Adaptações no processo adotado

Com relação ao processo de descoberta de conhecimento em textos proposto por Aranha, Vellasco e Passos (2007) e utilizado neste trabalho, duas etapas não serão tratadas. A etapa de coleta foi dispensada, pois os dados para o experimento já haviam sido obtidos pelo LABIC/UFES, laboratório parceiro do COMUNIC. Além disso, a etapa de indexação não será abordada, uma vez que os dados utilizados no experimento não serão armazenados em bancos de dados após a mineração.

6.2 Dados coletados

Para este experimento foram utilizados dois conjuntos de tweets coletados nos meses de janeiro e fevereiro de 2015, durante períodos em que ocorreram manifestações do Movimento Passe Livre (MPL). Estes datasets foram utilizados pelo grupo de pesquisadores do COMUNIC para estudar as ações deste coletivo ativista aplicando o desenho de pesquisa definido por eles. A Tabela 7 apresenta as informações referentes aos datasets utilizados neste experimento. Embora os termos usados para buscar e coletar os tweets sejam diferentes, ambos os datasets tratam do mesmo tema.

Tabela 7 – Datasets utilizados no experimento com mineração de textos

Dataset	Quantidade	Período de extração	Termo de pesquisa
1	18496	08/01/2015 - 10/01/2015	protesto
2	11538	31/01/2015 - 10/02/2015	passe livre

Fonte: Produzido pelo autor

6.3 Criação do conjunto de treinamento

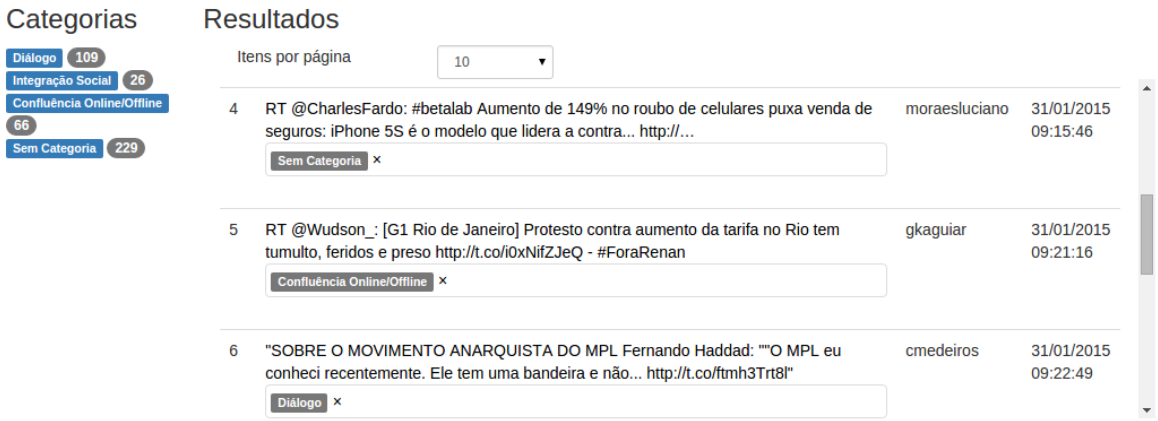
Um fragmento de cada dataset foi classificado manualmente por dois especialistas no domínio. Quanto aos conjuntos de treinamento criados, o dataset #2 foi rotulado por um especialista que detém maior conhecimento no domínio em relação ao analista que rotulou o dataset #1. A Figura 25 mostra a interface criada no próprio SoNDA para que os analistas pudessem realizar esta tarefa. Os datasets possuem vários tweets repetidos e estes foram filtrados para evitar itens duplicados no conjunto de treinamento. Esta interface foi integrada ao SoNDA apenas para os fins deste experimento, uma vez que a criação de um conjunto de treinamento não é uma funcionalidade da aplicação.

Cada tweet pode ser atribuído a uma ou mais categorias, com exceção dos tweets considerados irrelevantes. Assim, cada tweet poderia pertencer a qualquer combinação do conjunto de três categorias existentes: Diálogo, Integração Social e Confluência Online/Offline. Os tweets irrelevantes foram rotulados como “Sem categoria”.

A Tabela 8 e Tabela 9 apresentam a distribuição das categorias anotadas no primeiro e segundo dataset, respectivamente.

Estas categorias foram obtidas seguindo os passos que a metodologia prevê, ou seja, são provenientes das leituras de três trechos do início, meio e fim de cada um dos datasets por parte dos analistas, que identificaram estes temas-chave para investigar. O significado de cada categoria pode ser conferido em Coelho (2015). Por exemplo, tweets pertencentes a categoria “Confluência Online/Offline”, indicam manifestações ocorrendo tanto em espaços virtuais quanto nas ruas.

Figura 25 – Interface para criação do conjunto de treinamento



Fonte: Produzido pelo autor

Tabela 8 – Combinação de categorias anotadas manualmente no primeiro dataset

Combinação de categorias	**Quantidade**
Confluência Online/Offline	48
Confluência Online/Offline, Diálogo	2
Confluência Online/Offline, Integração Social	16
Diálogo	105
Diálogo, Integração Social	2
Integração Social	8
Sem categoria	229
Total	410

Fonte: Produzido pelo autor

6.4 Pré-processamento

As subseções a seguir tratam dos métodos e ferramentas utilizados para limpeza e transformação dos tweets para uma representação adequada à execução dos algoritmos de mineração. De acordo com Gokulakrishnan et al. (2012), a etapa de pré-processamento em textos de mídias sociais, como o Facebook e o Twitter, visa extrair o conteúdo relevante das mensagens e eliminar o que for dispensável. Algumas das técnicas utilizadas neste trabalho também foram aplicadas nos trabalhos de Gokulakrishnan et al. (2012) e Khan, Bashir e Qamar (2014).

Tabela 9 – Combinação de categorias anotadas manualmente no segundo dataset

Combinação de categorias	**Quantidade**
Confluência Online/Offline	37
Confluência Online/Offline, Diálogo	5
Confluência Online/Offline, Integração Social	25
Confluência Online/Offline, Diálogo, Integração Social	1
Diálogo	31
Diálogo, Integração Social	11
Integração Social	31
Sem categoria	226
Total	367

Fonte: Produzido pelo autor

6.4.1 Normalização

A normalização consiste em converter todas as letras maiúsculas para minúsculas, de forma que palavras escritas de maneiras diferentes possam ser tratadas da mesma forma. Por exemplo, o termo “ProTesTO” seria convertido para “protesto”.

6.4.2 Remoção de termos e símbolos irrelevantes

Utilizando expressões regulares, foram detectados URLs, menções a usuários, pontuações, caracteres especiais e emojis. Estes termos são considerados irrelevantes para a classificação e foram removidos. *Hashtags* não foram removidas, foi excluído apenas o símbolo # que serve para identificá-la. Para remoção das *stopwords*, foi utilizado a stoplist para o idioma português da biblioteca para Python NLTK (Bird, Klein e Loper (2009)). A seguir, um exemplo de tweet antes e depois de ser submetido a este processo:

```
olá @outrousuario, eu sou exemplo de tweet pré-processado!!! #tcc
#mineracao http://goo.gl/abc

olá exemplo tweet pré-processado tcc mineracao
```

No exemplo, as palavras “eu”, “sou” e de “foram” removidas por fazerem parte da lista de *stopwords*. Os demais itens eliminados foram a URL incluída na mensagem, a menção a outro usuário, pontuações e o símbolo # que identifica uma *hashtag*. Tanto a normalização quanto o processo de limpeza de termos irrelevantes foram implementados em um *script* na linguagem Python, que manipulou um arquivo CSV contendo o conjunto de treinamento.

6.4.3 Tokenização e transformação para vetor de atributos

Estas duas tarefas foram realizadas no módulo de pré-processamento do *software* utilizado para efetuar a mineração dos tweets, o MEKA (Read et al. (2016)). O MEKA é uma extensão do *software* para mineração de dados WEKA, que implementa métodos para classificação multirrótulo.

Tanto a tokenização quanto a transformação para vetor de atributos é realizada simultaneamente, através do método *StringToWordVector* existente no MEKA. Primeiramente, a lista de tokens de um tweet é obtida, quebrando a mensagem em partes utilizando os espaços em branco como delimitador de token.

Após a tokenização, o tweet é convertido para uma representação estruturada, de modo a estar preparado para ser submetido aos algoritmos de classificação. Cada tweet é representado como um vetor de atributos, onde cada entrada no vetor possui um valor numérico associado a cada token existente na coleção de tweets. Cada entrada do vetor pode assumir três tipos de representação: binária, indicando a presença ou ausência do token no tweet, como um número inteiro que representa a contagem de ocorrências do token no tweet, ou como número em ponto flutuante que indica o peso do token. No experimento, os tweets foram representados seguindo esta última representação. Foi utilizado a métrica TF-IDF para cálculo dos pesos, pois esta permite avaliar a importância de um termo no tweet para diferenciá-lo dos demais.

TF-IDF (é acrônimo para *term frequency–inverse document frequency*, que significa frequência do termo–inverso da frequência nos documentos) é uma métrica que avalia a importância de um termo num documento em relação aos outros documentos.

O cálculo do valor de TF-IDF do termo de um documento é composto pela multiplicação de outras duas medidas: TF e IDF, como mostra a fórmula a seguir:

$$tfidf(d,t) = tf(t,d) \cdot idf(t)$$

A métrica TF (*term frequency*), dada pela fórmula a seguir, calcula o número de vezes que o termo t ocorre no documento d.

$$tf(t,d) = \sum_{i \in d}^{|d|} 1\{d_i = t\}$$

A medida IDF (*inverse document frequency*), dada pela fórmula a seguir, mede a importância de um termo. Termos que aparecem com frequência em vários documentos não ajudam a distinguir um documento de outro, portanto, esta métrica visa diminuir o peso de um termo com muitas ocorrências na coleção de documentos e aumentar o peso

de termos com baixa frequência.

$$idf(t, d) = \log\left(\frac{\mid D \mid}{\sum\limits_{d \in D}^{|D|} 1\{t \in d\}}\right)$$

6.5 Mineração dos tweets

Os experimentos foram feitos com cada conjunto de tweets rotulado pelos especialistas submetidos individualmente aos algoritmos de classificação. Para validação e teste dos modelos, foi utilizada a técnica de validação cruzada particionando o conjunto através do método k-fold, com k=10. O método k-fold particiona o conjunto de treinamento em k subconjuntos, utilizando $k - 1$ subconjuntos para treinar o classificador e o conjunto restante para teste. O processo é repetido k vezes e cada um dos k subconjuntos é utilizado uma vez para avaliar o modelo. As métricas consideradas para avaliação foram a Acurácia, *Hamming Loss* e *One-error*.

Foram utilizados os métodos de classificação multirrótulo *Binary Relevance* e *Label Powerset*. Esta comparação pretende verificar se a dependência entre classes pode influenciar no desempenho do classificador. Para cada método de transformação do problema, foram aplicados os algoritmos de classificação monorrótulo SMO (Sequential minimal optimization) e Naive Bayes.

6.6 Resultados dos experimentos

A Tabela 10 e Tabela 11 apresentam o valor das métricas para os experimentos realizados com os dois conjuntos de dados rotulados. De um modo geral, a estratégia de transformação do problema *Label Powerset* teve resultados superiores com relação a de *Binary Relevance*, o que é possível concluir que há uma certa dependência entre as classes. O dataset #2, classificado pelo especialista com maior experiência obteve resultados melhores do que o dataset #1, o que pode indicar que o conhecimento do analista que rotulou as postagens também teve influência. A combinação de método de transformação e algoritmo de classificação que propiciou o melhor desempenho, foi *Label Powerset* com o algoritmo SMO.

Tabela 10 – Resultados do experimento com dataset #1

		Métricas		
Método de transformação	**Algoritmo**	**Acurácia**	**Hamming Loss**	**One-error**
Binary Reference	SMO	0.529	0.212	0.426
	Naive Bayes	0.563	0.243	0.391
Label Powerset	SMO	0.6	0.216	0.388
	Naive Bayes	0.586	0.223	0.396

Fonte: Produzido pelo autor

Tabela 11 – Resultados do experimento com dataset #2

		Métricas		
Método de transformação	**Algoritmo**	**Acurácia**	**Hamming Loss**	**One-error**
Binary Reference	SMO	0.709	0.122	0.237
	Naive Bayes	0.64	0.167	0.298
Label Powerset	SMO	0.757	0.125	0.237
	Naive Bayes	0.691	0.157	0.301

Fonte: Produzido pelo autor

7 Conclusão

O principal resultado deste trabalho foi o desenvolvimento do software SoNDA. O sistema foi criado de modo a atender uma demanda do COMUNIC, que necessitava de uma ferramenta computacional para realizar análise qualitativa sob um grande volume de dados provenientes de redes sociais. Foram revisados alguns dos softwares de mercado para análise qualitativa mais populares e identificado que eles não atendiam por completo os anseios dos pesquisadores do grupo de pesquisa.

Com base nas características do projeto e nas demandas dos pesquisadores, foi definido um processo de desenvolvimento para o SoNDA, que consistiu de uma etapa única para especificação da arquitetura do software e diversos ciclos iterativos onde as funcionalidades foram implementadas.

Além do sistema em si, este trabalho propôs um novo método para classificação das postagens. Inicialmente, a metodologia para análise prévia que esta categorização fosse feita através de pesquisa por termo ou palavra-chave. De modo a combater alguns dos problemas existentes nesta abordagem, foi seguido um processo de descoberta de conhecimento em textos, onde os tweets previamente coletados foram pré-processados e submetidos a algoritmos de classificação multirrótulo. Dentre os dois conjuntos de tweets rotulados usados no experimento, um deles obteve resultados satisfatórios, que ainda podem ser aperfeiçoados. A melhora pode ser atingida através de tarefas extras no pré-processamento, como a correção de palavras escritas incorretamente. Além disso, é necessário aumentar o conjunto de exemplos rotulados para evitar o desbalanceamento entre classes. A seção a seguir sugere algumas funcionalidades não-cobertas por este trabalho e que seriam úteis para as próximas versões.

7.1 Trabalhos futuros

As funcionalidades disponíveis nesta versão do SoNDA são suficientes para uma pesquisa aplicando a metodologia proposta por Lapa et al. (2015). Tomando como base outros *softwares* de mercado, é nítido que o SoNDA carece de alguns recursos que estão disponíveis em outras aplicações.

De modo a tornar o SoNDA um software mais completo e poderoso, para as futuras versões poderiam ser consideradas as seguintes melhorias:

- Extração de postagens diretamente pela aplicação: No momento, os dados a serem analisados no sistema devem ser coletados através de outras aplicações. A integração

da coleta inicial dos dados ao SoNDA dispensaria a utilização de outras ferramentas para esta tarefa.

- Dar suporte a dados de outras redes sociais: Atualmente, o sistema limita-se a importação de apenas posts do Twitter. Pode ser do interesse do pesquisador analisar posts de outras fontes, como o Facebook, LinkedIn, fóruns de discussão. Para isso, seria necessário adequar o sistema para lidar com dados de outros serviços de redes sociais.

- Flexibilização para outras metodologias: As operações e conceitos presentes no sistema estão fortemente relacionados a metodologia de análise do COMUNIC. Para viabilizar a utilização de outras metodologias de pesquisa, seria necessário tornar mais genéricas algumas nomenclaturas especificadas e efetuar uma análise a respeito de quais funcionalidades adicionais devem estar disponíveis.

- Integração com software de mineração: Neste trabalho, os experimentos com mineração de texto foram efetuados independente do SoNDA. Pode ser considerada a integração com o MEKA, por exemplo, para realizar a classificação dos posts diretamente pela aplicação.

Referências

AMPOFO, L. et al. Text mining and social media: When quantitative meets qualitative and software meets people. *Innovations in Digital Research Methods*, SAGE, p. 161–91, 2015. Citado na página 15.

ARANHA, C. N.; VELLASCO, M.; PASSOS, E. Uma abordagem de pré-processamento automático para mineração de textos em português: sob o enfoque da inteligência computacional. *Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ*, 2007. Citado 3 vezes nas páginas 27, 28 e 63.

ASUR, S.; HUBERMAN, B. A. Predicting the future with social media. In: IEEE. *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*. [S.l.], 2010. v. 1, p. 492–499. Citado na página 15.

BENEVENUTO, F.; ALMEIDA, J.; SILVA, A. Explorando redes sociais online: Da coleta e análise de grandes bases de dados às aplicações. *Mini-cursos do Simpósio Brasileiro de Redes de Computadores (SBRC)*, 2011. Citado na página 19.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python*. [S.l.]: "O'Reilly Media, Inc.", 2009. Citado na página 66.

BOYD, D. Social network sites as networked publics: Affordances, dynamics, and implications. In: PAPACHARISSI, Z. (Ed.). *A networked self: Identity, community, and culture on social network sites*. [S.l.]: Routledge, 2010. p. 39–58. Citado na página 14.

COELHO, I. C. *Internet e Educação: Aproximações inspiradas pelos movimentos sociais articulados em rede para a formação crítica de sujeitos*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2015. Citado na página 64.

ELLISON, N. B. et al. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, Wiley Online Library, v. 13, n. 1, p. 210–230, 2007. Citado na página 14.

FRANÇA, T. C. et al. Big social data: Princípios sobre coleta, tratamento e análise de dados sociais. 2014. Citado na página 19.

GOKULAKRISHNAN, B. et al. Opinion mining and sentiment analysis on a twitter data stream. In: IEEE. *Advances in ICT for Emerging Regions (ICTer), 2012 International Conference on*. [S.l.], 2012. p. 182–188. Citado na página 65.

JAVA, A. et al. Why we twitter: understanding microblogging usage and communities. In: ACM. *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*. [S.l.], 2007. p. 56–65. Citado na página 14.

KHAN, F. H.; BASHIR, S.; QAMAR, U. Tom: Twitter opinion mining framework using hybrid classification scheme. *Decision Support Systems*, Elsevier, v. 57, p. 245–257, 2014. Citado na página 65.

LAPA, A. B. et al. Fatores e circunstâncias para o empoderamento do sujeito nas redes sociais um desenho de pesquisa. *CIAIQ2015*, v. 2, 2015. Citado 2 vezes nas páginas 21 e 70.

MINER, G. *Practical text mining and statistical analysis for non-structured text data applications*. [S.l.]: Academic Press, 2012. Citado 3 vezes nas páginas 23, 27 e 28.

MISHNE, G.; GLANCE, N. S. et al. Predicting movie sales from blogger sentiment. In: *AAAI spring symposium: computational approaches to analyzing weblogs*. [S.l.: s.n.], 2006. p. 155–158. Citado na página 15.

MITRA, S.; ACHARYA, T. *Data mining: multimedia, soft computing, and bioinformatics*. [S.l.]: John Wiley & Sons, 2005. Citado na página 63.

ORESKOVIC, A. *Here's another area where Twitter appears to have stalled: tweets per day*. 2015. Disponível em: <http://www.businessinsider.com/twitter-tweets-per-day-appears-to-have-stalled-2015-6>. Acesso em: 06 jun 2016. Citado na página 15.

PAPACHARISSI, Z.; OLIVEIRA, M. de F. The rhythms of news storytelling on twitter: Coverage of the january 25th egyptian uprising on twitter. In: *World Association for Public Opinion Research Conference*. [S.l.: s.n.], 2011. v. 312, p. 3188. Citado na página 15.

READ, J. et al. MEKA: A multi-label/multi-target extension to Weka. *Journal of Machine Learning Research*, v. 17, n. 21, p. 1–5, 2016. Disponível em: <http://jmlr.org/papers/v17/12-164.html>. Citado na página 67.

REIS, L. P.; COSTA, A. P.; SOUZA, F. N. de. A survey on computer assisted qualitative data analysis software. In: AISTI. *Information Systems and Technologies (CISTI), 2016 11th Iberian Conference on*. [S.l.], 2016. p. 1–6. Citado 4 vezes nas páginas 18, 19, 30 e 31.

SCHWABER, K.; BEEDLE, M. Agile software development with scrum. 2002. Citado na página 39.

SILVA, E. M.; FERNEDA, E.; PRADO, H. A. D. Metodologia para descoberta de conhecimento em textos (text mining). 2007. Citado na página 27.

SILVER, C.; LEWINS, A. *Using software in qualitative research: A step-by-step guide*. [S.l.]: Sage, 2014. Citado na página 18.

SOARES, F. de A. *Categorização Automática de Textos Baseada em Mineração de Textos*. Tese (Doutorado) — PUC-Rio, 2013. Citado 3 vezes nas páginas 23, 28 e 29.

SOMMERVILLE, I. *Engenharia de software*. 9. ed. [S.l.]: PEARSON BRASIL, 2011. ISBN 9788579361081. Citado 2 vezes nas páginas 39 e 41.

SOROWER, M. S. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 2010. Citado 3 vezes nas páginas 24, 25 e 26.

SOUZA, F. N. d.; COSTA, A. P.; MOREIRA, A. Questionamento no processo de análise de dados qualitativos com apoio do software webqda. *EduSer-Revista de educação*, Instituto Politécnico de Bragança, Escola Superior de Educação, p. 19–30, 2011. Citado na página 55.

TAN, A.-H. et al. Text mining: The state of the art and the challenges. In: *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*. [S.l.: s.n.], 1999. v. 8, p. 65–70. Citado na página 23.

TAYLOR, C.; GIBBS, G. R. *What is Qualitative Data Analysis (QDA)?* 2010. Disponível em: <http://onlineqda.hud.ac.uk/Intro_QDA/what_is_qda.php>. Acesso em: 28 set 2016. Citado na página 17.

TSOUMAKAS, G.; KATAKIS, I. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006. Citado na página 24.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Mining multi-label data. In: *Data mining and knowledge discovery handbook*. [S.l.]: Springer, 2009. p. 667–685. Citado na página 26.

Apêndices

APÊNDICE A – Artigo

SoNDA: Um software para apoio à análise qualitativa de postagens de redes sociais

Cesar Smaniotto Júnior

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

cesar.junior@grad.ufsc.br

Abstract. *With the intention of observing the political action in social networks, Comunic research group developed a methodology of qualitative analysis of social networking posts. This methodology specifies a series of steps to filter a potentially large set of posts extracted from social networks, to get some instances of interest, where it's made content analysis in the dialogues. This work is the development of an application, called SoNDA (Social Network Data Analysis), which supports the three stages of analysis provided by the methodology. The work is not restricted only in the implementation of the software itself, but also aims to contribute to the methodology created by COMUNIC. It was proposed and carried out experiments with an alternative method for the categorization of the posts, through techniques of text mining. Posts were categorized using multi-label classification algorithms, which allow you to associate more than one category with a post.*

Resumo. *Com a intenção de observar a ação política nas redes sociais, o grupo de pesquisa COMUNIC do CED/UFSC elaborou uma metodologia de análise qualitativa de postagens de redes sociais. Esta metodologia especifica uma série de passos para filtrar um conjunto potencialmente grande de postagens extraídos das redes sociais, até obter algumas instâncias de interesse, onde é efetuada análise de conteúdo nos diálogos. O presente trabalho consiste no desenvolvimento de uma aplicação, denominada SoNDA (Social Network Data Analysis), que oferece suporte às três etapas de análise previstas pela metodologia. O trabalho não se restringe apenas na implementação do software, mas também visa contribuir com a metodologia criada pelo COMUNIC. Foi proposto e realizado experimentos com um método alternativo para a categorização das postagens, através de técnicas de mineração de textos. As postagens foram categorizadas utilizando algoritmos de classificação multirrótulo, que permitem associar mais de uma categoria a uma postagem.*

1. Introdução

A web 2.0 estabeleceu uma série de novos conceitos, além de ter promovido o usuário como gerador de conteúdo, ao invés de um mero consumidor da informação fornecida pelos sites. Esses avanços contribuíram para a evolução dos serviços de redes sociais online. Segundo [Boyd 2010], o perfil representa um indivíduo numa rede social e serve como local de interação com os demais integrantes da rede. As interações geralmente acontecem na forma de troca de mensagens privadas entre si, como um chat, e envio

de mensagens para o perfil virtual do usuário, chamadas postagens/publicações. Quanto ao caráter público ou semipúblico de um perfil, isso é uma restrição que o usuário pode impor a respeito de quais informações expostas por ele estão disponíveis para os demais visualizarem, e de que forma os outros membros da rede virtual podem interagir com ele. A rede de conexões de um usuário é, usualmente, constituída por pessoas que fazem parte do seu círculo social e indivíduos que compartilham interesses em comum.

Serviços de *microblogging* normalmente têm as mesmas características de redes sociais citadas acima, porém costumam impor restrição ao tamanho do texto que o usuário publica. Publicações em microblog são, tipicamente, expressões sucintas do estado do usuário em relação a algum tema, podendo este ser qualquer mensagem relacionada ao usuário, como, por exemplo, algo do seu cotidiano, uma notícia, um evento ou outros interesses.

[Java et al. 2007] diferenciam os *blogs* tradicionais de *microblogs* sob dois aspectos: primeiro, por incentivar a postagem de textos curtos, os *microblogs* requerem menos tempo do usuário para a geração de conteúdo; o outro fator que os difere é a frequência de atualização. Enquanto usuários de *blogs* costumam atualizar a sua página em um intervalo de dias, um usuário de *microblog* pode publicar várias atualizações por dia em seu perfil. Tornando, portanto, os serviços de *microblogging* uma ferramenta de comunicação extremamente ágil.

Uma das ferramentas de microblogging mais populares atualmente é o Twitter. O ranking Alexa[1] indica que é o oitavo site mais acessado no mundo. O Twitter permite os usuários enviarem mensagens de no máximo 140 caracteres, conhecidos como "tweets". Segundo dados divulgados pela própria empresa, o Twitter possui cerca de 313 milhões de usuários ativos mensalmente[2]. Não há dados atualizados sobre o número médio de tweets publicados por mês, mas a última vez que a empresa divulgou essa estatística foi em novembro de 2013, como informa [Oreskovic 2015]. Naquela altura, cerca de 500 milhões de tweets eram publicados por mês em média, e a tendência é que este número tenha crescido.

[Ampofo et al. 2015] apontam que a imensidão de conteúdo gerada por usuários de plataformas de redes sociais aliado ao surgimento de ferramentas e técnicas para armazenamento dos dados em tempo real e análise automatizada foram fundamentais para que o potencial dessas informações fossem exploradas por diversas áreas, como da inteligência de negócio, da área da saúde e das ciências sociais.

Ainda de acordo com [Ampofo et al. 2015], as empresas exploram o conteúdo das mídias sociais com o objetivo de obter ganho comercial. Exemplos deste tipo de aplicação são os trabalhos de [Mishne et al. 2006] e de [Asur and Huberman 2010] onde foram utilizados análise de conteúdo e de sentimentos em dados de mídias sociais para predizer o faturamento de filmes com bilheteria. No campo das ciências sociais, [Papacharissi and de Fatima Oliveira 2011] aplicaram análise de conteúdo automatizada e análise de discurso em posts do Twitter relacionados aos protestos no Egito que levaram a renúncia do presidente Hosni Mubarak, a fim de identificar e estudar as características das notícias reportadas.

[1]http://www.alexa.com/topsites. Acessado em 08 de outubro de 2016.

[2]https://about.twitter.com/company. Acessado em 08 de outubro de 2016.

Também da área de ciências sociais e humanas, o COMUNIC (Grupo de Pesquisa em Mídia-Educação e Comunicação Educacional), do Centro de Ciências da Educação da Universidade Federal de Santa Catarina (UFSC) elaborou uma metodologia para investigar a ação política promovida nas redes sociais por grupos ativistas. O desafio dos pesquisadores do COMUNIC é aplicar este desenho de pesquisa tendo como entrada um conjunto de posts de redes sociais, coletados em momentos de grande mobilização social e analisá-los qualitativamente, para identificar fatores e circunstâncias relevantes nos espaços sociais virtuais da internet para a formação crítica.

O presente trabalho visa dar suporte tecnológico aos pesquisadores, através da criação de um software que dê suporte às atividades especificadas na metodologia. Uma vez que o volume de dados extraído de redes sociais tende a ser grande, é necessário um software para auxiliar os pesquisadores em rotinas comuns do estudo, como na organização dos dados da pesquisa, filtragens e aquisição de novos dados a partir dos existentes. Este trabalho não se limita ao desenvolvimento do software que solucione o problema do COMUNIC, mas também visa aprimorar métodos de filtragem utilizados pela metodologia, propondo e realizando experimentos com um outro método para categorização das postagens, empregando técnicas de mineração de texto.

2. Sistema SoNDA

O SoNDA é um software para análise de posts de redes sociais, cujas funcionalidades foram projetadas para dar suporte às atividades descritas na metodologia criada pelo COMUNIC. Uma das principais razões de sua existência é o fato de outros softwares similares do mercado, expostos na Tabela 1, não satisfazerem as necessidades dos pesquisadores para apoiar seu estudo aplicando a metodologia. Tanto é, que todas as funcionalidades do SoNDA estão fortemente relacionadas à alguma etapa da metodologia, como será discutido mais adiante neste capítulo.

No comparativo entre as principais soluções de software para análise qualitativa descritos no 1, a mais próxima de corresponder às atividades da metodologia é o NVivo. O único aspecto não coberto por este software é a capacidade de recuperar todas as trocas comunicativas na qual um dado post está situado. Pois bem, o NVivo, assim como outros softwares comerciais analisados, não possibilitam a implementação de plug-ins e extensões que acrescentam funcionalidades à solução já existente, o que poderia resolver esse problema. Uma outra alternativa, seria recuperar o diálogo no qual um post está inserido, de forma manual ou automática, usando um software específico, importar estes dados para o NVivo, e criar um link entre o post e o seu diálogo. Desta maneira, o software semi-automatizaria as atividades previstas na metodologia. O principal objetivo do uso de um software para apoiar as pesquisas aplicando a metodologia do COMUNIC é, automatizar completamente tarefas que não dependam da interpretação do pesquisador. Este é um ponto importante, pois seria inconveniente para o usuário executar manualmente tarefas que poderiam ser realizadas automaticamente após poucos cliques em uma interface gráfica, sobretudo em um problema como o apresentado, onde o volume de dados a serem manipulados e analisados é abundante.

Podemos citar duas grandes diferenças do SoNDA para outras ferramentas do tipo CAQDAS. A primeira, é que a estrutura teórica é rígida, isto é, dependente do modelo de investigação do COMUNIC. Esta decisão levou em consideração que, em um

Tabela 1. Comparação resumida entre o SoNDA, NVivo, MAXQDA, Atlas.ti, Dedoose e webQDA

	SoNDA	NVivo	MAXQDA	Atlas.ti	Dedoose	webQDA
Extração de dados de redes sociais	✗	✓	✓	✗	✗	✗
Importação de dados de redes sociais via arquivo	✓	✓	✓	✓	✓	✗
Codificação automática de texto	✓	✓	✓	✓	✗	✗
Permitir trabalho colaborativo simultâneo e em tempo real	✓	✓	✗	✗	✓	✓
Compilação dos diálogos	✓	✗	✗	✗	✗	✗

primeiro momento, outros modelos de pesquisa não seriam utilizados no SoNDA. Em outros softwares do segmento, a estrutura é flexível, isto é, não impõe ao pesquisador seguir um determinado modelo de pesquisa para utilizar a ferramenta. Outra diferença está nos tipos de dados para análise suportados. Enquanto o SoNDA permite importar apenas posts de redes sociais, outros softwares de mercado permitem trabalhar com uma variedade de tipos de dados, como texto, áudios e vídeos em diversos formatos.

Em termos funcionais, o SoNDA é similar aos softwares analisados. Em todos eles, o pesquisador está apto a incluir os dados de entrada para análise, criar categorias, associar trechos dos dados de entrada a categorias, filtrar e questionar os dados, com o objetivo de responder às questões de sua pesquisa.

2.1. Escopo do sistema

O sistema desenvolvido neste trabalho apoia três etapas da metodologia especificada em [Lapa et al. 2015], são elas: etapas 2, 3 e 4. Portanto, a primeira etapa, conhecida como etapa de extração dos dados das redes sociais, não é contemplada no software, pois consideramos que há diversas ferramentas que atendem essa necessidade como, por exemplo, o TAGS (https://tags.hawksey.info/). Assim, o SoNDA foca na carga, pré-processamento e análise dos dados. Sendo assim, no contexto do SoNDA, a etapa 1 é denominada "configuração da pesquisa", enquanto as demais permanecem a mesma nomenclatura definida anteriormente.

Apesar do SoNDA pertencer ao segmento de aplicações CAQDAS, ele não dispõe de algumas características e funcionalidades comumente encontradas em outros *softwares* deste tipo, como por exemplo: possibilidade de criar estruturas hierárquicas de códigos e anexação de comentários e lembretes aos dados. O escopo do trabalho abrange apenas os requisitos necessários para o apoio à metodologia do COMUNIC.

Uma outra limitação do SoNDA é em relação aos serviços de redes sociais em que dá suporte. Atualmente, só é possível analisar dados extraídos do Twitter. Existe uma grande diferença no formato dos dados extraídos nas mais diversas redes sociais e, como

a metodologia foi elaborada para trabalhar sobre os dados do Twitter, focamos apenas nesta plataforma.

2.2. Arquitetura do sistema

O ponto de partida para o desenvolvimento foi a definição da arquitetura do sistema e das tecnologias empregadas na implementação. Na primeira elicitação de requisitos do sistema, foi estabelecida a restrição de que o *software* deveria ser executado a partir de um navegador web. Esta decisão de projeto impactou na escolha do modelo da arquitetura, onde foi feita a opção por um sistema cliente-servidor. [Sommerville 2011] explica que neste modelo, o usuário interage com uma aplicação executando em seu computador, por exemplo, um navegador web e comunica-se com um programa em execução em um computador remoto para adquirir os recursos fornecidos por ele, como páginas da web e imagens.

O SoNDA é constituído dessas duas camadas: cliente e servidor, onde cada uma pode ser tratada como um subsistema. Na aplicação servidor, estão implementadas a lógica da aplicação, a persistência dos dados e a interface para recuperar e modificar os dados. A aplicação cliente é responsável pela interação com o usuário, que envolve o carregamento das páginas, a validação inicial de entradas de formulários e a comunicação com o servidor para atender os objetivos do usuário.

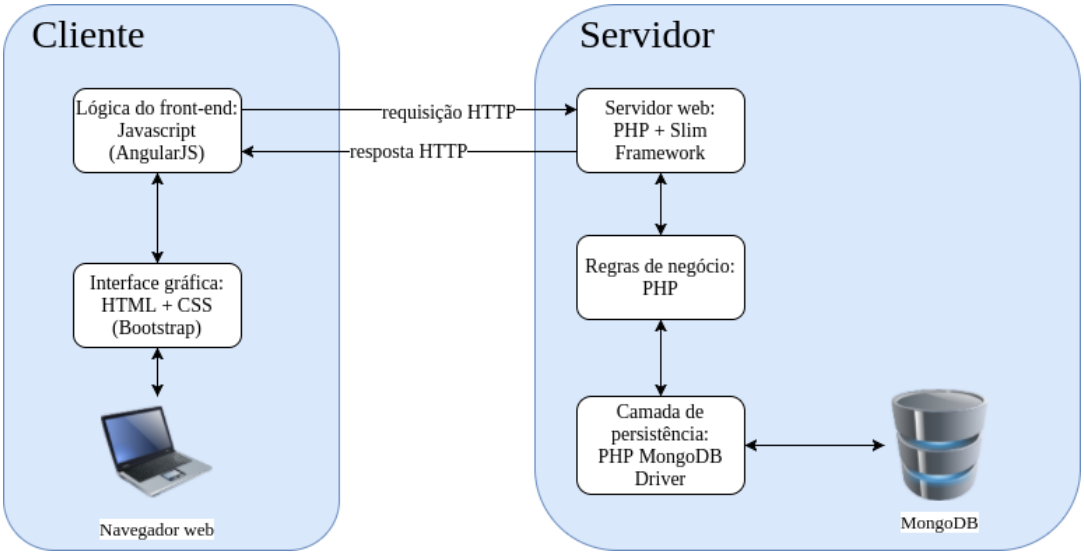
A comunicação entre os subsistemas é efetuada através de requisições HTTP a uma API REST. REST é acrônimo para *Representational State Transfer* (em português, Transferência de Estado Representacional). Para [Sommerville 2011], é um estilo arquitetural baseado na transferência de representação de recursos do servidor para o cliente, utilizando o protocolo HTTP para a comunicação entre as partes.

A Figura 1 ilustra os componentes da arquitetura do SoNDA e as respectivas tecnologias empregadas na implementação de cada um. A aplicação servidor foi escrita com a linguagem PHP. O *framework* Slim foi utilizado para criação da API, que acessa a camada de persistência para atender as requisições feitas pelos usuários. Para persistência dos dados, optou-se pelo MongoDB. Essa escolha se deve ao fato dos dados importados para o sistema não obedecerem uma estrutura rígida, o sistema é flexível quanto a nomes e quantidade de atributos de cada arquivo importado. O MongoDB atende este propósito, pois não impõe uma estrutura fixa para armazenar os registros.

A aplicação cliente é uma *single-page application* (aplicação de página única) implementada na linguagem Javascript e o *framework* AngularJS. Aplicações web deste gênero garantem uma experiência de uso similar às aplicações *desktop*, pois a mesma não é recarregada após uma ação do usuário. Para atender as solicitações dos usuários, a aplicação cliente efetua requisições à API REST implementada na aplicação servidor. Por fim, para o *layout* do sistema foi optado por um tema com design minimalista. De modo que o grupo de pesquisa não contava com um designer para criação do visual da aplicação, foi feita a opção por utilizar um template gratuito e de código aberto. Dentre as opções existentes, foi escolhido o tema construído com o *framework* Bootstrap SB Admin[3] para ser modificado e adaptado às necessidades do SoNDA.

[3]https://startbootstrap.com/template-overviews/sb-admin/

Figura 1. Arquitetura do SoNDA



2.3. Requisitos funcionais

Os requisitos funcionais de um sistema descrevem o que ele deve ser capaz de fazer. Os requisitos elicitados e implementados nos ciclos de desenvolvimento são expostos a seguir, agrupados pelas etapas da metodologia do COMUNIC que o SoNDA apoia, indicando quais funcionalidades são necessárias para atender as demandas de uma determinada fase.

- Etapa 1: Configuração da pesquisa
    1. O sistema deve permitir o cadastro de projetos.
    2. O sistema deve possibilitar a inclusão de membros ao projeto.
    3. O sistema deve permitir a importação de arquivo texto no formato CSV contendo tweets a um projeto.
- Etapa 2: Mineração por Espaços de possibilidades
    1. O sistema deve permitir a extração de amostras dos datasets importados no projeto.
    2. O sistema deve permitir o cadastro de categorias do tipo Espaços de possibilidades.
    3. O sistema deve ser capaz de filtrar os datasets por categorias de Espaços de possibilidades.
- Etapa 3: Mineração por Fatores e circunstâncias
    1. O sistema deve permitir o cadastro de categorias do tipo Fatores e circunstâncias.
    2. O sistema deve prover mecanismo para o pesquisador associar uma categoria do tipo Fator e circunstância a um tweet.
- Etapa 4: Recuperação dos diálogos
    1. O sistema deve fornecer mecanismos para o pesquisador interrogar os datasets codificados em Fatores e circunstâncias
    2. O sistema deve ser capaz de a partir de um tweet, recuperar o diálogo onde o mesmo está inserido.

2.4. Requisitos não-funcionais

Requisitos não-funcionais são restrições impostas aos serviços ou funcionalidades oferecidas pelo sistema. No SoNDA, estas restrições estão relacionados ao acesso às funcionalidades e dados do sistema.

1. Somente usuários registrados e identificados poderão ter acesso ao sistema.
2. Usuários só poderão ter acesso a projetos que criou e aos que foi convidado a colaborar.

3. Experimentos com Mineração de Textos

A metodologia do COMUNIC, inicialmente, prevê a pesquisa por termos e palavras-chave nos tweets como método de categorização dos posts por Espaços de Possibilidades. Porém, durante o progresso da aplicação da metodologia em um estudo, observou-se que o processo para consolidação do dicionário de palavras-chave de cada Espaço de Possibilidades, é um processo demorado que exige muitos refinamentos. Os pesquisadores precisavam reler os trechos dos datasets, seja para obter mais palavras-chave ou para verificar se algum termo incluído anteriormente é realmente significativo. Todo esse retrabalho onera os pesquisadores e acaba se tornando um gargalo da metodologia.

Além disso, [Mitra and Acharya 2005] citam dois problemas envolvendo a abordagem de categorização adotada para a etapa 2, que acaba não levando em consideração o significado semântico das palavras usadas na busca. Um dos problemas são os sinônimos, onde em um documento pode não existir um determinado termo, porém ser altamente relacionado ao termo em questão, já que na linguagem natural duas palavras diferentes podem ter o mesmo significado. Também é possível que uma palavra seja polissêmica, isto é, possua mais de um significado. Os experimentos com técnicas de mineração de textos serão utilizadas para diminuir o impacto dos problemas citados.

É importante destacar que na versão atual do SoNDA, este método alternativo não está disponível para uso, o trabalho contempla apenas a especificação e realização de experimentos.

3.1. Especificação dos experimentos

Para este experimento foram utilizados dois conjuntos de tweets coletados nos meses de janeiro e fevereiro de 2015, durante períodos em que ocorreram manifestações do Movimento Passe Livre (MPL). Estes datasets foram utilizados pelo grupo de pesquisadores do COMUNIC para estudar as ações deste coletivo ativista aplicando o desenho de pesquisa definido por eles.

Um fragmento de cada dataset foi classificado manualmente por dois especialistas no domínio. Quanto aos conjuntos de treinamento criados, o dataset #2 foi rotulado por um especialista que detém maior conhecimento no domínio em relação ao analista que rotulou o dataset #1. Cada tweet pode ser atribuído a uma ou mais categorias, com exceção dos tweets considerados irrelevantes. Assim, cada tweet poderia pertencer a qualquer combinação do conjunto de três categorias existentes: Diálogo, Integração Social e Confluência Online/Offline. Os tweets irrelevantes foram rotulados como “Sem categoria”.

3.2. Mineração dos tweets

Os experimentos foram feitos com cada conjunto de tweets rotulado pelos especialistas submetidos individualmente aos algoritmos de classificação. Para validação e teste dos modelos, foi utilizada a técnica de validação cruzada particionando o conjunto através do método k-fold, com k=10. O método k-fold particiona o conjunto de treinamento em k subconjuntos, utilizando $k - 1$ subconjuntos para treinar o classificador e o conjunto restante para teste. O processo é repetido k vezes e cada um dos k subconjuntos é utilizado uma vez para avaliar o modelo. As métricas consideradas para avaliação foram a Acurácia, *Hamming Loss* e *One-error*.

Foram utilizados os métodos de classificação multirrótulo *Binary Relevance* e *Label Powerset*. Esta comparação pretende verificar se a dependência entre classes pode influenciar no desempenho do classificador. Para cada método de transformação do problema, foram aplicados os algoritmos de classificação monorrótulo SMO (Sequential minimal optimization) e Naive Bayes.

3.3. Resultados dos experimentos

A 2 e 3 apresentam o valor das métricas para os experimentos realizados com os dois conjuntos de dados rotulados. De um modo geral, a estratégia de transformação do problema *Label Powerset* teve resultados superiores com relação a de *Binary Relevance*, o que é possível concluir que há uma certa dependência entre as classes. O dataset #2, classificado pelo especialista com maior experiência obteve resultados melhores do que o dataset #1, o que pode indicar que o conhecimento do analista que rotulou as postagens também teve influência. A combinação de método de transformação e algoritmo de classificação que propiciou o melhor desempenho, foi *Label Powerset* com o algoritmo SMO.

Tabela 2. Resultados do experimento com dataset #1

		Métricas		
Método de transformação	**Algoritmo**	**Acurácia**	**Hamming Loss**	**One-error**
Binary Reference	SMO	0.529	0.212	0.426
	Naive Bayes	0.563	0.243	0.391
Label Powerset	SMO	0.6	0.216	0.388
	Naive Bayes	0.586	0.223	0.396

Tabela 3. Resultados do experimento com dataset #2

		Métricas		
Método de transformação	**Algoritmo**	**Acurácia**	**Hamming Loss**	**One-error**
Binary Reference	SMO	0.709	0.122	0.237
	Naive Bayes	0.64	0.167	0.298
Label Powerset	SMO	0.757	0.125	0.237
	Naive Bayes	0.691	0.157	0.301

4. Conclusão

O principal resultado deste trabalho foi o desenvolvimento do software SoNDA. O sistema foi criado de modo a atender uma demanda do COMUNIC, que necessitava de uma

ferramenta computacional para realizar análise qualitativa sob um grande volume de dados provenientes de redes sociais. Foram revisados alguns dos softwares de mercado para análise qualitativa mais populares e identificado que eles não atendiam por completo os anseios dos pesquisadores do grupo de pesquisa.

Com base nas características do projeto e nas demandas dos pesquisadores, foi definido um processo de desenvolvimento para o SoNDA, que consistiu de uma etapa única para especificação da arquitetura do software e diversos ciclos iterativos onde as funcionalidades foram implementadas.

Além do sistema em si, este trabalho propôs um novo método para classificação das postagens. Inicialmente, a metodologia para análise prévia que esta categorização fosse feita através de pesquisa por termo ou palavra-chave. De modo a combater alguns dos problemas existentes nesta abordagem, foi seguido um processo de descoberta de conhecimento em textos, onde os tweets previamente coletados foram pré-processados e submetidos a algoritmos de classificação multirrótulo. Dentre os dois conjuntos de tweets rotulados usados no experimento, um deles obteve resultados satisfatórios, que ainda podem ser aperfeiçoados. A melhora pode ser atingida através de tarefas extras no pré-processamento, como a correção de palavras escritas incorretamente. Além disso, é necessário aumentar o conjunto de exemplos rotulados para evitar o desbalanceamento entre classes. A seção a seguir sugere algumas funcionalidades não-cobertas por este trabalho e que seriam úteis para as próximas versões.

Referências

Ampofo, L., Collister, S., O'Loughlin, B., and Chadwick, A. (2015). Text mining and social media: When quantitative meets qualitative and software meets people. *Innovations in Digital Research Methods*, pages 161–91.

Asur, S. and Huberman, B. A. (2010). Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE.

Boyd, D. (2010). Social network sites as networked publics: Affordances, dynamics, and implications. In Papacharissi, Z., editor, *A networked self: Identity, community, and culture on social network sites*, pages 39–58. Routledge.

Java, A., Song, X., Finin, T., and Tseng, B. (2007). Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM.

Lapa, A. B., Coelho, I. C., Ramos, V. F. C., and Malini, F. (2015). Fatores e circunstâncias para o empoderamento do sujeito nas redes sociais um desenho de pesquisa. *CIAIQ2015*, 2.

Mishne, G., Glance, N. S., et al. (2006). Predicting movie sales from blogger sentiment. In *AAAI spring symposium: computational approaches to analyzing weblogs*, pages 155–158.

Mitra, S. and Acharya, T. (2005). *Data mining: multimedia, soft computing, and bioinformatics*. John Wiley & Sons.

Oreskovic, A. (2015). Here's another area where twitter appears to have stalled: tweets per day.

Papacharissi, Z. and de Fatima Oliveira, M. (2011). The rhythms of news storytelling on twitter: Coverage of the january 25th egyptian uprising on twitter. In *World Association for Public Opinion Research Conference*, volume 312, page 3188.

Sommerville, I. (2011). *Engenharia de software*. PEARSON BRASIL, 9 edition.

APÊNDICE B – Código-fonte

```
<?php

ini_set('display_errors', true);

require_once __DIR__ . '/vendor/autoload.php';



use Slim\Slim;

use comunic\social_network_analyzer\model\facade\
   FacadeFactory;
use comunic\social_network_analyzer\model\repository\mongo\
   MongoRepository;
use comunic\social_network_analyzer\middlewares\TokenAuth;



$restapp = new Slim();

$mongo = new MongoRepository("workshoprio");
$factory = new FacadeFactory($mongo);

$restapp->add(new TokenAuth($factory->instantiateUsers()));

define('CONTROLLERS_PATH', __DIR__.'/src/comunic/
   social_network_analyzer/controllers/');

$controllerDir = opendir(CONTROLLERS_PATH);
while ($controller = readdir($controllerDir)) {
    if($controller != '.' && $controller != '..') {

        require CONTROLLERS_PATH . $controller;
    }
}
```

```
$restapp->get('/testarango/', function() {
    require_once 'testearango.php';
});



$restapp->run();

<?php

namespace sna\tests\model\entity\mappers{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\mappers\
   ArrayToDataset;
use comunic\social_network_analyzer\model\entity\Dataset;

class ArrayToDatasetTest extends PHPUnit_Framework_TestCase{


    public function testInvoke(){


        $arrayData = array("_id" => new \MongoId("54202
           c79d1c82dc01a000032"),
                                        "name" => "FooDataset
                                           ");

        $fArrayToDataset = new ArrayToDataset();
        $dataset = $fArrayToDataset($arrayData);

        $this->assertEquals($arrayData['_id']->{'$id'},
           $dataset->getId());
        $this->assertEquals($arrayData['name'], $dataset->
           getName());

    }
```

```
}


}

<?php

namespace sna\tests\model\entity\mappers{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\entity\mappers\
       DatasetToArray;
    use comunic\social_network_analyzer\model\entity\Dataset;

    class DatasetToArrayTest extends
       PHPUnit_Framework_TestCase{

        public function testInvoke(){
            $dataset = new Dataset();

            $expectedArray = array("_id" => new \MongoId("
               54202c79d1c82dc01a000032"),
                                    "name" => "FooDataset
                                       ");

            $dataset->setId("54202c79d1c82dc01a000032");
            $dataset->setName("FooDataset");

            $fDatasetToArray = new DatasetToArray();

            $resultArray = $fDatasetToArray($dataset);

            $this->assertEquals($expectedArray, $resultArray)
               ;



        }
```

```
    }

}
```

```
<?php

namespace sna\tests\model\entity\mappers{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\Category;
use comunic\social_network_analyzer\model\entity\mappers\
   ArrayToCategory;
use comunic\social_network_analyzer\model\util\
   IdMongoGenerator;

class ArrayToCategoryTest extends PHPUnit_Framework_TestCase{


    public function testeInvoke(){

        $expectedObj = new Category();
        $idMongo=new IdMongoGenerator();
        $catId = $idMongo();
        $expectedObj->setId($catId);
        $expectedObj->setName("FooCategory");
        $expectedObj->setKeywords(array("FooKw", "BarKw"));

        $arrayData = array(
            "_id" => new \MongoId($catId),
            "name" => "FooCategory",
            "keywords" => array("FooKw", "BarKw"),
            "included" => null,
            "excluded" => null
            );
```

```
        $fArrayToCategory = new ArrayToCategory();

        $this->assertEquals($expectedObj, $fArrayToCategory(
           $arrayData));

    }

}

}

<?php

namespace sna\tests\model\entity\mappers{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\entity\Tweet;
    use comunic\social_network_analyzer\model\entity\mappers\
       TweetToArray;
    use comunic\social_network_analyzer\model\util\
       IdMongoGenerator;

    class TweetToArrayTest extends PHPUnit_Framework_TestCase
       {


        public function testeInvoke(){

            $tweet = new Tweet();
            $idMongo=new IdMongoGenerator();
            $tweetId = $idMongo();
            $tweet->setId($tweetId);
            $tweet->setText("Luta␣contra␣a␣tarifa␣em␣Mau ,␣
               Grande␣S o␣Paulo:␣https://t.co/cYFaM8fWmD␣
               http://t.co/EifkoRBnXN");
            $tweet->setFromUser("mpl_sp");
            $tweet->setIdTweet("553315797247217664");
            $tweet->setCreatedAt("Thu␣Jan␣08␣22:22:20␣+0000␣
               2015");
```

```
            $tweet->setTime("1420755740");

            $expectedArray = array(
                "_id" => $tweetId,
                "text" => "Luta␣contra␣a␣tarifa␣em␣Mau ,␣
                   Grande␣S o␣Paulo:␣https://t.co/cYFaM8fWmD
                   ␣http://t.co/EifkoRBnXN",
                "fromUserId" => null,
                "idTweet" => "553315797247217664",
                "createdAt" => "Thu␣Jan␣08␣22:22:20␣+0000␣
                   2015",
                "time" => "1420755740",
                'toUserId' => null,
                'fromUser' => "mpl_sp",
                'isoLanguageCode' => null,
                'source' => null,
                'profileImageUrl' => null,
                'geoType' => null,
                'geoCoordinates0' => null,
                'geoCoordinates1' => null,

                );

            $fTweetToArray = new TweetToArray();

            $this->assertEquals($expectedArray,
               $fTweetToArray($tweet));

        }

    }

}

<?php

namespace sna\tests\model\entity\mappers{

    use PHPUnit_Framework_TestCase;
```

```
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToProject;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToDataset;

    class ArrayToProjectTest extends
       PHPUnit_Framework_TestCase{


        public function testInvoke(){
            $projectArray = array(
                "_id" => new \MongoId("54202
                   c79d1c82dc01a000034"),
                "name" => "FooProject",
                "datasetsIds" => "54202c79d1c82dc01a000032"
                );

            $farrayToProject = new ArrayToProject();

            $projectObj = $farrayToProject($projectArray);


            $this->assertEquals($projectArray['_id']->{'$id'
               }, $projectObj->getId());
            $this->assertEquals($projectArray['name'],
               $projectObj->getName());
            $this->assertEquals($projectArray['datasetsIds'],
               $projectObj->getDatasetsIds());
        }

    }


}


<?php
```

```
namespace sna\tests\model\entity\mappers\ProjectToArrayTest{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\entity\mappers\
       ProjectToArray;
    use comunic\social_network_analyzer\model\entity\Project;

    class ProjectToArrayTest extends
       PHPUnit_Framework_TestCase{


        public function testInvoke(){
            $arrayExpected = array(
                "_id" => new \MongoId("54202
                   c79d1c82dc01a000034"),
                "name" => "FooProject",
                "datasetsIds"=> "54202c79d1c82dc01a000032"
                );

            $project = new Project();
            $project->setId("54202c79d1c82dc01a000034");
            $project->setName("FooProject");
           $project->setDatasetsIds("54202c79d1c82dc01a000032
              ");

            $fProjectToArray = new ProjectToArray();
            $arrayResult = $fProjectToArray($project);

            $this->assertEquals($arrayExpected, $arrayResult)
               ;

        }

    }



}
```

```
<?php

namespace sna\tests\model\entity\mappers{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\Category;
use comunic\social_network_analyzer\model\entity\mappers\
   CategoryToArray;
use comunic\social_network_analyzer\model\util\
   IdMongoGenerator;

class CategoryToArrayTest extends PHPUnit_Framework_TestCase{


    public function testeInvoke(){

        $category = new Category();
        $idMongo=new IdMongoGenerator();
        $catId = $idMongo();
        $category->setId($catId);
        $category->setName("FooCategory");
        $category->setKeywords(array("FooKw", "BarKw"));

        $expectedArray = array(
            "_id" => new \MongoId($catId),
            "name" => "FooCategory",
            "keywords" => array("FooKw", "BarKw"),
            "included" => null,
            "excluded" => null
            );

        $fCategoryToArray = new CategoryToArray();

        $this->assertEquals($expectedArray, $fCategoryToArray
           ($category));

    }
```

```
}

}

<?php

namespace sna\tests\model\entity\mappers{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToTweet;


    class ArrayToTweetTest extends PHPUnit_Framework_TestCase
       {


        public function testInvoke(){
            $tweetAsArray = array(
                "_id" => new \MongoId("54202
                   c79d1c82dc01a000034"),
                "text" => "Luta␣contra␣a␣tarifa␣em␣Mau ,␣
                   Grande␣S o␣Paulo:␣https://t.co/cYFaM8fWmD
                   ␣http://t.co/EifkoRBnXN",
                "fromUserId" => null,
                "idTweet" => "553315797247217664",
                "createdAt" => "Thu␣Jan␣08␣22:22:20␣+0000␣
                   2015",
                "time" => "1420755740",
                 'toUserId' => null,
                'fromUser' => "mpl_sp",
                'isoLanguageCode' => null,
                'source' => null,
                'profileImageUrl' => null,
                'geoType' => null,
                'geoCoordinates0' => null,
                'geoCoordinates1' => null
                );
```

```
            $farrayToTweet = new ArrayToTweet();

            $tweetAsObj = $farrayToTweet($tweetAsArray);


            $this->assertEquals($tweetAsArray['_id']->{'$id'
               }, $tweetAsObj->getId());
            $this->assertEquals($tweetAsArray['text'],
               $tweetAsObj->getText());
            $this->assertEquals($tweetAsArray['fromUser'],
               $tweetAsObj->getFromUser());
        }

    }



}


<?php

namespace sna\tests\model\entity{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\Project;

class ProjectTest extends PHPUnit_Framework_TestCase{

    protected $project;

    protected function setUp(){
        $this->project = new Project();
    }

    public function testName(){

        $this->project->setName("Name");
```

```
        $this->assertEquals("Name", $this->project->getName()
           );


    }


    public function testId(){

        $this->project->setId("123456");

        $this->assertEquals("123456", $this->project->getId()
           );


    }




}
}



<?php

namespace sna\tests\model\entity\mappers{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\Category;
use comunic\social_network_analyzer\model\entity\format\json\
   JsonCategoryFormatter;
use comunic\social_network_analyzer\model\util\
   IdMongoGenerator;

class JsonCategoryFormatterTest extends
   PHPUnit_Framework_TestCase{
```

```
    public function testFormat(){

        $category = new Category();
        $idMongo=new IdMongoGenerator();
        $catId = $idMongo();
        $category->setId($catId);
        $category->setName("FooCategory");
        $category->setKeywords(array("FooKw", "BarKw"));

        $jsonExpected = '{"name":"FooCategory","keywords":["
           FooKw","BarKw"],"included":null,"excluded":null,"
           id":"'.$catId.'"}';

        $formatter = new JsonCategoryFormatter();

        $this->assertEquals($jsonExpected, $formatter->format
           ($category));


    }

}

}
```

```
<?php

namespace sna\tests\model\entity\format\json\
   JsonDatasetFormatterTest{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\Dataset;
use comunic\social_network_analyzer\model\entity\format\json\
   JsonDatasetFormatter;

class JsonDatasetFormatterTest extends
   PHPUnit_Framework_TestCase{
```

```
    public function testFormat(){

        $expectJson = '{"name":"FooDataset","id":"54202
           c79d1c82dc01a000032"}';
        $dataset = new Dataset();

        $dataset->setId("54202c79d1c82dc01a000032");
        $dataset->setName("FooDataset");

        $formatter = new JsonDatasetFormatter();

        $resultJson = $formatter->format($dataset);

        $this->assertEquals($expectJson, $resultJson);


    }

}


}

<?php

namespace sna\tests\model\entity\mappers{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\Tweet;
use comunic\social_network_analyzer\model\entity\format\json\
   JsonTweetFormatter;
use comunic\social_network_analyzer\model\util\
   IdMongoGenerator;

class JsonTweetFormatterTest extends
   PHPUnit_Framework_TestCase{

    public function testFormat(){
```

```
        $tweet = new Tweet();
        $idMongo=new IdMongoGenerator();
        $tweetId = $idMongo();
        $tweet->setId($tweetId);
        $tweet->setText("Luta␣contra␣a␣tarifa␣em␣Maua,␣Grande
           ␣Sao␣Paulo:");
        $tweet->setFromUser("mpl_sp");
        $tweet->setIdTweet("553315797247217664");
        $tweet->setCreatedAt("Thu␣Jan␣08␣22:22:20␣+0000␣2015"
           );
        $tweet->setTime("1420755740");

        $jsonExpected ='{"text":"Luta␣contra␣a␣tarifa␣em␣Maua
           ,␣Grande␣Sao␣Paulo:","toUserId":null,"fromUser":"
           mpl_sp","idTweet":"553315797247217664","fromUserId
           ":null,"isoLanguageCode":null,"source":null,"
           profileImageUrl":null,"geoType":null,"
           geoCoordinates0":null,"geoCoordinates1":null,"
           createdAt":"Thu␣Jan␣08␣22:22:20␣+0000␣2015","time
           ":"1420755740","id":"'.$tweetId.'"}';

        $formatter = new JsonTweetFormatter();

        $this->assertEquals($jsonExpected, $formatter->format
           ($tweet));


    }

}

}

<?php

namespace sna\tests\model\entity\format\json{

    use PHPUnit_Framework_TestCase;
```

```
    use comunic\social_network_analyzer\model\entity\Project;
    use comunic\social_network_analyzer\model\entity\Dataset;
    use comunic\social_network_analyzer\model\entity\format\
       json\JsonProjectFormatter;

    class JsonProjectFormatterTest extends
       PHPUnit_Framework_TestCase{

        protected $format;

        protected function setUp(){
            $this->formattter = new JsonProjectFormatter();

        }

        public function testFormat(){
            $project = new Project();
            $project->setId("54202c79d1c82dc01a000034");
            $project->setName("FooProject");
            $project->setDatasetsIds("54202
               c79d1c82dc01a000032");

            $expectjson = '{"name":"FooProject","datasetsIds
               ":"54202c79d1c82dc01a000032","id":"54202
               c79d1c82dc01a000034"}';
            $result = $this->formattter->format($project);

            $this->assertEquals($expectjson,$result);


        }



    }
}
```

```
<?php

namespace sna\tests\model\entity\parse\json{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\entity\parse\
       json\JsonDatasetParser;
    use comunic\social_network_analyzer\model\entity\Dataset;
    use comunic\social_network_analyzer\model\util\
       IdMongoGenerator;

    class JsonDatasetParserTest extends
       PHPUnit_Framework_TestCase{

        public function testParse(){

            $jsonTextDataset = '{"id":"54202
               c79d1c82dc01a000032","name":"FooDataset"}';
            $jsonObjDataset = \json_decode($jsonTextDataset);

            $idMongoGen = new IdMongoGenerator();

            $parser = new JsonDatasetParser($idMongoGen());
            $objDataset = $parser->parse($jsonTextDataset);

            $this->assertEquals($jsonObjDataset->{'id'},
               $objDataset->getId());
            $this->assertEquals($jsonObjDataset->{'name'},
               $objDataset->getName());




            $jsonTextDataset = '{"name":"BarDataset"}';
            $jsonObjDataset = \json_decode($jsonTextDataset);

            $parser = new JsonDatasetParser($idMongoGen());
            $objDataset = $parser->parse($jsonTextDataset);
```

```
            $this->assertNotEquals("54202c79d1c82dc01a000032"
               , $objDataset->getId());
            $this->assertEquals($jsonObjDataset->{'name'},
               $objDataset->getName());

        }


    }

}


<?php

namespace sna\tests\model\entity\mappers{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\entity\Category;
use comunic\social_network_analyzer\model\entity\parse\json\
   JsonCategoryParser;
use comunic\social_network_analyzer\model\util\
   IdMongoGenerator;

class JsonCategoryParserTest extends
   PHPUnit_Framework_TestCase{

    public function testParse(){

        $category = new Category();
        $idMongo=new IdMongoGenerator();
        $catId = $idMongo();
        $category->setId($catId);
        $category->setName("FooCategory");
        $category->setKeywords(array("FooKw", "BarKw"));

        $category2 = new Category();
        $catId2 = $idMongo();
```

```
        $category2->setId($catId2);
        $category2->setName("FooCategory");
        $category2->setKeywords(array("FooKw", "BarKw"));

        $jsonCategory = '[{"name":"FooCategory","keywords":["
           FooKw","BarKw"],"included":null,"excluded":null,"
           id":"'.$catId.'"},{"name":"FooCategory","keywords
           ":["FooKw","BarKw"],"included":null,"excluded":
           null,"id":"'.$catId2.'"}]';

        $parser = new JsonCategoryParser();

        $this->assertEquals(array($category, $category2),
           $parser->parse($jsonCategory));


    }

}

}

<?php

namespace sna\tests\model\entity\parse\json{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\entity\parse\
       json\JsonProjectParser;
    use comunic\social_network_analyzer\model\entity\Project;

    class JsonProjectParserTest extends
       PHPUnit_Framework_TestCase{

        public function testParse(){

            $jsonTextProject = '{"id":"54202
               c79d1c82dc01a000034","datasetsIds":"54202
               c79d1c82dc01a000032","name":"FooDataset"}';
```

```
            $jsonObjProject = \json_decode($jsonTextProject);
            $parser = new JsonProjectParser();
            $projectObj = $parser->parse($jsonTextProject);

            $this->assertEquals($jsonObjProject->{'id'},
               $projectObj->getId());
            $this->assertEquals($jsonObjProject->{'name'},
               $projectObj->getName());
            $this->assertEquals($jsonObjProject->{'
               datasetsIds'}, $projectObj->getDatasetsIds());
        }

    }
}


<?php

namespace sna\tests\model\entity\mappers{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\entity\Tweet;
    use comunic\social_network_analyzer\model\entity\parse\
       json\JsonTweetParser;
    use comunic\social_network_analyzer\model\util\
       IdMongoGenerator;

    class JsonTweetParserTest extends
       PHPUnit_Framework_TestCase{

        public function testParse(){

            $tweet = new Tweet();
            $idMongo=new IdMongoGenerator();
            $tweetId = $idMongo();
            $tweet->setId($tweetId);
            $tweet->setText("Luta␣contra␣a␣tarifa␣em␣Maua,␣
               Grande␣Sao␣Paulo");
            $tweet->setFromUser("mpl_sp");
```

```
            $tweet->setIdTweet("553315797247217664");
            $tweet->setCreatedAt("Thu␣Jan␣08␣22:22:20␣+0000␣
               2015");
            $tweet->setTime("1420755740");

            $tweetAsText ='[{"text":"Luta␣contra␣a␣tarifa␣em␣
               Maua,␣Grande␣Sao␣Paulo","toUserId":null,"
               fromUser":"mpl_sp","idTweet
               ":"553315797247217664","fromUserId":null,"
               isoLanguageCode":null,"source":null,"
               profileImageUrl":null,"geoType":null,"
               geoCoordinates0":null,"geoCoordinates1":null,"
               createdAt":"Thu␣Jan␣08␣22:22:20␣+0000␣2015","
               time":"1420755740","id":"'.$tweetId.'"}]';

            $parser = new JsonTweetParser();

            $this->assertEquals($tweet, $parser->parse(
               $tweetAsText)[0]);


        }

    }

}

<?php

namespace sna\tests\model\repository\mongo{

use Zumba\PHPUnit\Extensions\Mongo\Client\Connector;
use Zumba\PHPUnit\Extensions\Mongo\DataSet\DataSet;

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\repository\mongo\
   DatasetsRepository;
use comunic\social_network_analyzer\model\entity\Dataset as
   DatasetModel;
```

```
class DatasetsRepositoryTest extends
   PHPUnit_Framework_TestCase{

      const DEFAULT_DATABASE = "development";

         protected $connection;
         protected $dataset;
         protected $fixture ;

         protected $repository;

         public function getMongoConnection() {
             if (empty($this->connection)) {
                 $this->connection = new Connector(new \
                    MongoClient());
                 $this->connection->setDb(static::
                    DEFAULT_DATABASE);
             }
             return $this->connection;
         }
         public function getMongoDataSet() {
             if (empty($this->dataset)) {
                 $this->dataset = new DataSet($this->
                    getMongoConnection());
                 $this->dataset->setFixture($this->fixture);
                 $this->dataset->buildCollections();
             }
             return $this->dataset;
         }

         public function setUp(){
             $this->repository = new DatasetsRepository(static
                ::DEFAULT_DATABASE);

             $this->fixture = array(
                 "datasets" => array(
                     array(
                         "_id" => new \MongoId("54202
```

```
                        c79d1c82dc01a000032"),
                    "name" => "FooDataset",
                    ),
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000033"),
                    "name" => "BarDataset",
                    )
                ),

            "projects" => array(
                array(
                "_id" => new \MongoId("14202
                    c79d1c82dc01a000032"),
                "name" => "FooProject",
                "datasets_id" => array("54202
                    c79d1c82dc01a000032","54202
                    c79d1c82dc01a000033")
                )
                )

            );

        $this->getMongoConnection();
        $this->getMongoDataSet();
    }

    public function tearDown(){
        $this->dataset->dropAllCollections();
    }

public function testInsert(){
    $datasetObj = new DatasetModel();
    $datasetObj->setName("aDataset");

    $this->repository->insert($datasetObj);

    $result = $this->connection->collection("datasets")->
        findOne(array("name" => "aDataset"));
```

```
        $this->assertEquals($datasetObj->getName(), $result['
            name']);

    }

    public function testUpdate(){
        $datasetObj = new DatasetModel();
        $datasetObj->setId("54202c79d1c82dc01a000032");
        $datasetObj->setName("new␣name␣Dataset");

        $this->repository->insert($datasetObj);

        $result = $this->connection->collection("datasets")->
            findOne(array("_id" => new \MongoId("54202
            c79d1c82dc01a000032")));

        $this->assertEquals($datasetObj->getId(), $result['
            _id']->{'$id'});
        $this->assertEquals($datasetObj->getName(), $result['
            name']);

    }

    public function testDelete(){
        $this->assertEquals(2, $this->connection->collection(
            'datasets')->count());
        $this->repository->delete("54202c79d1c82dc01a000032")
            ;

        $this->assertEquals(1, $this->connection->collection(
            'datasets')->count());
        $this->assertNotEquals(0, $this->connection->
            collection('datasets')->count());
        $this->assertNotEquals(2, $this->connection->
            collection('datasets')->count());


    }
```

```
    public function testListAll(){

        $results = $this->repository->listAll();

        $this->assertEquals(\count($results), $this->
           connection->collection('datasets')->count());
        $this->assertNotEquals(\count($results)-1, $this->
           connection->collection('datasets')->count());
        $this->assertNotEquals(\count($results)+1, $this->
           connection->collection('datasets')->count());

    }

    public function testFindById(){

        $datasetObj = $this->repository->findById("54202
           c79d1c82dc01a000032");

        $this->assertEquals("FooDataset", $datasetObj->
           getName());

        $datasetsArray =  $this->repository->findById(array("
           54202c79d1c82dc01a000032","54202
           c79d1c82dc01a000033"));

        $this->assertEquals(2, \count($datasetsArray));
    }




}



}

<?php
```

```
namespace sna\tests\model\repository\mongo{

    use PHPUnit_Framework_TestCase;

    use Zumba\PHPUnit\Extensions\Mongo\Client\Connector;
    use Zumba\PHPUnit\Extensions\Mongo\DataSet\DataSet;

    use comunic\social_network_analyzer\model\repository\
       mongo\TweetsRepository;
    use comunic\social_network_analyzer\model\entity\Tweet;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToTweet;

    class TweetsRepositoryTest extends
       PHPUnit_Framework_TestCase{

        const DEFAULT_DATABASE = "development";

        protected $connection;
        protected $dataset;
        protected $fixture ;

        protected $repo;

        public function getMongoConnection() {
            if (empty($this->connection)) {
                $this->connection = new Connector(new \
                   MongoClient());
                $this->connection->setDb(static::
                   DEFAULT_DATABASE);
            }
            return $this->connection;
        }
        public function getMongoDataSet() {
            if (empty($this->dataset)) {
                $this->dataset = new DataSet($this->
                   getMongoConnection());
```

```
            $this->dataset->setFixture($this->fixture);
            $this->dataset->buildCollections();
        }
        return $this->dataset;
    }

    public function setUp(){
        $this->repo = new TweetsRepository(static::
            DEFAULT_DATABASE);

        $this->fixture = array(
            "tweets" => array(
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000032"),
                    "text" => "Luta␣contra␣a␣tarifa␣em␣
                        Maua,␣Grande␣S  o␣Paulo",
                    "fromUserId" => null,
                    "idTweet" => "553315797247217664",
                    "createdAt" => "Thu␣Jan␣08␣22:22:20␣
                        +0000␣2015",
                    "time" => "1420755740",
                    'toUserId' => null,
                    'fromUser' => "mpl_sp",
                    'isoLanguageCode' => null,
                    'source' => null,
                    'profileImageUrl' => null,
                    'geoType' => null,
                    'geoCoordinates0' => null,
                    'geoCoordinates1' => null
                    ),
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000033"),
                    "text" => "Luta␣contra␣a␣tarifa␣em␣
                        Niteroi,␣Rio␣de␣Janeiro",
                    "fromUserId" => null,
                    "idTweet" => "553315797247217665",
                    "createdAt" => "Thu␣Jan␣08␣22:22:20␣
```

```
                            +0000␣2015",
                        "time" => "1420755740",
                        'toUserId' => null,
                        'fromUser' => "mpl_rj",
                        'isoLanguageCode' => null,
                        'source' => null,
                        'profileImageUrl' => null,
                        'geoType' => null,
                        'geoCoordinates0' => null,
                        'geoCoordinates1' => null
                        )
                    )
);

$this->getMongoConnection();
$this->getMongoDataSet();
}

public function tearDown(){
    $this->dataset->dropAllCollections();
}

public function testFindById(){

    $tweet = $this->repo->findById("54202c79d1c82dc01a000032"
       );

    $this->assertEquals("54202c79d1c82dc01a000032", $tweet->
       getId());
    $this->assertEquals("Luta␣contra␣a␣tarifa␣em␣Maua,␣Grande
       ␣S o␣Paulo", $tweet->getText());

    $this->assertNotEquals("54202c79d1c82dc01a000031", $tweet
       ->getId());

}


public function testInsert(){
```

```
        $tweet = new Tweet();
        $tweet->setText("Luta␣contra␣a␣tarifa␣em␣Salvador,␣Bahia"
            );

        $this->repo->insert($tweet);

        $result = $this->connection->collection('tweets')->
            findOne(['text' => "Luta␣contra␣a␣tarifa␣em␣Salvador,␣
            Bahia"]);

        $this->assertEquals($tweet->getText(), $result['text']);

    }


    public function testListAll(){
        $count = $this->connection->collection('tweets')->count()
            ;
        $this->assertEquals(2, $count);

        $this->assertEquals(2,count($this->repo->listAll()));

        $this->assertNotEquals($count+1, count($this->repo->
            listAll()));

    }




}
}
```

```
<?php
```

```
namespace sna\tests\model\repository\mongo{

    use PHPUnit_Framework_TestCase;

    use Zumba\PHPUnit\Extensions\Mongo\Client\Connector;
    use Zumba\PHPUnit\Extensions\Mongo\DataSet\DataSet;

    use comunic\social_network_analyzer\model\repository\
       mongo\ProjectsRepository;
    use comunic\social_network_analyzer\model\entity\Project;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToProject;

    class ProjectsRepositoryTest extends
       PHPUnit_Framework_TestCase{

        const DEFAULT_DATABASE = "development";

        protected $connection;
        protected $dataset;
        protected $fixture ;

        protected $repo;

        public function getMongoConnection() {
            if (empty($this->connection)) {
                $this->connection = new Connector(new \
                   MongoClient());
                $this->connection->setDb(static::
                   DEFAULT_DATABASE);
            }
            return $this->connection;
        }
        public function getMongoDataSet() {
            if (empty($this->dataset)) {
                $this->dataset = new DataSet($this->
                   getMongoConnection());
                $this->dataset->setFixture($this->fixture);
                $this->dataset->buildCollections();
```

```
        }
        return $this->dataset;
    }

    public function setUp(){
        $this->repo = new ProjectsRepository(static::
            DEFAULT_DATABASE);

        $this->fixture = array(
            "projects" => array(
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000032"),
                    "name" => "FooProject"
                    ),
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000033"),
                    "name" => "BarProject"
                    )
                )
            );

        $this->getMongoConnection();
        $this->getMongoDataSet();
    }

    public function tearDown(){
        $this->dataset->dropAllCollections();
    }

    public function testFindById(){

        $project = $this->repo->findById("54202
            c79d1c82dc01a000032");

        $this->assertEquals("54202c79d1c82dc01a000032",
            $project->getId());
        $this->assertEquals("FooProject", $project->
```

```
        getName());

    $this->assertNotEquals("54202c79d1c82dc01a000031"
        , $project->getId());

}


public function testUpdate(){
    $project = new Project();
    $project->setId("54202c79d1c82dc01a000034");
    $project->setName("umProjeto");

    $this->repo->insert($project);

    $result = $this->connection->collection('projects
        ')->findOne(['_id' => new \MongoId('54202
        c79d1c82dc01a000034')]);

    $this->assertEquals($project->getName(), $result[
        'name']);
    $this->assertEquals($project->getId(), $result['
        _id']->{'$id'});

}

public function testInsert(){
    $project = new Project();
    $project->setName("BarProject");

    $this->repo->insert($project);
    $result = $this->connection->collection('projects
        ')->findOne(['name' => "BarProject"]);

    $this->assertEquals($project->getName(), $result[
        'name']);

}
```

```
        public function testDelete(){
            $count = $this->connection->collection('projects'
               )->count();
            $this->assertEquals(2, $count);

            $this->repo->delete("54202c79d1c82dc01a000032");

            $count = $this->connection->collection('projects'
               )->count();
            $this->assertEquals(1, $count);
            $this->assertNotEquals(0, $count);
            $this->assertNotEquals(2, $count);
        }

        public function testListAll(){
            $count = $this->connection->collection('projects'
               )->count();
            $this->assertEquals(2, $count);

            $this->assertEquals(2,count($this->repo->listAll
               ()));

            $this->assertNotEquals($count+1, count($this->
               repo->listAll()));

        }




    }
}



<?php
```

```
namespace sna\tests\model\repository\mongo{

    use PHPUnit_Framework_TestCase;

    use Zumba\PHPUnit\Extensions\Mongo\Client\Connector;
    use Zumba\PHPUnit\Extensions\Mongo\DataSet\DataSet;

    use comunic\social_network_analyzer\model\repository\
       mongo\CategoriesRepository;
    use comunic\social_network_analyzer\model\entity\Category
       ;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToCategory;

    class CategoriesRepositoryTest extends
       PHPUnit_Framework_TestCase{

        const DEFAULT_DATABASE = "development";

        protected $connection;
        protected $dataset;
        protected $fixture ;

        protected $repo;

        public function getMongoConnection() {
            if (empty($this->connection)) {
                $this->connection = new Connector(new \
                   MongoClient());
                $this->connection->setDb(static::
                   DEFAULT_DATABASE);
            }
            return $this->connection;
        }
        public function getMongoDataSet() {
            if (empty($this->dataset)) {
                $this->dataset = new DataSet($this->
                   getMongoConnection());
                $this->dataset->setFixture($this->fixture);
```

```
            $this->dataset->buildCollections();
        }
        return $this->dataset;
    }

    public function setUp(){
        $this->repo = new CategoriesRepository(static::
            DEFAULT_DATABASE);

        $this->fixture = array(
            "categories" => array(
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000032"),
                    "name" => "FooCategory",
                    "keywords"=> array("FooKw", "BarKw"),
                    "included"=>null,
                    "excluded"=>null
                    ),
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000033"),
                    "name" => "BarCategory",
                    "keywords"=> array("FooKw", "BarKw"),
                    "included"=>null,
                    "excluded"=>null
                    )
                )
            );

        $this->getMongoConnection();
        $this->getMongoDataSet();
    }

    public function tearDown(){
        $this->dataset->dropAllCollections();
    }

    public function testFindById(){
```

```
        $category = $this->repo->findById("54202
            c79d1c82dc01a000032");

        $this->assertEquals("54202c79d1c82dc01a000032",
            $category->getId());
        $this->assertEquals("FooCategory", $category->
            getName());

        $this->assertNotEquals("54202c79d1c82dc01a000031"
            , $category->getId());

    }


    public function testUpdate(){
        $category = new Category();
        $category->setId("54202c79d1c82dc01a000034");
        $category->setName("umaCategoria");
        $category->setKeywords(array("umakw","duaskw"));

        $this->repo->insert($category);

        $result = $this->connection->collection('
            categories')->findOne(['_id' => new \MongoId('
            54202c79d1c82dc01a000034')]);

        $this->assertEquals($category->getName(), $result
            ['name']);
        $this->assertEquals($category->getId(), $result['
            _id']->{'$id'});

    }

    public function testInsert(){
        $category = new Category();
        $category->setName("umaCategoria");
        $category->setKeywords(array("umakw","duaskw"));
        $this->repo->insert($category);
```

```
        $result = $this->connection->collection('
            categories')->findOne(['name' => "umaCategoria
            "]);

        $this->assertEquals($category->getName(), $result
            ['name']);

    }

    public function testDelete(){
        $count = $this->connection->collection('
            categories')->count();
        $this->assertEquals(2, $count);

        $this->repo->delete("54202c79d1c82dc01a000032");

        $count = $this->connection->collection('
            categories')->count();
        $this->assertEquals(1, $count);
        $this->assertNotEquals(0, $count);
        $this->assertNotEquals(2, $count);
    }

    public function testListAll(){
        $count = $this->connection->collection('
            categories')->count();
        $this->assertEquals(2, $count);

        $this->assertEquals(2,count($this->repo->listAll
            ()));

        $this->assertNotEquals($count+1, count($this->
            repo->listAll()));

    }
```

```
    }
}
```

```
<?php

namespace sna\tests\model\facade{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\facade\
       ProjectsFacade;
    use comunic\social_network_analyzer\model\repository\
       mongo\ProjectsRepository;
    use comunic\social_network_analyzer\model\repository\
       mongo\DatasetsRepository;
    use comunic\social_network_analyzer\model\entity\Dataset
       as DatasetModel;
    use comunic\social_network_analyzer\model\entity\format\
       json\JsonDatasetFormatter;
    use comunic\social_network_analyzer\model\entity\format\
       json\JsonProjectFormatter;
    use comunic\social_network_analyzer\model\entity\parse\
       json\JsonProjectParser;

    use Zumba\PHPUnit\Extensions\Mongo\Client\Connector;
    use Zumba\PHPUnit\Extensions\Mongo\DataSet\DataSet;

    class ProjectsFacadeTest extends
       PHPUnit_Framework_TestCase{

        const DEFAULT_DATABASE = "development";

        protected $connection;
        protected $dataset;
        protected $fixture ;
```

```
protected $facade;
protected $projectRepo;
protected $datasetRepo;


public function getMongoConnection() {
    if (empty($this->connection)) {
        $this->connection = new Connector(new \
           MongoClient());
        $this->connection->setDb(static::
           DEFAULT_DATABASE);
    }
    return $this->connection;
}
public function getMongoDataSet() {
    if (empty($this->dataset)) {
        $this->dataset = new DataSet($this->
           getMongoConnection());
        $this->dataset->setFixture($this->fixture);
        $this->dataset->buildCollections();
    }
    return $this->dataset;
}

public function setUp(){
    $this->projectRepo = new ProjectsRepository(
       static::DEFAULT_DATABASE);
    $this->datasetRepo = new DatasetsRepository(
       static::DEFAULT_DATABASE);
    $this->facade = new ProjectsFacade($this->
       projectRepo, $this->datasetRepo);

    $this->fixture = array(
        "datasets" => array(
            array(
                "_id" => new \MongoId("54202
                   c79d1c82dc01a000032"),
                "name" => "FooDataset",
                ),
```

```
                array(
                    "_id" => new \MongoId("54202
                       c79d1c82dc01a000033"),
                    "name" => "BarDataset",
                    )
                ),

            "projects" => array(
                array(
                    "_id" => new \MongoId("14202
                       c79d1c82dc01a000032"),
                    "name" => "FooProject",
                    "datasetsIds" => array("54202
                       c79d1c82dc01a000032","54202
                       c79d1c82dc01a000033")
                    )
                )

            );

        $this->getMongoConnection();
        $this->getMongoDataSet();
    }

    public function tearDown(){
        $this->dataset->dropAllCollections();
    }

    public function testGetDatasets(){

        $datasetsArrayExpected = $this->datasetRepo->
           findById(array("54202c79d1c82dc01a000032","
           54202c79d1c82dc01a000033"));
        $formatter = new JsonDatasetFormatter();

        $this->assertEquals($formatter->format(
           $datasetsArrayExpected), $this->facade->
           getDatasets("14202c79d1c82dc01a000032",
           $formatter));
```

```
}

public function testInsert(){

    $projectAsText =  '{"name":"BarProject","
       datasetsIds":"54202c79d1c82dc01a000032"}';

    $parser = new JsonProjectParser();
    $this->facade->insert($projectAsText, $parser);

    $this->assertEquals(1, $this->connection->
       collection('projects')->count(array("name" =>
       "BarProject")));

}

public function testUpdate(){

    $projectAsText =  '{"name":"BarProject","
       datasetsIds":"54202c79d1c82dc01a000032","id
       ":"54202c79d1c82dc01a000035"}';

    $parser = new JsonProjectParser();
    $this->facade->insert($projectAsText, $parser);

    $this->assertEquals($parser->parse($projectAsText
       ), $this->projectRepo->findById("54202
       c79d1c82dc01a000035"));

}

public function testDelete(){

    $this->assertEquals(1, $this->connection->
       collection('projects')->count());

    $this->facade->delete("14202c79d1c82dc01a000032")
       ;
```

```
            $this->assertEquals(0, $this->connection->
               collection('projects')->count());

        }

        public function testListAll(){
            $projects = $this->projectRepo->listAll();

            $formatter = new JsonProjectFormatter();

            $this->assertEquals($formatter->format($projects)
               , $this->facade->listAll($formatter));
        }

        public function testFindById(){
            $project = $this->projectRepo->findById("14202
               c79d1c82dc01a000032");

            $formatter = new JsonProjectFormatter();

            $this->assertEquals($formatter->format($project),
                $this->facade->findById("14202
               c79d1c82dc01a000032",$formatter));
        }


    }


}

<?php

namespace sna\tests\model\facade{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\facade\
       CategoriesFacade;
```

```
use comunic\social_network_analyzer\model\repository\
   mongo\CategoriesRepository;
use comunic\social_network_analyzer\model\entity\Category
   ;
use comunic\social_network_analyzer\model\entity\format\
   json\JsonCategoryFormatter;
use comunic\social_network_analyzer\model\entity\parse\
   json\JsonCategoryParser;
use comunic\social_network_analyzer\model\util\
   IdMongoGenerator;

use Zumba\PHPUnit\Extensions\Mongo\Client\Connector;
use Zumba\PHPUnit\Extensions\Mongo\DataSet\DataSet;

class CategoriesFacadeTest extends
   PHPUnit_Framework_TestCase{

    const DEFAULT_DATABASE = "development";

    protected $connection;
    protected $dataset;
    protected $fixture ;

    protected $facade;
    protected $catRepo;

    public function getMongoConnection() {
        if (empty($this->connection)) {
            $this->connection = new Connector(new \
               MongoClient());
            $this->connection->setDb(static::
               DEFAULT_DATABASE);
        }
        return $this->connection;
    }
    public function getMongoDataSet() {
        if (empty($this->dataset)) {
            $this->dataset = new DataSet($this->
               getMongoConnection());
```

```
            $this->dataset->setFixture($this->fixture);
            $this->dataset->buildCollections();
        }
        return $this->dataset;
    }

    public function setUp(){
        $this->catRepo = new CategoriesRepository(static
            ::DEFAULT_DATABASE);
        $this->facade = new CategoriesFacade($this->
            catRepo);

        $this->fixture = array(
            "categories" => array(
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000032"),
                    "name" => "FooCategory",
                    "keywords" => array("FooKw", "BarKw")
                        ,
                    "included" => null,
                    "excluded" => null
                    ),
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000033"),
                    "name" => "BarCategory",
                    "keywords" => array("FooKw", "BarKw")
                        ,
                    "included" => null,
                    "excluded" => null
                    )
                )


            );

        $this->getMongoConnection();
        $this->getMongoDataSet();
```

```
}

public function tearDown(){
    $this->dataset->dropAllCollections();
}

public function testInsert(){

    $categoryAsText = '{"name":"New␣Category","
       keywords":["FooKw","BarKw"],"included":null,"
       excluded":null}';


    $parser = new JsonCategoryParser();
    $this->facade->update($categoryAsText, $parser);

    $this->assertEquals(1, $this->connection->
       collection("categories")->count(["name" => "
       New␣Category"]));
}

 public function testUpdate(){
    $idMongoGen = new IdMongoGenerator();
    $catId = $idMongoGen();

    $categoryAsText = '{"name":"FooCategory","
       keywords":["FooKw","BarKw"],"included":null,"
       excluded":null,"id":"'.$catId.'"}';


    $parser = new JsonCategoryParser();
    $this->facade->update($categoryAsText, $parser);

    $this->assertEquals($parser->parse(
       $categoryAsText), $this->catRepo->findById(
       $catId));
```

```
    }

    public function testDelete(){

        $this->assertEquals(2, $this->connection->
            collection('categories')->count());

        $this->facade->delete("54202c79d1c82dc01a000032")
            ;

        $this->assertEquals(1, $this->connection->
            collection('categories')->count());

    }

    public function testListAll(){
        $categories = $this->catRepo->listAll();

        $formatter = new JsonCategoryFormatter();

        $this->assertEquals($formatter->format(
            $categories), $this->facade->listAll(
            $formatter));
    }

    public function testFindById(){
        $category = $this->catRepo->findById("54202
            c79d1c82dc01a000032");

        $formatter = new JsonCategoryFormatter();

        $this->assertEquals($formatter->format($category)
            , $this->facade->findById("54202
            c79d1c82dc01a000032",$formatter));
    }


}
```

```
}

<?php

namespace sna\tests\model\facade{

    use PHPUnit_Framework_TestCase;
    use comunic\social_network_analyzer\model\facade\
       TweetsFacade;
    use comunic\social_network_analyzer\model\repository\
       mongo\TweetsRepository;
    use comunic\social_network_analyzer\model\repository\
       mongo\CategoriesRepository;
    use comunic\social_network_analyzer\model\entity\Tweet;
    use comunic\social_network_analyzer\model\entity\format\
       json\JsonTweetFormatter;
    use comunic\social_network_analyzer\model\entity\parse\
       json\JsonTweetParser;
    use comunic\social_network_analyzer\model\util\
       IdMongoGenerator;

    use Zumba\PHPUnit\Extensions\Mongo\Client\Connector;
    use Zumba\PHPUnit\Extensions\Mongo\DataSet\DataSet;

    class CategoriesFacadeTest extends
       PHPUnit_Framework_TestCase{

        const DEFAULT_DATABASE = "development";

        protected $connection;
        protected $dataset;
        protected $fixture ;

        protected $facade;
        protected $tweetsRepo;
        protected $catRepo;

        public function getMongoConnection() {
```

```
        if (empty($this->connection)) {
            $this->connection = new Connector(new \
                MongoClient());
            $this->connection->setDb(static::
                DEFAULT_DATABASE);
        }
        return $this->connection;
    }
    public function getMongoDataSet() {
        if (empty($this->dataset)) {
            $this->dataset = new DataSet($this->
                getMongoConnection());
            $this->dataset->setFixture($this->fixture);
            $this->dataset->buildCollections();
        }
        return $this->dataset;
    }

    public function setUp(){
        $this->catRepo = new CategoriesRepository(static
            ::DEFAULT_DATABASE);
        $this->tweetsRepo = new TweetsRepository(static::
            DEFAULT_DATABASE);

        $this->facade = new TweetsFacade($this->
            tweetsRepo, $this->catRepo);

        $this->fixture = array(
            "tweets" => array(
                array(
                    "_id" => new \MongoId("54202
                        c79d1c82dc01a000032"),
                    "text" => "Luta␣contra␣a␣tarifa␣em␣
                        Maua,␣Grande␣S o␣Paulo",
                    "fromUserId" => null,
                    "idTweet" => "553315797247217664",
                    "createdAt" => "Thu␣Jan␣08␣22:22:20␣
                        +0000␣2015",
                    "time" => "1420755740",
```

```
                        'toUserId' => null,
                        'fromUser' => "mpl_sp",
                        'isoLanguageCode' => null,
                        'source' => null,
                        'profileImageUrl' => null,
                        'geoType' => null,
                        'geoCoordinates0' => null,
                        'geoCoordinates1' => null
                        ),
                    array(
                        "_id" => new \MongoId("54202
                            c79d1c82dc01a000033"),
                        "text" => "Luta␣contra␣a␣tarifa␣em␣
                            Niteroi,␣Rio␣de␣Janeiro",
                        "fromUserId" => null,
                        "idTweet" => "553315797247217665",
                        "createdAt" => "Thu␣Jan␣08␣22:22:20␣
                            +0000␣2015",
                        "time" => "1420755740",
                        'toUserId' => null,
                        'fromUser' => "mpl_rj",
                        'isoLanguageCode' => null,
                        'source' => null,
                        'profileImageUrl' => null,
                        'geoType' => null,
                        'geoCoordinates0' => null,
                        'geoCoordinates1' => null
                        )
                    )
);

$this->getMongoConnection();
$this->getMongoDataSet();
}

public function tearDown(){
    $this->dataset->dropAllCollections();
}
```

```
public function testInsertAll(){
 $tweetAsText ='[{"text":"FooText","toUserId":null,"fromUser
    ":"mpl_sp","idTweet":"553315797247217664","fromUserId":
    null,"isoLanguageCode":null,"source":null,"
    profileImageUrl":null,"geoType":null,"geoCoordinates0":
    null,"geoCoordinates1":null,"createdAt":"Thu␣Jan␣08␣
    22:22:20␣+0000␣2015","time":"1420755740"},{"text":"Luta␣
    contra␣a␣tarifa␣em␣Maua,␣Grande␣Sao␣Paulo","toUserId":
    null,"fromUser":"mpl_sp","idTweet":"553315797247217664","
    fromUserId":null,"isoLanguageCode":null,"source":null,"
    profileImageUrl":null,"geoType":null,"geoCoordinates0":
    null,"geoCoordinates1":null,"createdAt":"Thu␣Jan␣08␣
    22:22:20␣+0000␣2015","time":"1420755740"}]';



 $parser = new JsonTweetParser();
 $this->facade->insertAll($tweetAsText, $parser);

 $this->assertEquals(1, $this->connection->collection("tweets
    ")->count(["text" => "FooText"]));
 $this->assertEquals(1, $this->connection->collection("tweets
    ")->count(["text" => "Luta␣contra␣a␣tarifa␣em␣Maua,␣
    Grande␣Sao␣Paulo"]));
 $this->assertEquals(\count($this->tweetsRepo->listAll()),
    $this->connection->collection("tweets")->count());
}

public function testListAll(){
    $tweets = $this->tweetsRepo->listAll();

    $formatter = new JsonTweetFormatter();

    $this->assertEquals($formatter->format($tweets), $this->
       facade->listAll($formatter));
}

public function testFindById(){
    $tweet = $this->tweetsRepo->findById("54202
```

```
        c79d1c82dc01a000032");

    $formatter = new JsonTweetFormatter();

    $this->assertEquals($formatter->format($tweet), $this->
        facade->findById("54202c79d1c82dc01a000032",$formatter
        ));
}

public function testFindInAnInterval(){
    $this->markTestSkipped();
}

public function testFindByCategory(){
    $this->markTestSkipped();
}


}


}

<?php

namespace sna\tests\model\entity\mappers{

use PHPUnit_Framework_TestCase;
use comunic\social_network_analyzer\model\util\
    IdMongoGenerator;

class IdMongoGeneratorTest extends PHPUnit_Framework_TestCase
    {


    public function testeInvoke(){

        $idMongo=new IdMongoGenerator();
```

```
        $generateId = $idMongo();

        $mongoId = new \MongoId($generateId);

        $this->assertEquals($generateId, $mongoId->{'$id'});

    }

}

}
```

```
<?php
use comunic\social_network_analyzer\model\entity\parse\json\
   BasicObjectParser;
use comunic\social_network_analyzer\model\entity\format\json\
   BasicObjectFormatter;
use comunic\social_network_analyzer\model\entity\mappers\
   ArrayToCategory;
use comunic\social_network_analyzer\model\entity\mappers\
   CategoryToArray;
use comunic\social_network_analyzer\model\exception\
   ValidationException;
use comunic\social_network_analyzer\model\exception\
   UniqueConstraintException;

$categoryFacade = $factory->instantiateCategories();


$restapp->get('/categories/json/:idProject/:id', function(
   $idProject,$id) use ($categoryFacade) {

    echo $categoryFacade->findById($id, new
       BasicObjectFormatter(new CategoryToArray()));
});

$restapp->post('/categories/json/:projectId', function(
   $projectId) use ($categoryFacade, $restapp) {
```

```
        try {
                $token = \explode("␣", $restapp->request->
                   headers->get('Authorization'))[1];
                echo $categoryFacade->insert($restapp->
                   request()->getBody(), new
                   BasicObjectParser(new ArrayToCategory()),
                   $token, $projectId);
        } catch (ValidationException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        } catch (UniqueConstraintException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        }

});

$restapp->put('/categories/json/:idProject', function(
   $idProject) use ($categoryFacade, $restapp) {

        try {
                $token = \explode("␣", $restapp->request->
                   headers->get('Authorization'))[1];
                echo $categoryFacade->update($restapp->
                   request()->getBody(), new
                   BasicObjectParser(new ArrayToCategory()),
                   $token,$idProject);
        } catch (ValidationException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        } catch (UniqueConstraintException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        }
```

```
});

$restapp->delete('/categories/json/:idProject/:id', function(
   $idProject, $id) use ($categoryFacade) {

    echo $categoryFacade->delete($id,$idProject);

});

<?php
use comunic\social_network_analyzer\model\entity\parse\json\
   BasicObjectParser;
use comunic\social_network_analyzer\model\entity\format\json\
   BasicObjectFormatter;
use comunic\social_network_analyzer\model\exception\
   ValidationException;
use comunic\social_network_analyzer\model\exception\
   UniqueConstraintException;
use \comunic\social_network_analyzer\model\entity\mappers\
   MatrixToArray;
use \comunic\social_network_analyzer\model\entity\mappers\
   ArrayToMatrix;

$matrixFacade = $factory->instantiateMatrices();

$restapp->get('/matrices/json/:idProject/:id', function(
   $idProject,$id) use ($restapp,$matrixFacade) {

    $populated = $restapp->request()->params()["populated"];

    if(!$populated){
        echo $matrixFacade->findById($id, new
           BasicObjectFormatter(new MatrixToArray()));
    }else{
        echo $matrixFacade->populateMatrix($id, new
           BasicObjectFormatter(new MatrixToArray()));
    }
```

```
});

$restapp->post('/matrices/json/:projectId', function(
   $projectId) use ($matrixFacade, $restapp) {

    try {
        $token = \explode("␣", $restapp->request->headers->
           get('Authorization'))[1];
        echo $matrixFacade->insert($restapp->request()->
           getBody(), new BasicObjectParser(new ArrayToMatrix
           ()),$token, $projectId);
    } catch (ValidationException $e) {
        $restapp->response->status(400);
        $restapp->response->write(\json_encode(["error" => $e
           ->getMessage()]));
    } catch (UniqueConstraintException $e) {
        $restapp->response->status(400);
        $restapp->response->write(\json_encode(["error" => $e
           ->getMessage()]));
    }



});

$restapp->put('/matrices/json/:idProject', function(
   $idProject) use ($matrixFacade, $restapp) {

    try {
        $token = \explode("␣", $restapp->request->headers->
           get('Authorization'))[1];
        echo $matrixFacade->update($restapp->request()->
           getBody(), new BasicObjectParser(new ArrayToMatrix
           ()),$token,$idProject);
    } catch (ValidationException $e) {
        $restapp->response->status(400);
        $restapp->response->write(\json_encode(["error" => $e
           ->getMessage()]));
```

```
    } catch (UniqueConstraintException $e) {
        $restapp->response->status(400);
        $restapp->response->write(\json_encode(["error" => $e
           ->getMessage()]));
    }


});

$restapp->delete('/matrices/json/:idProject/:id', function(
   $idProject, $id) use ($matrixFacade) {

    echo $matrixFacade->delete($id,$idProject);

});

<?php
use comunic\social_network_analyzer\model\entity\parse\json\
   BasicObjectParser;
use comunic\social_network_analyzer\model\entity\format\json\
   BasicObjectFormatter;
use comunic\social_network_analyzer\model\entity\mappers\
   ArrayToProject;
use comunic\social_network_analyzer\model\entity\mappers\
   ProjectToArray;
use comunic\social_network_analyzer\model\exception\
   ValidationException;
use comunic\social_network_analyzer\model\exception\
   UniqueConstraintException;


$projectFacade = $factory->instantiateProjects();

$restapp->get('/projects/json/:id', function($id) use (
   $projectFacade) {
        echo $projectFacade->findById($id, new
           BasicObjectFormatter(new ProjectToArray()));
});
```

```
$restapp->get('/projects/json', function() use (
   $projectFacade, $restapp) {
        echo $projectFacade->listAll(new BasicObjectFormatter
           (new ProjectToArray()));
});

$restapp->post('/projects/json', function() use (
   $projectFacade, $restapp) {

        try {
                $token = \explode("␣", $restapp->request->
                   headers->get('Authorization'))[1];
                echo $projectFacade->insert($restapp->request
                   ()->getBody(), new BasicObjectParser(new
                   ArrayToProject()),$token);
        } catch (ValidationException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        } catch (UniqueConstraintException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        }

});

$restapp->put('/projects/json', function() use (
   $projectFacade, $restapp) {

        try {
                $token = \explode("␣", $restapp->request->
                   headers->get('Authorization'))[1];
                echo $projectFacade->update($restapp->request
                   ()->getBody(), new BasicObjectParser(new
                   ArrayToProject()),$token);
        } catch (ValidationException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
```

```
                error" => $e->getMessage()]));
        } catch (UniqueConstraintException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                error" => $e->getMessage()]));
        }

});

$restapp->delete('/projects/json/:id', function($id) use (
   $projectFacade) {
        echo $projectFacade->delete($id);
});


<?php
use comunic\social_network_analyzer\model\entity\format\json\
   BasicObjectFormatter;
use comunic\social_network_analyzer\model\entity\mappers\
   UserToArray;


$adminFacade = $factory->instantiateAdmin();

$restapp->get('/users/json', function() use ($adminFacade) {
    echo $adminFacade->listUsers(new BasicObjectFormatter(new
        UserToArray));
});


$restapp->post('/users/json/enable/:id', function($id) use (
   $adminFacade, $restapp) {

        $enabled = $restapp->request()->params()["enabled"];

    echo $adminFacade->enableUser($id, $enabled);
});

$restapp->post('/users/json/set_role/:id', function($id) use
```

```
    ($adminFacade, $restapp) {

        $role = $restapp->request()->params()["role"];

    echo $adminFacade->setRole($id, $role);
});
```

```
<?php
use comunic\social_network_analyzer\model\entity\parse\csv\
    CSVTweetParser;
use comunic\social_network_analyzer\model\entity\parse\json\
    JsonTweetParser;
use comunic\social_network_analyzer\model\entity\format\json\
    JsonTweetFormatter;
use comunic\social_network_analyzer\model\entity\format\json\
    JsonPaginator;
use comunic\social_network_analyzer\model\entity\mappers\
    TweetToArray;
use comunic\social_network_analyzer\model\entity\mappers\
    TweetToArrayCompact;
use comunic\social_network_analyzer\model\entity\mappers\
    ArrayToTweet;
use comunic\social_network_analyzer\model\entity\parse\json\
    BasicObjectParser;
use comunic\social_network_analyzer\model\entity\format\json\
    BasicObjectFormatter as FormatterJSON;
use comunic\social_network_analyzer\model\entity\format\json\
    BasicFormatter;
use comunic\social_network_analyzer\model\entity\mappers\
    TweetThreadToArray;
use comunic\social_network_analyzer\model\entity\format\csv\
    BasicObjectFormatter as FormatterCSV;
use \comunic\social_network_analyzer\model\entity\parse\csv\
    CSVTweetParser2;
```

```
$tweetsFacade = $factory->instantiateTweets();

$restapp->post('/tweets/csv_to_json/:idDataset', function(
   $idDataset) use($restapp, $tweetsFacade) {
    $body = \json_decode($restapp->request()->getBody(), TRUE
       );
//    echo $tweetsFacade->insert($body["data"], new
   CSVTweetParser(new ArrayToTweet(),"|"), $idDataset);

    $parser = new CSVTweetParser2($body["delimiter"], $body["
       enclosure"]);
    $tweetsFacade->insert($body["data"], $parser, $idDataset,
        $body["idName"], $body["textName"], $body["
       fromUserName"], $body["createdAtName"] );
});

$restapp->get('/tweet/json/:idTweet', function($idTweet) use(
   $tweetsFacade) {

    echo $tweetsFacade->findById($idTweet, new FormatterJSON(
       new TweetToArray()));

});

$restapp->put('/tweet/json', function() use($tweetsFacade,
   $restapp) {

    echo $tweetsFacade->update($restapp->request()->getBody()
       , new BasicObjectParser(new ArrayToTweet()));

});

$restapp->get('/tweet/json/get_conversation/:idTweet',
   function($idTweet) use($tweetsFacade) {

    echo $tweetsFacade->getConversation($idTweet, new
       FormatterJSON(new TweetThreadToArray()));

});
```

```
$restapp->get('/tweet/json/get_more_conversation/:idRootTweet
   /:idLastTweet', function($idRootTweet,$idLastTweet) use(
   $tweetsFacade) {

    echo $tweetsFacade->getMoreConversation($idRootTweet,
       $idLastTweet, new FormatterJSON(new TweetThreadToArray
       ()));

});

$restapp->post('/tweet/json/set_visibility/:idTweet',
   function($idTweet) use($tweetsFacade, $restapp) {

    $visible = $restapp->request->params()["visible"];

    $visible = \filter_var($visible, FILTER_VALIDATE_BOOLEAN)
       ;
    echo $tweetsFacade->setVisibility($idTweet, $visible);

});

$restapp->post('/tweet/json/add_tag/:idTweet/:idCategory',
   function($idTweet,$idCategory) use($tweetsFacade,$restapp)
    {

    $token = \explode("␣", $restapp->request->headers->get('
       Authorization'))[1];
    echo $tweetsFacade->addTag($idTweet, $idCategory,$token);


});

$restapp->post('/tweet/json/remove_tag/:idTweet/:idCategory',
    function($idTweet,$idCategory) use($tweetsFacade,$restapp
   ) {

    $token = \explode("␣", $restapp->request->headers->get('
       Authorization'))[1];
```

```
    echo $tweetsFacade->removeTag($idTweet, $idCategory,
       $token);


});

$restapp->post('/tweet/json/add_class/:idTweet/:idCategory',
   function($idTweet,$idCategory) use($tweetsFacade,$restapp)
    {

    $token = \explode("␣", $restapp->request->headers->get('
       Authorization'))[1];
    echo $tweetsFacade->addClass($idTweet, $idCategory,$token
       );


});

$restapp->post('/tweet/json/remove_class/:idTweet/:idCategory
   ', function($idTweet,$idCategory) use($tweetsFacade,
   $restapp) {

    $token = \explode("␣", $restapp->request->headers->get('
       Authorization'))[1];
    echo $tweetsFacade->removeClass($idTweet, $idCategory,
       $token);


});

$restapp->get('/tweet/json/count_classes_in_dataset/:
   idProject/:idDataset',function($idProject,$idDataset) use
   ($tweetsFacade){
    echo $tweetsFacade->countClassesInDataset($idProject,
       $idDataset);
});

$restapp->get('/tweets/json/:idDataset', function($idDataset)
    use($restapp, $tweetsFacade) {
```

```
$params = $restapp->request()->params();

$idsDatasets = \explode(",",$idDataset);

$includeRepeated = isset($params["includeRepeated"]) ? \
   filter_var($params["includeRepeated"],
   FILTER_VALIDATE_BOOLEAN) : true;


if(isset($params['skip']) and isset($params['amount'])){

    $options = array(
        'skip' => intval($params['skip']),
        'amount' => intval($params['amount']),
        'sortBy' => isset($params['sortBy']) ? $params['
           sortBy'] : 'unixTimestamp',
        'direction' => isset($params['direction']) ?
           $params['direction'] : 1
    );
    switch ($params['filter']) {
        case 'byCategory':

            echo $tweetsFacade->findByCategory(
               $idsDatasets, $params['idCategory'],new
               JsonPaginator(new TweetToArray()),
               $options);
            break;

        case 'default':
            echo $tweetsFacade->listByDataset(
               $idsDatasets,$includeRepeated,new
               JsonPaginator(new TweetToArray()),
               $options);
            break;

        case 'searchByTerm':
            echo $tweetsFacade->searchInTheText(
               $idsDatasets,$params['searchBy'],new
```

```
            JsonPaginator(new TweetToArray()),$options
            );
        break;

    case 'extractFragment':
        echo $tweetsFacade->getDatasetSlice(
            $idsDatasets,$params['cutPoint'],$params['
            fragmentSize'],new JsonPaginator(new
            TweetToArray()), $options);
        break;

    case 'byCategoryAndUser':
        echo $tweetsFacade->findByCategoryAndUser(
            $idsDatasets,$params['idCategory'],$params
            ['idUser'],new JsonPaginator(new
            TweetToArray()),$options);
        break;

    case 'byCategoryAndQttTags':
        echo $tweetsFacade->findByCategoryAndQttTags(
            $idsDatasets,$params['idCategory'],$params
            ['qtt'],new JsonPaginator(new TweetToArray
            ()),$options);
        break;

    case 'byDistinctCategories':
        echo $tweetsFacade->
            findByQttDistinctCategories($idsDatasets,
            $params['qtt'],new JsonPaginator(new
            TweetToArray()),$options);
        break;
    case 'byRelation':
        echo $tweetsFacade->
            findByRelationBetweenCategories(
            $idsDatasets,$params['rowId'],$params['
            columnId'],$params['relation'],new
            JsonPaginator(new TweetToArray()),$options
            );
        break;
```

```
            default:
                # code...
                break;
        }

    }


});

$restapp->get('/tweets/csv/:idDataset', function($idDataset)
   use($restapp, $tweetsFacade) {

    $params = $restapp->request()->params();

    $idsDatasets = \explode(",",$idDataset);

    $includeRepeated = isset($params["includeRepeated"]) ?
       $params["includeRepeated"] : true;

    $options = array(
        'sortBy' => isset($params['sortBy']) ? $params['
           sortBy'] : 'unixTimestamp',
        'direction' => isset($params['direction']) ? $params[
           'direction'] : 1
    );


//    switch ($params['filter']) {
//        case 'byCategory':
//            $jsonResponse = ['values' => $tweetsFacade->
   findByCategory($idsDatasets, $params['idCategory'],new
   FormatterCSV(new TweetToArray()),$options)];
//            echo \json_encode($jsonResponse);
//            break;
//
//        case 'default':
//            $jsonResponse = ['values' => $tweetsFacade->
```

```
   listByDataset($idsDatasets,$includeRepeated,new
   FormatterCSV(new TweetToArray()), $options)];
//             echo \json_encode($jsonResponse);
//             break;
//
//         case 'search':
//             $jsonResponse = ['values' => $tweetsFacade->
   searchInTheText($params['searchBy'],new FormatterCSV(new
   TweetToArray()),$options)];
//             echo \json_encode($jsonResponse);
//
//             break;
//
//     }

    switch ($params['filter']) {
        case 'byCategory':

            echo $tweetsFacade->findByCategory($idsDatasets,
               $params['idCategory'],new FormatterCSV(new
               TweetToArrayCompact()), $options);
            break;

        case 'default':
            echo $tweetsFacade->listByDataset($idsDatasets,
               $includeRepeated,new FormatterCSV(new
               TweetToArrayCompact()), $options);
            break;

        case 'searchByTerm':
            echo $tweetsFacade->searchInTheText($idsDatasets,
               $params['searchBy'],new FormatterCSV(new
               TweetToArrayCompact()),$options);
            break;

        case 'extractFragment':
            echo $tweetsFacade->listByDataset($idsDatasets,
               $includeRepeated,new FormatterCSV(new
               TweetToArrayCompact()), $options);
```

```
            break;

        case 'byCategoryAndUser':
            echo $tweetsFacade->findByCategoryAndUser(
               $idsDatasets,$params['idCategory'],$params['
               idUser'],new FormatterCSV(new
               TweetToArrayCompact()),$options);
            break;

        case 'byCategoryAndQttTags':
            echo $tweetsFacade->findByCategoryAndQttTags(
               $idsDatasets,$params['idCategory'],$params['
               qtt'],new FormatterCSV(new TweetToArrayCompact
               ()),$options);
            break;

        case 'byDistinctCategories':
            echo $tweetsFacade->findByQttDistinctCategories(
               $idsDatasets,$params['qtt'],new FormatterCSV(
               new TweetToArrayCompact()),$options);
            break;
        case 'byRelation':
            echo $tweetsFacade->
               findByRelationBetweenCategories($idsDatasets,
               $params['rowId'],$params['columnId'],$params['
               relation'],new FormatterCSV(new
               TweetToArrayCompact()),$options);
            break;

        default:
            # code...
            break;
    }

});


<?php
use comunic\social_network_analyzer\model\entity\parse\json\
```

```
   BasicObjectParser;
use comunic\social_network_analyzer\model\entity\format\json\
   BasicObjectFormatter;
use comunic\social_network_analyzer\model\entity\mappers\
   ArrayToDataset;
use comunic\social_network_analyzer\model\entity\mappers\
   DatasetToArray;
use comunic\social_network_analyzer\model\exception\
   ValidationException;
use comunic\social_network_analyzer\model\exception\
   UniqueConstraintException;

$datasetFacade = $factory->instantiateDatasets();

$restapp->get('/datasets/json/:idProject/:id', function(
   $idProject ,$id) use ($datasetFacade) {
        echo $datasetFacade->findById($id, new
           BasicObjectFormatter(new DatasetToArray()));
});

$restapp->get('/datasets/json', function() use (
   $datasetFacade) {
        echo $datasetFacade->listAll(new BasicObjectFormatter
           (new DatasetToArray()));
});

$restapp->post('/datasets/json/:idProject', function(
   $idProject) use ($datasetFacade, $restapp) {

        try {
                $token = \explode("␣", $restapp->request->
                   headers->get('Authorization'))[1];
                echo $datasetFacade->insert($restapp->request
                   ()->getBody(), new BasicObjectParser(new
                   ArrayToDataset()), $token,$idProject);
        } catch (ValidationException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
```

```
        } catch (UniqueConstraintException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        }

});

$restapp->put('/datasets/json/:idProject', function(
   $idProject) use ($datasetFacade, $restapp) {

        try {
                $token = \explode("␣", $restapp->request->
                   headers->get('Authorization'))[1];
                echo $datasetFacade->update($restapp->request
                   ()->getBody(), new BasicObjectParser(new
                   ArrayToDataset()), $token,$idProject);
        } catch (ValidationException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        } catch (UniqueConstraintException $e) {
                $restapp->response->status(400);
                $restapp->response->write(\json_encode(["
                   error" => $e->getMessage()]));
        }

});

$restapp->delete('/datasets/json/:idProject/:id', function(
   $idProject, $id) use ($datasetFacade) {
        echo $datasetFacade->delete($id, $idProject);
});

<?php
use comunic\social_network_analyzer\model\entity\parse\json\
   BasicObjectParser;
use comunic\social_network_analyzer\model\entity\format\json\
   BasicObjectFormatter;
```

```
use comunic\social_network_analyzer\model\entity\mappers\
   ArrayToUser;
use comunic\social_network_analyzer\model\entity\mappers\
   UserToArray;
use comunic\social_network_analyzer\model\exception\
   UniqueConstraintException;


$userFacade = $factory->instantiateUsers();

$restapp->get('/users/json/:id', function($id) use (
   $userFacade) {
    echo $userFacade->findById($id, new BasicObjectFormatter(
       new UserToArray));
});

$restapp->post('/auth/google', function() use ($userFacade,
   $restapp) {


    $client = new GuzzleHttp\Client();

    $body = \json_decode($restapp->request->getBody(), true);


    $params = [
    'code' => $body["code"],
    'client_id' => $body["clientId"],
    'client_secret' => "jls52-wr8A_WJAy4eYipbsym",
    'redirect_uri' => $body['redirectUri'],
    'grant_type' => 'authorization_code'
    ];
            // Step 1. Exchange authorization code for access
                token.
    $accessTokenResponse = $client->request('POST', 'https://
       accounts.google.com/o/oauth2/token', [
        'form_params' => $params
        ]);
    $accessToken = json_decode($accessTokenResponse->getBody
```

```
        (), true);
            // Step 2. Retrieve profile information about the
                current user.
    $profileResponse = $client->request('GET', 'https://www.
        googleapis.com/plus/v1/people/me/openIdConnect', [
         'headers' => array('Authorization' => 'Bearer ' .
            $accessToken['access_token'])
        ]);
    $profile = $profileResponse->getBody();
            // Step 3a. If user is already signed in then
               link accounts.

    $email = json_decode($profileResponse->getBody(), true)["
        email"];

    if($userFacade->existUser($email)){

        echo \json_encode(['token' => $userFacade->
            updateAndGetToken($email)]);

    }else{
        $token = $userFacade->signup($profile, new
            BasicObjectParser(new ArrayToUser()));

        echo \json_encode(['token' => $token]);
    }

});


$restapp->post('/users/json/signup', function() use (
    $userFacade, $restapp) {

    try{
        echo $userFacade->signup($restapp->request()->getBody
            (), new BasicObjectParser(new ArrayToUser()));
    } catch (UniqueConstraintException $e) {
        $restapp->response->status(400);
        $restapp->response->write(\json_encode(["error" => $e
```

```
            ->getMessage()]));
    }


});

$restapp->put('/users/json/change_password/:id', function($id
   ) use ($userFacade, $restapp) {

    $passwd = \json_decode($restapp->request()->getBody(),
       true)["password"];

    $userFacade->changePassword($id,$passwd);

});

$restapp->put('/users/json', function() use ($userFacade,
   $restapp) {
    echo $userFacade->update($restapp->request()->getBody(),
       new BasicObjectParser(new ArrayToUser()));
});


$restapp->post('/users/json/login', function() use (
   $userFacade,$restapp) {

        $data = \json_decode($restapp->request()->getBody(),
           TRUE);


    $result = $userFacade->login($data['email'], $data['
       password']);

    if($result["success"]){
        $restapp->response->status(200);
    }else{
        $restapp->response->status(400);
    }
```

```
    $restapp->response->write(\json_encode($result));


});


<?php

namespace comunic\social_network_analyzer\model\entity{

        class Word{

                private $id;
                private $word;

                function __construct($id=null, $word=null){

                        $this->id = strval($id);
                        $this->word = $word;
                }

                public function getId()
                {
                    return $this->id;
                }

                public function setId($id)
                {
                    return $this->id = $id;
                }

                public function getWord()
                {
                    return $this->word;
                }

                public function setWord($word)
                {
                    return $this->word = $word;
```

```
                }


                public function __toString()
                    {
                        return $this->word;
                   }



        }


}

<?php

namespace comunic\social_network_analyzer\model\entity{

        class Project{


                private $id;
                private $name;
                private $description;
                private $datasets;
                private $categories;
        private $matrices;
                private $createdBy;
                private $updatedBy;


                public function getId(){
                        return $this->id;
                }

                public function setId($id){
                        return $this->id = $id;
                }
```

```
public function getName(){
        return $this->name;
}

public function setName($name){
        return $this->name = $name;
}

public function getDescription()
{
    return $this->description;
}

public function setDescription($description)
{
    return $this->description = $description;
}

public function getDatasets()
{
    return $this->datasets;
}

public function setDatasets($datasets)
{
    return $this->datasets = $datasets;
}

public function getCategories()
{
    return $this->categories;
}

public function setCategories($categories)
{
    return $this->categories = $categories;

}
```

```
/**
 * @return mixed
 */
public function getMatrices()
{
    return $this->matrices;
}

/**
 * @param mixed $matrices
 */
public function setMatrices($matrices)
{
    $this->matrices = $matrices;
}
```

```
        /**
         * @return mixed
         */
        public function getCreatedBy()
        {
                return $this->createdBy;
        }

        /**
         * @param mixed $createdBy
         */
        public function setCreatedBy($createdBy)
        {
                $this->createdBy = $createdBy;
        }

        /**
         * @return mixed
         */
        public function getUpdatedBy()
```

```
                {
                        return $this->updatedBy;
                }

                /**
                 * @param mixed $updatedBy
                 */
                public function setUpdatedBy($updatedBy)
                {
                        $this->updatedBy = $updatedBy;
                }

        }

}



<?php

namespace comunic\social_network_analyzer\model\entity\
   validator{

        class CategoryValidator extends Validator{

                public function verifyErrors($category){


                        if(empty($category->getName())){

                                $this->setError("The␣name␣of␣
                                   category␣can't␣be␣empty");
                        }

                        if(strlen($category->getName()) <  3)
                           {

                                $this->setError("The␣name␣of␣
                                   category␣must␣have␣at␣
```

```
                                        least␣three␣digits");
                        }

                        if(empty($category->getKeywords())){

                                $this->setError("The␣category
                                    ␣must␣have␣at␣least␣one␣
                                    keyword");
                        }

                }


        }

}



<?php

namespace comunic\social_network_analyzer\model\entity\
   validator{

        abstract class Validator{

                protected $errors = [];

                public function validate($object)
                {
                        $this->verifyErrors($object);
                        return $this->isValid();
                }

                protected function setError($error){
                        $this->errors[] = $error;
                }

                public function printErrors(){
```

```
                        $err = "";

                        if(!empty($this->errors)){
                                foreach ($this->errors as
                                   $error) {
                                        $err .= $error . "\n"
                                           ;
                                }
                        }
                        return \trim($err);
                }

                protected abstract function verifyErrors(
                   $object);


                protected function isValid(){
                        return \count($this->errors) == 0;
                }

        }

}



<?php

namespace comunic\social_network_analyzer\model\entity\
   validator{

        class DatasetValidator extends Validator{

                public function verifyErrors($dataset){

                        if(empty($dataset->getName())){

                                $this->setError("The␣name␣of␣
```

```
                                dataset␣can't␣be␣empty");
                        }

                        if(strlen($dataset->getName()) <  3){

                                $this->setError("The␣name␣of␣
                                   dataset␣must␣have␣at␣least
                                   ␣three␣digits");
                        }

                }

        }

}



<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 15:55
 */

namespace comunic\social_network_analyzer\model\entity\
   validator;


class MatrixValidator extends Validator
{

    public function verifyErrors($matrix){

        if(empty($matrix->getName())){

            $this->setError("The␣name␣of␣category␣can't␣be␣
```

```
                empty");
        }


    }

}<?php

namespace comunic\social_network_analyzer\model\entity\
   validator{

        class ProjectValidator extends Validator{

                public function verifyErrors($project){

                        if(empty($project->getName())){

                                $this->setError("The␣name␣of␣
                                   project␣can't␣be␣empty");
                        }

                        if(strlen($project->getName()) <  3){

                                $this->setError("The␣name␣of␣
                                   project␣must␣have␣at␣least
                                   ␣three␣digits");
                        }


                }


        }

}



<?php
```

```
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 20/12/16
 * Time: 14:55
 */

namespace comunic\social_network_analyzer\model\entity\
   validator;

use InvalidArgumentException;

class UserValidator
{

    public function validate($params){

        if(!isset($params['name']) || \trim($params['name'])
           == ''){

            throw new InvalidArgumentException("O␣campo␣nome␣
               n o␣pode␣ser␣vazio");

        }

        if(!isset($params['email']) || \trim($params['email'
           ]) == ''){

            throw new InvalidArgumentException("O␣campo␣e-
               mail␣n o␣pode␣ser␣vazio");

        }

        if (!\filter_var($params['email'],
           FILTER_VALIDATE_EMAIL)) {

            throw new InvalidArgumentException("O␣campo␣e-
               mail␣deve␣conter␣um␣endere o␣v lido");
        }
```

```
        if(!isset($params['password']) || \trim($params['
           password']) == ''){

            throw new InvalidArgumentException("O␣campo␣senha
               ␣  ␣obrigat rio");

        }

        return true;
    }

}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 28/09/16
 * Time: 00:52
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;


class ObjectToArrayCompact
{
    private $attributes;

    /**
     * @return mixed
     */
    public function getAttributes()
    {
        return $this->attributes;
    }

    /**
     * @param mixed $attributes
     */
```

```
public function setAttributes($attributes)
{
    $this->attributes = $attributes;
}

public function __invoke($obj){

    $arr = [];

    foreach ($this->getAttributes() as $attr){

        $attrName = $attr["name"];
        $getMethod = "get" . \ucfirst($attrName);



        if(!isset($attr["mapper"])){

            if(\is_array($obj->$getMethod())){
                $arr[$attrName] =  \json_encode($obj->
                   $getMethod());
            }else{
                $arr[$attrName] = $obj->$getMethod();
            }




        }
        else{

            if(!empty($obj->$getMethod())){


                if(isset($attr["isArray"]) && $attr["
                   isArray"]){
```

```
                        $output = array_map(function ($object
                            ) { return \strval($object); },
                            $obj->$getMethod());

                        $arr[$attrName] = \implode(",␣",
                            $output);


                    }else{

                        $arr[$attrName] = \strval($obj->
                            $getMethod());

                    }

                }else{
                    $arr[$attrName] = "";
                }



            }

        }

        return $arr;
    }
}<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{

        use comunic\social_network_analyzer\model\entity\Tag;


        class ArrayToTag extends ArrayToObject{

        public function __construct()
        {
```

```
            $this->setEntityName("Tag");
            $attrs = [

                ["name" => "id"],
                ["name" => "category",
                "mapper" => "ArrayToCategory",
                "isArray" => false],
                ["name" => "addedBy",
                "mapper" => "ArrayToModification",
                "isArray" => true],

            ];

            $this->setAttributes($attrs);

        }

//              function __invoke($arrayData){
//
//                      $tag = new Tag();
//
//                      if(isset($arrayData['id'])){
//                              $tag->setId($arrayData['id'])
   ;
//                      }
//
//                      if(isset($arrayData['category'])){
//
//                              $toCategory = new
   ArrayToCategory();
//
//                              $tag->setCategory($toCategory
   ($arrayData['category']));
//                      }
//
//                      if(isset($arrayData['addedBy'])){
//
//                              $modifications = array();
```

```
//
//                                  foreach ($arrayData['addedBy
   '] as $addUser) {
//
//                                          $toModification = new
    ArrayToModification();
//                                          $modifications[] =
   $toModification($addUser);
//                                  }
//
//
//
//                                  $tag->setAddedBy(
   $modifications);
//                          }
//
//                          return $tag;
//
//                  }

        }

}

<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{

        class UserToArray extends ObjectToArray{

                public function __construct()
                {
                        $attrs = [

                                ["name" => "id"],
                                ["name" => "name"],
                                ["name" => "email"],
                                ["name" => "role"],
```

```
                                ["name" => "password"],
                                ["name" => "enabled"],

                            ];

                            $this->setAttributes($attrs);

                    }

//                  public function __invoke($obj){
//
//                          return \array_filter(array(
//                                  'id' => $obj->getId(),
//                                  'name' => $obj->getName(),
//                                  'email' => $obj->getEmail(),
//                                  'role' => $obj->getRole(),
//                                  'password' => $obj->
    getPassword(),
//                                  'enabled' => $obj->getEnabled
    ()
//                    ));
//                  }
        }
}

<?php

namespace comunic\social_network_analyzer\model\entity\
    mappers{

    use comunic\social_network_analyzer\model\entity\Tweet;

    class ArrayToTweet extends ArrayToObject{

        public function __construct()
        {

            $this->setEntityName("Tweet");
//              $attrs = [
```

```
//
//                  ["name" => "id"],
//                  ["name" => "text"],
//                  ["name" => "idTweet"],
//                  ["name" => "toUserId"],
//                  ["name" => "fromUser"],
//                  ["name" => "fromUserId"],
//                  ["name" => "isoLanguageCode"],
//                  ["name" => "source"],
//                  ["name" => "profileImageUrl"],
//                  ["name" => "geoType"],
//                  ["name" => "geoCoordinates0"],
//                  ["name" => "geoCoordinates1"],
//                  ["name" => "createdAt"],
//                  ["name" => "time"],
//                  ["name" => "idDataset"],
//                  ["name" => "classes",
//                      "mapper" => "ArrayToTag",
//                      "isArray" =>true],
//                  ["name" => "textNormalized"],
//                  ["name" => "isVisible"],
//                  ["name" => "existsSimilarPostedPreviously
   "],
//                  ["name" => "textNormalized"],
//                  ["name" => "cleartext"],
//                  ["name" => "tags",
//                      "mapper" => "ArrayToTag",
//                  "isArray" =>true],
//
//
//              ];

            $attrs = [

                ["name" => "id"],
                ["name" => "text"],
                ["name" => "idTweet"],
                ["name" => "fromUser"],
                ["name" => "unixTimestamp"],
```

```
                ["name" => "idDataset"],
                ["name" => "otherAttributes"],
                ["name" => "classes",
                    "mapper" => "ArrayToTag",
                    "isArray" =>true],
                ["name" => "isVisible"],
                ["name" => "textPreProcessed"],
                ["name" => "normalizedText"],
                ["name" => "tags",
                    "mapper" => "ArrayToTag",
                "isArray" =>true],
            ];

            $this->setAttributes($attrs);

        }

//          public function __invoke($arrayData){
//
//              $tweet = new Tweet();
//
//              if(isset($arrayData['id'])){
//                  $tweet->setId($arrayData['id']);
//              }
//
//              if(isset($arrayData['text'])){
//                  $tweet->setText(\trim($arrayData['text']));
//              }
//
//              if(isset($arrayData['idTweet'])){
//                  $tweet->setIdTweet($arrayData['idTweet']);
//              }
//
//              if(isset($arrayData['toUserId'])){
//                  $tweet->setToUserId($arrayData['toUserId'])
    ;
//              }
//
//              if(isset($arrayData['fromUser'])){
```

```
//                  $tweet->setFromUser($arrayData['fromUser'])
    ;
//              }
//
//              if(isset($arrayData['fromUserId'])){
//                  $tweet->setFromUserId($arrayData['
    fromUserId']);
//              }
//
//              if(isset($arrayData['isoLanguageCode'])){
//                  $tweet->setIsoLanguageCode($arrayData['
    isoLanguageCode']);
//              }
//
//              if(isset($arrayData['source'])){
//                  $tweet->setSource($arrayData['source']);
//              }
//
//              if(isset($arrayData['profileImageUrl'])){
//                  $tweet->setProfileImageUrl($arrayData['
    profileImageUrl']);
//              }
//
//              if(isset($arrayData['geoType'])){
//                  $tweet->setGeoType($arrayData['geoType']);
//              }
//
//              if(isset($arrayData['geoCoordinates0'])){
//                  $tweet->setGeoCoordinates0($arrayData['
    geoCoordinates0']);
//              }
//
//              if(isset($arrayData['geoCoordinates1'])){
//                  $tweet->setGeoCoordinates1($arrayData['
    geoCoordinates1']);
//              }
//
//              if(isset($arrayData['createdAt'])){
//                  $tweet->setCreatedAt($arrayData['createdAt
```

```
    ']);
//              }
//
//
//              if(isset($arrayData['time'])){
//                  $tweet->setTime($arrayData['time']);
//              }
//
//              if(isset($arrayData['idDataset'])){
//                  $tweet->setIdDataset($arrayData['idDataset
    ']);
//              }
//
//              if(isset($arrayData['class'])){
//                  $tweet->setClass($arrayData['class']);
//              }
//
//              if(isset($arrayData['textNormalized'])){
//                  $tweet->setTextNormalized($arrayData['
    textNormalized']);
//              }
//
//              if(isset($arrayData['isVisible'])){
//                  $tweet->setIsVisible($arrayData['isVisible
    ']);
//              }
//
//              if(isset($arrayData['hasSimilar'])){
//                  $tweet->setHasSimilar($arrayData['
    hasSimilar']);
//              }
//
//              if(isset($arrayData['cleartext'])){
//                  $tweet->setCleartext($arrayData['cleartext
    ']);
//              }
//
//              if(isset($arrayData['tags'])){
//
```

```
//                  $tags = array();
//
//                  foreach ($arrayData['tags'] as $tag) {
//
//                      $toTag = new ArrayToTag();
//                      $tags[] = $toTag($tag);
//                  }
//
//
//
//                  $tweet->setTags($tags);
//              }
//
//              return $tweet;
//          }
//
//
    }




}



<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 13:52
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;


class ObjectToArray
```

```
{

    private $attributes;

    /**
     * @return mixed
     */
    public function getAttributes()
    {
        return $this->attributes;
    }

    /**
     * @param mixed $attributes
     */
    public function setAttributes($attributes)
    {
        $this->attributes = $attributes;
    }

    public function __invoke($obj){

        $arr = [];

        foreach ($this->getAttributes() as $attr){

            $attrName = $attr["name"];
            $getMethod = "get" . \ucfirst($attrName);



            if(!isset($attr["mapper"])){

                $arr[$attrName] = $obj->$getMethod();


            }
            else{
```

```
                if(!empty($obj->$getMethod())){

                $mapperName = 'comunic\\
                   social_network_analyzer\\model\\entity\\
                   mappers\\' . $attr["mapper"];

                if(isset($attr["isArray"]) && $attr["isArray"
                   ]){

                    $items = array();

                    foreach ($obj->$getMethod() as $item) {
                        $toArr = new $mapperName();
                        $items[] = $toArr($item);
                    }

                    $arr[$attrName] = $items;

                }else{

                    $toArr = new $mapperName();
                    $arr[$attrName] = $toArr($obj->$getMethod
                       ());

                }

                }else{
                    $arr[$attrName] = [];
                }



            }

        }

        return $arr;
    }
```

```
}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 09:31
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;



class ArrayToMatrix extends ArrayToObject
{
    public function __construct()
    {
        $this->setEntityName("Matrix");

        $attrs = [

            ["name" => "id"],
            ["name" => "name"],
            ["name" => "datasetsIds"],

            ["name" => "rowsIds"],
            ["name" => "columnsIds"],
            ["name" => "relation"],

            ["name" => "elements",
            "mapper" => "ArrayToMatrixElement",
            "isArray" => true],

            ["name" => "createdBy",
                "mapper" => "ArrayToModification",
                "isArray" => false],
            ["name" => "updatedBy",
                "mapper" => "ArrayToModification",
                "isArray" => false]
```

```
        ];

        $this->setAttributes($attrs);
    }


}<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{

        use comunic\social_network_analyzer\model\entity\
           mappers\DatasetToArray;
        use comunic\social_network_analyzer\model\entity\
           mappers\CategoryToArray;

        class ProjectToArray extends ObjectToArray{

                public function __construct()
                {

                        $attrs = [

                                ["name" => "id"],
                                ["name" => "name"],
                                ["name" => "description"],
                                ["name" => "datasets",
                                        "mapper" => "
                                           DatasetToArray",
                                        "isArray" => true],
                                ["name" => "categories",
                                        "mapper" => "
                                           CategoryToArray",
                                        "isArray" => true],
                ["name" => "matrices",
                    "mapper" => "MatrixToArray",
                    "isArray" => true],
                                ["name" => "createdBy",
```

```
                                        "mapper" => "
                                           ModificationToArray
                                           ",
                                        "isArray" => false],
                                ["name" => "updatedBy",
                                        "mapper" => "
                                           ModificationToArray
                                           ",
                                        "isArray" => false]



                        ];

                        $this->setAttributes($attrs);
                }

//              public function __invoke($obj){
//
//                      $datasets = array();
//
//                      if(!\is_null($obj->getDatasets())){
//                              foreach ($obj->getDatasets()
   as $dataset) {
//                                      $toArray = new
   DatasetToArray();
//                                      $datasets[] =
   $toArray($dataset);
//                              }
//                      }
//
//                      $categories = array();
//
//                      if(!\is_null($obj->getCategories())){
//                              foreach ($obj->getCategories
   () as $category) {
//                                      $toArray = new
   CategoryToArray();
//                                      $categories[] =
```

```
   $toArray($category);
//                                  }
//                          }
//
//                          $modToArray = new ModificationToArray
   ();
//
//                          if(!empty($obj->getCreatedBy())){
//
//
//                                  $createdBy = $modToArray($obj
   ->getCreatedBy());
//                          }else{
//                                  $createdBy = null;
//                          }
//
//                          if(!empty($obj->getUpdatedBy())){
//                                  $updatedBy = $modToArray($obj
   ->getUpdatedBy());
//                          }else{
//                                  $updatedBy = null;
//                          }
//
//                          return array(
//                                  'id' => $obj->getId(),
//                                  'name' => $obj->getName(),
//                                  'description' => $obj->
   getDescription(),
//                  'datasets' => $datasets,
//                  'categories' => $categories,
//                                  'createdBy' => $createdBy,
//                                  'updatedBy' => $updatedBy
//
//
//                          );
//
//
//
//                }
```

```
        }


}


<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 09:31
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;


class ArrayToMatrixElement extends ArrayToObject
{

public function __construct()
{
    $this->setEntityName("MatrixElement");

    $attrs = [


        ["name" => "rowId"],
        ["name" => "rowName"],
        ["name" => "columnId"],
        ["name" => "columnName"],
        ["name" => "symbol"]

    ];

    $this->setAttributes($attrs);
```

```
}

}<?php


namespace comunic\social_network_analyzer\model\entity\
   mappers{

    class TweetToArray extends ObjectToArray{


        public function __construct()
        {

//            $attrs = [
//
//                ["name" => "id"],
//                ["name" => "text"],
//                ["name" => "idTweet"],
//                ["name" => "toUserId"],
//                ["name" => "fromUser"],
//                ["name" => "fromUserId"],
//                ["name" => "isoLanguageCode"],
//                ["name" => "source"],
//                ["name" => "profileImageUrl"],
//                ["name" => "geoType"],
//                ["name" => "geoCoordinates0"],
//                ["name" => "geoCoordinates1"],
//                ["name" => "createdAt"],
//                ["name" => "time"],
//                ["name" => "idDataset"],
//                ["name" => "classes",
//                    "mapper" => "TagToArray",
//                    "isArray" =>true],
//                ["name" => "textNormalized"],
//                ["name" => "isVisible"],
//                ["name" => "existsSimilarPostedPreviously
   "],
//                ["name" => "textNormalized"],
```

```
//                  ["name" => "cleartext"],
//                  ["name" => "tags",
//                      "mapper" => "TagToArray",
//                      "isArray" =>true],
//
//
//              ];

            $attrs = [

                ["name" => "id"],
                ["name" => "text"],
                ["name" => "idTweet"],
                ["name" => "fromUser"],
                ["name" => "unixTimestamp"],
                ["name" => "idDataset"],
                ["name" => "otherAttributes"],
                ["name" => "classes",
                    "mapper" => "TagToArray",
                    "isArray" =>true],
                ["name" => "isVisible"],
                ["name" => "textPreProcessed"],
                ["name" => "normalizedText"],
                ["name" => "tags",
                    "mapper" => "TagToArray",
                    "isArray" =>true],
            ];

            $this->setAttributes($attrs);

        }


    }
```

```
}



<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{


use \comunic\social_network_analyzer\model\entity\User;

    class ArrayToUser extends ArrayToObject{

        public function __construct()
        {

            $this->setEntityName("User");
            $attrs = [

                ["name" => "id"],
                ["name" => "name"],
                ["name" => "email"],
                ["name" => "role"],
                ["name" => "password"],
                ["name" => "enabled"],


            ];

            $this->setAttributes($attrs);

        }

//         public function __invoke($arrayData){
//             $user = new User();
//
//             if (isset($arrayData['id'])) {
//                   $user->setId($arrayData['id']);
//             }
```

```
//
//              if (isset($arrayData['name'])) {
//                  $user->setName($arrayData['name']);
//              }
//
//              if (isset($arrayData['email'])) {
//                  $user->setEmail($arrayData['email']);
//              }
//
//              if (isset($arrayData['role'])) {
//                  $user->setRole($arrayData['role']);
//              }
//
//              if (isset($arrayData['password'])) {
//                  $user->setPassword($arrayData['password'])
   ;
//              }
//
//              if (isset($arrayData['enabled'])) {
//                  $user->setEnabled($arrayData['enabled']);
//              }
//
//              return $user;
//
//          }


    }



}



<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{
```

```
        class ArrayToModification extends ArrayToObject{

        public function __construct()
        {

            $this->setEntityName("Modification");
            $attrs = [

                ["name" => "id"],
                ["name" => "idUser"],
                ["name" => "nameUser"],
                ["name" => "time"],

            ];

            $this->setAttributes($attrs);

        }

//              function __invoke($arrayData){
//
//                      $modification = new Modification();
//
//                      if(isset($arrayData['id'])){
//                              $modification->setId(
   $arrayData['id']);
//                      }
//
//                      if(isset($arrayData['idUser'])){
//                              $modification->setIdUser(
   $arrayData['idUser']);
//                      }
//
//                      if(isset($arrayData['nameUser'])){
//                              $modification->setNameUser(
   $arrayData['nameUser']);
//                      }
```

```
//
//                      if(isset($arrayData['time'])){
//                              $modification->setTime(
   $arrayData['time']);
//                      }
//
//                      return $modification;
//
//              }

        }

}

<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 15:43
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;


class MatrixToArray extends ObjectToArray
{

    public function __construct()
    {

        $attrs = [

            ["name" => "id"],
            ["name" => "name"],
            ["name" => "datasetsIds"],

            ["name" => "rowsIds"],
```

```
            ["name" => "columnsIds"],
            ["name" => "relation"],

            ["name" => "elements",
                "mapper" => "MatrixElementToArray",
                "isArray" => true],

            ["name" => "createdBy",
                "mapper" => "ModificationToArray",
                "isArray" => false],
            ["name" => "updatedBy",
                "mapper" => "ModificationToArray",
                "isArray" => false]

        ];

        $this->setAttributes($attrs);
    }

}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 15:44
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;


class MatrixElementToArray extends ObjectToArray
{

    public function __construct()
    {

        $attrs = [
```

```
            ["name" => "rowId"],
            ["name" => "rowName"],
            ["name" => "columnId"],
            ["name" => "columnName"],
            ["name" => "symbol"]

        ];

        $this->setAttributes($attrs);
    }

}<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{

    class ModificationToArray extends ObjectToArray{


        public function __construct()
        {


            $attrs = [

                ["name" => "id"],
                ["name" => "idUser"],
                ["name" => "nameUser"],
                ["name" => "time"],

            ];

            $this->setAttributes($attrs);

        }
//          function __invoke($obj){
//
//              return array(
```

```
//                "id" => $obj->getId(),
//                  "idUser" => $obj->getIdUser(),
//                  "nameUser" => $obj->getNameUser(),
//                  "time" => $obj->getTime()
//                  );
//        }

    }

}


<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{

    class TagToArray extends ObjectToArray{

        public function __construct()
        {


            $attrs = [

                ["name" => "id"],
                ["name" => "category",
                    "mapper" => "CategoryToArray",
                    "isArray" => false],
                ["name" => "addedBy",
                    "mapper" => "ModificationToArray",
                    "isArray" => true],

            ];

            $this->setAttributes($attrs);

        }
```

```
//         function __invoke($obj){
//
//               $modifications = [];
//
//             if(!\is_null($obj->getAddedBy())){
//                 foreach ($obj->getAddedBy() as $addUser) {
//                     $toModificationArr = new
   ModificationToArray();
//                     $modifications[] = $toModificationArr(
   $addUser);
//                 }
//             }
//
//             $category = null;
//
//             if(!\is_null($obj->getCategory())){
//                 $toCategoryArr = new CategoryToArray();
//                 $category = $toCategoryArr($obj->
   getCategory());
//             }
//
//             return array(
//                 "id" => $obj->getId(),
//                 "category" => $category,
//                 "addedBy" => $modifications
//
//                 );
//         }

    }

}


<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{
```

```
    class ArrayToCategory extends ArrayToObject{

        public function __construct()
        {

            $this->setEntityName("Category");
            $attrs = [

                ["name" => "id"],
            ["name" => "name"],
            ["name" => "description"],
            ["name" => "keywords"],
            ["name" => "color"],
            ["name" => "type"],
            ["name" => "createdBy",
                "mapper" => "ArrayToModification",
                "isArray" => false],
            ["name" => "updatedBy",
                "mapper" => "ArrayToModification",
                "isArray" => false]




            ];

            $this->setAttributes($attrs);

        }

//         public function __invoke($arrayData)
//         {
//             return parent::__invoke($arrayData); // TODO:
   Change the autogenerated stub
//         }

//         public function __invoke($arrayData){
//             $category = new Category();
//
```

```
//              if (isset($arrayData['id'])) {
//                  $category->setId($arrayData['id']);
//              }
//
//              if (isset($arrayData['name'])) {
//                  $category->setName($arrayData['name']);
//              }
//
//              if (isset($arrayData['description'])) {
//                  $category->setDescription($arrayData['
    description']);
//              }
//
//              if (isset($arrayData['keywords'])) {
//                  $category->setKeywords($arrayData['keywords
    ']);
//              }
//
//              if (isset($arrayData['color'])) {
//                  $category->setColor($arrayData['color']);
//              }
//
//              if(isset($arrayData['type'])){
//                  $category->setType($arrayData['type']);
//              }
//
//              if(isset($arrayData['createdBy'])){
//                  $toModification = new ArrayToModification()
    ;
//                  $category->setCreatedBy($toModification(
    $arrayData['createdBy']));
//              }
//
//              if(isset($arrayData['updatedBy'])){
//                  $toModification = new ArrayToModification()
    ;
//                  $category->setUpdatedBy($toModification(
    $arrayData['updatedBy']));
//              }
```

```
//
//             return $category;
//
//         }


    }



}
```

```
<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 10/04/16
 * Time: 19:25
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;


class TweetThreadToArray extends ObjectToArray
{


    public function __construct()
    {

        $attrs = [

            ["name" => "id"],
            ["name" => "text"],
            ["name" => "name"],
            ["name" => "username"],
```

```
            ["name" => "date"]



        ];

        $this->setAttributes($attrs);

    }
//
//public function __invoke($obj){
//    return [
//        "id" => $obj->getId(),
//        "name" =>$obj->getName(),
//        "text" =>$obj->getText(),
//        "username" => $obj->getUsername(),
//        "date" => $obj->getDate()
//    ];
//}
}<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{


        class ArrayToDataset extends ArrayToObject{

        public function __construct()
        {

            $this->setEntityName("Dataset");
            $attrs = [

                ["name" => "id"],
                ["name" => "name"],
                ["name" => "description"],
                ["name" => "hasTweets"],
                ["name" => "createdBy",
                    "mapper" => "ArrayToModification",
```

```
                    "isArray" => false],
                ["name" => "updatedBy",
                    "mapper" => "ArrayToModification",
                    "isArray" => false]
            ];

            $this->setAttributes($attrs);

        }

//              function __invoke($arrayData){
//
//                      $dataset = new Dataset();
//
//                      if(isset($arrayData['id'])){
//                              $dataset->setId($arrayData['
   id']);
//                      }
//
//                      if(isset($arrayData['name'])){
//                              $dataset->setName($arrayData
   ['name']);
//                      }
//
//                      if(isset($arrayData['description'])){
//                              $dataset->setDescription(
   $arrayData['description']);
//                      }
//
//                      if(isset($arrayData['hasTweets'])){
//                              $dataset->setHasTweets(
   $arrayData['hasTweets']);
//                      }
//
//                      if(isset($arrayData['createdBy'])){
//                              $toModification = new
   ArrayToModification();
//                              $dataset->setCreatedBy(
   $toModification($arrayData['createdBy']));
```

```
//                      }
//
//                      if(isset($arrayData['updatedBy'])){
//                              $toModification = new
   ArrayToModification();
//                              $dataset->setUpdatedBy(
   $toModification($arrayData['updatedBy']));
//                      }
//
//                      return $dataset;
//
//              }

        }

}

<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{


        class ArrayToProject extends ArrayToObject{

        public function __construct()
        {

            $this->setEntityName("Project");
            $attrs = [

                ["name" => "id"],
            ["name" => "name"],
            ["name" => "description"],
            ["name" => "datasets",
                "mapper" => "ArrayToDataset",
                "isArray" => true],
            ["name" => "categories",
                "mapper" => "ArrayToCategory",
```

```
                "isArray" => true],
            ["name" => "matrices",
                 "mapper" => "ArrayToMatrix",
                 "isArray" => true],
            ["name" => "createdBy",
                "mapper" => "ArrayToModification",
                "isArray" => false],
            ["name" => "updatedBy",
                "mapper" => "ArrayToModification",
                "isArray" => false]



            ];

            $this->setAttributes($attrs);

        }

//              public function __invoke($arrayData){
//
//                      $project = new Project();
//
//                      if(isset($arrayData['id'])){
//                              $project->setId($arrayData['
   id']);
//                      }
//
//                      if(isset($arrayData['name'])){
//                              $project->setName($arrayData
   ['name']);
//                      }
//
//                      if(isset($arrayData['description'])){
//                              $project->setDescription(
   $arrayData['description']);
//                      }
//
//                      if(isset($arrayData['datasets'])){
```

```
//
//                                  $datasets = array();
//
//                                  foreach ($arrayData['datasets
   '] as $dataset) {
//
//                                          $toDataset = new
   ArrayToDataset();
//                                          $datasets[] =
   $toDataset($dataset);
//                                  }
//
//                                  $project->setDatasets(
   $datasets);
//                          }
//
//                          if(isset($arrayData['categories'])){
//
//                                  $categories = array();
//
//                                  foreach ($arrayData['
   categories'] as $category) {
//                                          $toCategory = new
   ArrayToCategory();
//                                          $categories[] =
   $toCategory($category);
//                                  }
//
//                                  $project->setCategories(
   $categories);
//                          }
//
//                          if(isset($arrayData['createdBy'])){
//                                  $toModification = new
   ArrayToModification();
//                                  $project->setCreatedBy(
   $toModification($arrayData['createdBy']));
//                          }
//
```

```
//                      if(isset($arrayData['updatedBy'])){
//                              $toModification = new
   ArrayToModification();
//                              $project->setUpdatedBy(
   $toModification($arrayData['updatedBy']));
//                      }
//
//                      return $project;
//
//              }


        }




}
```

```
<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{

    class DatasetToArray extends ObjectToArray{

        public function __construct()
        {

            $attrs = [

                ["name" => "id"],
                ["name" => "name"],
                ["name" => "description"],
                ["name" => "hasTweets"],
                ["name" => "createdBy",
```

```
                    "mapper" => "ModificationToArray",
                    "isArray" => false],
                ["name" => "updatedBy",
                    "mapper" => "ModificationToArray",
                    "isArray" => false]
            ];

            $this->setAttributes($attrs);

        }

//          function __invoke($obj){
//
//              $modToArray = new ModificationToArray();
//
//              if(!empty($obj->getCreatedBy())){
//
//
//                  $createdBy = $modToArray($obj->getCreatedBy
    ());
//              }else{
//                  $createdBy = null;
//              }
//
//              if(!empty($obj->getUpdatedBy())){
//                  $updatedBy = $modToArray($obj->getUpdatedBy
    ());
//              }else{
//                  $updatedBy = null;
//              }
//
//              return array(
//                  "id" => $obj->getId(),
//                  "name" => $obj->getName(),
//                  'description' => $obj->getDescription(),
//                  "hasTweets" => $obj->getHasTweets(),
//                  'createdBy' => $createdBy,
//                  'updatedBy' => $updatedBy
//
```

```
//             );
//         }

    }

}


<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 28/09/16
 * Time: 01:05
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;


class TweetToArrayCompact extends ObjectToArrayCompact
{

    public function __construct()
    {

//         $attrs = [
//
//             ["name" => "id"],
//             ["name" => "text"],
//             ["name" => "idTweet"],
//             ["name" => "toUserId"],
//             ["name" => "fromUser"],
//             ["name" => "fromUserId"],
//             ["name" => "isoLanguageCode"],
//             ["name" => "source"],
//             ["name" => "profileImageUrl"],
//             ["name" => "geoType"],
//             ["name" => "geoCoordinates0"],
```

```
//              ["name" => "geoCoordinates1"],
//              ["name" => "createdAt"],
//              ["name" => "time"],
//              ["name" => "idDataset"],
//              ["name" => "classes",
//                  "mapper" => "TagToArray",
//                  "isArray" =>true],
//              ["name" => "textNormalized"],
//              ["name" => "isVisible"],
//              ["name" => "existsSimilarPostedPreviously"],
//              ["name" => "textNormalized"],
//              ["name" => "cleartext"],
//              ["name" => "tags",
//                  "mapper" => "TagToArray",
//                  "isArray" =>true],
//
//
//          ];

        $attrs = [

            ["name" => "id"],
            ["name" => "text"],
            ["name" => "idTweet"],
            ["name" => "fromUser"],
            ["name" => "unixTimestamp"],
            ["name" => "idDataset"],
            ["name" => "classes",
                "mapper" => "TagToArray",
                "isArray" =>true],
            ["name" => "isVisible"],
            ["name" => "textPreProcessed"],
            ["name" => "tags",
                "mapper" => "TagToArray",
                "isArray" =>true],
            ["name" => "otherAttributes"]
        ];

        $this->setAttributes($attrs);
```

```
    }

}<?php

namespace comunic\social_network_analyzer\model\entity\
   mappers{



    class CategoryToArray extends ObjectToArray{

        public function __construct()
        {


            $attrs = [

                ["name" => "id"],
                ["name" => "name"],
                ["name" => "description"],
                ["name" => "keywords"],
                ["name" => "color"],
                ["name" => "type"],
                ["name" => "createdBy",
                    "mapper" => "ModificationToArray",
                    "isArray" => false],
                ["name" => "updatedBy",
                    "mapper" => "ModificationToArray",
                    "isArray" => false]



            ];

            $this->setAttributes($attrs);

        }
```

```
//          public function __invoke($obj){
//
//              $modToArray = new ModificationToArray();
//
//              if(!empty($obj->getCreatedBy())){
//
//
//                  $createdBy = $modToArray($obj->getCreatedBy
    ());
//              }else{
//                  $createdBy = null;
//              }
//
//              if(!empty($obj->getUpdatedBy())){
//                  $updatedBy = $modToArray($obj->getUpdatedBy
    ());
//              }else{
//                  $updatedBy = null;
//              }
//
//              return array(
//                  'id' => $obj->getId(),
//                  'name' => $obj->getName(),
//                  'description' => $obj->getDescription(),
//                  'keywords' => $obj->getKeywords(),
//                  'color' => $obj->getColor(),
//                  'type' => $obj->getType(),
//                  'createdBy' => $createdBy,
//                  'updatedBy' => $updatedBy
//
//                  );
//          }


    }



}
```

```
<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 09:36
 */

namespace comunic\social_network_analyzer\model\entity\
   mappers;
use comunic\social_network_analyzer\model\entity as entities;

class ArrayToObject
{

    private $entityName, $attributes;

    /**
     * @return mixed
     */
    public function getEntityName()
    {
        return $this->entityName;
    }

    /**
     * @param mixed $entityName
     */
    public function setEntityName($entityName)
    {
        $this->entityName = $entityName;
    }

    /**
     * @return mixed
     */
```

```
public function getAttributes()
{
    return $this->attributes;
}

/**
 * @param mixed $attributes
 */
public function setAttributes($attributes)
{
    $this->attributes = $attributes;
}

public function __invoke($arrayData){

    $className = 'comunic\\social_network_analyzer\\model
       \\entity\\' . $this->entityName;

    $obj = new $className();

    foreach ($this->getAttributes() as $attr){

        $attrName = $attr["name"];
        $attrMethod = "set" . \ucfirst($attrName);

        if (isset($arrayData[$attrName])) {

            if(!isset($attr["mapper"])){

                $obj->$attrMethod($arrayData[$attrName]);

            }else{

                $mapperName = 'comunic\\
                   social_network_analyzer\\model\\entity
                   \\mappers\\' . $attr["mapper"];

                if(isset($attr["isArray"]) && $attr["
                   isArray"]){
```

```
                        $items = array();

                        foreach ($arrayData[$attrName] as
                           $item) {
                            $toAttr = new $mapperName();
                            $items[] = $toAttr($item);
                        }

                        $obj->$attrMethod($items);

                    }else{

                        $toAttr = new $mapperName();

                        $obj->$attrMethod($toAttr($arrayData[
                           $attrName]));
                    }
                }
            }

        }

        return $obj;

    }



}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 14/07/16
 * Time: 15:10
 */

namespace comunic\social_network_analyzer\model\entity;
```

```
class TweetThread

{

    private $id, $name, $username, $text, $date;


    public function __construct($id,$username, $name, $text,
       $date)
    {
        $this->text = $text;
        $this->date = $date;
        $this->name = $name;
        $this->username = $username;
        $this->id = $id;

    }

    /**
     * @return mixed
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * @return mixed
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * @return mixed
     */
```

```
    public function getUsername()
    {
        return $this->username;
    }
    /**
     * @return mixed
     */
    public function getText()
    {
        return $this->text;
    }

    /**
     * @return mixed
     */
    public function getDate()
    {
        return $this->date;
    }

}<?php

namespace comunic\social_network_analyzer\model\entity{
    /**
     * class Dataset
     *
     */
    class Dataset
    {

        /** Aggregations: */

        /** Compositions: */

        /*** Attributes: ***/
        /**
         *
         * @access private
         */
```

```
private $id;
private $name;
private $description;
private $hasTweets;
private $createdBy;
private $updatedBy;

function __construct(){
    $this->hasTweets = false;
}


public function getId()
{
    return $this->id;
}

public function setId($id)
{
    return $this->id = $id;
}

public function getName()
{
    return $this->name;
}

public function setName($name)
{
    return $this->name = $name;
}

public function getDescription()
{
    return $this->description;
}

public function setDescription($description)
{
```

```
        return $this->description = $description;
    }

    public function getHasTweets()
    {
        return $this->hasTweets;
    }

    public function setHasTweets($hasTweets)
    {
        return $this->hasTweets = $hasTweets;
    }

    /**
     * @return mixed
     */
    public function getCreatedBy()
    {
        return $this->createdBy;
    }

    /**
     * @param mixed $createdBy
     */
    public function setCreatedBy($createdBy)
    {
        $this->createdBy = $createdBy;
    }

    /**
     * @return mixed
     */
    public function getUpdatedBy()
    {
        return $this->updatedBy;
    }

    /**
     * @param mixed $updatedBy
```

```
         */
        public function setUpdatedBy($updatedBy)
        {
            $this->updatedBy = $updatedBy;
        }

    }



    // end of Dataset
}

<?php

namespace comunic\social_network_analyzer\model\entity{

        class Tag{

                private $id;
                private $category;
                private $addedBy;

        /**
         * Tag constructor.
         * @param $categoryId
         */
        public function __construct($category=null)
        {
            $this->category = $category;
        }

        public function __toString()
        {
            return $this->category->getName();
        }


        /**
```

```
         * @return mixed
         */
        public function getId()
        {
                return $this->id;
        }

        /**
         * @param mixed $id
         */
        public function setId($id)
        {
                $this->id = $id;
        }

/**
 * @return mixed
 */
public function getCategory()
{
    return $this->category;
}

/**
 * @param mixed $category
 */
public function setCategory($category)
{
    $this->category = $category;
}



        /**
         * @return mixed
         */
        public function getAddedBy()
        {
                return $this->addedBy;
```

```
}

/**
 * @param mixed $addedBy
 */
public function setAddedBy($addedBy)
{
        $this->addedBy = $addedBy;
}


public function addModification(Modification
   $mod)
{
    if($this->searchModification($mod->
       getIdUser())==-1){
                $this->addedBy[] = $mod;
        }
}

public function removeModification($userId){

        $index = $this->searchModification(
           $userId);
        unset($this->addedBy[$index]);

}

private function searchModification($idUser){

        $index = -1;

        for($i = 0; $i < count($this->addedBy
           ); $i++){
              if($this->addedBy[$i]->
                 getIdUser() == $idUser){
                      $index = $i;
                      break;
              }
```

```
                        }
                        return $index;
                }

                public function addedByIsEmpty(){
                        return count($this->addedBy) == 0;
                }




        }


}

<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 14/07/16
 * Time: 11:02
 */

namespace comunic\social_network_analyzer\model\entity;


class ConversationCrawler
{


    private function getConversationRoot($idTweet){

        $content = $this->getPageContent($idTweet);

        if($content){

            $result = $content->getElementsByTagName("div");
```

```
            foreach ($result as $item) {

                if ($item->hasAttribute("data-id")) {
                    return $item->getAttribute("data-id");
                }
            }
        }

    }

    public function getPageContent($idTweet){

        \libxml_use_internal_errors(true);
        $result = \DOMDocument::loadHTMLFile("https://mobile.
           twitter.com/string/status/" . $idTweet);
        \libxml_use_internal_errors(false);

        return $result;
    }

    public function retrieveConversation($idTweet){

        $tweets = [];

        $root = $this->getConversationRoot($idTweet);

        if($root){

            $dom = $this->getPageContent($root);
            $xpath = new \DomXPath($dom);

            $rootNode = $xpath->query("//table[@class='main-
               tweet']")[0];

            $htmlString = $dom->saveHTML($rootNode);

            $tweets[] = $this->getDetails($htmlString,"root")
               ;
```

```
            $replies = $xpath->query("//table[@class='tweet␣␣
                ']");

            foreach ($replies as $reply){

                $htmlString = $dom->saveHTML($reply);
                $tweets[] = $this->getDetails($htmlString,"
                    reply");
            }
        }

        return $tweets;

    }

    private function getDetails($node, $type){

        $dom = new \DOMDocument();
        $dom->loadHTML(mb_convert_encoding($node, 'HTML-
            ENTITIES', 'UTF-8'));

        $xpath = new \DomXPath($dom);

        $text = \trim($xpath->query("//div[@class=\"dir-ltr
            \"]")[0]->textContent);
        $id = \trim($xpath->query("//div[@class=\"tweet-text
            \"]/@data-id")[0]->textContent);

        switch ($type){

            case 'root':

                $name = \trim($xpath->query("//div[@class=\"
                    fullname\"]")[0]->textContent);
                $username = \trim($xpath->query("//span[
                    @class=\"username\"]")[0]->textContent);
                $date = $xpath->query("//div[@class=\"
                    metadata\"]/a")[0]->textContent;
                $date = \trim(\explode("-",$date)[1]);
```

```
                break;


            case 'reply':

                $name = \trim($xpath->query("//strong[@class
                    =\"fullname\"]")[0]->textContent);
                $username = \trim($xpath->query("//div[@class
                    =\"username\"]")[0]->textContent);

                $date = \trim($xpath->query("//a[@name='
                    tweet_" . $id ."']")[0]->textContent);

                break;
        }

        return new TweetThread($id,$username,$name,$text,
            $date);
    }



    public function retrieveMoreConversation($root,
        $lastTweet){

        $tweets = [];

        $result = \file_get_contents("https://mobile.twitter.
            com/i/rw/conversation/$root/$root/descendants/
            $lastTweet");
        $html = \json_decode($result, true)["html"];

        if($html){

            $dom = new \DOMDocument();

            libxml_use_internal_errors(true);
            $dom->loadHTML(mb_convert_encoding($html, 'HTML-
```

```
            ENTITIES', 'UTF-8'));
        libxml_use_internal_errors(false);

        $xpath = new \DomXPath($dom);

        $replies = $xpath->query("//div[@class='Tweet-
            body']");


        foreach ($replies as $reply){
            $htmlString = $dom->saveHTML($reply);
            $tweets[] = $this->getMoreDetails($htmlString
                );

        }
    }

    return $tweets;
}


private function getMoreDetails($node){


    $dom = new \DOMDocument();

    libxml_use_internal_errors(true);
    $dom->loadHTML(mb_convert_encoding($node, 'HTML-
        ENTITIES', 'UTF-8'));
    libxml_use_internal_errors(false);

    $xpath = new \DomXPath($dom);

    $username = \trim($xpath->query("//div/span/span[
        @class=\"UserNames-screenName␣u-dir\"]")[0]->
        textContent);
    $username = \substr($username,1);

    $name = \trim( $xpath->query("//div/span/b")[0]->
```

```
            textContent);

        $text = \trim($xpath->query("//div[@class=\"Tweet-
            text␣TweetText␣u-textBreak␣u-dir\"]")[0]->
            textContent);

        $date = \trim($xpath->query("//a[@class=\"Tweet-
            timestamp␣u-floatRight\"]/time")[0]->textContent);

        $twLink = \trim($xpath->query("//a[@class=\"Tweet-
            timestamp␣u-floatRight\"]/@href")[0]->textContent)
            ;
        $twLinkExp = \explode("/",$twLink);

        $id = $twLinkExp[count($twLinkExp) - 1];

        return new TweetThread($id,$username,$name,$text,
            $date);

    }



}<?php

/**
 * class IObjectFormatter
 *
 */

namespace comunic\social_network_analyzer\model\entity\format
    {

    interface IObjectFormatter {
        /** Aggregations: */
        /** Compositions: */

        /**
```

```
         *
         *
         * @param  obj
         * @return string
         * @access public
         */
        public function format($obj);
    }

    // end of IObjectFormatter
}

<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 20/12/16
 * Time: 13:43
 */

namespace comunic\social_network_analyzer\model\entity\format
   \assoc_array;

use Doctrine\ODM\MongoDB\Cursor;

class MongoDBDoctrineFormatter
{

    public function map($entityName,$object)
    {

        $class = "comunic\\social_network_analyzer\\model\\
           entity\\" . $entityName;

        if($object instanceof $class){
            return $object->toArray();
        }

        $data = [];
```

```
        if($object instanceof Cursor){

            foreach ($object as $obj){
                $data[] = $obj->toArray();
            }

        }

        return $data;


    }


}<?php

namespace comunic\social_network_analyzer\model\entity\format
   \json {

    use comunic\social_network_analyzer\model\entity\format\
       IObjectFormatter;

    class BasicObjectFormatter implements IObjectFormatter {

        private $funcToArray;

        function __construct($funcToArray){
            $this->funcToArray = $funcToArray;
        }


        public function format($obj) {
            if (\is_array($obj)) {

                $data = array();

                foreach ($obj as $item) {
                    $data[] = $this->funcToArray->__invoke(
```

```
                        $item);
                }

                return \json_encode($data);
            }

            return \json_encode($this->funcToArray->__invoke(
               $obj));
        }
    }

}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 19/12/16
 * Time: 18:51
 */

namespace comunic\social_network_analyzer\model\entity\format
   \json;


use comunic\social_network_analyzer\model\entity\format\
   IObjectFormatter;
use Doctrine\ODM\MongoDB\Cursor;

class NewBasicObjectFormatter implements IObjectFormatter
{

    public function format($data)
    {

        return \json_encode($data);


    }
}<?php
```

```
namespace comunic\social_network_analyzer\model\entity\format
   \json{

    use \comunic\social_network_analyzer\model\entity\format\
       IObjectFormatter;

    class JsonPaginator implements IObjectFormatter{

        private $objToMap;

        public function __construct($objToMap){
            $this->objToMap = $objToMap;
        }

        public function format($paginator){

            $document = array();


            $document["count"] = $paginator->getCount();
            $document["page"] = $paginator->getPage();
            $document["numberPages"] =  $paginator->
               getNumberPages();
            $document["amountPerPage"] = $paginator->
               getAmountPerPage();

            $data = array();


            foreach ($paginator->getObjectList() as $obj) {
                $data[] = $this->objToMap->__invoke($obj);
            }

            $document["data"] = $data;

            return \json_encode($document);

        }
```

```
    }



}

<?php

namespace comunic\social_network_analyzer\model\entity\format
   \csv{

        use comunic\social_network_analyzer\model\entity\
           format\IObjectFormatter;


        class BasicObjectFormatter implements
           IObjectFormatter{

                private $enclosure;
                private $delimiter;
                private $funcToArray;

                function __construct($funcToArray, $delimiter
                   ='|',$enclosure='"'){
                        $this->funcToArray = $funcToArray;
                        $this->enclosure = $enclosure;
                        $this->delimiter = $delimiter;

                }

                public function format($objects){


                        $fp = \fopen('php://temp', 'r+b');
                        //header
                        \fputcsv($fp, \array_keys($this->
                           funcToArray->__invoke(\current(
                           $objects))), $this->delimiter,
                           $this->enclosure);
```

```
                        foreach ($objects as $obj) {


                                $values=\array_values($this->
                                   funcToArray->__invoke($obj
                                   ));

                                \fputcsv($fp, $values, $this
                                   ->delimiter, $this->
                                   enclosure);
                        }

                        \rewind($fp);
                        $data = \rtrim(\stream_get_contents(
                           $fp), "\n");
                        \fclose($fp);

                        return $data;

                }

        }

}



<?php

namespace comunic\social_network_analyzer\model\entity{

        class Modification{

                private $id;
                private $idUser;
                private $nameUser;
                private $time;
```

```
/**
 * Modification constructor.
 * @param $idUser
 * @param $nameUser
 * @param $time
 */
public function __construct($idUser=null,
   $nameUser=null, $time=null)
{
        $this->idUser = $idUser;
        $this->nameUser = $nameUser;
        $this->time = $time;
}


/**
 * @return mixed
 */
public function getId()
{
        return $this->id;
}

/**
 * @param mixed $id
 */
public function setId($id)
{
        $this->id = $id;
}

public function getIdUser()
{
    return $this->idUser;
}

public function setIdUser($idUser)
{
```

```
                return $this->idUser = $idUser;
            }

            /**
             * @return mixed
             */
            public function getNameUser()
            {
                    return $this->nameUser;
            }

            /**
             * @param mixed $nameUser
             */
            public function setNameUser($nameUser)
            {
                    $this->nameUser = $nameUser;
            }



            public function getTime()
            {
                return $this->time;
            }

            public function setTime($time)
            {
                return $this->time = $time;
            }



        }

}


<?php
```

```
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 23/05/16
 * Time: 15:47
 */

namespace comunic\social_network_analyzer\model\entity;


use comunic\social_network_analyzer\model\util\types\EnumType
   ;

class CategoryTypes extends EnumType
{

    public function getFields()
    {
        return ["SPACES_OF_POSSIBILITY", "RELEVANT_PROCESS"];
    }
}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 05/08/16
 * Time: 09:13
 */

namespace comunic\social_network_analyzer\model\entity;


class MatrixElement
{

    private $rowId, $rowName, $columnId, $columnName, $symbol
       ;

    /**
     * MatrixElement constructor.
```

```
     * @param $rowId
     * @param $rowName
     * @param $columnId
     * @param $columnName
     * @param $symbol
     */
    public function __construct($rowId=null, $rowName=null,
       $columnId=null, $columnName=null, $symbol=null)
    {
        $this->rowId = $rowId;
        $this->rowName = $rowName;
        $this->columnId = $columnId;
        $this->columnName = $columnName;
        $this->symbol = $symbol;
    }


    /**
     * @return mixed
     */
    public function getRowId()
    {
        return $this->rowId;
    }

    /**
     * @param mixed $rowId
     */
    public function setRowId($rowId)
    {
        $this->rowId = $rowId;
    }

    /**
     * @return mixed
     */
    public function getRowName()
    {
        return $this->rowName;
```

```
}

/**
 * @param mixed $rowName
 */
public function setRowName($rowName)
{
    $this->rowName = $rowName;
}

/**
 * @return mixed
 */
public function getColumnId()
{
    return $this->columnId;
}

/**
 * @param mixed $columnId
 */
public function setColumnId($columnId)
{
    $this->columnId = $columnId;
}

/**
 * @return mixed
 */
public function getColumnName()
{
    return $this->columnName;
}

/**
 * @param mixed $columnName
 */
public function setColumnName($columnName)
{
```

```
        $this->columnName = $columnName;
    }

    /**
     * @return mixed
     */
    public function getSymbol()
    {
        return $this->symbol;
    }

    /**
     * @param mixed $symbol
     */
    public function setSymbol($symbol)
    {
        $this->symbol = $symbol;
    }



}<?php

/**
 * class IObjectParser
 *
 */

namespace comunic\social_network_analyzer\model\entity\parse
   {

    interface IObjectParser {
        /** Aggregations: */
        /** Compositions: */

        /**
         *
         *
         * @param string text
```

```
         * @return void
         * @access public
         */
        public function parse($text);
    }

    // end of IObjectParser
}

<?php

namespace comunic\social_network_analyzer\model\entity\parse\
   json{

    use comunic\social_network_analyzer\model\entity\parse\
       IObjectParser;
    use comunic\social_network_analyzer\model\util\ArrayUtil;

    class BasicObjectParser implements IObjectParser{

        private $funcToObj;

        function __construct($funcToObj){
            $this->funcToObj = $funcToObj;
        }

        public function parse($jsonText){

            $arrayData = \json_decode($jsonText, true);

            if(ArrayUtil::is_assoc_array($arrayData)){

                return $this->funcToObj->__invoke($arrayData)
                   ;

            }

            if(\is_array($arrayData)){
```

```
                $listObjects = array();

                foreach ($arrayData as $item) {
                    $listObjects[] = $this->funcToObj->
                       __invoke($item);
                }

                return $listObjects;

            }
        }
    }
}

<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 19/12/16
 * Time: 21:05
 */

namespace comunic\social_network_analyzer\model\entity\parse\
   json;


use comunic\social_network_analyzer\model\entity\parse\
   IObjectParser;
use comunic\social_network_analyzer\model\entity\parse\text;

class NewBasicObjectParser implements IObjectParser
{

    public function parse($text)
    {
        return \json_decode($text,true);
    }
}<?php
```

```
namespace comunic\social_network_analyzer\model\entity\parse\
   csv {

  class CSVTweetParser2 {

    private $delimiter;
      private $enclosure;

    function __construct($delimiter, $enclosure){
      $this->delimiter = $delimiter;
        $this->enclosure = $enclosure;
    }

    public function parse($csvString){

        $fp = fopen('php://temp','r+');
        fwrite($fp, $csvString);
        rewind($fp); //rewind to process CSV

        $header = $row = fgetcsv($fp, 0, $this->delimiter,
           $this->enclosure);
        $data = [];

        while (($row = fgetcsv($fp, 0, $this->delimiter,
           $this->enclosure)) !== FALSE) {
            $data[] = \array_combine($header, $row);
        }

        fclose($fp);

        return $data;


    }



 }
```

```
}<?php

namespace comunic\social_network_analyzer\model\entity\parse\
   csv{

    use comunic\social_network_analyzer\model\entity\parse\
       IObjectParser;

    abstract class BasicCSVObjectParser implements
       IObjectParser{


        public function parse($csvData){
            $csvData = \trim($csvData);
            $csvDataArray = \explode("\n", $csvData);

            $csvDataArray = $this->prepareCSV($csvDataArray);

            $fieldNames = \str_getcsv(\trim($csvDataArray[0])
               , "|");
            unset($csvDataArray[0]);

            $objects = array();

            foreach ($csvDataArray as $csvRow) {

                $csvRowArray = \str_getcsv(\trim($csvRow), "|
                   ");

                $rowDataMap = array();

                if (\count($fieldNames) == \count(
                   $csvRowArray)) {

                    for ($i = 0; $i < \count($fieldNames); $i
                       ++) {
                        $f_name = $fieldNames[$i];
                        $csv_fd_data = $csvRowArray[$i];
                        $rowDataMap[$f_name] = $csv_fd_data;
```

```
                    }

                    $objects[] = $this->arrayToObject(
                        $rowDataMap);
                }

            }

            return $objects;



        }

        abstract protected function arrayToObject($arrayData)
            ;

        abstract protected function prepareCSV($data);


    }




}


<?php

namespace comunic\social_network_analyzer\model\entity\parse\
    csv {

  use comunic\social_network_analyzer\model\entity\parse\csv\
      BasicCSVObjectParser;
  use comunic\social_network_analyzer\model\entity\Tweet;

  class CSVTweetParser {
```

```
private $funcToObj;
private $delimiter;

function __construct($funcToObj, $delimiter){
  $this->funcToObj = $funcToObj;
  $this->delimiter = $delimiter;
}

public function parse($csvData){
  $csvData = \trim($csvData);
  $csvDataArray = \explode("\n", $csvData);

  $csvDataArray = $this->removeLineBreaksFromText(
     $csvDataArray);

  $fieldNames = \str_getcsv(\trim($csvDataArray[0]),
     $this->delimiter);
  unset($csvDataArray[0]);

  $objects = array();

  foreach ($csvDataArray as $csvRow) {

    $csvRowArray = \str_getcsv(\trim($csvRow), $this->
       delimiter);

    if (\count($fieldNames) == \count($csvRowArray)) {

      $arrayData = \array_combine($fieldNames,
         $csvRowArray);

      $objects[] = $this->funcToObj->__invoke($arrayData)
         ;
    }

  }

  return $objects;
```

```
    }

    private function removeLineBreaksFromText($data){
     for ($i=0; $i <\count($data) ; $i++) {
       if(\strlen($data[$i]) < 140 && (\count(\explode($this
          ->delimiter, $data[$i]))==1)){
         $data[$i+1] = $data[$i] . $data[$i+1];
         unset($data[$i]);
       }
     }

     return $data;
   }

 }

}<?php

namespace comunic\social_network_analyzer\model\entity{

    use comunic\social_network_analyzer\model\auth\Crypt;
    /**
     * class User
     *
     */
    class User
    {

        /** Aggregations: */

        /** Compositions: */

        /*** Attributes: ***/

        private $id;
        private $name;
```

```
private $email;
private $role;
private $enabled;
private $password;

/**
 * @return mixed
 */
public function getPassword()
{
    return $this->password;
}

/**
 * @param mixed $password
 */
public function setPassword($password)
{
    $this->password = $password;
}

/**
 * @return $id
 * @access public
 */


public function getId(){
    return $this->id;
}

/**
 * @return $name
 * @access public
 */

public function getName(){
    return $this->name;
}
```

```
/**
 * @param $id
 * @return void
 * @access public
 */

public function setId($id){
    $this->id = $id;
}

/**
 * @param $id
 * @return void
 * @access public
 */

public function setName($name){
    $this->name = $name;
}

public function getEmail()
{
    return $this->email;
}

public function setEmail($email)
{
    return $this->email = $email;
}

public function getRole()
{
    return $this->role;
}

public function setRole($role)
{
    return $this->role = $role;
```

```
        }

        public function getEnabled()
        {
            return $this->enabled;
        }

        public function setEnabled($enabled)
        {
            return $this->enabled = $enabled;
        }


    }

} // end of User

<?php

namespace comunic\social_network_analyzer\model\entity{

        use comunic\social_network_analyzer\model\util\types\
           EnumType;

        class SlicingPoint extends EnumType{


                public function getFields(){
                        return ["BEGINNING", "HALF", "END"];
                }


        }

}


<?php
```

```
namespace comunic\social_network_analyzer\model\entity{

    use comunic\social_network_analyzer\model\util\StringUtil
       ;
    /**
     * class Tweet
     *
     */
    class Tweet
    {

        /** Aggregations: */

        /** Compositions: */

        /*** Attributes: ***/

        /**
         * attribute id is the id of object on database
         * attribute idT$isVisibleweet is the id of tweet
         */

        private $text;
        private $fromUser;
        private $id;
        private $idTweet;
        private $unixTimestamp;
        private $otherAttributes;
        private $idDataset;
        private $classes;
        private $textPreProcessed;
        private $isVisible;
        private $tags;
        private $normalizedText;



        public function __construct(){
        }
```

```
/**
 * @return mixed
 */
public function getText()
{
    return $this->text;
}

/**
 * @param mixed $text
 */
public function setText($text)
{
    $this->text = $text;
}

/**
 * @return mixed
 */
public function getFromUser()
{
    return $this->fromUser;
}

/**
 * @param mixed $fromUser
 */
public function setFromUser($fromUser)
{
    $this->fromUser = $fromUser;
}

/**
 * @return mixed
 */
public function getId()
{
    return $this->id;
```

```
}

/**
 * @param mixed $id
 */
public function setId($id)
{
    $this->id = $id;
}

/**
 * @return mixed
 */
public function getIdTweet()
{
    return $this->idTweet;
}

/**
 * @param mixed $idTweet
 */
public function setIdTweet($idTweet)
{
    $this->idTweet = $idTweet;
}

/**
 * @return mixed
 */
public function getUnixTimestamp()
{
    return $this->unixTimestamp;
}

/**
 * @param mixed $unixTimestamp
 */
public function setUnixTimestamp($unixTimestamp)
{
```

```
        $this->unixTimestamp = $unixTimestamp;
    }

    /**
     * @return mixed
     */
    public function getOtherAttributes()
    {
        return $this->otherAttributes;
    }

    /**
     * @param mixed $otherAttributes
     */
    public function setOtherAttributes($otherAttributes)
    {
        $this->otherAttributes = $otherAttributes;
    }

    /**
     * @return mixed
     */
    public function getIdDataset()
    {
        return $this->idDataset;
    }

    /**
     * @param mixed $idDataset
     */
    public function setIdDataset($idDataset)
    {
        $this->idDataset = $idDataset;
    }

    /**
     * @return mixed
     */
    public function getClasses()
```

```
{
    return $this->classes;
}

/**
 * @param mixed $classes
 */
public function setClasses($classes)
{
    $this->classes = $classes;
}

/**
 * @return mixed
 */
public function getTextPreProcessed()
{
    return $this->textPreProcessed;
}

/**
 * @param mixed $textNormalized
 */
public function setTextPreProcessed($textPreProcessed
  )
{
    $this->textPreProcessed = $textPreProcessed;
}

/**
 * @return mixed
 */
public function getIsVisible()
{
    return $this->isVisible;
}

/**
 * @param mixed $isVisible
```

```
     */
    public function setIsVisible($isVisible)
    {
        $this->isVisible = $isVisible;
    }

    /**
     * @return mixed
     */
    public function getTags()
    {
        return $this->tags;
    }

    /**
     * @param mixed $tags
     */
    public function setTags($tags)
    {
        $this->tags = $tags;
    }



    public function searchClass($idCategory){

        $index = -1;
        for($i=0; $i<count($this->classes); $i++){

            if($this->classes[$i]->getCategory()->getId()
                == $idCategory){
                $index = $i;
                break;
            }
        }

        return $index;
    }
```

```
public function addClass($category, $idUser,$nameUser
   ){


    $index = $this->searchClass($category->getId());

    if($index==-1){

        $class = new Tag($category);

        $class->addModification(new Modification(
           $idUser,$nameUser,\time()));
        $this->classes[] = $class;

    }
    else{

        $this->classes[$index]->addModification(new
           Modification($idUser,$nameUser, \time()));
    }

}

public function removeClass($idCategory,$idUser){

    $indexTag = $this->searchClass($idCategory);

    if($indexTag > -1){
        $class = $this->classes[$indexTag];
        $class->removeModification($idUser);

        if($class->addedByIsEmpty()){
            unset($this->classes[$indexTag]);
        }
    }
}

public function searchTag($idCategory){
```

```
        $index = -1;
        for($i=0; $i<count($this->tags); $i++){

            if($this->tags[$i]->getCategory()->getId() ==
                $idCategory){
                $index = $i;
                break;
            }
        }

        return $index;
    }


    public function addTag($category, $idUser,$nameUser){


        $index = $this->searchTag($category->getId());

        if($index==-1){

            $tag = new Tag($category);

            $tag->addModification(new Modification(
               $idUser,$nameUser,\time()));
            $this->tags[] = $tag;

        }
        else{

            $this->tags[$index]->addModification(new
               Modification($idUser,$nameUser, \time()));
        }

    }

    public function removeTag($idCategory,$idUser=null){
```

```
        $indexTag = $this->searchTag($idCategory);

        if($indexTag > -1){

            if(\is_null($idUser)){
                unset($this->tags[$indexTag]);
            }else{
            $tag = $this->tags[$indexTag];
            $tag->removeModification($idUser);

            if($tag->addedByIsEmpty()){
                unset($this->tags[$indexTag]);
            }
            }
        }
    }

    public function normalizeTweet(){
        $this->setNormalizedText(StringUtil::
           removeAccents($this->text));
    }

    /**
     * @return mixed
     */
    public function getNormalizedText()
    {
        return $this->normalizedText;
    }

    /**
     * @param mixed $normalizedText
     */
    public function setNormalizedText($normalizedText)
    {
        $this->normalizedText = $normalizedText;
    }
```

```
    }

} // end of Tweet

<?php

namespace comunic\social_network_analyzer\model\entity{


    class Paginator{

        private $objectList;
        private $count;
        private $page;
        private $amountPerPage;
        private $numberPages;

        public function __construct($objectList, $count,
           $skip, $amountPerPage){
            $this->objectList = $objectList;
            $this->count = $count;
            $this->page = ($skip/$amountPerPage) + 1;
            $this->amountPerPage = $amountPerPage;
            $this->numberPages = \ceil($count /
               $amountPerPage);

        }

        public function getObjectList()
        {
            return $this->objectList;
        }

        public function setObjectList($objectList)
        {
            return $this->objectList = $objectList;
        }
```

```
public function getCount()
{
    return $this->count;
}

public function setCount($count)
{
    return $this->count = $count;
}

public function getPage()
{
    return $this->page;
}

public function setPage($page)
{
    return $this->page = $page;
}

public function getAmountPerPage()
{
    return $this->amountPerPage;
}

public function setAmountPerPage($amountPerPage)
{
    return $this->amountPerPage = $amountPerPage;
}

public function getNumberPages()
{
    return $this->numberPages;
}

public function setNumberPages($numberPages)
{
    return $this->numberPages = $numberPages;
}
```

```
    }

}



<?php

namespace comunic\social_network_analyzer\model\entity{

    use comunic\social_network_analyzer\model\util\StringUtil
       ;
    /**
     * class Category
     *
     */
    class Category
    {

        /** Aggregations: */

        /** Compositions: */

        /*** Attributes: ***/

        private $id;
        private $name;
        private $description;
        private $keywords;
        private $type;
        private $color;
        private $createdBy;
        private $updatedBy;


        /**
```

```
     * @return $id
     * @access public
     */


    public function getId(){
        return $this->id;
    }

    /**
     * @return $name
     * @access public
     */

    public function getName(){
        return $this->name;
    }


    /**
     * @return $keywords
     * @access public
     */

    public function getKeywords(){
        return $this->keywords;
    }

    /**
     * @param $id
     * @return void
     * @access public
     */

    public function setId($id){
        $this->id = $id;
    }

    /**
```

```
     * @param $name
     * @return void
     * @access public
     */

    public function setName($name){
        $this->name = $name;
    }

    /**
     * @param $keywords
     * @return void
     * @access public
     */

    public function setKeywords($keywords){
        $this->keywords = $keywords;
    }

    public function getDescription()
    {
        return $this->description;
    }

    public function setDescription($description)
    {
        return $this->description = $description;
    }

    public function getColor()
    {
        return $this->color;
    }

    public function setColor($color)
    {
        return $this->color = $color;
    }
```

```
    /**
     * @return mixed
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * @param mixed $type
     */
    public function setType($type)
    {
        $this->type = $type;
    }

    /**
     * @return mixed
     */
    public function getCreatedBy()
    {
        return $this->createdBy;
    }

    /**
     * @param mixed $createdBy
     */
    public function setCreatedBy($createdBy)
    {
        $this->createdBy = $createdBy;
    }

    /**
     * @return mixed
     */
    public function getUpdatedBy()
    {
        return $this->updatedBy;
    }
```

```
/**
 * @param mixed $updatedBy
 */
public function setUpdatedBy($updatedBy)
{
    $this->updatedBy = $updatedBy;
}



public function toRegex(){
    $kwAsRegex = array();

    foreach ($this->keywords as $kw) {
        // // $kw = str_replace('?', '', $kw);
        // $wordAsRegex = StringUtil::accentToRegex(
           $kw);
        // // $wordAsRegex = str_replace('.', '.?',
           $wordAsRegex);
        $kwAsRegex[] = "/\b$kw\b/iu";
    }

    return $kwAsRegex;
}

public function matchWithKeywords($words){
    $kwAsRegex = $this->toRegex();
    $matchWords=array();

    foreach ($kwAsRegex as $kw) {
        $matchWords = \array_merge($matchWords,
           preg_grep($kw, $words));
    }

    return $matchWords;
}
```

```
    }


} // end of Category

<?php

namespace comunic\social_network_analyzer\model\entity{

        use comunic\social_network_analyzer\model\util\types\
           EnumType;

        class UserRole extends EnumType{


                public function getFields(){
                        return ["ADMINISTRATOR", "MODERATOR",
                            "RESEARCHER"];
                }


        }

}


<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 03/08/16
 * Time: 11:41
 */

namespace comunic\social_network_analyzer\model\entity;


class Matrix
{
```

```
private $id, $name, $datasetsIds, $rowsIds, $columnsIds,
   $relation, $elements;
private $createdBy;
private $updatedBy;

/**
 * @return mixed
 */
public function getId()
{
    return $this->id;
}

/**
 * @param mixed $id
 */
public function setId($id)
{
    $this->id = $id;
}


/**
 * @return mixed
 */
public function getName()
{
    return $this->name;
}

/**
 * @param mixed $name
 */
public function setName($name)
{
    $this->name = $name;
}
```

```
/**
 * @return mixed
 */
public function getDatasetsIds()
{
    return $this->datasetsIds;
}

/**
 * @param mixed $datasetsIds
 */
public function setDatasetsIds($datasetsIds)
{
    $this->datasetsIds = $datasetsIds;
}

/**
 * @return mixed
 */
public function getRowsIds()
{
    return $this->rowsIds;
}

/**
 * @param mixed $rowsIds
 */
public function setRowsIds($rowsIds)
{
    $this->rowsIds = $rowsIds;
}

/**
 * @return mixed
 */
public function getColumnsIds()
{
    return $this->columnsIds;
}
```

```
/**
 * @param mixed $columnsIds
 */
public function setColumnsIds($columnsIds)
{
    $this->columnsIds = $columnsIds;
}

/**
 * @return mixed
 */
public function getRelation()
{
    return $this->relation;
}

/**
 * @param mixed $relation
 */
public function setRelation($relation)
{
    $this->relation = $relation;
}

/**
 * @return mixed
 */
public function getElements()
{
    return $this->elements;
}

/**
 * @param mixed $elements
 */
public function setElements($elements)
{
    $this->elements = $elements;
```

```
}

/**
 * @return mixed
 */
public function getCreatedBy()
{
    return $this->createdBy;
}

/**
 * @param mixed $createdBy
 */
public function setCreatedBy($createdBy)
{
    $this->createdBy = $createdBy;
}

/**
 * @return mixed
 */
public function getUpdatedBy()
{
    return $this->updatedBy;
}

/**
 * @param mixed $updatedBy
 */
public function setUpdatedBy($updatedBy)
{
    $this->updatedBy = $updatedBy;
}
```

```
}<?php

namespace comunic\social_network_analyzer\model\exception{

        class ValidationException extends \Exception{

        }

}


<?php

namespace comunic\social_network_analyzer\model\exception{

        class NoResultException extends \Exception{

        }

}


<?php

namespace comunic\social_network_analyzer\model\exception{

        class UniqueConstraintException extends \Exception{

        }

}


<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 19/12/16
```

```
 * Time: 17:21
 */

namespace comunic\social_network_analyzer\model\repository\
   doctrine;


use comunic\social_network_analyzer\model\repository\
   IRepositoryFactory;
use Doctrine\ODM\MongoDB\DocumentManager;

class DoctrineRepository implements IRepositoryFactory
{

    private $dm;

    /**
     * DoctrineRepository constructor.
     * @param $dm
     */
    public function __construct(DocumentManager $dm)
    {
        $this->dm = $dm;
    }

    public function instantiateProject(){
        return new ProjectRepository($this->dm);
    }


    public function instantiateTweet()
    {
        // TODO: Implement instantiateTweet() method.
    }

    public function instantiateUser()
    {
        return new UserRepository($this->dm);
    }
```

```
    public function instantiateCategory()
    {
        // TODO: Implement instantiateCategory() method.
    }

    public function instantiateDataset()
    {
        // TODO: Implement instantiateDataset() method.
    }

    public function instantiateMatrix(){
        // TODO: Implement instantiateDataset() method.
    }
}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 19/12/16
 * Time: 16:54
 */

namespace comunic\social_network_analyzer\model\repository\
   doctrine;

use comunic\social_network_analyzer\model\entity\Project;
use comunic\social_network_analyzer\model\repository\
   IProjectsRepository;
use Doctrine\ODM\MongoDB\DocumentManager;


class ProjectRepository implements IProjectsRepository
{

    private $dm;
    private $queryBuilder;

    /**
     * ProjectRepository constructor.
```

```
     * @param $dm
     */
    public function __construct(DocumentManager $dm)
    {
        $this->dm = $dm;
        $this->queryBuilder = $dm->createQueryBuilder(Project
           ::class);
    }

    public function save($project){
        $this->dm->persist($project);
        $this->dm->flush();
    }

    public function findById($id){

        return $this
            ->queryBuilder
            ->field("_id")->equals($id)
            ->getQuery()
            ->getSingleResult();
    }

    public function listAll(){

        return $this->queryBuilder
            ->getQuery()
            ->execute();
    }

    public function delete($id){

        return $this
            ->queryBuilder
            ->remove()
            ->field("_id")->equals($id)
            ->getQuery()
            ->execute();
```

```
    }

    public function findByUser($userId){

        $this->queryBuilder->addOr($this->queryBuilder->expr
           ()->field("admin")->equals($userId));
        $this->queryBuilder->addOr($this->queryBuilder->expr
           ()->field("managers")->equals($userId));
        $this->queryBuilder->addOr($this->queryBuilder->expr
           ()->field("members")->equals($userId));
        return $this->queryBuilder->getQuery()->execute();

    }


}<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 20/12/16
 * Time: 14:32
 */

namespace comunic\social_network_analyzer\model\repository\
   doctrine;

use Doctrine\ODM\MongoDB\DocumentManager;
use comunic\social_network_analyzer\model\entity\User;
use comunic\social_network_analyzer\model\repository\
   IUsersRepository;

class UserRepository implements IUsersRepository
{
    private $dm;
    private $queryBuilder;

    /**
     * UserRepository constructor.
     * @param $dm
```

```
 */
public function __construct(DocumentManager $dm)
{
    $this->dm = $dm;
    $this->queryBuilder = $dm->createQueryBuilder(User::
       class);
}

public function save($project){
    $this->dm->persist($project);
    $this->dm->flush();
}

public function findById($id){

    return $this
        ->queryBuilder
        ->field("_id")->equals($id)
        ->getQuery()
        ->getSingleResult();
}

public function findByEmail($email){

    return $this
        ->queryBuilder
        ->field("email")->equals($email)
        ->getQuery()
        ->getSingleResult();
}

public function listAll(){

    return $this->queryBuilder
        ->getQuery()
        ->execute();
}

public function delete($id){
```

```
        return $this
            ->queryBuilder
            ->remove()
            ->field("_id")->equals($id)
            ->getQuery()
            ->execute();

    }

}<?php

namespace comunic\social_network_analyzer\model\repository{

interface IProjectsRepository{

        public function insert($project);

        public function update($project);

        public function delete($id);

        public function listAll();

        public function findById($id);


}

}


<?php

namespace comunic\social_network_analyzer\model\repository{
/**
 * class ITweetsRepository
 *
 */
```

```
interface ITweetsRepository
{

  /** Aggregations: */

  /** Compositions: */

  /**
   *
   *
   * @param  \comunic\social_network_analyzer\model\entity\
      Tweet $tweet
   * @return void
   * @access public
   */
  public function insert( $tweets);

  /**
   *
   *
   * @return array An array of Tweet's instances
   * @see \comunic\social_network_analyzer\model\entity\Tweet
   * @access public
   */
  public function listAll($options);

  /**
   *
   *
   * @param  $id
   * @return \comunic\social_network_analyzer\model\entity\
      Tweet
   * @access public
   */
  public function findById( $id);

  /**
   *
   * @param \comunic\social_network_analyzer\model\entity\
```

```
      Category  $category
    * @return array An array of Tweet's instances
    * @see \comunic\social_network_analyzer\model\entity\Tweet
    * @access public
    */
   public function findByCategory($datasetId, $category,
      $options);




}



} // end of ITweetsRepository

<?php

namespace comunic\social_network_analyzer\model\repository {

    /**
     * class ICategoriesRepository
     *
     */
    interface ICategoriesRepository {
        /** Aggregations: */
        /** Compositions: */

        /**
         *
         *
         * @param  \comunic\social_network_analyzer\model\
            entity\Category $category
         * @return void
         * @access public
         */
        public function insert($category, $projectId);
```

```
/**
 *
 *
 * @param   \comunic\social_network_analyzer\model\
    entity\Category $category
 * @return void
 * @access public
 */
public function update($category, $projectId);

/**
 *
 *
 * @param $id int
 * @return void
 * @access public
 */
public function delete($id, $projectId);

/**
 *
 *
 * @param  int $id
 * @return \comunic\social_network_analyzer\model\
    entity\Category
 * @access public
 */
public function findById($id);

/**
 *
 *
 * @return array An array of Category's instances
 * @see \comunic\social_network_analyzer\model\entity
    \Category
 * @access public
 */
public function listAll($projectId);
```

```
    }

} // end of ICategoriesRepository

<?php

namespace comunic\social_network_analyzer\model\repository{
/**
 * class IUsersRepository
 *
 */
interface IUsersRepository
{

  /** Aggregations: */

  /** Compositions: */

  /**
   *
   *
   * @param  $user \comunic\social_network_analyzer\entity\
      User
   * @return void
   * @access public
   */
  public function insert( $user);

  /**
   *
   *
   * @param  $user \comunic\social_network_analyzer\entity\
      User
   * @return void
   * @access public
   */
  public function update( $user);
```

```
  /**
   *
   *
   * @param $id int
   * @return void
   * @access public
   */
  public function delete( $id);

  /**
   *
   *
   * @param $id int
   * @return \comunic\social_network_analyzer\entity\User
   * @access public
   */
  public function findById( $id,$fields);

  /**
   *
   *
   * @return array An array of User instances
   * @see \comunic\social_network_analyzer\entity\User
   * @access public
   */
  public function listAll();



}

} // end of IUserRepository

<?php

namespace comunic\social_network_analyzer\model\repository\
   arango{
```

```
use comunic\social_network_analyzer\model\repository\
   IProjectsRepository;
use comunic\social_network_analyzer\model\entity\
   mappers\ArrayToProject;
use comunic\social_network_analyzer\model\entity\
   mappers\ProjectToArray;

class ProjectsRepository extends
   AbstractArangoRepository implements
   IProjectsRepository{

        function __construct(){
                parent::__construct();

                $this->entityName = "projects";
        }

        public function insert($project){
                return $this->graphHandler->
                   createVertex($project, new
                   ProjectToArray(), $this->
                   entityName);
        }

        public function update($project){
                $id = $this->buildId($this->
                   entityName, $project->getId());
                return $this->graphHandler->
                   updateVertex($id, $project, new
                   ProjectToArray());
        }

        public function delete($id){
                $id = $this->buildId($this->
                   entityName, $id);
                return $this->graphHandler->
                   deleteVertex($id);
        }
```

```
                public function listAll(){

                        return $this->graphHandler->
                           listVertex($this->entityName, new
                           ArrayToProject());
                }

                public function findById($id){
                        $id = $this->buildId($this->
                           entityName, $id);
                        return $this->graphHandler->getVertex
                           ($id, new ArrayToProject());

                }

        }



}

<?php

namespace comunic\social_network_analyzer\model\repository\
   arango\mappers{


    class ArrayWithArangoKeyToObject{

        public function __invoke($arrayData, $arrayToObj){

            $fArrayToObj = $arrayToObj;

            $obj = $fArrayToObj($arrayData);

            if(isset($arrayData['_key'])){
                $obj->setId($arrayData['_key']);
            }
```

```
            return $obj;


        }


    }


}


<?php

namespace comunic\social_network_analyzer\model\repository\
   arango\mappers{

    class ObjectToArrayWithArangoKey{

        public function __invoke($obj, $objToArray){

            $fObjToArray = $objToArray;

            $arrayData = $fObjToArray($obj);

            // unset($arrayData["id"]);

            if($obj->getId() != null){
                $arrayData["_key"] = $obj->getId();
            }

            return $arrayData;
        }
```

```
    }



}



<?php

namespace comunic\social_network_analyzer\model\repository\
   arango{

        abstract class AbstractArangoRepository{

                protected $entityName;
                protected $graphHandler;

                function __construct(){

                        $this->graphHandler = new
                           ArangoGraphHandler();

                }

                protected function buildId($collName,$objId){
                        return "$collName/$objId";
                }

        }

}



<?php

namespace comunic\social_network_analyzer\model\repository\
   arango{
```

```
require "vendor/autoload.php";

use triagens\ArangoDb\GraphHandler;
use triagens\ArangoDb\CollectionHandler;
use triagens\ArangoDb\Edge;
use triagens\ArangoDb\Cursor;
use triagens\ArangoDb\Vertex;
use triagens\ArangoDb\Graph;
use triagens\ArangoDb\EdgeDefinition;
use triagens\ArangoDb\Statement;
use triagens\ArangoDb\Urls;
use triagens\ArangoDb\UrlHelper;
use triagens\ArangoDb\AqlUserFunction;
use comunic\social_network_analyzer\model\repository\
   arango\mappers\ArrayWithArangoKeyToObject;
use comunic\social_network_analyzer\model\repository\
   arango\mappers\ObjectToArrayWithArangoKey;
use comunic\social_network_analyzer\model\util\
   ArrayUtil;
use comunic\social_network_analyzer\model\entity\
   Paginator;

class ArangoGraphHandler{

        private $graphHandler;
        private $graph;
        private $connection;

        function __construct(){
                $conn = new ConnectionArango();
                $this->connection = $conn->
                   getConnection();

                $this->graphHandler = new
                   GraphHandler($this->connection);

                try {

                        /*
```

```
TODO criar arquivo com arrays
    e constantes de
   configura  o

*/

// $colHandler = new
   CollectionHandler($this->
   connection);
// $colHandler->create("words
   ", array('keyOptions'=>
   array('type'=>'
   autoincrement','
   allowUserKeys'=>true)));

$this->graphHandler->
   createGraph(new Graph("api
   "));
$this->setGraph("api");
$this->addEdgeDefinition("
   projects_datasets_has","
   projects","datasets");
$this->addEdgeDefinition("
   datasets_tweets_belong","
   datasets","tweets");
$this->addEdgeDefinition("
   projects_categories_belong
   ","projects","categories")
   ;
$this->addEdgeDefinition("
   categories_words_contains"
   ,"categories","words");
$this->addEdgeDefinition("
   tweets_categories_belong",
   "tweets","categories");
$this->addEdgeDefinition("
   tweets_categories_not_belong
   ","tweets","categories");
$this->addEdgeDefinition("
```

```
                                    tweets_words_contains","
                                    tweets","words");


                                $aqlFunc = new
                                    AqlUserFunction($this->
                                    connection);

                                $aqlFunc->register("
                                    myfunctions::
                                    tweetsByCategory","
                                    function␣(config,␣result,␣
                                    vertex,␣path)␣{

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(config.data.
    indexOf(String(vertex._key))>␣-1){
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣return␣path.
    vertices[path.vertices.length␣-␣2];
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣}

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣}");

                                $aqlFunc->register("
                                    myfunctions::pruneProjects
                                    ","function␣(config,vertex
                                    ,path){
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣var␣str␣=␣vertex._id;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(str.split('/'')
    [0]===’projects’){
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣return␣[␣’
    prune’,␣’exclude’␣];}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣}");


                            } catch (\Exception $e) {

                            }
                            $this->setGraph("api");
```

```
}

public function importObjects($collection,
   $objects,$toArrayFunc){

        $fObjectToArray = new
           ObjectToArrayWithArangoKey();

        $arrayData = array();
        foreach ($objects as $obj) {
                $arrayData[] =
                   $fObjectToArray($obj,
                   $toArrayFunc);
        }

        return $this->import($collection,
           $arrayData);

}

public function lastInserted($collName,
   $toObjFunc){
        $collHandler = new CollectionHandler(
           $this->connection);
        $lastWord = $collHandler->last(
           $collName);

        return !\is_null($lastWord) ? $this->
           createObject($lastWord,$toObjFunc)
            : -1;

}

public function import($collection,
   $arrayData){
        $params = array(
                "type"=>"list",
                "collection"=>$collection,
                "createCollection"=>false,
```

```
                "overwrite"=>false,
                "waitForSync"=>false,
                "onDuplicate"=>"ignore",
                "complete"=>false,
                "details"=>true);

        $url      = UrlHelper::
           appendParamsUrl(Urls::URL_IMPORT,
           $params);

        $response = $this->connection->post(
           $url, $this->connection->
           json_encode_wrapper($arrayData));

        // echo var_dump($response->getJson()
           );

}




public function setGraph($graphName){
        $this->graph = $this->graphHandler->
           getGraph($graphName);
}

public function addEdgeDefinition(
   $relationName, $colFromName, $colToName){
        $this->graphHandler->
           addEdgeDefinition($this->graph,
           new EdgeDefinition($relationName,
           $colFromName, $colToName));
}

public function createVertex($obj,
   $toArrayDataFunction, $collection=null){
        $fObjectToArray = new
           ObjectToArrayWithArangoKey();
```

```
        $arrayData = $fObjectToArray($obj,
            $toArrayDataFunction);

        return $this->graphHandler->
            saveVertex($this->graph,
            $arrayData, $collection);
}

public function deleteVertex($vertexId,
    $revision = null, $options = array(),
    $collection = null){
        return $this->graphHandler->
            removeVertex($this->graph,
            $vertexId, $revision, $options,
            $collection);
}

public function getVertex($vertexId,
    $toObjectFunction, array $options = array
    (), $collection = null){
        $arrayData = $this->graphHandler->
            getVertex($this->graph, $vertexId,
             $options, $collection)->getAll();
        return $this->createObject($arrayData
            , $toObjectFunction);

}

public function hasVertex($vertexId){
        return $this->graphHandler->hasVertex
            ($this->graph, $vertexId);
}

public function updateVertex($vertexId, $obj,
    $toArrayDataFunction, $options = array(),
    $collection = null){
        $fObjectToArray = new
            ObjectToArrayWithArangoKey();
```

```
            $arrayData = $fObjectToArray($obj,
               $toArrayDataFunction);
            $documentVertex = Vertex::
               createFromArray($arrayData);

            return $this->graphHandler->
               updateVertex($this->graph,
               $vertexId, $documentVertex,
               $options, $collection);
    }

    public function saveEdge($fromVertex,
       $toVertex, $collectionEdge = null,
       $labelEdge = null, $documentEdge = array()
       ){
            return $this->graphHandler->saveEdge(
               $this->graph, $fromVertex,
               $toVertex, $labelEdge,
               $documentEdge, $collectionEdge);
    }

    public function createEdge($fromVertex,
       $toVertex, $label=null,$document=array()){

            if (is_array($document)) {
                    $document = Edge::
                       createFromArray($document)
                       ;
            }
            if (!is_null($label)) {
                    $document->set('$label',
                       $label);
            }
            $document->setFrom($fromVertex);
            $document->setTo($toVertex);
            $data = $document->getAll();
            $data["_from"] = $fromVertex;
            $data["_to"] = $toVertex;
```

```
        return $data;
}

public function getEdge($edgeId, array
   $options = array(), $collection = null){
        return $this->graphHandler->getEdge(
           $this->graph, $edgeId, $options,
           $collection);
}

public function hasEdge($edgeId){
        return $this->graphHandler->hasEdge(
           $this->graph, $edgeId);
}

public function updateEdge($edgeId, $label,
   $arrayData, $options = array(),
   $collection = null){
        $document = Edge::createFromArray(
           $arrayData);

        return $this->graphHandler->
           updateEdge($this->graph, $edgeId,
           $label, $document, $options =
           array(), $collection = null);
}

public function deleteEdge($edgeId, $revision
    = null, $options = array(), $collection =
    null){
        return $this->graphHandler->
           removeEdge($this->graph, $edgeId,
           $revision = null, $options = array
           (), $collection = null);
}

public function getEdges($vertexId,
   $edgeCollection){
```

```
            return $this->graphHandler->getEdges(
               $this->graph, array('vertexExample
               '=> $vertexId, '
               edgeCollectionRestriction'=>
               $edgeCollection));
      }

      public function getEdgesWithVertices(
         $vertexId,$edgeCollection,$toObjectFunc,
         $params,$orientation='any',$edgeExample=
         null){
            // $params['vertexId']=$vertexId;
            // $params['@edgeCollection']=
               $edgeCollection;
            // $params['orientation']=
               $orientation;
            // $params['edgeExample']=
               $edgeExample;

            // $params['option']=array('
               includeVertices'=>true);

            // $cursor = $this->executeStatement
               ("FOR e in EDGES(@@edgeCollection,
               @vertexId,@orientation,
               @edgeExample,@option) SORT e.
               vertex.@sortBy @direction LIMIT
               @skip,@amount RETURN e",
            // $params,true);

            // $result = $cursor->getAll();

            // $objects = array();
            // foreach ($cursor as $item) {
            //       $objects[]=$this->
               createObject($item->vertex,
               $toObjectFunc);
            // }
            // return new Paginator($objects,
```

```
            $cursor->getFullCount(), $params['
               skip'],$params['amount']);

            $params['vertexId']=$vertexId;
            $params['graph']="api";
            $params['@edgeCollection']=
               $edgeCollection;
            // $params['orientation']=
               $orientation;

            // $params['option']=array('
               includeVertices'=>true);

            $cursor = $this->executeStatement("
               for␣e␣in␣graph_edges(@graph,
               @vertexId,{
               edgeCollectionRestriction:
               @@edgeCollection,includeData:true
               })␣return␣(for␣u␣in␣tweets␣filter␣
               e._to␣==␣u._id␣sort␣u.@sortBy␣
               @direction␣limit␣@skip,@amount␣
               return␣u␣)"
                     ,$params,true);

            $objects = $this->createObject(
               $cursor->getAll(),
               $toObjectFunction);

            return new Paginator($objects,
               $cursor->getFullCount(), $skip,
               $amount);
      }

      public function executeStatement($query,
         $params, $fullCount=false){

            $stmt = new Statement($this->
               connection, array('query' =>
               $query, 'bindVars' => $params, '
```

```
            fullCount' => $fullCount));

        return $stmt->execute();

    }

    private function createObject($arrayData,
        $toObjectFunction){

        $fArrayToObject = new
            ArrayWithArangoKeyToObject();


        if(ArrayUtil::is_assoc_array(
            $arrayData)){
                return $fArrayToObject(
                    $arrayData,
                    $toObjectFunction);
        }else{
                $objects = array();
                foreach($arrayData as $item){
                        $objects[]=
                            $fArrayToObject(
                            $item->getAll(),
                            $toObjectFunction)
                            ;
                }
                return $objects;
        }

    }

    public function listVertex($collectionName,
        $toObjectFunction){
        $options['@collection'] =
            $collectionName;
        $cursor = $this->executeStatement("
            FOR u in @@collection RETURN u",
                $options,true);
```

```
            if($cursor->getCount()==0){
                    return [];
            }

            return $this->createObject($cursor->
               getAll(),$toObjectFunction);
    }

    public function listInAnInterval(
       $collectionName, $toObjectFunction,
       $options){
            $options['@collection'] =
               $collectionName;
            $cursor = $this->executeStatement("
               FOR␣u␣in␣@@collection␣SORT␣u.
               @sortBy␣@direction␣LIMIT␣@skip,
               @amount␣RETURN␣u",
                    $options, true);

            $objects = $this->createObject(
               $cursor->getAll(),
               $toObjectFunction);

            return new Paginator($objects,
               $cursor->getFullCount(), $options[
               'skip'], $options['amount']);

    }

    public function getByIds($idList,
       $collectionName,$toObjectFunction,$options
       ){
            $options['@collection'] =
               $collectionName;
            $options['idList']=$idList;
            $cursor = $this->executeStatement("
               FOR␣u␣in␣@idList␣FOR␣v␣in␣
               @@collection␣FILTER␣u␣==␣v._id␣
```

```
                SORT␣v.@sortBy␣@direction␣RETURN␣v
                ",
                        $options,true);

        if($cursor->getCount()==0){
                return [];
        }

        return $this->createObject($cursor->
            getAll(), $toObjectFunction);
}

public function getByIdsInAnInterval($idList,
    $collectionName,$toObjectFunction,$options
    ){
        $options['@collection'] =
            $collectionName;
        $options['idList'] = $idList;
        $cursor = $this->executeStatement("
            FOR␣u␣in␣@idList␣FOR␣v␣in␣
            @@collection␣FILTER␣u␣==␣v._id␣
            SORT␣v.@sortBy␣@direction␣LIMIT␣
            @skip,@amount␣RETURN␣v",
                $options,true);

        $objects = $this->createObject(
            $cursor->getAll(),
            $toObjectFunction);

        return new Paginator($objects,
            $cursor->getFullCount(), $options[
            'skip'], $options['amount']);
}

public function queryLikeInAnInterval(
    $collection,$attrib, $search,
    $toObjectFunc, $options){

        $options['@collection']=$collection;
```

```
            $options['attrib']=$attrib;
            $options['search']="%$search%";

            $cursor = $this->executeStatement("
               FOR␣u␣in␣@@collection␣FILTER␣LIKE(
               u.@attrib,@search,true)␣SORT␣u.
               @sortBy␣@direction␣LIMIT␣@skip,
               @amount␣RETURN␣u",$options,true);
            $objects = $this->createObject(
               $cursor->getAll(), $toObjectFunc);
            return new Paginator($objects,
               $cursor->getFullCount(), $options[
               'skip'], $options['amount']);


      }

      public function queryLike($collection,$attrib
         , $search, $toObjectFunc, $options){

            $options['@collection']=$collection;
            $options['attrib']=$attrib;
            $options['search']="%$search%";

            $cursor = $this->executeStatement("
               FOR␣u␣in␣@@collection␣FILTER␣LIKE(
               u.@attrib,@search,true)␣SORT␣u.
               @sortBy␣@direction␣RETURN␣u",
               $options);
            return $this->createObject($cursor->
               getAll(), $toObjectFunc);



      }

      public function getCommonNeighbors($vertex1,
         $vertex2, $options1=array(), $options2=
         array()){
```

```
            $params = array(
                    "graph" => "api",
                    "vertex1" => $vertex1,
                    "vertex2" => $vertex2,
                    "options1" => $options1,
                    "options2" => $options2
                    );

            $cursor = $this->executeStatement("
               FOR␣e␣IN␣GRAPH_COMMON_NEIGHBORS(
               @graph,␣@vertex1,@vertex2,
               @options1,@options2)␣RETURN␣e.
               neighbors", $params);

            $neighbors=array();

            foreach ($cursor->getAll() as $item)
               {

                    $neighbors = \array_merge(
                       $neighbors,$item->getAll()
                       );

            }
            return ArrayUtil::eliminates_repeated
               ($neighbors);

    }

    // public function importDocuments($objects,
       $collectionName,$toArrayFunc){

    //      $fObjectToArray = new
       ObjectToArrayWithArangoKey();

    //      $arrayData = array();
    //      foreach ($objects as $obj) {
    //              $arrayData[] =
       $fObjectToArray($obj, $toArrayFunc);
```

```
//      }


//      $query = "FOR u in @arrayData INSERT
   u IN @@collection RETURN NEW._id";
//      $params = array('arrayData'=>
   $arrayData, '@collection'=>$collectionName
   );

//      $cursor = $this->executeStatement(
   $query,$params);

//      return $cursor->getAll();

// }

// public function returnsExistingIds(
   $idsList,$collectionName){

//      $query = "FOR u in @idsList FOR v in
   @@collection FILTER u == v._id RETURN u";
//      $params = array('idsList'=>$idsList,
   '@collection'=>$collectionName);

//      return $this->executeStatement($query
   ,$params)->getAll();

// }

// public function createEdgeToManyFrom($to,
   $listFrom,$edgeCollection){
//      $query = "FOR u in @listFrom
//                              INSERT {
//                                      _to:
   @to,
//                                      _from
   : u
//                              }
//                              IN
```

```
                @@collection
            //                                  RETURN NEW";
            //      $params = array('to'=>$to,'listFrom
                '=>$listFrom, '@collection'=>
                $edgeCollection);

            //      $cursor = $this->executeStatement(
                $query,$params);

            //      return $cursor->getAll();

            // }

        }

}

<?php

namespace comunic\social_network_analyzer\model\repository\
    arango{

        use comunic\social_network_analyzer\model\repository\
            IDatasetsRepository;
        use comunic\social_network_analyzer\model\entity\
            mappers\ArrayToDataset;
        use comunic\social_network_analyzer\model\entity\
            mappers\DatasetToArray;


        class DatasetsRepository extends
            AbstractArangoRepository implements
            IDatasetsRepository{

                function __construct(){
                        parent::__construct();

                        $this->entityName = "datasets";
                }
```

```
public function insert($dataset, $projectId){
        $datasetVertex = $this->graphHandler
           ->createVertex($dataset, new
           DatasetToArray(), $this->
           entityName);

        $projectId = $this->buildId("projects
           ", $projectId);

        return $this->graphHandler->saveEdge(
           $projectId, $datasetVertex, "
           projects_datasets_has","
           projects_datasets_has");
}

public function filterByProject($projectId){
        $projectId = $this->buildId("projects
           ", $projectId);

        $edges = $this->graphHandler->
           getEdges($projectId,"
           projects_datasets_has");

        $datasets = array();
        foreach ($edges->getAll() as $edge) {
                $datasets[] = $this->
                   graphHandler->getVertex(
                   $edge->getTo() ,new
                   ArrayToDataset());
        }

        return $datasets;
}

public function update($dataset){
        $id = $this->buildId($this->
           entityName, $dataset->getId());
        return $this->graphHandler->
```

```
                        updateVertex($id, $dataset, new
                        DatasetToArray());
                }

                public function delete($id){
                        $id = $this->buildId($this->
                        entityName, $id);
                        return $this->graphHandler->
                        deleteVertex($id);
                }

                public function listAll(){

                        return $this->graphHandler->
                        listVertex($this->entityName,new
                        ArrayToDataset());
                }

                public function findById($id){
                        $id = $this->buildId($this->
                        entityName, $id);
                        return $this->graphHandler->getVertex
                        ($id, new ArrayToDataset());

                }

        }



}

<?php

namespace comunic\social_network_analyzer\model\repository\
   arango{

        use comunic\social_network_analyzer\model\util\
```

```
    StringUtil;
use comunic\social_network_analyzer\model\util\
    ArrayUtil;
use comunic\social_network_analyzer\model\entity\Word
    ;
use comunic\social_network_analyzer\model\entity\
    mappers\WordToArray;
use comunic\social_network_analyzer\model\entity\
    mappers\ArrayToWord;


class WordsRepository extends
    AbstractArangoRepository {

        function __construct(){
                parent::__construct();

                $this->entityName = "words";

        }



        public function importFromTweet($tweets){

                $edges = array();
                $newWords = array();
                $existingWords = $this->toAssocArray
                    ();
                $idAtual = $this->graphHandler->
                    lastInserted("words",new
                    ArrayToWord())->getId() + 1;

                foreach ($tweets as $tweet) {
                        $textWithoutPunctuation =
                            StringUtil::
                            removePunctuation($tweet->
                            getText(),"_-");
                        $wordsTw = ArrayUtil::
```

```
            eliminates_repeated(\
            explode("␣", \
            mb_strtolower(
            $textWithoutPunctuation)))
            ;

        $tweetId = $this->buildId("
            tweets",$tweet->getId());

        foreach($wordsTw as $word){

                if(isset(
                    $existingWords[
                    $word])){

                        $wordObj =
                            $existingWords
                            [$word];

                }else if(isset(
                    $newWords[$word]))
                    {

                        $wordObj =
                            $newWords[
                            $word];

                }else{

                        $wordObj =
                            new Word(
                            $idAtual
                            ++,$word);
                        $newWords[
                            $word]=
                            $wordObj;

                }
```

```
                    $wordId = $this->
                       buildId("words",
                       $wordObj->getId())
                       ;

                    $edges[] = $this->
                       graphHandler->
                       createEdge(
                       $tweetId,$wordId,"
                       tweets_words_contains
                       ");

                }


            }
            $this->graphHandler->importObjects("
               words",\array_values($newWords),
               new WordToArray());

            $slicer = ArrayUtil::slicer($edges
               ,20000);

            foreach ($slicer as $slice) {
                    $this->graphHandler->import("
                       tweets_words_contains",
                       $slice);
            }

            return $newWords;
```

```
// $edges = array();
// $words = array();
// $id = 0;
// foreach ($tweets as $tweet) {
//       $textWithoutPunctuation =
   StringUtil::removePunctuation(
   $tweet->getText(),"_-");
//       $wordsTw = ArrayUtil::
   eliminates_repeated(\explode(" ",
   $textWithoutPunctuation));

//       $tweetIdArango = $this->
   buildId("tweets",$tweet->getId());
//       $wordIdArango ="";

//       foreach($wordsTw as $word){

//             if(!isset($words[
   $word])){
//                   $wordObj =
   new Word($id++,$word);
//                   $words[$word
   ]= $wordObj;
//                   $wordIdArango
    = $this->buildId("words",$wordObj
   ->getId());
//             }else{
//                   $wordIdArango
    = $this->buildId("words",$words[
   $word]->getId());
//             }

//             $edges[] = $this->
   graphHandler->createEdge(
   $tweetIdArango,$wordIdArango,"
   tweets_words_contains");

//       }
```

```
// }
// $this->graphHandler->importObjects
   ("words",\array_values($words),new
    WordToArray());

// $slicer = ArrayUtil::slicer($edges
   ,20000);

// foreach ($slicer as $slice) {
//      $this->graphHandler->import("
   tweets_words_contains",$slice);
// }




// $textWithoutPunctuation =
   StringUtil::removePunctuation(
   $tweet->getText());
// $words = ArrayUtil::
   eliminates_repeated(\explode(" ",
   $textWithoutPunctuation));

// $wordObjects=array();

// foreach($words as $word){
//      $wordObjects[]= new Word(
   $word);
// }

// $this->graphHandler->importObjects
   ("words",$wordObjects,new
   WordToArray());

// $edges = array();
// $tweetIdArango = $this->buildId("
   tweets",$tweet->getId());
// foreach ($wordObjects as $word) {
```

```
            //      $wordIdArango = $this->
               buildId("words",$word->getId());
            //      $edges[] = $this->
               graphHandler->createEdge(
               $tweetIdArango,$wordIdArango,"
               tweets_words_contains",array("_key
               "=>$tweet->getId().$word->getId())
               );
            // }

            // $this->graphHandler->import("
               tweets_words_contains",$edges);
    }

    public function getWordsAsString(){
            $words = $this->graphHandler->
               listVertex($this->entityName,new
               ArrayToWord());
            $strWords = array();

            foreach ($words as $word) {
                    $strWords[]=$word->getWord();
            }

            return $strWords;
    }

    public function listAll(){
            return $this->graphHandler->
               listVertex($this->entityName,new
               ArrayToWord());
    }

    public function toAssocArray(){
            $words = $this->listAll();
            $assoc_words = array();

            foreach ($words as $word) {
```

```
                                $assoc_words[$word->getWord()
                                    ]=$word;
                        }

                        return $assoc_words;
                }



        }



}

<?php

namespace comunic\social_network_analyzer\model\repository\
    arango{

        use comunic\social_network_analyzer\model\repository\
            IRepositoryFactory;

        class ArangoRepository implements IRepositoryFactory{

                public function instantiateTweet(){

                        return new TweetsRepository();
                }
                public function instantiateUser(){

                }

                public function instantiateCategory(){

                        return new CategoriesRepository();
                }
```

```
                public function instantiateProject(){

                        return new ProjectsRepository();

                }
                public function instantiateDataset(){

                        return new DatasetsRepository();
                }

        }


}



<?php

namespace comunic\social_network_analyzer\model\repository\
   arango{

        use comunic\social_network_analyzer\model\repository\
           ITweetsRepository;
        use comunic\social_network_analyzer\model\entity\
           mappers\ArrayToTweet;
        use comunic\social_network_analyzer\model\entity\
           mappers\TweetToArray;
        use comunic\social_network_analyzer\model\util\
           StringUtil;
        use comunic\social_network_analyzer\model\util\
           ArrayUtil;
        use comunic\social_network_analyzer\model\entity\Word
           ;
        use comunic\social_network_analyzer\model\entity\
           mappers\WordToArray;
        use comunic\social_network_analyzer\model\entity\
           mappers\ArrayToWord;
        use comunic\social_network_analyzer\model\entity\
```

```
    Paginator;
use comunic\social_network_analyzer\model\repository\
    arango\mappers\ArrayWithArangoKeyToObject;
class TweetsRepository extends
    AbstractArangoRepository implements
    ITweetsRepository{


        function __construct(){
                parent::__construct();

                $this->entityName = "tweets";

        }

        public function import($tweets, $datasetId){
                // $slices = ArrayUtil::slicer(
                    $tweets,1000);

                // foreach ($slices as $slice) {
                        $this->graphHandler->
                            importObjects($this->
                            entityName,$tweets,new
                            TweetToArray());

                        $edges = array();
                        $datasetIdArango = $this->
                            buildId("datasets",
                            $datasetId);
                        foreach ($tweets as $tweet) {

                                $tweetIdArango =
                                    $this->buildId(
                                    $this->entityName,
                                    $tweet->getId());
                                $edges[] = $this->
                                    graphHandler->
                                    createEdge(
                                    $datasetIdArango,
```

```
                                $tweetIdArango,"
                                datasets_tweets_belong
                                ",array("_key"=>
                                $datasetId.$tweet
                                ->getId()));

                        }
                        $this->graphHandler->import("
                           datasets_tweets_belong",
                           $edges);
                // }

                $wordsRepo = new WordsRepository();
                // foreach ($tweets as $tweet) {

                return $wordsRepo->importFromTweet(
                   $tweets);

                // }

        }

        public function listAll(){
                return $this->graphHandler->
                   listVertex($this->entityName,new
                   ArrayToTweet());
        }

        public function findById($id){
                $id = $this->buildId($this->
                   entityName, $id);
                return $this->graphHandler->getVertex
                   ($id, new ArrayToTweet());

        }

        public function addToTheCategory($tweetId,
           $categoryId){
                return $this->
```

```
                    connectTweetWithCategory($tweetId,
                     $categoryId,"
                    tweets_categories_belong");
        }

        public function removeOfCategory($tweetId,
           $categoryId){
                return $this->
                    connectTweetWithCategory($tweetId,
                     $categoryId,"
                    tweets_categories_not_belong");
        }

        private function connectTweetWithCategory(
           $tweetId, $categoryId, $relationName){
                return $this->graphHandler->
                    createEdge($tweetId,$categoryId,
                    $relationName,$relationName);
        }

        public function findByCategory($datasetId,
           $category,$options){

                $query = 'LET␣params␣=␣{
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣filterVertices:@filter,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣visitor:␣@visitorFunc,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣visitorReturnsResults␣:␣true␣,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data:@data,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣maxDepth:3
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣FOR␣result␣IN␣graph_traversal("api",␣
   @category,␣"any",␣params)␣sort␣result.time␣asc
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣RETURN␣result';

                $params = array(
                        'filter' => "myfunctions::
                           pruneProject",
                        'visitorFunc' => "myfunctions
                           ::tweetsByCategory",
```

```
            'data' => \is_array(
               $datasetId) ? $datasetId :
                array($datasetId),
            'category' => $this->buildId(
               "categories",$category->
               getId()),


            );

$stmt = $this->graphHandler->
   executeStatement($query,$params);


$fArrayToObject = new
   ArrayWithArangoKeyToObject();

$data = $stmt->getAll();

return $data;


// if(ArrayUtil::is_assoc_array($data
   )){
//       return $fArrayToObject($data,
    new ArrayToTweet());
// }else{
//       $objects = array();
//       foreach($data as $item){
//                $objects[]=
   $fArrayToObject($item->getAll(),
   new ArrayToTweet());
//       }
//       return $objects;
// }



// $wordsRepo = new WordsRepository()
```

```
            ;


        // $words = $wordsRepo->listAll();

        // $matchWords = $category->
           matchWithKeywords($words);

        // $wordsIds = array();
        // foreach ($matchWords as $word) {
        //      $wordsIds[] = array("_id" =>
           $this->buildId("words", $word->
           getId()));
        // }

        // return $wordsIds;

        // $neighbors = $this->graphHandler->
           getCommonNeighbors(array("_id"=>
           $this->buildId("datasets",
           $datasetId)), $wordsIds);

        // if(\count($neighbors)==0){
        //      return [];
        // }

        // return $this->graphHandler->
           getByIds($neighbors,$this->
           entityName,new ArrayToTweet(),
           $options);
    }


    public function findbyCategoryInAnInterval(
       $datasetId, $category, $options){

        $wordsRepo = new WordsRepository();

        $words = $wordsRepo->listAll();
```

```
            $matchWords = $category->
               matchWithKeywords($words);

            $wordsIds = array();
            foreach ($matchWords as $word) {
                    $wordsIds[] = array("_id" =>
                       $this->buildId("words",
                       $word->getId()));
            }



            $neighbors = $this->graphHandler->
               getCommonNeighbors(array("_id"=>
               $this->buildId("datasets",
               $datasetId)), $wordsIds);

            return $this->graphHandler->
               getByIdsInAnInterval($neighbors,
               $this->entityName,new ArrayToTweet
               (),$options);

      }

      public function listByDataset($datasetId,
         $options){
            $datasetId = $this->buildId("datasets
               ", $datasetId);

            $edges = $this->graphHandler->
               getEdges($datasetId, "
               datasets_tweets_belong");

            $tweetsIds = array();
            foreach ($edges->getAll() as $edge) {
                    $tweetsIds[] =$edge->getTo();
            }
```

```
            return $this->graphHandler->getByIds(
               $tweetsIds,$this->entityName,new
               ArrayToTweet(),$options);

    }

    public function searchInTheTextInAnInterval(
       $search,$options){
            return $this->graphHandler->
               queryLikeInAnInterval($this->
               entityName,"text", $search, new
               ArrayToTweet(), $options);
    }

    public function searchInTheText($search,
       $options){
            return $this->graphHandler->queryLike
               ($this->entityName,"text", $search
               , new ArrayToTweet(), $options);
    }

    public function listByDatasetInAnInterval(
       $datasetId, $options){
            $datasetId = $this->buildId("datasets
               ", $datasetId);

            $edges = $this->graphHandler->
               getEdges($datasetId, "
               datasets_tweets_belong");

            $tweetsIds = array();
            foreach ($edges->getAll() as $edge) {
                    $tweetsIds[] =$edge->getTo();
            }

            return $this->graphHandler->
               getByIdsInAnInterval($tweetsIds,
               $this->entityName,new ArrayToTweet
               (),$options);
```

```
                        // $datasetId = $this->buildId("
                           datasets", $datasetId);

                        // return $this->graphHandler->
                           getEdgesWithVertices($datasetId,"
                           datasets_tweets_belong",new
                           ArrayToTweet(),$options);

                }



        }



}

<?php

namespace comunic\social_network_analyzer\model\repository\
   arango{

        require "vendor/autoload.php";

        use triagens\ArangoDb\Connection;
        use triagens\ArangoDb\ConnectionOptions;
        use triagens\ArangoDb\UpdatePolicy;

        class ConnectionArango{

                private $connection;

                function __construct(){

                        $connectionOptions =array(
                            // server endpoint to connect to
```

```
    ConnectionOptions::
        OPTION_ENDPOINT => 'tcp
        ://127.0.0.1:8529',
// authorization type to use (
   currently supported: 'Basic')
    ConnectionOptions::
        OPTION_AUTH_TYPE => 'Basic
        ',
// user for basic authorization
    ConnectionOptions::
        OPTION_AUTH_USER => 'root'
        ,
// password for basic
   authorization
    ConnectionOptions::
        OPTION_AUTH_PASSWD => '',
// connection persistence on
   server. can use either 'Close'
    (one-time connections) or '
   Keep-Alive' (re-used
   connections)
    ConnectionOptions::
        OPTION_CONNECTION => '
        Close',
// connect timeout in seconds
    ConnectionOptions::
        OPTION_TIMEOUT => 30,
// whether or not to reconnect
   when a keep-alive connection
   has timed out on server
    ConnectionOptions::
        OPTION_RECONNECT => true,
// optionally create new
   collections when inserting
   documents
    ConnectionOptions::
        OPTION_CREATE => true,
// optionally create new
   collections when inserting
```

```
                                    documents
                                     ConnectionOptions::
                                        OPTION_UPDATE_POLICY =>
                                        UpdatePolicy::LAST,
                                     ConnectionOptions::
                                        OPTION_DATABASE
                                        => 'api'
                                     );

$this->connection = new Connection($connectionOptions);

}

public function getConnection(){
        return $this->connection;
}


}

}


<?php

namespace comunic\social_network_analyzer\model\repository\
   arango{

        use comunic\social_network_analyzer\model\repository\
           ICategoriesRepository;
        use comunic\social_network_analyzer\model\entity\
           mappers\ArrayToCategory;
        use comunic\social_network_analyzer\model\entity\
           mappers\CategoryToArray;

        class CategoriesRepository extends
           AbstractArangoRepository implements
           ICategoriesRepository{
```

```
function __construct(){
        parent::__construct();

        $this->entityName = "categories";

}

public function insert($category, $projectId)
   {

        $wordsRepo = new WordsRepository();

        $words = $wordsRepo->listAll();



        $catId = $this->graphHandler->
           createVertex($category, new
           CategoryToArray(), $this->
           entityName);

        $this->graphHandler->saveEdge($this->
           buildId("projects",$projectId),
           $catId,"projects_categories_belong
           ","projects_categories_belong");

        if(!empty($words)){
                $edges = array();
                $matchWords = $category->
                   matchWithKeywords($words);
                foreach ($matchWords as $word
                   ){

                        $edges[] = $this->
                           graphHandler->
                           createEdge($catId,
                           "words/".$word->
                           getId()."
                           categories_words_contains
```

```
                                    ");

                        }

                        $this->graphHandler->import("
                            categories_words_contains"
                            ,$edges);
                }

        }

        public function update($category){
                $id = $this->buildId($this->
                    entityName, $category->getId());
                return $this->graphHandler->
                    updateVertex($id, $category, new
                    CategoryToArray());
        }

        public function filterByProject($projectId){
                $projectId = $this->buildId("projects
                    ", $projectId);

                $edges = $this->graphHandler->
                    getEdges($projectId,"
                    projects_categories_belong");

                $categories = array();
                foreach ($edges->getAll() as $edge) {
                        $categories[] = $this->
                            graphHandler->getVertex(
                            $edge->getTo() ,new
                            ArrayToCategory());
                }

                return $categories;
        }

        public function delete($id){
```

```
                        $id = $this->buildId($this->
                           entityName, $id);
                        return $this->graphHandler->
                           deleteVertex($id);
                }

                public function listAll(){

                        return $this->graphHandler->
                           listVertex($this->entityName, new
                           ArrayToCategory());
                }

                public function findById($id){
                        $id = $this->buildId($this->
                           entityName, $id);
                        return $this->graphHandler->getVertex
                           ($id, new ArrayToCategory());

                }

        }



}

<?php

namespace comunic\social_network_analyzer\model\repository\
   fake {

    use comunic\social_network_analyzer\model\repository\
       ITweetsRepository;

    /**
     * Description of tweetsrepository
     *
```

```
     * @author jean
     */
    class TweetsRepository extends DiskRepository implements
       ITweetsRepository {



        function __construct() {
            parent::__construct('tweets');
        }


        public function listByCategory($category) {

            return $this->listAll();

        }

    }

}<?php


namespace comunic\social_network_analyzer\model\repository\
   fake {

    use comunic\social_network_analyzer\model\repository\
       IUsersRepository;

    /**
     * Description of userrepository
     *
     * @author jean
     */
    class UserRepository extends DiskRepository implements
       IUsersRepository {

         function __construct() {
            parent::__construct('users');
```

```
        }

    }

}<?php

namespace comunic\social_network_analyzer\model\repository\
   fake {

    use comunic\social_network_analyzer\model\repository\
       IRepositoryFactory;

    /**
     * Description of fakerepository
     *
     * @author jean
     */
    class FakeRepository implements IRepositoryFactory {

        public function instantiateCategory() {
            return new CategoriesRepository();
        }

        public function instantiateTweet() {
            return new TweetsRepository();
        }

        public function instantiateUser() {
            return new UserRepository();
        }

//put your code here
    }

}<?php


namespace comunic\social_network_analyzer\model\repository\
   fake {
```

```
    use comunic\social_network_analyzer\model\repository\
       ICategoriesRepository;

    /**
     * Description of categoriesrepository
     *
     * @author jean
     */
    class CategoriesRepository extends DiskRepository
       implements ICategoriesRepository {


        function __construct() {
            parent::__construct('categories');
        }

    }

}<?php

namespace comunic\social_network_analyzer\model\repository\
   fake {


    class DiskRepository {

        private $data;
        private $dataFile;

        public function __construct($dataFile) {
            $this->dataFile = __DIR__ . DIRECTORY_SEPARATOR .
                $dataFile.'.fakedata';
        }

        public function delete($id) {
            $this->loadData();
            $this->findById($id);
            unset($this->data[$id]);
```

```
        $this->storeData();
    }

    private function loadData() {

        if (\file_exists($this->dataFile)) {
            $this->data = \file_get_contents($this->
               dataFile);
            $this->data = \unserialize($this->data);
        } else {
            $this->data = array();
        }
    }

    private function storeData() {
        \file_put_contents($this->dataFile, \serialize(
           $this->data));
    }

    public function findById($id) { {
            $this->loadData();
            if (\array_key_exists($id, $this->data)) {
                return $this->data[$id];
            }

            throw new \Exception("Nao␣existe␣entidade␣com
               ␣id␣=␣$id");
        }
    }

    public function insert($entity) {
        $this->loadData();
        $id = 0;

        if (\count($this->data) != 0) {

            $entityKeys = \array_keys($this->data);
            \sort($entityKeys);
            $id = \array_pop($entityKeys) + 1;
```

```
            }


            $entity->setId($id);
            $this->data[$id] = $entity;
            $this->storeData();
        }

        public function listAll() {
            $this->loadData();
            return $this->data;
        }

        public function update($entity) {
            $this->loadData();
            $id = $entity->getId();
            $this->findById($id);
            $this->data[$id] = $entity;
            $this->storeData();
        }

    }

}<?php


namespace comunic\social_network_analyzer\model\repository{
/**
 * class IRepositoryFactory
 *
 */
interface IRepositoryFactory
{

  /** Aggregations: */

  /** Compositions: */

  /**
```

```
     *
     *
     * @return  \comunic\social_network_analyzer\model\
        repository\ITweetsRepository
     * @access public
     */
    public function instantiateTweet();

    /**
     *
     *
     * @return \comunic\social_network_analyzer\model\
        repository\IUsersRepository
     * @access public
     */
    public function instantiateUser();

    /**
     *
     *
     * @return  \comunic\social_network_analyzer\model\
        repository\ICategoriesRepository
     * @access public
     */
    public function instantiateCategory();

      /**
     *
     *
     * @return  \comunic\social_network_analyzer\model\
        repository\IProjectsRepository
     * @access public
     */
    public function instantiateProject();

      /**
     *
     *
     * @return  \comunic\social_network_analyzer\model\
```

```
      repository\IDatasetsRepository
    * @access public
    */
   public function instantiateDataset();



}

} // end of IRepositoryFactory

<?php

namespace comunic\social_network_analyzer\model\repository{
/**
 * class IDatasetsRepository
 *
 */
interface IDatasetsRepository
{

   /** Aggregations: */

   /** Compositions: */

   /**
    *
    *
    * @param comunic::social_network_analyzer::model::entity::
      Dataset dataset
    * @return void
    * @access public
    */
   public function insert( $dataset, $projectId);

   /**
    *
    *
    * @param comunic::social_network_analyzer::model::entity::
```

```
     Dataset dataset
    * @return void
    * @access public
    */
  public function update( $dataset, $projectId);

  /**
   *
   *
   * @param int id
   * @return void
   * @access public
   */
  public function delete( $id, $projectId);

  /**
   *
   *
   * @return void
   * @access public
   */
  public function listAll($projectId);

  /**
   *
   *
   * @param int id
   * @return void
   * @access public
   */
  public function findById( $id);


}


} // end of IDatasetsRepository

<?php
```

```
namespace comunic\social_network_analyzer\model\repository\
   mongo{

        use \comunic\social_network_analyzer\model\repository
           \IProjectsRepository;
        use comunic\social_network_analyzer\model\entity\
           mappers\ArrayToProject;
        use comunic\social_network_analyzer\model\entity\
           mappers\ProjectToArray;
        use comunic\social_network_analyzer\model\util\
           MongoUtil;
        use comunic\social_network_analyzer\model\exception\
           UniqueConstraintException;

        class ProjectsRepository implements
           IProjectsRepository{

              private $collection;

              public function __construct($connectionType){


                      $conn = new ConnectionMongo();
                      $conn = $conn->getConnection(
                         $connectionType);
                      $this->collection = $conn->
                         selectCollection("projects");

                      $this->collection->createIndex(["name
                         " => 1],["unique" =>true]);
              }

              public function insert($project){

                      try {

                              $toArray = new ProjectToArray
                                 ();
```

```
                $arrayData = MongoUtil::
                    includeMongoIdObject(
                    $toArray($project));

                return $this->collection->
                    save($arrayData, $options=
                    array("w" =>1));

            } catch (\MongoDuplicateKeyException
                $e) {

                throw new
                    UniqueConstraintException(
                    "Already␣exists␣a␣project␣
                    with␣name␣" . $project->
                    getName());

            }catch (\MongoException $e) {

                echo $e->getMessage();
            }
        }

        public function listAll(){

            $cursor=$this->collection->find(
                $query=array(),$fields=array());

            $outputObjects=array();

            $toProject = new ArrayToProject();

            foreach ($cursor as $item) {

                $outputObjects[]=$toProject(
                    MongoUtil::
                    removeMongoIdObject($item)
                    );
```

```
            }

            return $outputObjects;

    }

    public function findById($id){

            $arrayData=$this->collection->findOne
               (array('_id' => new \MongoId($id))
               ,$fields=array());

            $toProject = new ArrayToProject();

            return $toProject(MongoUtil::
               removeMongoIdObject($arrayData));

    }

    public function delete($id){

            try {

                    return $this->collection->
                       remove(array('_id' => new
                       \MongoId($id)), $options=
                       array());

            } catch (\MongoCursorException $e) {

                    echo $e->getMessage();
            }
    }

    public function update($project){
            try {


                    $toArray = new ProjectToArray
```

```
                                    ();

                                $arrayData = MongoUtil::
                                    includeMongoIdObject(
                                    $toArray($project));

                                return $this->collection->
                                    save($arrayData, $options=
                                    array());

                            }catch (\MongoDuplicateKeyException
                                $e) {

                                throw new
                                    UniqueConstraintException(
                                    "Already exists a project
                                    with name " . $project->
                                    getName());

                            }catch (\MongoException $e) {

                                echo $e->getMessage();
                            }
                    }



            }

}

<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 08/08/16
 * Time: 15:48
 */
```

```
namespace comunic\social_network_analyzer\model\repository\
   mongo;


use comunic\social_network_analyzer\model\entity\mappers\
   ArrayToMatrix;
use comunic\social_network_analyzer\model\entity\mappers\
   MatrixToArray;
use comunic\social_network_analyzer\model\util\MongoUtil;
use comunic\social_network_analyzer\model\exception\
   UniqueConstraintException;

class MatricesRepository
{

    private $collection;

    function __construct($connectionType){
        $conn = new ConnectionMongo();
        $conn = $conn->getConnection($connectionType);
        $this->collection = $conn->selectCollection("projects
           ");

    }

    public function insert($matrix, $projectId){

        try {

            if(!$this->isUnique($matrix->getName())){
                throw new UniqueConstraintException("Already␣
                   exists␣a␣matrix␣with␣name␣" . $matrix->
                   getName());
            }

            $toArray = new MatrixToArray();

            $arrayData = MongoUtil::includeMongoIdObject(
               $toArray($matrix));
```

```
            return $this->collection->update(array('_id' =>
                new \MongoId($projectId)), array('$addToSet'
                => array("matrices" => $arrayData)), $options=
                array());

        }catch (\MongoException $e) {

            echo $e->getMessage();
        }
    }

    public function update($matrix, $projectId){

        try {

            if(!$this->isUnique($matrix->getName(),$matrix->
                getId())){
                 throw new UniqueConstraintException("Already␣
                    exists␣a␣matrix␣with␣name␣" . $matrix->
                    getName());
            }

            $toArray = new MatrixToArray();

            $arrayData = MongoUtil::includeMongoIdObject(
                $toArray($matrix));

            return $this->collection->update(['$and' => [['
                _id' => new \MongoId($projectId)],["matrices.
                _id" => new \MongoId($matrix->getId())]]], ['
                $set' => ["matrices.$" => $arrayData]],
                $options=array());
        }catch (\MongoException $e) {

            echo $e->getMessage();
        }

    }
```

```
public function delete($id, $projectId){

    return $this->collection->update(array('_id' => new \
       MongoId($projectId)), array('$pull' => array("
       matrices" => array("_id" => new \MongoId($id)))),
       $options=array());

}

public function findById($id){

    $arrayData=$this->collection->findOne(array('matrices
       ._id' => new \MongoId($id)), array("matrices.$" =>
        true, "_id" => false), $fields=array());

    $toMatrix = new ArrayToMatrix();

    return $toMatrix(MongoUtil::removeMongoIdObject(
       $arrayData["matrices"][0]));

}

public function isUnique($name,$id=null){
    /*
     * se id for nulo, opera ao de insert
     * so verifica se existe outro nome de matriz igual
     *
     *
     * se for update, primeiro verifica se o nome foi
        trocado
     * se foi trocado, verifica se o mesmo e repetido ou
        nao
     *
     * */

    $cursorFindByName = $this->collection->find(array('
       matrices.name' => $name), array("matrices.$" =>
       true, "_id" => false));
```

```
        if($cursorFindByName->count()==0){
            return true;
        }
        if(!\is_null($id)){
            $cursorFindByNameAndId = $this->collection->find
               (['matrices' => ['$elemMatch' => ['_id' => new
                \MongoId($id),"name" => $name]]],["matrices.$
               " => true, "_id" => false]);

            if($cursorFindByNameAndId->count()==1){
                return true;
            }
        }
        return false;

    }


}<?php

namespace comunic\social_network_analyzer\model\repository\
   mongo{

    use \comunic\social_network_analyzer\model\repository\
       IUsersRepository;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToUser;
    use comunic\social_network_analyzer\model\entity\mappers\
       UserToArray;
    use comunic\social_network_analyzer\model\util\MongoUtil;
    use comunic\social_network_analyzer\model\exception\
       NoResultException;
    use comunic\social_network_analyzer\model\exception\
       UniqueConstraintException;

    class UsersRepository implements IUsersRepository{

        private $collection;
```

```
public function __construct($connectionType){


    $conn = new ConnectionMongo();
    $conn = $conn->getConnection($connectionType);
    $this->collection = $conn->selectCollection("
       users");
    $this->collection->createIndex(["email" => 1],["
       unique" =>true]);
}

public function insert($user){

    try {
        $toArray = new UserToArray();

        $arrayData = MongoUtil::includeMongoIdObject(
           $toArray($user));

        return $this->collection->save($arrayData,
           $options=array());

    }catch (\MongoDuplicateKeyException $e) {

        throw new UniqueConstraintException("Already␣
           exists␣an␣user␣with␣email␣" . $user->
           getEmail());

    } catch (\MongoCursorException $e) {

        echo $e->getMessage();

    }catch (\MongoException $e) {

        echo $e->getMessage();
    }
}
```

```
public function listAll(){

    $cursor=$this->collection->find($query=array(),['
       password' => false]);

    $outputObjects=array();

    $toUser = new ArrayToUser();

    foreach ($cursor as $item) {

        $outputObjects[]=$toUser(MongoUtil::
           removeMongoIdObject($item));
    }

    return $outputObjects;

}

public function findById($id, $fields=[]){

    $arrayData=$this->collection->findOne(array('_id'
        => new \MongoId($id)),$fields);

    $toUser = new ArrayToUser();

    return $toUser(MongoUtil::removeMongoIdObject(
       $arrayData));

}

public function delete($id){

    try {

        return $this->collection->remove(array('_id'
           => new \MongoId($id)), $options=array());

    } catch (\MongoCursorException $e) {
```

```
            echo $e->getMessage();
        }
    }

    public function update($user){
        try {

            $toArray = new UserToArray();

            $arrayData = MongoUtil::includeMongoIdObject(
               $toArray($user));

            return $this->collection->save($arrayData,
               $options=array());

        }catch (\MongoDuplicateKeyException $e) {

            throw new UniqueConstraintException("Already␣
               exists␣an␣user␣with␣email␣" . $user->
               getEmail());

        } catch (\MongoCursorException $e) {

            echo $e->getMessage();

        }catch (\MongoException $e) {

            echo $e->getMessage();
        }
    }


    public function findByEmail($email,$fields=array()){

        $arrayData=$this->collection->findOne(array('
           email' => $email),$fields);

        if(\is_null($arrayData)){
```

```
                throw new NoResultException("Nenhum␣usu rio␣
                    encontrado␣com␣o␣email␣" + $email);
            }

            $toUser = new ArrayToUser();

            return $toUser(MongoUtil::removeMongoIdObject(
                $arrayData));

        }

        public function countUsers(){
            $cursor=$this->collection->find();
            return $cursor->count();
        }



    }

}

<?php


namespace comunic\social_network_analyzer\model\repository\
    mongo{

    use comunic\social_network_analyzer\model\repository\
        IRepositoryFactory;

    class MongoRepository implements IRepositoryFactory{

        private $connectionType;

        public function __construct($connectionType){
            $this->connectionType = $connectionType;
        }
```

```
        public function instantiateCategory() {
            return new CategoriesRepository($this->
               connectionType);
        }

        public function instantiateTweet() {
            return new TweetsRepository($this->connectionType
               );
        }

        public function instantiateUser() {
            return new UsersRepository($this->connectionType)
               ;
        }

        public function instantiateProject() {
            return new ProjectsRepository($this->
               connectionType);
        }

        public function instantiateDataset() {
            return new DatasetsRepository($this->
               connectionType);
        }

        public function instantiateMatrix(){
            return new MatricesRepository($this->
               connectionType);
        }



    }



}
```

```
<?php

namespace comunic\social_network_analyzer\model\repository\
   mongo{

use \comunic\social_network_analyzer\model\entity\Paginator;
use comunic\social_network_analyzer\model\repository\mongo\
   mappers\ArrayWithMongoIdToObject;
use comunic\social_network_analyzer\model\repository\mongo\
   mappers\ObjectToArrayWithMongoId;
use comunic\social_network_analyzer\model\util\ArrayUtil;

    class MongoCollectionHandler{

        private $collection;

        function __construct($collectionName, $connectionType
           ){

            $conn = new ConnectionMongo();
            $conn = $conn->getConnection($connectionType);
            $this->collection = $conn->selectCollection(
               $collectionName);

        }

        public function find($toObjectFunction,$query = array
           (), $fields=array()){
            $cur=$this->collection->find($query,$fields);
            $outputObject=$this->cursorToObjectArray(
               $toObjectFunction,$cur);
            return $outputObject;

        }

        public function count($query){
            return $this->collection->count($query);
        }
```

```
private function cursorToObjectArray(
   $toObjectFunction,$cursor){
    $outputObjects=array();

    $fArrayWithMongoIdToObj = new
       ArrayWithMongoIdToObject();

    foreach ($cursor as $item) {
        $outputObjects[]=$fArrayWithMongoIdToObj(
           $item, $toObjectFunction);
    }

    return $outputObjects;

}

public function findInAnInterval($indPage, $amount,
   $toObjectFunction,$query = array(), $fields=array
   ()){
    // $cursor = $this->collection->find($query,
       $fields);
    // $cursor = $cursor->skip($initial);
    // $cur=$cursor->limit($final);
    // $outputObject=$this->cursorToObjectArray(
       $toObjectFunction,$cur);
    // return $outputObject;
    $cursor = $this->collection->find($query,$fields)
       ;
    $count = $cursor->count();
    $cursor->skip(($indPage-1 ) * $amount);
    $cursor->limit($amount);
    $outputObject = $this->cursorToObjectArray(
       $toObjectFunction, $cursor);

    return new Paginator($outputObject, $count,
       $indPage, $amount);
}
```

```
//retorna um array com os dados de um documento
public function findOne($toObjectFunction,$query =
   array(), $fields=array()){
    $arrayData=$this->collection->findOne($query,
       $fields);

    if(\count($arrayData)==1 && ArrayUtil::
       is_assoc_array($arrayData)){

        $arrayData = \array_values($arrayData)[0][0];
    }

    $fArrayWithMongoIdToObj = new
       ArrayWithMongoIdToObject();

    return $fArrayWithMongoIdToObj($arrayData,
       $toObjectFunction);

}

public function save($obj, $toArrayDataFunction,
   $options=array()){

    try {
        $fObjToArrayWithMongoId = new
           ObjectToArrayWithMongoId();

       $arrayData=$fObjToArrayWithMongoId($obj,
          $toArrayDataFunction);

        return $this->collection->save($arrayData,
           $options);

    } catch (\MongoCursorException $e) {

        echo $e->getMessage();

    }catch (\MongoException $e) {
```

```
            echo $e->getMessage();
        }
    }


    public function update($criteria, $operator,
       $attribute, $obj, $toArrayDataFunction, $options=
       array()){


        try {
            $fObjToArrayWithMongoId = new
               ObjectToArrayWithMongoId();

           $arrayData=$fObjToArrayWithMongoId($obj,
              $toArrayDataFunction);


            return $this->collection->update($criteria,
               array($operator => array($attribute =>
               $arrayData)) , $options);



        } catch (\MongoCursorException $e) {

            echo $e->getMessage();

        }catch (\MongoException $e) {

            echo $e->getMessage();
        }

    }



    public function delete($criteria=array(), $options=
       array()){
```

```
            try {

                return $this->collection->remove($criteria,
                   $options);

            } catch (\MongoCursorException $e) {

                echo $e->getMessage();
            }
        }


    }
}

<?php

namespace comunic\social_network_analyzer\model\repository\
   mongo{

    use comunic\social_network_analyzer\model\repository\
       IDatasetsRepository;
    use comunic\social_network_analyzer\model\entity\mappers\
       DatasetToArray;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToDataset;
    use comunic\social_network_analyzer\model\util\MongoUtil;
    use comunic\social_network_analyzer\model\exception\
       UniqueConstraintException;

    class DatasetsRepository implements IDatasetsRepository{

        private $collection;
        private $connectionType;

        function __construct($connectionType){
            $this->connectionType = $connectionType;
            $this->setCollection("projects");
```

```
}

public function setCollection($collName){
    $conn = new ConnectionMongo();
    $conn = $conn->getConnection( $this->
       connectionType);
    $this->collection = $conn->selectCollection(
       $collName);
}

public function insert($object,$projectId){

    try {

        if(!$this->isUnique($object->getName())){
            throw new UniqueConstraintException("
               Already␣exists␣a␣category␣with␣name␣"
               . $object->getName());
        }

        $toArray = new DatasetToArray();

        $arrayData = MongoUtil::includeMongoIdObject(
           $toArray($object));

        return $this->collection->update(array('_id'
           => new \MongoId($projectId)), array('
           $addToSet' => array("datasets" =>
           $arrayData)), $options=array());

    }
    catch (\MongoDuplicateKeyException $e) {

        throw new UniqueConstraintException("Already␣
           exists␣a␣dataset␣with␣name␣" . $object->
           getName());

    }catch (\MongoException $e) {
```

```
            echo $e->getMessage();
        }

    }



    public function update($object, $projectId){

        try {

            if(!$this->isUnique($object->getName(),
               $object->getId())){
                throw new UniqueConstraintException("
                   Already␣exists␣a␣dataset␣with␣name␣" .
                    $object->getName());
            }

            $toArray = new DatasetToArray();

            $arrayData = MongoUtil::includeMongoIdObject(
               $toArray($object));

            return $this->collection->update(['$and' =>
               [['_id' => new \MongoId($projectId)],["
               datasets._id" => new \MongoId($object->
               getId())]]], ['$set' => ["datasets.$" =>
               $arrayData]], $options=array());
        }catch (\MongoException $e) {

            echo $e->getMessage();
        }

    }

    public function delete($id, $projectId){

        $this->collection->update(array('_id' => new \
```

```
        MongoId($projectId)), array('$pull' => array("
        datasets" => array("_id" => new \MongoId($id))
        )), $options=array());
    $this->setCollection("tweets");
    return $this->collection->remove(['idDataset' =>
        $id], []);
}



public function isUnique($name,$id=null){
    /*
     * se id for nulo, opera ao de insert
     * so verifica se existe outro nome de categoria
        igual
     *
     *
     * se for update, primeiro verifica se o nome foi
        trocado
     * se foi trocado, verifica se o mesmo e repetido
        ou nao
     *
     * */

    $cursorFindByName = $this->collection->find(array
        ('datasets.name' => $name), array("datasets.$"
         => true, "_id" => false));

    if($cursorFindByName->count()==0){
        return true;
    }
    if(!\is_null($id)){
        $cursorFindByNameAndId = $this->collection->
            find(['datasets' => ['$elemMatch' => ['_id
            ' => new \MongoId($id),"name" => $name
            ]]],["datasets.$" => true, "_id" => false
            ]);

        if($cursorFindByNameAndId->count()==1){
```

```
                    return true;
                }
            }
            return false;

        }

     public function findById($id){

        $arrayData=$this->collection->findOne(array('datasets
           ._id' => new \MongoId($id)), array("datasets.$" =>
            true, "_id" => false), $fields=array());

        $toDataset = new ArrayToDataset();

        return $toDataset(MongoUtil::removeMongoIdObject(
           $arrayData["datasets"][0]));
    }

    public function listAll($projectId){

        return $this->mongoch->find(new ArrayToDataset());
    }

}

}


<?php

namespace comunic\social_network_analyzer\model\repository\
   mongo{



class ConnectionMongo{

    //TODO criar construtor p/ receber por par metros dados
```

```
   de conex o
//talvez singleton

// private $dbUser = "";
// private $dbPwd = "";
// private $dbName = 'unity_test';
// private $dbPort = "27017";
// private $dbHost ="localhost";

private $connectionInfo = array(
    "development" => array(
        "dbName" => "development"
        ),
    "workshoprio" => array("dbName" => "workshoprio"),

    "comunic" => array(
        "dbName" => "teste"
        )

    );

private function getURL(){
     return "mongodb://".$this->dbUser.":".$this->dbPwd."
        @".$this->dbHost.":".$this->dbPort;
}

public function getConnection($type){

    try {

       //$conn = new \Mongo($this->getURL());
        $conn = new \MongoClient();
        return $conn->selectDB($this->connectionInfo[
           $type]["dbName"]);

    } catch (\MongoConnectionException $e) {
        echo $e->getMessage();
    }
}
```

```
}

}
```

```
<?php

namespace comunic\social_network_analyzer\model\repository\
   mongo {

    use \comunic\social_network_analyzer\model\repository\
       mongo\MongoCollectionHandler;
    use \comunic\social_network_analyzer\model\repository\
       ITweetsRepository;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToTweet;
    use comunic\social_network_analyzer\model\entity\mappers\
       TweetToArray;
    use comunic\social_network_analyzer\model\util\StringUtil
       ;
    use comunic\social_network_analyzer\model\util\MongoUtil;
    use comunic\social_network_analyzer\model\entity\
       Paginator;



    class TweetsRepository implements ITweetsRepository {

        private $collection;

        function __construct($connectionType) {

            $conn = new ConnectionMongo();
            $conn = $conn->getConnection($connectionType);
            $this->collection = $conn->selectCollection("
               tweets");
```

```
}

public function insert($tweet) {


    try {
        $toArray = new TweetToArray();

        $arrayData = MongoUtil::includeMongoIdObject(
           $toArray($tweet));

        return $this->collection->save($arrayData,
           $options=array());

    } catch (\MongoCursorException $e) {

        echo $e->getMessage();

    }catch (\MongoException $e) {

        echo $e->getMessage();
    }


}

public function update($tweet) {


    try {
        $toArray = new TweetToArray();

        $arrayData = MongoUtil::includeMongoIdObject(
           $toArray($tweet));

        return $this->collection->save($arrayData,
           $options=array());

    } catch (\MongoCursorException $e) {
```

```
            echo $e->getMessage();

        }catch (\MongoException $e) {

            echo $e->getMessage();
        }


    }

    public function findById($id) {

        $arrayData=$this->collection->findOne(array('_id'
            => new \MongoId($id)),$fields=array());

        $toTweet = new ArrayToTweet();

        return $toTweet(MongoUtil::removeMongoIdObject(
           $arrayData));
    }

    public function listAll($options){

        $cursor = $this->collection->find($query=array(),
           $fields=array());
        return $this->getTweets($cursor,$options);

    }

    public function listByDataset($idDataset,
       $includeRepeated, $options){

        if(!$includeRepeated){
            $cursor=$this->collection->find(['$and'=>[["
               idDataset" => ['$in' => $idDataset]],["
               existsSimilarPostedPreviously"=>false],["
               isVisible"=>true]]],$fields=array());
```

```
        }else{
            $cursor=$this->collection->find(['$and'=>[["
                idDataset" => ['$in' => $idDataset]],["
                isVisible"=>true]]],$fields=array());

        }

        return $this->getTweets($cursor,$options);

    }

    private function getTweets($cursor,$options){

        \extract($options);

        $count = $cursor->count();
        $cursor->sort(array($sortBy => $direction));
        $paginate = false;

        if(isset($skip) && isset($amount)){
            $paginate = true;
            $cursor->skip($skip);
            $cursor->limit($amount);
        }

        $outputObjects=array();
        $toTweet = new ArrayToTweet();
        foreach ($cursor as $item) {

            $outputObjects[]=$toTweet(MongoUtil::
                removeMongoIdObject($item));
        }

        if($paginate){
            return new Paginator($outputObjects, $count,
                $skip, $amount);
        }

        return $outputObjects;
```

```
}

public function findbyCategory($idDataset, $kwAsRegex
   , $options){

    $mongoRegex = array();
    foreach ($kwAsRegex as $kw) {
        $mongoRegex[]= new \MongoRegex(StringUtil::
           removeAccents($kw));
    }

    $cursor = $this->collection->find(array('$and' =>
        array(array('idDataset' => array('$in' =>
       $idDataset)),array('normalizedText' => array('
       $in' => $mongoRegex)),["isVisible"=>true])),
       $fields=array());

    return $this->getTweets($cursor,$options);
}


public function searchInTheText($idDataset,$term,
   $options){

    $cursor = $this->collection->find(['$and' => [
       array('idDataset' => array('$in' => $idDataset
       )),["isVisible"=>true],array('normalizedText'
       => array('$regex' => new \MongoRegex("/$term/i
       "))))]]);

    return $this->getTweets($cursor,$options);
}

public function searchSimilar($idDataset,$idTweet,
   $term, $options){

    if($this->findById($idTweet)->
       getExistsSimilarPostedPreviously()){
```

```
            return [];
        }

        $cursor = $this->collection->find(['$and' => [
            array('idDataset' => array('$in' => $idDataset
            )),["_id"=>['$ne'=> new \MongoId($idTweet)]],
            array('cleartext' => $term)]]);

        return $this->getTweets($cursor,$options);
    }


    public function countByDataset($idDataset){
        $cursor=$this->collection->find(array("idDataset"
            => $idDataset),$fields=array());
        return $cursor->count();
    }

    public function existSimilar($idDataset, $cleartext){
        $cursor=$this->collection->find(['$and' =>[["
            idDataset" => $idDataset],["cleartext" =>
            $cleartext]]]);

        return $cursor->count() > 1;
    }

    //lista tweets marcados por um usu rio em uma
        categoria
    public function findByCategoryAndUser($idDataset,
        $idCategory,$idUser,$options){

        $cursor = $this->collection->find(['$and' => [['
            idDataset' => ['$in' => $idDataset]],["
            isVisible"=>true],["tags" => ['$elemMatch'=>["
            category._id"=> new \MongoId($idCategory), "
            addedBy.idUser" => $idUser]]]]]);

         return $this->getTweets($cursor,$options);
    }
```

```
//tweets por categoria marcada um determinado n mero
    de vezes
public function findByCategoryAndQttTags($idDataset,
   $idCategory,$qtt,$options){
    $index = $qtt - 1;

    $cursor = $this->collection->find(['$and' => [['
       idDataset' => ['$in' => $idDataset]],["
       isVisible"=>true],["tags" => ['$elemMatch'=>["
       category._id"=> new \MongoId($idCategory), "
       addedBy.$index"=>['$exists'=>true]]]]]);
    return $this->getTweets($cursor,$options);

}

//n mero de categorias distintas marcadas em um
   tweet
public function findByQttDistinctCategories(
   $idDataset,$qtt,$options){
    $cursor = $this->collection->find(['$and' => [['
       idDataset' => ['$in' => $idDataset]],["
       isVisible"=>true],["tags" =>['$exists'=>true],
        '$where'=>"this.tags.length>=$qtt"]]]);
    return $this->getTweets($cursor,$options);

}

public function getRelationBetweenCategories(
   $idDataset,$idCatRow,$idCatColumn,$relation){

    switch ($relation){
        case "not":
            return $this->getNotBetweenCategories(
               $idDataset,$idCatRow,$idCatColumn);
            break;
        default:
            return $this->getAndOrBetweenCategories(
               $idDataset,$idCatRow,$idCatColumn,
```

```
                $relation);
            break;
    }


}

private function getAndOrBetweenCategories($idDataset
   ,$idCatRow,$idCatColumn,$relation) {

    switch ($relation){
        case 'and':
            $relation = '$and';
            break;
        case 'or':
            $relation = '$or';
            break;
    }

    $res = $this->collection->aggregate(

        [
            ['$match' =>
            ['$and' => [['idDataset' => ['$in' =>
               $idDataset]],["isVisible"=>true],
                [$relation=>
                    [
                    ["tags"=>['$elemMatch'=>["
                       category._id"=> new \MongoId(
                       $idCatRow)]]],
                    ["tags"=>['$elemMatch'=>["
                       category._id"=> new \MongoId(
                       $idCatColumn)]]]
                ]
                ]
                ]
                ]
            ],
```

```
                ['$project'=>
                    ['_id'=>1]]
            ]

        );

        return \count($res["result"]);

    }

    private function getNotBetweenCategories($idDataset,
        $idCatRow,$idCatColumn){

        $res = $this->collection->aggregate(

            [
                ['$match'=>
                    ['$and'=>[
                        ['idDataset' => ['$in' =>
                            $idDataset]],["isVisible"=>
                            true],
                        ["tags"=>['$elemMatch'=>["
                            category._id"=>new \MongoId(
                            $idCatRow)]]],
                        ["tags"=>['$not'=>['$elemMatch'
                            =>["category._id"=>new \
                            MongoId($idCatColumn)]]]]]],

                ['$project'=>['_id'=>true]]]


        );

        return \count($res["result"]);

    }

    public function findByRelationBetweenCategories(
        $idDataset,$idCatRow,$idCatColumn,$relation,
```

```
    $options){
     switch ($relation){
         case "not":
             return $this->findByNotBetweenCategories(
                $idDataset,$idCatRow,$idCatColumn,
                $options);
             break;
         default:
             return $this->
                findByAndOrBetweenCategories(
                $idDataset,$idCatRow,$idCatColumn,
                $relation,$options);
             break;
     }
}

private function findByNotBetweenCategories(
    $idDataset,$idCatRow,$idCatColumn,$options){

     $cursor = $this->collection->find(['$and'=>[
         ['idDataset' => ['$in' => $idDataset]],["
            isVisible"=>true],
         ["tags"=>['$elemMatch'=>["category._id"=>new
            \MongoId($idCatRow)]]],
         ["tags"=>['$not'=>['$elemMatch'=>["category.
            _id"=>new \MongoId($idCatColumn)]]]]]]);


     return $this->getTweets($cursor,$options);
}

private function findByAndOrBetweenCategories(
    $idDataset,$idCatRow,$idCatColumn,$relation,
    $options){

     switch ($relation){
         case 'and':
             $relation = '$and';
             break;
```

```
            case 'or':
                $relation = '$or';
                break;
        }

        $cursor = $this->collection->find(

            ['$and' => [['idDataset' => ['$in' =>
               $idDataset]],["isVisible"=>true],
                [$relation=>
                    [
                        ["tags"=>['$elemMatch'=>["
                           category._id"=> new \MongoId(
                           $idCatRow)]]],
                        ["tags"=>['$elemMatch'=>["
                           category._id"=> new \MongoId(
                           $idCatColumn)]]]
                    ]
                ]
            ]


        );

        return $this->getTweets($cursor,$options);
    }

    public function countClasses($idDataset, $idCategory)
       {

        $res = $this->collection->aggregate(

            [
                ['$match'=>
                    ['$and'=>[
                        ['idDataset' => $idDataset],
                        ["classes"=>['$elemMatch'=>["
                           category._id"=>new \MongoId(
```

```
                                $idCategory)]]]]]],

                        ['$project'=>['_id'=>true]]]


            );

            return \count($res["result"]);

        }

    }

}



<?php

namespace comunic\social_network_analyzer\model\repository\
   mongo{

    use \comunic\social_network_analyzer\model\repository\
       ICategoriesRepository;
    use comunic\social_network_analyzer\model\entity\mappers\
       ArrayToCategory;
    use comunic\social_network_analyzer\model\entity\mappers\
       CategoryToArray;
    use comunic\social_network_analyzer\model\util\MongoUtil;
    use comunic\social_network_analyzer\model\exception\
       UniqueConstraintException;


    class CategoriesRepository implements
       ICategoriesRepository{

        private $collection;

        function __construct($connectionType){
```

```
        $conn = new ConnectionMongo();
        $conn = $conn->getConnection($connectionType);
        $this->collection = $conn->selectCollection("
           projects");

    }

    public function insert($category, $projectId){

        try {

            if(!$this->isUnique($category->getName())){
                throw new UniqueConstraintException("
                   Already␣exists␣a␣category␣with␣name␣"
                   . $category->getName());
            }

            $toArray = new CategoryToArray();

            $arrayData = MongoUtil::includeMongoIdObject(
               $toArray($category));

            return $this->collection->update(array('_id'
               => new \MongoId($projectId)), array('
               $addToSet' => array("categories" =>
               $arrayData)), $options=array());

        }catch (\MongoException $e) {

            echo $e->getMessage();
        }
    }

    public function update($category, $projectId){

        try {

            if(!$this->isUnique($category->getName(),
               $category->getId())){
```

```
                throw new UniqueConstraintException("
                    Already␣exists␣a␣category␣with␣name␣"
                    . $category->getName());
        }

            $toArray = new CategoryToArray();

            $arrayData = MongoUtil::includeMongoIdObject(
                $toArray($category));

            return $this->collection->update(['$and' =>
                [['_id' => new \MongoId($projectId)],["
                categories._id" => new \MongoId($category
                ->getId())]]], ['$set' => ["categories.$"
                => $arrayData]], $options=array());
        }catch (\MongoException $e) {

            echo $e->getMessage();
        }

    }

    public function delete($id, $projectId){

        return $this->collection->update(array('_id' =>
            new \MongoId($projectId)), array('$pull' =>
            array("categories" => array("_id" => new \
            MongoId($id)))), $options=array());

    }

    public function findById($id){

       $arrayData=$this->collection->findOne(array('
            categories._id' => new \MongoId($id)), array("
            categories.$" => true, "_id" => false), $fields
            =array());

       $toCategory = new ArrayToCategory();
```

```
        return $toCategory(MongoUtil::removeMongoIdObject(
            $arrayData["categories"][0]));

    }

    public function isUnique($name,$id=null){
         /*
          * se id for nulo, opera ao de insert
          * so verifica se existe outro nome de categoria
             igual
          *
          *
          * se for update, primeiro verifica se o nome foi
             trocado
          * se foi trocado, verifica se o mesmo e repetido
             ou nao
          *
          * */

        $cursorFindByName = $this->collection->find(array(
            'categories.name' => $name), array("categories.
            $" => true, "_id" => false));

        if($cursorFindByName->count()==0){
            return true;
        }
        if(!\is_null($id)){
            $cursorFindByNameAndId = $this->collection->
                find(['categories' => ['$elemMatch' => ['
                _id' => new \MongoId($id),"name" => $name
                ]]],["categories.$" => true, "_id" => false
                ]);

            if($cursorFindByNameAndId->count()==1){
                return true;
            }
        }
         return false;
```

```
    }

    public function listAll($projectId){

        return $this->mongoch->find(new ArrayToCategory());

    }


}


}


<?php

namespace comunic\social_network_analyzer\model\auth{

        use \Firebase\JWT\JWT;

        class JsonWebToken{

                private $secret_key;

                function __construct(){
                        $this->secret_key = "key";
                }

                public function createToken($userInfo = array
                   ()){

                        $token = array(
                            "iat" => \time(),
                            "exp" => \time() + (60*60*24),
```

```
                "data" => $userInfo
                        );

                        return JWT::encode($token, $this->
                           secret_key);

                }

                private function decode($token){

                        $decoded = JWT::decode($token, $this
                           ->secret_key, array('HS256'));

                        return (array) $decoded;

                }

                public function isExpired($token){
                        $expireTime = $this->decode($token)["
                           exp"];

                        return $expireTime > \time();
                }

                public function getUserInfo($token){
                    return (array) $this->decode($token)["
                       data"];
        }


        }

}


<?php
/**
 * Created by PhpStorm.
 * User: cesar
```

```
 * Date: 28/07/16
 * Time: 11:46
 */

namespace comunic\social_network_analyzer\model\auth;

use comunic\social_network_analyzer\model\auth\JsonWebToken;
use comunic\social_network_analyzer\model\exception\
   NoResultException;
class Authenticator
{

    private $repository;
    private $jwt;

    function __construct($repository){
        $this->repository = $repository;
        $this->jwt = new JsonWebToken();
    }

    public function authenticateByToken($token){

        return $this->jwt->isExpired($token);

    }

    public function authenticateByEmailPassword($email,
       $password){

        $success = true;
        $message = "";
        $token = "";

       try{
           $user = $this->repository->findByEmail($email);

           $crypt = new Crypt();

//            if(!$user->getEnabled()){
```

```
//                 $message = "Usu rio  n  o  habilitado";
//                 $success = false;
//             }

            if(!$crypt->verify($password,$user->getPassword())
               ){
                $message = "Senha incorreta";
                $success = false;
            }

           if($success){
               $data = [
                   "name" => $user->getName(),
                   "id" => $user->getId(),
                   "email" => $user->getEmail(),
                   "role" => $user->getRole()

               ];

               $token = $this->jwt->createToken($data);
           }
        }catch (NoResultException $e){
            $message = "Usu rio n o cadastrado";
            $success = false;

        }finally{
            $response = [

                "success" => $success,
                "message" => $message,
                "token" => $token

            ];


            return $response;
        }
```

```
    }



}<?php

namespace comunic\social_network_analyzer\model\auth{

        class Crypt{



                public function applyHash($password){

                        return \password_hash($password,
                           PASSWORD_DEFAULT);

                }

                public function verify($password, $hash){
                        return \password_verify($password,
                           $hash);
                }




        }

}


<?php

namespace comunic\social_network_analyzer\model\facade{

/**
 * class DatasetsFactory
```

```
 *
 */
class DatasetsFacade extends CrudFacade
{
        //TODO sobrescrever delete para deletar tds os tweets
            de um dataset

}


} // end of DatasetsFactory

<?php

namespace comunic\social_network_analyzer\model\facade{

        class ProjectsFacade extends CrudFacade{


        }


}



<?php

namespace comunic\social_network_analyzer\model\facade {

    /**
     * class CategoriesFacade
     *
     */
    class CategoriesFacade extends CrudFacade{

    }

} // end of CategoriesFacade
```

```
<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 03/08/16
 * Time: 11:55
 */

namespace comunic\social_network_analyzer\model\facade;


use comunic\social_network_analyzer\model\entity\
   MatrixElement;
use comunic\social_network_analyzer\model\repository\mongo\
   CategoriesRepository;
use comunic\social_network_analyzer\model\repository\mongo\
   TweetsRepository;

class MatricesFacade extends CrudFacade
{


    private $tweetsRepository, $catRepository;

    public function __construct($tweetsRepository,
       $catRepository, $repository,$validator,$userFacade)
    {

        $this->catRepository = $catRepository;
        $this->tweetsRepository = $tweetsRepository;
        parent::__construct($repository,$validator,
           $userFacade);

    }

    public function populateMatrix($idMatrix,$fmt){

        $matrix = $this->repository->findById($idMatrix);
```

```
//          $matrix = $idMatrix;

        $items = [];

        $rows = $matrix->getRowsIds();
        $columns = $matrix->getColumnsIds();
        $rowsNames = $this->getCategoriesName($rows);
        $columnsNames = $this->getCategoriesName($columns);

        foreach (\range(0,count($rows) -1) as $iRow){

            foreach (\range(0,count($columns) - 1) as $iCol){

                $rowId = $rows[$iRow];
                $columnId = $columns[$iCol];
                $symbol =  $this->tweetsRepository->
                   getRelationBetweenCategories($matrix->
                   getDatasetsIds(),$rowId,$columnId,$matrix
                   ->getRelation());
                $items[] = new MatrixElement($rowId,
                   $rowsNames[$iRow],$columnId,$columnsNames[
                   $iCol],$symbol);

            }

        }

        $matrix->setElements($items);
        return $fmt->format($matrix);
    }

    private function getCategoriesName($ids){

        $names = [];

        foreach ($ids as $id){
            $category = $this->catRepository->findById($id);
            $names[] = $category->getName();
        }
```

```
        return $names;
    }




}<?php


namespace comunic\social_network_analyzer\model\facade{


        use comunic\social_network_analyzer\model\entity\
           UserRole;

class AdminFacade{

        private $repository;


        function __construct($repository){
                $this->repository = $repository;
        }

        public function listUsers($formatter){

                return $formatter->format($this->repository->
                   listAll());

        }

        public function enableUser($idUser,$status){

                $user = $this->repository->findById($idUser);
                $user->setEnabled($status);
                if(!$status){
                        $user->setToken("");
                }
```

```
                $this->repository->update($user);
        }

        public function setRole($idUser, $permission){
                $user = $this->repository->findById($idUser);
                $user->setRole(UserRole::$permission()->value
                   ());
                $this->repository->update($user);
        }

}


} // end of UsersFacade

<?php
namespace comunic\social_network_analyzer\model\facade {

    use comunic\social_network_analyzer\model\entity\
       validator\CategoryValidator;
    use comunic\social_network_analyzer\model\entity\
       validator\DatasetValidator;
    use comunic\social_network_analyzer\model\entity\
       validator\MatrixValidator;
    use comunic\social_network_analyzer\model\entity\
       validator\ProjectValidator;

    class FacadeFactory {

        private $repositoryFactory;



        public function __construct($repositoryFactory) {
            $this->repositoryFactory = $repositoryFactory;
        }

        public function instantiateTweets() {
            return new TweetsFacade($this->repositoryFactory
```

```
        ->instantiateTweet(), $this->repositoryFactory
        ->instantiateCategory(),$this->
        instantiateUsers(),$this->repositoryFactory->
        instantiateProject());
}


public function instantiateCategories() {
    return new CategoriesFacade($this->
        repositoryFactory->instantiateCategory(), new
        CategoryValidator(),$this->instantiateUsers())
        ;
}


public function instantiateUsers() {
    return new UsersFacade($this->repositoryFactory->
        instantiateUser());
}


public function instantiateProjects() {
    return new ProjectsFacade($this->
        repositoryFactory->instantiateProject(), new
        ProjectValidator(),$this->instantiateUsers());
}

public function instantiateDatasets() {
    return new DatasetsFacade($this->
        repositoryFactory->instantiateDataset(), new
        DatasetValidator(),$this->instantiateUsers());
}

public function instantiateAdmin() {
    return new AdminFacade($this->repositoryFactory->
        instantiateUser());
}

public function instantiateMatrices(){
```

```
            return new MatricesFacade($this->
               repositoryFactory->instantiateTweet(),$this->
               repositoryFactory->instantiateCategory(),
                $this->repositoryFactory->instantiateMatrix()
                   , new MatrixValidator(), $this->
                   instantiateUsers());
        }


// end of member function instantiateUsers
    }

} // end of FacadeFactory

<?php


namespace comunic\social_network_analyzer\model\facade{

        use comunic\social_network_analyzer\model\entity\
           Modification;
        use comunic\social_network_analyzer\model\exception\
           ValidationException;
        use comunic\social_network_analyzer\model\repository\
           mongo\MongoRepository;


        class CrudFacade{

                protected $repository;
                protected $validator;
                protected $userFacade;

                function __construct($repository,$validator,
                   $userFacade){
                        $this->repository = $repository;
                        $this->validator = $validator;
                        $this->userFacade = $userFacade;
                }
```

```
public function insert($obj_json, $parser,
   $tokenUser, $auxId = null){

        $obj = $parser->parse($obj_json);

        if(!$this->validator->validate($obj))
           {
                throw new ValidationException
                   ($this->validator->
                   printErrors());

        }
        $user = $this->userFacade->getByToken
           ($tokenUser);
        $modification = new Modification(
           $user->getId(), $user->getName(),
           \time());
        $obj->setCreatedBy($modification);
        $this->repository->insert($obj,
           $auxId);
}

public function update($obj_json,$parser,
   $tokenUser, $auxId = null){
        $obj = $parser->parse($obj_json);

        if(!$this->validator->validate($obj))
           {
                throw new ValidationException
                   ($this->validator->
                   printErrors());

        }

        $user = $this->userFacade->getByToken
           ($tokenUser);
        $modification = new Modification(
           $user->getId(), $user->getName(),
```

```
                        \time());
                    $obj->setUpdatedBy($modification);
                    $this->repository->update($obj,
                        $auxId);
                }

                public function findById($id, $formatter,
                    $auxId = null){
                        return $formatter->format($this->
                            repository->findById($id,$auxId));
                }

                public function listAll($formatter){
                        return $formatter->format($this->
                            repository->listAll());
                }

                public function delete($id, $auxId =null){
                        $this->repository->delete($id, $auxId
                            );
                }

        }

}

<?php


namespace comunic\social_network_analyzer\model\facade{

        use comunic\social_network_analyzer\model\auth\Crypt;
        use comunic\social_network_analyzer\model\auth\
            JsonWebToken;
        use comunic\social_network_analyzer\model\entity\
            UserRole;
        use comunic\social_network_analyzer\model\entity\User
            ;
        use comunic\social_network_analyzer\model\exception\
```

```
            NoResultException;
    use comunic\social_network_analyzer\model\auth\
       Authenticator;
/**
 * class UsersFacade
 *
 */
class UsersFacade{

        private $repository;
    private $authenticator;

        function __construct($repository){
                $this->repository = $repository;
        $this->authenticator = new Authenticator($repository)
           ;
        }

        public function authenticate($token){


        return $this->authenticator->authenticateByToken(
           $token);

        }

        public function login($email,$password){
            return $this->authenticator->
               authenticateByEmailPassword($email,$password);
    }

        public function signup($user_json, $parser){

                $user = $parser->parse($user_json);
        $crypt = new Crypt();

                if($this->repository->countUsers() >=1){
                        $user->setRole(UserRole::RESEARCHER()
                           ->value());
```

```
            }else{
                    $user->setRole(UserRole::
                       ADMINISTRATOR()->value());

            }
            $user->setPassword($crypt->applyHash($user->
               getPassword()));
            $user->setEnabled(false);

            $this->repository->insert($user);


    }

    public function changePassword($idUser, $password){
    $user = $this->repository->findById($idUser);
    $crypt = new Crypt();
    $user->setPassword($crypt->applyHash($password));

    $this->repository->insert($user);
}


    public function findById($id, $formatter){

            $user = $this->repository->findById($id);
            return $formatter->format($user);

    }


    public function getByToken($token){

        $jwt = new JsonWebToken();
        $idUser = $jwt->getUserInfo($token)["id"];
    return $this->repository->findById($idUser);

    }
```

```
}

} // end of UsersFacade

<?php


namespace comunic\social_network_analyzer\model\facade{

    use comunic\social_network_analyzer\model\entity\
       SlicingPoint;
    use comunic\social_network_analyzer\model\entity\
       ConversationCrawler;
    use comunic\social_network_analyzer\model\entity\Tweet;
    use comunic\social_network_analyzer\model\util\StringUtil
       ;
    use comunic\social_network_analyzer\model\util\TwitterApi
       ;
    use comunic\social_network_analyzer\model\util\
       TwitterCrawler;
    use comunic\social_network_analyzer\model\entity\
       Modification;
    use comunic\social_network_analyzer\model\entity\Tag;

    /**
     * class TweetsFacade
     *
     */
    class TweetsFacade
    {


        function __construct($repository, $categoryRep,
           $userRepository,$projRepository){
            $this->repository = $repository;
```

```
        $this->categoryRep = $categoryRep;
        $this->userRepository = $userRepository;
        $this->projectRepository = $projRepository;
    }

    public function insert( $text,  $parser, $idDataset,
       $idName, $textName, $fromUserName, $createdAtName)
        {

        $assocArrayTweets = $parser->parse($text);

        foreach ($assocArrayTweets as $arrayTw) {

            $tweet = new Tweet();
            $tweet->setIdDataset($idDataset);
            $tweet->setIsVisible(true);

            $tweet->setText($arrayTw[$textName]);
            unset($arrayTw[$textName]);
            $tweet->setIdTweet($arrayTw[$idName]);
            unset($arrayTw[$idName]);
            $tweet->setFromUser($arrayTw[$fromUserName]);
            unset($arrayTw[$fromUserName]);
            $tweet->setUnixTimestamp(\strtotime($arrayTw[
               $createdAtName]));

            $tweet->setOtherAttributes($arrayTw);


            $this->repository->insert($tweet);
        }



    }

    public function findById( $id,  $fmt) {
        return $fmt->format($this->repository->findById(
           $id));
```

```
} // end of member function findById

public function listByDataset($idDataset,
   $includeRepeated,$fmt, $options){
    return $fmt->format($this->repository->
       listByDataset($idDataset, $includeRepeated,
       $options));
}

public function getDatasetSlice($idDataset,$cutPoint,
   $fragmentSize,$fmt,$options){

    $skip = $options["skip"];
    $count = 0;

    foreach ($idDataset as $dataset){
        $count += $this->repository->countByDataset(
           $dataset);
    }
    switch ($cutPoint) {
        case SlicingPoint::BEGINNING():
            $options["skip"] = 0 + $skip;
            break;

        case SlicingPoint::HALF():
            $options["skip"]  = (int) \floor($count /
                2) - (\floor($fragmentSize / 2)) +
               $skip;
            break;

        case SlicingPoint::END():
            $options["skip"]  = ($count -
               $fragmentSize) + $skip;
            break;
    }

    return $this->listByDataset($idDataset,true,$fmt,
       $options);
```

```
}


public function findByCategory($idDataset,
   $idCategory, $fmt, $options) {
    $category = $this->categoryRep->findById(
       $idCategory);

    $kwAsRegex = $category->toRegex();

    $tweets=$this->repository->findbyCategory(
       $idDataset, $kwAsRegex, $options);
    return $fmt->format($tweets);
} // end of member function listByCategory


public function searchInTheText($idDataset,$term,
   $fmt, $options){
    return $fmt->format($this->repository->
       searchInTheText($idDataset,StringUtil::
       removeAccents($term), $options));
}


public function getConversation($id, $fmt){
    $tweet = $this->repository->findById($id);
    $ttCrawler = new ConversationCrawler();

    $conversation = $ttCrawler->retrieveConversation(
       $tweet->getIdTweet());

    return $fmt->format($conversation);

}

public function getMoreConversation($idRootTweet,
   $idLastTweet, $fmt){

    $ttCrawler = new ConversationCrawler();
```

```
        $conversation = $ttCrawler->
            retrieveMoreConversation($idRootTweet,
            $idLastTweet);

        return $fmt->format($conversation);

    }

    public function setVisibility($id, $isVisible){
        $tweet = $this->repository->findById($id);
        $tweet->setIsVisible($isVisible);
        $this->repository->insert($tweet);
    }

    public function update($tweet_text,$parser){

        $tweet = $parser->parse($tweet_text);

        $this->repository->update($tweet);

    }

    public function addTag($tweetId, $idCategory,
       $tokenUser){

        $category = $this->categoryRep->findById(
            $idCategory);
        $tweet = $this->repository->findById($tweetId);
        $user = $this->userRepository->getByToken(
            $tokenUser);

        $tweet->addTag($category,$user->getId(),$user->
            getName());


        $this->repository->insert($tweet);

    }
```

```
public function removeTag($idTweet, $idCategory,
   $tokenUser){

    $tweet = $this->repository->findById($idTweet);
    $user = $this->userRepository->getByToken(
       $tokenUser);
    $tweet->removeTag($idCategory,$user->getId());
    $this->repository->insert($tweet);

}

public function addClass($tweetId, $idCategory,
   $tokenUser){

    $category = $this->categoryRep->findById(
       $idCategory);
    $tweet = $this->repository->findById($tweetId);
    $user = $this->userRepository->getByToken(
       $tokenUser);

    $tweet->addClass($category,$user->getId(),$user->
       getName());

    $this->repository->insert($tweet);

}

public function removeClass($idTweet, $idCategory,
   $tokenUser){

    $tweet = $this->repository->findById($idTweet);
    $user = $this->userRepository->getByToken(
       $tokenUser);
    $tweet->removeClass($idCategory,$user->getId());
    $this->repository->insert($tweet);

}
```

```
public function findByCategoryAndUser($idDataset,
   $idCategory, $idUser, $fmt, $options) {

    $tweets=$this->repository->findByCategoryAndUser(
       $idDataset,$idCategory,$idUser,$options);
    return $fmt->format($tweets);
}

public function findByCategoryAndQttTags($idDataset,
   $idCategory,$qtt,$fmt,$options) {

    $tweets=$this->repository->
       findByCategoryAndQttTags($idDataset,
       $idCategory,$qtt,$options);
    return $fmt->format($tweets);
}

public function findByQttDistinctCategories(
   $idDataset,$qtt,$fmt,$options){

    $tweets=$this->repository->
       findByQttDistinctCategories($idDataset,$qtt,
       $options);
    return $fmt->format($tweets);
}

public function findByRelationBetweenCategories(
   $idDataset,$idCatRow,$idCatColumn,$relation,$fmt,
   $options){
    $tweets = $this->repository->
       findByRelationBetweenCategories($idDataset,
       $idCatRow,$idCatColumn,$relation,$options);
    return $fmt->format($tweets);
}

public function countClassesInDataset($projectId,
   $datasetId){

    $categories = $this->projectRepository->findById(
```

```
            $projectId)->getCategories();


        $result = [];

        foreach ($categories as $category){
            if($category->getType() == "
               SPACES_OF_POSSIBILITY"){
                $result[] = ["id" => $category->getId(),
                    "count"=>$this->repository->
                       countClasses($datasetId,$category
                       ->getId())];
            }
        }

        return \json_encode($result);

    }

}
```

```
} // end of TweetsFacade

<?php

namespace comunic\social_network_analyzer\model\util{

    mb_internal_encoding('UTF-8');
    mb_regex_encoding("UTF-8");

    class StringUtil {



/**
* Returna uma string com acento em uma express o REGEX para
   encontrar todas as variantes
* em formato n o -sensitivo ao acento.
```

```
* Obtido em: http://tech.rgou.net/php/pesquisas-nao-sensiveis
   -ao-caso-e-acento-no-mongodb-e-php/
*
* @param string $text O texto.
* @return string O texto em REGEX.
*/
static public function accentToRegex($text)
{
    $from = str_split(utf8_decode(self::ACCENT_STRINGS));
    $to   = str_split(strtolower(self::NO_ACCENT_STRINGS));
    $text = utf8_decode($text);
    $regex = array();
    foreach ($to as $key => $value){
        if (isset($regex[$value])){
            $regex[$value] .= $from[$key];
        } else {
            $regex[$value] = $value;
            $regex[$value] .= $from[$key];
        }
    }
    foreach ($regex as $rg_key => $rg){
/**
* Marca os caracteres acentuados que ser o substitu dos com
    suas outras variantes
*/
$text = preg_replace("/[$rg]/", "_{$rg_key}_", $text);
}
foreach ($regex as $rg_key => $rg){
    $text = preg_replace("/_{$rg_key}_/", "[$rg]", $text);
}
return utf8_encode($text);
}

static public function removePunctuation($string,$except){

    return \mb_ereg_replace("/[^a-zA-Z0-9\s$except]/u", "",
       $string);

}
```

```
static public function removeLineBreaks($string){
    return \str_replace(array("\n","\r"), "", $string);
}

static public function removeURL($string){
    return \preg_replace("/(((ftp|https?):\/\/)(www\.)?|www
       \.)([\da-z-_\.]+)([a-z\.]{2,7})([\/\w\.-_\?\&]*)*\/?/"
       ,"",$string);
}

static public function removeRT($string){
    return \preg_replace("/^RT\s*@[^:]*:\s*/","",$string);
}




}



}
<?php


namespace comunic\social_network_analyzer\model\util;


/**
 * Created by PhpStorm.
 * Author: Anis Ahmad <anisniit@gmail.com>
 * Date: 5/11/14
 * Time: 10:44 PM
 * https://gist.github.com/ajaxray/94b27439ba9c3840d420
 */

trait DocumentSerializer {
    private $_ignoreFields = array();
    /**
```

```
     * Convert Doctrine\ODM Document to Array
     *
     * @return array
     */
    function toArray() {
        $document = $this->toStdClass();
        return get_object_vars($document);
    }
    /**
     * Convert Doctrine\ODM Document to Array
     *
     * @return string
     */
    function toJSON() {
        $document = $this->toStdClass();
        return json_encode($document);
    }
    /**
     * Set properties to ignore when serializing
     *
     * @example  $this->setIgnoredFields(array('createdDate',
         'secretFlag'));
     *
     * @param array $fields
     */
    function setIgnoredFields(array $fields) {
        $this->_ignoreFields = $fields;
    }
    /**
     * Convert Doctrine\ODM Document to plain simple stdClass
     *
     * @return \stdClass
     */
    function toStdClass()
    {
        $document = new \stdClass();
        foreach($this->findGetters() as $getter) {
            $prop = lcfirst(substr($getter, 3));
            if(! in_array($prop, $this->_ignoreFields)) {
```

```
                $value = $this->$getter();
                $document->$prop = $this->formatValue($value)
                    ;
            }
        }
        return $document;
    }
    private function findGetters()
    {
        $funcs = get_class_methods(get_class($this));
        $getters = array();
        foreach($funcs as $func) {
            if(strpos($func, 'get') === 0) {
                $getters[] = $func;
            }
        }
        return $getters ;
    }
    private function formatValue($value) {
        if(is_scalar($value)) {
            return $value;
            // If the object uses this trait
        } elseif (in_array(__TRAIT__, class_uses(ltrim(
            get_class($value),"Proxies\\__CG__\\")))) {//
            GAMBIARRAAAAA
            return $value->toStdClass();
            // If it's a collection, format each value
        } elseif (is_a($value, 'Doctrine\ODM\MongoDB\
            PersistentCollection')) {
            $prop = array();
            foreach($value as $k => $v) {
                $prop[] = $this->formatValue($v);
            }
            return $prop;
            // If it's a Date, convert to unix timestamp
        } else if(is_a($value, 'DateTime')) {
            return $value->getTimestamp();
            // Otherwise leave a note that this type is not
                formatted
```

```
            // So that I can add formatting for this missed
               class
        } else {
            return 'Not formatted in DocumentSerializer: '.
               get_class($value);
        }
    }
}
<?php


namespace comunic\social_network_analyzer\model\util{


        class MongoUtil{


                public static function removeMongoIdObject(
                   array $array){

                        foreach ($array as $k => $v) {
                                if (is_array($v)) {
                                        $array[$k] = self::
                                           removeMongoIdObject
                                           ($v);
                                } else {
                                        if ("_id" === $k) {
                                                unset($array[
                                                   $k]);
                                                $array["id"]
                                                   = (string)
                                                    $v;
                                        }
                                }
                        }

                        return $array;

                }
```

```
                public static function includeMongoIdObject(
                   array $array){

                        foreach ($array as $k => $v) {
                                if (is_array($v)) {
                                        $array[$k] = self::
                                           includeMongoIdObject
                                           ($v);
                                } else {
                                        if ("id" === $k) {
                                                unset($array[
                                                   $k]);
                                                $array["_id"]
                                                    = new \
                                                   MongoId($v
                                                   );
                                        }
                                }
                        }

                        return $array;

                }




        }
}<?php

namespace comunic\social_network_analyzer\model\util{

  class ArrayUtil{

    public static function is_assoc_array($array){

     return \array_keys($array) !== \range(0, count($array) -
```

```
        1);

   }

   public static function eliminates_repeated($array){
     $withoutRepeated = array();

     foreach ($array as $item) {
      $withoutRepeated[$item] = $item;
    }
    return array_values($withoutRepeated);

  }

  public static function slicer($array, $qttPerSlice){
    $slices = array();

    $nSlices = ceil(count($array) / $qttPerSlice);

    for ($i=0; $i <$nSlices ; $i++) {

      $slices[] = \array_slice($array, $i*($qttPerSlice),
         $qttPerSlice);

    }

    return $slices;
  }

}

}

<?php


namespace comunic\social_network_analyzer\model\util\types{

        abstract class EnumType {
```

```
                private $val;

                public abstract function getFields();

                final function __construct( $str ) {
                        if ( ! in_array( $str,  $this->
                           getFields() ) ) {
                                throw new \Exception("unknown
                                   ␣type␣value:␣$str");
                        }
                        $this->val = $str;
                }

                public static function __callStatic( $func,
                   $args ) {
                        return new static( $func );
                }

                public function value() {
                        return $this->val;
                }

                public function __toString() {
                        return $this->value();
                }
        }


}


<?php
/**
 * Created by PhpStorm.
 * User: cesar
 * Date: 19/12/16
 * Time: 19:13
 */
```

```
namespace comunic\social_network_analyzer\model\builder;


use comunic\social_network_analyzer\model\entity\Project;

class ProjectBuilder
{

    private $project;

    /**
     * ProjectBuilder constructor.
     * @param $project
     */
    public function __construct($project = null)
    {
        if(\is_null($project)){
            $this->project = new Project();
        }else{
            $this->project = $project;
        }

    }

    public function build($params){

        $this->project->setName($params['name']);
        $this->project->setDescription($params['description'
           ]);

        return $this->project;

    }


}<?php
/**
 * Created by PhpStorm.
```

```
 * User: cesar
 * Date: 19/12/16
 * Time: 19:13
 */

namespace comunic\social_network_analyzer\model\builder;


use comunic\social_network_analyzer\model\entity\User;

class UserBuilder
{

    private $user;

    /**
     * ProjectBuilder constructor.
     * @param $user
     */
    public function __construct($user = null)
    {
        if(\is_null($user)){
            $this->user = new User();
        }else{
            $this->user = $user;
        }

    }

    public function build($params){

        $this->user->setName($params['name']);
        $this->user->setPassword($params['password']);
        $this->user->setEmail($params['email']);
        $this->user->setEnabled(true);
        return $this->user;

    }
```

```
}<?php

namespace comunic\social_network_analyzer\middlewares{

        use Slim\Middleware;


        class TokenAuth extends Middleware{

                private $usersFacade;

                function __construct($usersFacade){

                        $this->usersFacade = $usersFacade;

                        $this->public_urls = ["\/auth\/google
                           ","\/users\/json\/signup","\/users
                           \/json\/login"];

                        $this->admin_urls = ['\/users\/json\/
                           enable\/([a-z0-9]+)','\/users\/
                           json\/set_role\/([a-z0-9]+)'];

                }

                private function authenticate($token){
                        return $this->usersFacade->
                           authenticate($token);

                }

                private function deny_access() {
                        $res = $this->app->response();
                        $res->status(401);
                }

                private function unauthorized(){
                        $res = $this->app->response();
```

```
            $res->status(403);
        }

        private function containUrl($url, $urls){
            $patterns_flattened = implode('|',
                $urls);
            $matches = [];



            preg_match('/' . $patterns_flattened
                . '/', $url, $matches);
            return count($matches) > 0;
        }

        private function hasAccess($role, $url,
            $method){
            //NECESSITA TESTAR TODOS OS CASOS
            $is_admin_url = $this->is_admin_url(
                $url);

            switch ($role) {
                    case 'ADMINISTRATOR':
                    return true;

                    case 'MODERATOR':
                    if($is_admin_url){
                            return false;
                    }
                    return true;

                    case 'RESEARCHER':
                    $entity = \explode("/", $url)
                        [1];
                    $protected_entities = ["
                        datasets","tweets","
                        projects","users"];

                    if($is_admin_url){
```

```
                        return false;
                    }

                    if(\in_array($entity,
                       $protected_entities) &&
                       $method != "GET"){
                        return false;
                    }

                    return true;
                }

            }

            private function is_public_url($url){
                return $this->containUrl($url, $this
                   ->public_urls);
            }

            private function is_admin_url($url){
                return $this->containUrl($url, $this
                   ->admin_urls);
            }

            public function call() {

                $url = $this->app->request->
                   getPathInfo();

                if($this->is_public_url($url)) {
                        $this->next->call();

                }else{

                        $tokenAuth = \explode("␣",
                           $this->app->request->
                           headers->get('
                           Authorization'))[1];
```

```
                                    if ($this->authenticate(
                                       $tokenAuth)) {

                                          $user = $this->
                                             usersFacade->
                                             getByToken(
                                             $tokenAuth);
                            $role = $user->getRole();
                                          $method = $this->app
                                             ->request->
                                             getMethod();

                                          if($this->hasAccess(
                                             $role,$url,
                                             $method)){
                                                $this->next->
                                                   call();
                                          }else{
                                                $this->
                                                   unauthorized
                                                   ();
                                          }
                                    } else {
                                          $this->deny_access();
                                    }
                              }



                        }


                }


}
```

```
<?php


require_once __DIR__ . '/vendor/autoload.php';


use Doctrine\ODM\MongoDB\Configuration,
    Doctrine\ODM\MongoDB\Mapping\Driver\AnnotationDriver,
    Doctrine\ODM\MongoDB\DocumentManager,
    comunic\social_network_analyzer\model\entity\Project,
    Doctrine\MongoDB\Connection;



$config = new Configuration();
$config->setProxyDir(__DIR__ . '/cache');
$config->setProxyNamespace('Proxies');

$config->setHydratorDir(__DIR__ . '/cache');
$config->setHydratorNamespace('Hydrators');
$config->setDefaultDB('sonda_doctrine');

$annotationDriver = $config->newDefaultAnnotationDriver(array
   (__DIR__ . '/src/comunic/social_network_analyzer/model/
   entity'));
$config->setMetadataDriverImpl($annotationDriver);
AnnotationDriver::registerAnnotationClasses();

$dm = DocumentManager::create(new Connection(), $config);


// Document classes
//$classLoader = new ClassLoader('comunic\
   social_network_analyzer\model\entity', __DIR__);
//$classLoader->register();
//
//$project = new Project("aaa",null);
//
//$project->setId("58583a913cbd95e74b8b4567");
```

```
////$project->setName("Primeiro");
//$project->setDescription("Segundaaaaaooooo");
//
//$dm->persist($project);
//$dm->flush();<?php

namespace Hydrators;

use Doctrine\ODM\MongoDB\DocumentManager;
use Doctrine\ODM\MongoDB\Mapping\ClassMetadata;
use Doctrine\ODM\MongoDB\Hydrator\HydratorInterface;
use Doctrine\ODM\MongoDB\Query\Query;
use Doctrine\ODM\MongoDB\UnitOfWork;
use Doctrine\ODM\MongoDB\Mapping\ClassMetadataInfo;

/**
 * THIS CLASS WAS GENERATED BY THE DOCTRINE ODM. DO NOT EDIT
    THIS FILE.
 */
class comunicsocial_network_analyzermodelentityUserHydrator
   implements HydratorInterface
{
    private $dm;
    private $unitOfWork;
    private $class;

    public function __construct(DocumentManager $dm,
       UnitOfWork $uow, ClassMetadata $class)
    {
        $this->dm = $dm;
        $this->unitOfWork = $uow;
        $this->class = $class;
    }

    public function hydrate($document, $data, array $hints =
       array())
    {
        $hydratedData = array();
```

```
/** @Field(type="id") */
if (isset($data['_id']) || (! empty($this->class->
   fieldMappings['id']['nullable']) &&
   array_key_exists('_id', $data))) {
    $value = $data['_id'];
    if ($value !== null) {
        $return = $value instanceof \MongoId ? (
           string) $value : $value;
    } else {
        $return = null;
    }
    $this->class->reflFields['id']->setValue(
       $document, $return);
    $hydratedData['id'] = $return;
}

/** @Field(type="string") */
if (isset($data['name']) || (! empty($this->class->
   fieldMappings['name']['nullable']) &&
   array_key_exists('name', $data))) {
    $value = $data['name'];
    if ($value !== null) {
        $return = (string) $value;
    } else {
        $return = null;
    }
    $this->class->reflFields['name']->setValue(
       $document, $return);
    $hydratedData['name'] = $return;
}

/** @Field(type="string") */
if (isset($data['email']) || (! empty($this->class->
   fieldMappings['email']['nullable']) &&
   array_key_exists('email', $data))) {
    $value = $data['email'];
    if ($value !== null) {
        $return = (string) $value;
    } else {
```

```
            $return = null;
        }
        $this->class->reflFields['email']->setValue(
           $document, $return);
        $hydratedData['email'] = $return;
    }

    /** @Field(type="string") */
    if (isset($data['password']) || (! empty($this->class
       ->fieldMappings['password']['nullable']) &&
       array_key_exists('password', $data))) {
        $value = $data['password'];
        if ($value !== null) {
            $return = (string) $value;
        } else {
            $return = null;
        }
        $this->class->reflFields['password']->setValue(
           $document, $return);
        $hydratedData['password'] = $return;
    }

    /** @Field(type="boolean") */
    if (isset($data['enabled']) || (! empty($this->class
       ->fieldMappings['enabled']['nullable']) &&
       array_key_exists('enabled', $data))) {
        $value = $data['enabled'];
        if ($value !== null) {
            $return = (bool) $value;
        } else {
            $return = null;
        }
        $this->class->reflFields['enabled']->setValue(
           $document, $return);
        $hydratedData['enabled'] = $return;
    }

    /** @Many */
    $mongoData = isset($data['projects']) ? $data['
```

```
            projects'] : null;
        $return = $this->dm->getConfiguration()->
            getPersistentCollectionFactory()->create($this->dm
            , $this->class->fieldMappings['projects']);
        $return->setHints($hints);
        $return->setOwner($document, $this->class->
            fieldMappings['projects']);
        $return->setInitialized(false);
        if ($mongoData) {
             $return->setMongoData($mongoData);
        }
        $this->class->reflFields['projects']->setValue(
            $document, $return);
        $hydratedData['projects'] = $return;
        return $hydratedData;
    }
}<?php

namespace Proxies\__CG__\comunic\social_network_analyzer\
    model\entity;

/**
 * DO NOT EDIT THIS FILE - IT WAS CREATED BY DOCTRINE'S PROXY
     GENERATOR
 */
class User extends \comunic\social_network_analyzer\model\
    entity\User implements \Doctrine\ODM\MongoDB\Proxy\Proxy
{
    /**
     * @var \Closure the callback responsible for loading
        properties in the proxy object. This callback is
        called with
     *      three parameters, being respectively the proxy
        object to be initialized, the method that triggered
        the
     *      initialization process and an array of ordered
        parameters that were passed to that method.
     *
     * @see \Doctrine\Common\Persistence\Proxy::
```

```
    __setInitializer
 */
public $__initializer__;

/**
 * @var \Closure the callback responsible of loading
    properties that need to be copied in the cloned
    object
 *
 * @see \Doctrine\Common\Persistence\Proxy::__setCloner
 */
public $__cloner__;

/**
 * @var boolean flag indicating if this object was
    already initialized
 *
 * @see \Doctrine\Common\Persistence\Proxy::
    __isInitialized
 */
public $__isInitialized__ = false;

/**
 * @var array properties to be lazy loaded, with keys
    being the property
 *            names and values being their default values
 *
 * @see \Doctrine\Common\Persistence\Proxy::
    __getLazyProperties
 */
public static $lazyPropertiesDefaults = [];



/**
 * @param \Closure $initializer
 * @param \Closure $cloner
 */
public function __construct($initializer = null, $cloner
```

```
   = null)
{

    $this->__initializer__ = $initializer;
    $this->__cloner__      = $cloner;
}









/**
 *
 * @return array
 */
public function __sleep()
{
    if ($this->__isInitialized__) {
        return ['__isInitialized__', '' . "\0" . 'comunic
           \\social_network_analyzer\\model\\entity\\User
           ' . "\0" . 'id', '' . "\0" . 'comunic\\
           social_network_analyzer\\model\\entity\\User'
           . "\0" . 'name', '' . "\0" . 'comunic\\
           social_network_analyzer\\model\\entity\\User'
           . "\0" . 'email', '' . "\0" . 'comunic\\
           social_network_analyzer\\model\\entity\\User'
           . "\0" . 'password', '' . "\0" . 'comunic\\
           social_network_analyzer\\model\\entity\\User'
           . "\0" . 'enabled', '' . "\0" . 'comunic\\
           social_network_analyzer\\model\\entity\\User'
           . "\0" . 'projects', '' . "\0" . 'comunic\\
           social_network_analyzer\\model\\entity\\User'
           . "\0" . '_ignoreFields'];
    }

    return ['__isInitialized__', '' . "\0" . 'comunic\\
       social_network_analyzer\\model\\entity\\User' . "
```

```
        \0" . 'id', '' . "\0" . 'comunic\\
        social_network_analyzer\\model\\entity\\User' . "
        \0" . 'name', '' . "\0" . 'comunic\\
        social_network_analyzer\\model\\entity\\User' . "
        \0" . 'email', '' . "\0" . 'comunic\\
        social_network_analyzer\\model\\entity\\User' . "
        \0" . 'password', '' . "\0" . 'comunic\\
        social_network_analyzer\\model\\entity\\User' . "
        \0" . 'enabled', '' . "\0" . 'comunic\\
        social_network_analyzer\\model\\entity\\User' . "
        \0" . 'projects', '' . "\0" . 'comunic\\
        social_network_analyzer\\model\\entity\\User' . "
        \0" . '_ignoreFields'];
}

/**
 *
 */
public function __wakeup()
{
    if ( ! $this->__isInitialized__) {
        $this->__initializer__ = function (User $proxy) {
            $proxy->__setInitializer(null);
            $proxy->__setCloner(null);

            $existingProperties = get_object_vars($proxy)
               ;

            foreach ($proxy->__getLazyProperties() as
               $property => $defaultValue) {
                if ( ! array_key_exists($property,
                   $existingProperties)) {
                    $proxy->$property = $defaultValue;
                }
            }
        };

    }
}
```

```
/**
 *
 */
public function __clone()
{
    $this->__cloner__ && $this->__cloner__->__invoke(
       $this, '__clone', []);
}

/**
 * Forces initialization of the proxy
 */
public function __load()
{
    $this->__initializer__ && $this->__initializer__->
       __invoke($this, '__load', []);
}

/**
 * {@inheritDoc}
 * @internal generated method: use only when explicitly
    handling proxy specific loading logic
 */
public function __isInitialized()
{
    return $this->__isInitialized__;
}

/**
 * {@inheritDoc}
 * @internal generated method: use only when explicitly
    handling proxy specific loading logic
 */
public function __setInitialized($initialized)
{
    $this->__isInitialized__ = $initialized;
}
```

```
/**
 * {@inheritDoc}
 * @internal generated method: use only when explicitly
    handling proxy specific loading logic
 */
public function __setInitializer(\Closure $initializer =
   null)
{
    $this->__initializer__ = $initializer;
}

/**
 * {@inheritDoc}
 * @internal generated method: use only when explicitly
    handling proxy specific loading logic
 */
public function __getInitializer()
{
    return $this->__initializer__;
}

/**
 * {@inheritDoc}
 * @internal generated method: use only when explicitly
    handling proxy specific loading logic
 */
public function __setCloner(\Closure $cloner = null)
{
    $this->__cloner__ = $cloner;
}

/**
 * {@inheritDoc}
 * @internal generated method: use only when explicitly
    handling proxy specific cloning logic
 */
public function __getCloner()
{
    return $this->__cloner__;
```

```
}

/**
 * {@inheritDoc}
 * @internal generated method: use only when explicitly
    handling proxy specific loading logic
 * @static
 */
public function __getLazyProperties()
{
    return self::$lazyPropertiesDefaults;
}


/**
 * {@inheritDoc}
 */
public function getId()
{
    if ($this->__isInitialized__ === false) {
        return  parent::getId();
    }


    $this->__initializer__ && $this->__initializer__->
       __invoke($this, 'getId', []);

    return parent::getId();
}

/**
 * {@inheritDoc}
 */
public function setId($id)
{

    $this->__initializer__ && $this->__initializer__->
       __invoke($this, 'setId', [$id]);
```

```
        return parent::setId($id);
    }

    /**
     * {@inheritDoc}
     */
    public function getName()
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'getName', []);

        return parent::getName();
    }

    /**
     * {@inheritDoc}
     */
    public function setName($name)
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'setName', [$name]);

        return parent::setName($name);
    }

    /**
     * {@inheritDoc}
     */
    public function getEmail()
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'getEmail', []);

        return parent::getEmail();
    }
```

```
    /**
     * {@inheritDoc}
     */
    public function setEmail($email)
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'setEmail', [$email]);

        return parent::setEmail($email);
    }

    /**
     * {@inheritDoc}
     */
    public function getPassword()
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'getPassword', []);

        return parent::getPassword();
    }

    /**
     * {@inheritDoc}
     */
    public function setPassword($password)
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'setPassword', [$password]);

        return parent::setPassword($password);
    }

    /**
     * {@inheritDoc}
     */
```

```
public function getEnabled()
{

    $this->__initializer__ && $this->__initializer__->
       __invoke($this, 'getEnabled', []);

    return parent::getEnabled();
}

/**
 * {@inheritDoc}
 */
public function setEnabled($enabled)
{

    $this->__initializer__ && $this->__initializer__->
       __invoke($this, 'setEnabled', [$enabled]);

    return parent::setEnabled($enabled);
}

/**
 * {@inheritDoc}
 */
public function getProjects()
{

    $this->__initializer__ && $this->__initializer__->
       __invoke($this, 'getProjects', []);

    return parent::getProjects();
}

/**
 * {@inheritDoc}
 */
public function setProjects($projects)
{
```

```
        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'setProjects', [$projects]);

        return parent::setProjects($projects);
    }

    /**
     * {@inheritDoc}
     */
    public function addProject($project)
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'addProject', [$project]);

        return parent::addProject($project);
    }

    /**
     * {@inheritDoc}
     */
    public function toArray()
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'toArray', []);

        return parent::toArray();
    }

    /**
     * {@inheritDoc}
     */
    public function toJSON()
    {

        $this->__initializer__ && $this->__initializer__->
            __invoke($this, 'toJSON', []);
```

```
        return parent::toJSON();
    }

    /**
     * {@inheritDoc}
     */
    public function setIgnoredFields(array $fields)
    {

        $this->__initializer__ && $this->__initializer__->
           __invoke($this, 'setIgnoredFields', [$fields]);

        return parent::setIgnoredFields($fields);
    }

    /**
     * {@inheritDoc}
     */
    public function toStdClass()
    {

        $this->__initializer__ && $this->__initializer__->
           __invoke($this, 'toStdClass', []);

        return parent::toStdClass();
    }

}

//Dependencias API

{
    "require": {
        "triagens/arangodb": "^2.6",
        "slim/slim": "^2.6",
        "phpunit/phpunit": "^4.7",
        "zumba/mongounit": "^2.0",
        "abraham/twitteroauth": "^0.6.2",
        "firebase/php-jwt": "^3.0",
        "guzzlehttp/guzzle": "^6.1",
```

```
        "fabpot/goutte": "^3.1"
    },

    "autoload":{
        "psr-0" : {
            "comunic\\" : "src/",
            "sna\\tests" : "tests/"

        }
    }
}

angular.module('snaApp2', ['ui.utils','ngRoute','ngAnimate','
   ngTouch','ui.bootstrap','ngCookies','ngResource','
   ngSanitize', 'ngCsv', 'ngStorage', 'satellizer','
   ngDragDrop','ngMessages',"checklist-model",'ngFileSaver',
   'angularSpinners']);

angular.module('snaApp2').constant('USER_ROLES', {
    all: '*',
    admin: 'ADMINISTRATOR',
    moderator: 'MODERATOR',
    researcher: 'RESEARCHER'
});

angular.module('snaApp2').constant('BASE_URL', '/projetos/
   sonda/api/');

angular.module('snaApp2').config(['$routeProvider','
   $httpProvider','USER_ROLES', '$authProvider', function(
   $routeProvider,$httpProvider,USER_ROLES,$authProvider) {

    /* Add New Routes Above */
    $routeProvider.when('/project/add',{
        controller:"ProjectAddCtrl",
        templateUrl:"app/components/project/controllers/add/
           project-add.html",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin, USER_ROLES.
```

```
            moderator]
        }

    }).when('/',{
            controller:"HomeCtrl",
            templateUrl:"app/components/home/home.html",
            title:"Nome",
            data: {
                needsAuth : false,
            }

        })
        .when('/set_project',{
            controller:"ProjectSetCtrl",
            templateUrl:"app/components/project/controllers/
               set/project-set.html",
            title:"Nome",
            data: {
                needsAuth : false,
            }

        })

        .when('/login',{
            controller:"UserLoginCtrl",
            templateUrl:"app/components/user/controllers/user
               -login/user-login.html",
            title:"Login",
            data: {
                needsAuth : false,
            }

        })

        .when('/error',{
            controller:"ErrorCtrl",
            templateUrl:"app/components/error/error.html",
            title:"Error",
            data: {
```

```
            needsAuth : false,
        }

    })
    .when('/signup',{
        controller:"UserSignupCtrl",
        templateUrl:"app/components/user/controllers/
           signup/user-signup.html",
        title:"Sign up",
        data: {
            needsAuth : false,
        }

    })
    .when('/change_password',{
        controller:"ChangePasswordCtrl",
        templateUrl:"app/components/user/controllers/
           change-password/change-password.html",
        title:"Sign up",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator, USER_ROLES.
               researcher]
        }

    })
    .when('/filters',{
        controller:"FiltersCtrl",
        templateUrl:"app/components/filters/controllers/
           filters/filters.html",
        title:"Nome",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator, USER_ROLES.
               researcher]
        }
```

```
})

.when('/predefined_questions',{
    controller:"QuestioningCtrl",
    templateUrl:"app/components/questioning/
       controllers/questioning/questioning.html",
    title:"Nome",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.researcher,
           USER_ROLES.admin, USER_ROLES.moderator]
    }

})
.when('/project/:idProject/training_set',{
    controller:"TrainingSetCtrl",
    templateUrl:"app/components/training-set/training
       -set.html",
    title:"Nome",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
.when('/project/:idProject/matrices',{
    controller:"MatrixListCtrl",
    templateUrl:"app/components/matrix/controllers/
       list/matrix-list.html",
    title:"Nome",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }
```

```
})
.when('/project/:idProject/matrix/add',{
    controller:"MatrixAddCtrl",
    templateUrl:"app/components/matrix/controllers/
       add/matrix-add.html",
    title:"Nome",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
.when('/project/:idProject/matrix/:id',{
    controller:"MatrixViewCtrl",
    templateUrl:"app/components/matrix/controllers/
       view/matrix-view.html",
    title:"Nome",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
.when('/project/:idProject/matrix/:id/update',{
    controller:"MatrixUpdateCtrl",
    templateUrl:"app/components/matrix/controllers/
       update/matrix-update.html",
    title:"Nome",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }
```

```
})

.when('/user/profile',{
    controller:"UserViewCtrl",
    templateUrl:"app/components/user/controllers/user
       -view/user-view.html",
    title:"User profile",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
.when('/admin/users',{
    controller:"AdminViewUsersCtrl",
    templateUrl:"app/components/admin/controllers/
       view-users/admin-view-users.html",
    title:"Admin - Users",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin]
    }

})
.when('/project/:id/update',{
    controller:"ProjectUpdateCtrl",
    templateUrl:"app/components/project/controllers/
       update/project-update.html",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator]
    }

})
.when('/project/:id',{
    controller:"ProjectViewCtrl",
```

```
        templateUrl:"app/components/project/controllers/
           view/project-view.html",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator, USER_ROLES.
               researcher]
        }

    })
    .when('/project/:idProject/datasets',{
        controller:"DatasetListCtrl",
        templateUrl:"app/components/dataset/controllers/
           list/dataset-list.html",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator, USER_ROLES.
               researcher]
        }

    })
    .when('/project/:idProject/dataset/add',{
        controller:"DatasetAddCtrl",
        templateUrl:"app/components/dataset/controllers/
           add/dataset-add.html",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator]
        }

    })
    .when('/project/:idProject/dataset/:idDataset',{
        controller:"DatasetViewCtrl",
        templateUrl:"app/components/dataset/controllers/
           view/dataset-view.html",
        data: {
            needsAuth : true,
```

```
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator, USER_ROLES.
               researcher]
        }

    })
    .when('/project/:idProject/dataset/:idDataset/tweets/
       import',{
        controller:"ImportTweetCtrl",
        templateUrl:"app/components/tweet/controllers/
           import/tweet-import.html",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator]
        }

    })
    .when('/project/:idProject/dataset/:idDataset/tweet/:
       idTweet',{
        controller:"TweetViewCtrl",
        templateUrl:"app/components/tweet/controllers/
           view/tweet-view.html",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
               USER_ROLES.moderator, USER_ROLES.
               researcher]
        }

    })
    .when('/project/:idProject/dataset/:idDataset/tweet/:
       idTweet/view-conversation',{
        controller:"ViewConversationCtrl",
        templateUrl:"app/components/tweet/controllers/
           view-conversation/view-conversation.html",
        data: {
            needsAuth : true,
            authorizedRoles: [USER_ROLES.admin,
```

```
            USER_ROLES.moderator, USER_ROLES.
            researcher]
    }

})
.when('/project/:idProject/dataset/:idDataset/update
   ',{
    controller:"DatasetUpdateCtrl",
    templateUrl:"app/components/dataset/controllers/
       update/dataset-update.html",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator]
    }

})
.when('/project/:idProject/categories',{
    controller:"CategoryListCtrl",
    templateUrl:"app/components/category/controllers/
       list/category-list.html",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
.when('/project/:idProject/category/add',{
    controller:"CategoryAddCtrl",
    templateUrl:"app/components/category/controllers/
       add/category-add.html",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }
```

```
})
.when('/project/:idProject/category/:id/update',{
    controller:"CategoryUpdateCtrl",
    templateUrl:"app/components/category/controllers/
       update/category-update.html",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
.when('/project/:idProject/category/:id',{
    controller:"CategoryViewCtrl",
    templateUrl:"app/components/category/controllers/
       view/category-view.html",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
.when('/projects',{
    controller:"ProjectListCtrl",
    templateUrl:"app/components/project/controllers/
       list/project-list.html",
    title:"Projects",
    data: {
        needsAuth : true,
        authorizedRoles: [USER_ROLES.admin,
           USER_ROLES.moderator, USER_ROLES.
           researcher]
    }

})
```

```
        .otherwise({redirectTo:'/'});

    $authProvider.loginUrl = 'projetos/sonda/api/users/json/
       login';




    $httpProvider.interceptors.push(['$q', '$location', '
       $localStorage','$rootScope',function ($q, $location,
       $localStorage,$rootScope) {
        return {
            // 'request': function (config) {
            //     config.headers = config.headers || {};
            //     if ($localStorage.token) {
            //         config.headers.apikey = $localStorage.
               token;
            //     }
            //     return config;
            // },
            'responseError': function (response) {
                if (response.status === 401) {
                    delete $localStorage.token;
                    $location.path('/login');
                }
                if(response.status === 403){
                    delete $localStorage.token;
                    $location.path('/error');
                }
                return $q.reject(response);
            }
        };
    }]);


}]);

angular.module('snaApp2').run(function($rootScope, $location,
```

```
    $localStorage,$auth,Session) {


    $rootScope.$on( "$routeChangeStart", function(event, next
       ) {

        // var needsAuth = next.data.needsAuth;
        // var authorizedRoles = next.data.authorizedRoles;
        window.onbeforeunload = function(event) {
            $rootScope.$broadcast('savestate');
        }

        if(next.data.needsAuth){
            if (!Session.isAuthorized(next.data.
               authorizedRoles)) {
                event.preventDefault();
                if ($auth.isAuthenticated()) {
                    $location.path("/error");
                } else {
                    $location.path("/login");
                }
            }

        }



    });
});
angular.module('snaApp2').controller('CategoryUpdateCtrl',['
   $scope','$timeout','$routeParams','$location','Category',
   function($scope, $timeout,$routeParams, $location,Category
   ){

        $scope.id = $routeParams.id;
        $scope.error = false;
        $scope.msg = "";
        $scope.categoriesTypes = {"SPACES_OF_POSSIBILITY":"
           Espa os de possibilidade", "RELEVANT_PROCESS":"
```

```
        Fatores e circunst ncias"};

    $scope.category = Category.get({idProject:
        $routeParams.idProject, id:$routeParams.id});

    $scope.update = function(){
            console.log($scope.category.color);

            if(!Array.isArray($scope.category.keywords)){
                    var str = $scope.category.keywords;

                    var res = str.split(',');

                    for (var i = 0; i < res.length - 1; i
                        ++) {
                            res[i] = res[i].trim();
                    }

                    $scope.category.keywords = res;
            }

            $scope.category.$update({idProject:
                $routeParams.idProject},function sucess(){
                    $location.path('/project/'+
                        $routeParams.idProject+"/
                        categories");
            }, function error(res){
                    $timeout(function(){
                            $scope.error = true;
                            $scope.msg = res.data.error;
                    },0);
            });

    };

}]);describe('CategoryUpdateCtrl', function() {

    beforeEach(module('snaApp2'));
```

```
        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('CategoryUpdateCtrl', {$scope: scope
         });
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('CategoryViewCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('CategoryViewCtrl', {$scope: scope})
         ;
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('CategoryViewCtrl',
   ['$scope','$routeParams', 'Category', '$location','
   modalService',function($scope,$routeParams, Category,
   $location,modalService){

        $scope.id = $routeParams.id;
```

```
        $scope.idProject = $routeParams.idProject;

        $scope.category = Category.get({idProject:
           $routeParams.idProject, id:$routeParams.id});

        $scope.categoriesTypes = {"SPACES_OF_POSSIBILITY":"
           Espa os de possibilidade", "RELEVANT_PROCESS":"
           Fatores e circunst ncias"};



        $scope.delete = function () {

                var modalOptions = {
                        closeButtonText: 'Cancelar',
                        actionButtonText: 'Excluir categoria
                           ',
                        headerText: 'Excluir ' + $scope.
                           category.name + '?',
                        bodyText: 'Tem certeza que deseja
                           excluir esta categoria?'
                };

                modalService.showModal({}, modalOptions).then
                   (function (result) {
                        Category.delete({idProject: $scope.
                           idProject, id:$scope.category.id},
                            function(){
                                $location.path("/project/"+
                                   $scope.idProject+"/
                                   categories");
                        });

                });
        };

}]);describe('CategoryListCtrl', function() {

        beforeEach(module('snaApp2'));
```

```
        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('CategoryListCtrl', {$scope: scope})
         ;
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});
angular.module('snaApp2').controller('CategoryListCtrl', ['
   $scope', '$route','Category','Project','StorageProject', '
   modalService','$location',function($scope, $route,Category
   ,Project,StorageProject,modalService,$location){

        $scope.projectId = StorageProject.getProjectId();

        Project.get({id:$scope.projectId},function sucess(
           data){
                $scope.categories = data.categories;
        });

        $scope.delete = function (category) {

                var modalOptions = {
                        closeButtonText: 'Cancelar',
                        actionButtonText: 'Excluir categoria
                           ',
                        headerText: 'Excluir ' + category.
                           name + '?',
                        bodyText: 'Tem certeza que deseja
                           excluir esta categoria?'
                };
```

```
                modalService.showModal({}, modalOptions).then
                   (function (result) {
                        Category.delete({idProject:$scope.
                           projectId , id: category.id},
                           function(){
                                $route.reload();
                        });

                });
        };



}]);angular.module('snaApp2').controller('CategoryAddCtrl',['
   $scope','$timeout','$location','$routeParams','Category',
   function($scope,$timeout, $location,  $routeParams,
   Category){

        $scope.category = new Category();
        $scope.error = false;
        $scope.msg = "";
        $scope.categoriesTypes = {"SPACES_OF_POSSIBILITY":"
           Espa os de possibilidades", "RELEVANT_PROCESS":"
           Fatores e circunst ncias"};

        $scope.add = function(){

                if(!Array.isArray($scope.category.keywords)){
                        var str = $scope.category.keywords;

                        var res = str.split(',');

                        for (var i = 0; i < res.length - 1; i
                           ++) {
                                res[i] = res[i].trim();
                        }

                        $scope.category.keywords = res;
```

```
                }



                $scope.category.$saveCategory({idProject:
                   $routeParams.idProject},function sucess(){
                        $location.path('/project/'+
                           $routeParams.idProject+"/
                           categories");

                }, function error(res){
                        $timeout(function(){
                                $scope.error = true;
                                $scope.msg = res.data.error;
                        },0);
                });



        };


}]);describe('CategoryAddCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('CategoryAddCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));
```

```
});describe('categoryService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(categoryService) {

        //expect(categoryService.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').factory('Category', ['$resource
   ','BASE_URL', function($resource,BASE_URL){
        return $resource(BASE_URL + 'categories/json/:
           idProject/:id',{idProject:'@idProject', id:'@id
           '},{

                update:{
                        method:"PUT",
                        url:BASE_URL + "categories/json/:
                           idProject",
                        params:{idProject:'@_idProject'}
                },
                saveCategory:{
                        method:"POST",
                        url:BASE_URL + "categories/json/:
                           idProject",
                        params:{idProject:'@_idProject'}
                },
                filterByProject:{
                        method:"GET",
                        url:BASE_URL + "categories/json/
                           filter_by_project/:idProject",
                        params:{idProject:'@_idProject'},
                        isArray:true
                }

        });
```

```
}]);


describe('MatrixUpdateCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('MatrixUpdateCtrl', {$scope: scope})
         ;
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('MatrixUpdateCtrl',['
   $scope','StorageProject','Project','Matrix','$location','
   $routeParams','$timeout',function($scope,StorageProject,
   Project,Matrix,$location,$routeParams,$timeout){


    $scope.projectId = StorageProject.getProjectId();

    Project.get({id:$scope.projectId},function sucess(data){
        $scope.project = data;
    });

    $scope.matrix = Matrix.get({idProject : $routeParams.
       idProject, id: $routeParams.id, populated:0});
    $scope.error = false;
    $scope.msg = "";

    $scope.update = function(){
```

```
        $scope.matrix.$update({idProject:$routeParams.
           idProject},function sucess(){
            $location.path('/project/'+$routeParams.idProject
               +"/matrices");

        }, function error(res){
            $timeout(function(){
                $scope.error = true;
                $scope.msg = res.data.error;
            },0);
        });
    }

}]);angular.module('snaApp2').controller('MatrixViewCtrl',['
   $scope','$routeParams','Matrix','modalService','$location
   ','StorageProject','Project','Tweet','FileSaver', 'Blob','
   spinnerService',function($scope,$routeParams,Matrix,
   modalService,$location,StorageProject,Project,Tweet,
   FileSaver, Blob,spinnerService){


    $scope.idProject = StorageProject.getProjectId();
    $scope.id = $routeParams.id;

    Project.get({id:$scope.idProject},function sucess(data){
        $scope.project = data;

    });


   Matrix.get({idProject : $routeParams.idProject, id:
      $routeParams.id, populated:1},function sucess(data){
       $scope.matrix = data;
   });

    $scope.retrieveNames = function(tipo,id){

     var index = arrayObjectIndexOf($scope.project[tipo], id,
         "id");
```

```
 return $scope.project[tipo][index].name;

};

$scope.retrieveRows = function(index){

    var init = index * $scope.matrix.columnsIds.length;
    var result = [];

    for(var i=init; i < init + $scope.matrix.columnsIds.
       length; i++){
        result.push($scope.matrix.elements[i]);
    }

    return result;

};


function arrayObjectIndexOf(myArray, searchTerm, property
   ) {
    for(var i = 0, len = myArray.length; i < len; i++) {
        if (myArray[i][property] === searchTerm) return i
           ;
    }
    return -1;
}
$scope.delete = function () {

var modalOptions = {
    closeButtonText: 'Cancelar',
    actionButtonText: 'Excluir matriz',
    headerText: 'Excluir ' + $scope.matrix.name + '?',
    bodyText: 'Tem certeza que deseja excluir esta matriz
       ?'
};

modalService.showModal({}, modalOptions).then(function (
```

```
    result) {
      Matrix.delete({idProject: $scope.idProject, id:$scope
          .matrix.id}, function(){
          $location.path("/project/"+$scope.idProject+"/
              matrices");
      });

      });
};

/***
 *
 *
 *
 *
 *
 *
 * */

$scope.itensPerPage = 10;

var paramsDefault = function(){
    return {
        filter : "byRelation",
        idDataset: $scope.matrix.datasetsIds,
        skip:($scope.currentPage -1)*$scope.itensPerPage,
        amount:$scope.itensPerPage
    }
};

$scope.configRequest = function(indexRow,indexCol){

    var eIndex = indexRow * $scope.matrix.rowsIds.length
        + indexCol;

        $scope.currentPage = 1;
        var params = paramsDefault();
        params["rowId"] = $scope.matrix.elements[eIndex].
            rowId;
```

```
        params["columnId"] = $scope.matrix.elements[
           eIndex].columnId;
        params["relation"] = $scope.matrix.relation;

        $scope.lastConfig = params;
        $scope.getPage();

};


$scope.getPage = function(){

    $scope.disablePagination = true;
    spinnerService.show('matrixSpinner');

    Tweet.listInAnInterval($scope.lastConfig,function(
       results){
        $scope.tweets = results.data;


        for (var i = 0; i < $scope.tweets.length; i++) {
            $scope.tweets[i].selTags = [];

            if($scope.tweets[i].tags.length > 0){
                $scope.tweets[i].selTags = [];

                for(var j=0; j < $scope.tweets[i].tags.
                   length; j++){

                    $scope.tweets[i].selTags.push($scope.
                       tweets[i].tags[j].category);

                }


            }
        }

        $scope.count=results.count;
```

```
            $scope.showTweets = true;
            $scope.skipped = ($scope.currentPage -1)*$scope.
               itensPerPage;
            $scope.disablePagination = false;
            spinnerService.hide('matrixSpinner');

        },function error(err){

            console.log(err);
            $scope.disablePagination = false;
            spinnerService.hide('matrixSpinner');
        });
    };

    $scope.changePage = function(){

        $scope.lastConfig["skip"] = ($scope.currentPage -1)*
           $scope.itensPerPage;

        $scope.getPage();


    };

    $scope.changeAmount = function(_qtt){
        console.log(_qtt);
        $scope.currentPage = 1;
        $scope.lastConfig["skip"] = 0;
        $scope.lastConfig["amount"] = _qtt;
        $scope.getPage();
    };

    $scope.toCSV = function(){

        Tweet.exportToCSV($scope.lastConfig,function(results)
           {
            var blob = new Blob([results.values], {type: "
               application/csv;charset=utf-8"});
            FileSaver.saveAs(blob, 'export.csv');
```

```
        });

    };

    //tirei a parte de add remover tag, se for preciso, pegar
        do que j  est  feito.

}]);describe('MatrixViewCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('MatrixViewCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('MatrixListCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('MatrixListCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);
```

```
        }));

});angular.module('snaApp2').controller('MatrixListCtrl',['
   $scope','StorageProject','Project','Matrix','modalService
   ','$route',function($scope,StorageProject,Project,Matrix,
   modalService,$route){


    $scope.projectId = StorageProject.getProjectId();

    Project.get({id:$scope.projectId},function sucess(data){
        $scope.matrices = data.matrices;
    });

    $scope.delete = function (matrix) {

        var modalOptions = {
            closeButtonText: 'Cancelar',
            actionButtonText: 'Excluir matriz',
            headerText: 'Excluir ' + matrix.name + '?',
            bodyText: 'Tem certeza que deseja excluir esta
               matriz?'
        };

        modalService.showModal({}, modalOptions).then(
           function (result) {

            Matrix.delete({idProject: $scope.projectId, id:
               matrix.id},function(){
                $route.reload();
            });

        });
    };




}]);angular.module('snaApp2').controller('MatrixAddCtrl',['
```

```
    $scope','StorageProject','Project','Matrix','$location','
    $routeParams','$timeout',function($scope,StorageProject,
    Project,Matrix,$location,$routeParams,$timeout){


      $scope.projectId = StorageProject.getProjectId();

      Project.get({id:$scope.projectId},function sucess(data){
          $scope.project = data;
      });



      $scope.matrix = new Matrix();
      $scope.error = false;
      $scope.msg = "";

      $scope.add = function(){
          $scope.matrix.$saveMatrix({idProject:$routeParams.
             idProject},function sucess(){
              $location.path('/project/'+$routeParams.idProject
                 +"/matrices");

          }, function error(res){
              $timeout(function(){
                  $scope.error = true;
                  $scope.msg = res.data.error;
              },0);
          });
      }

}]);describe('MatrixAddCtrl', function() {

          beforeEach(module('snaApp2'));

          var scope,ctrl;

      beforeEach(inject(function($rootScope, $controller) {
        scope = $rootScope.$new();
        ctrl = $controller('MatrixAddCtrl', {$scope: scope});
```

```
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').factory('Matrix', ['$resource','
   BASE_URL', function($resource,BASE_URL){
        return $resource(BASE_URL + 'matrices/json/:idProject
           /:id',{idProject:'@idProject', id:'@id'},{

                update:{
                        method:"PUT",
                        url:BASE_URL + "matrices/json/:
                           idProject",
                        params:{idProject:'@_idProject'}
                },
                saveMatrix:{
                        method:"POST",
                        url:BASE_URL + "matrices/json/:
                           idProject",
                        params:{idProject:'@_idProject'}
                },


        });

}]);


describe('matrixService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(matrixService) {

        //expect(matrixService.doSomething()).toEqual('
```

```
            something');

  }));

});describe('ImportTweetCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ImportTweetCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('ImportTweetCtrl',['
   $scope','$routeParams','$location','Tweet','Dataset','
   spinnerService',function($scope,$routeParams,$location,
   Tweet,Dataset,spinnerService){

        $scope.idProject = $routeParams.idProject;
        $scope.idDataset = $routeParams.idDataset;

        $scope.dataset = Dataset.get({idProject: $routeParams
           .idProject, id:$routeParams.idDataset});

        $scope.error = false;
        $scope.msg = "";

        $scope.header = [];
        $scope.delimiters = [",", ";", "|"];
        $scope.enclosures = ["\"" , "'"];
```

```
$scope.getHeader = function(){

        if($scope.delimiter != undefined){

                var splitted = $scope.fileContent.
                   split("\n", 2);

                var header = splitted[0].split($scope
                   .delimiter);

                if(header.length > 1){

                        $scope.header = header;
                        $scope.error = false;

                }else{

                        $scope.delimiter = undefined;
                        $scope.error = true;
                        $scope.msg = "Delimitador
                           incorreto";
                }


        }

};

$scope.importEnabled = function(){

        return $scope.id == undefined || $scope.text
           == undefined || $scope.fromUser ==
           undefined || $scope.createdAt == undefined
           ;

}

$scope.importTweets=function(){
```

```
                spinnerService.show('importSpinner');

                var body = {data:$scope.fileContent,
                        delimiter:$scope.delimiter,
                        enclosure : $scope.enclosure,
                        idName : $scope.id,
                        fromUserName : $scope.fromUser,
                        createdAtName : $scope.createdAt,
                        textName : $scope.text};

                Tweet.importTweets({idDataset:$routeParams.
                   idDataset},body, function sucess(){

                        $scope.dataset.hasTweets = true;

                        $scope.dataset.$update({idProject:
                           $scope.idProject}, function(){
                                $location.path('/project/'+
                                   $routeParams.idProject+'/
                                   dataset/'+$routeParams.
                                   idDataset);
                        });

                        spinnerService.hide('importSpinner');
                },function error(err){
                        console.log(err);
                        spinnerService.hide('importSpinner');
                });


    };

    // $scope.csvToJson=function(csv){
//
    //      var lines=csv.trim().split("\n");
//
//
    // //retirando quebras de linha do texto
    // for(var k=1;k<lines.length;k++){
```

```
    //
        //      if((lines[k].length < 140) || (lines[k].split
           ("|").length===1)){
        //              lines[k+1] = lines[k].concat(" ").
           concat(lines[k+1]);
        //      }
    //
        // }
    //
        // var result = [];
    //
        // var headers=lines[0].split("|");
    //
        // for(var i=1;i<lines.length;i++){
    //
        //      var obj = {};
        //      var currentline=lines[i].split("|");
    //
    //
        //      if(headers.length === currentline.length){
        //              for(var j=0;j<headers.length;j++){
        //                      obj[headers[j]] = currentline
           [j];
        //              }
    //
        //              result.push(obj);
        //      }
    //
        // }
    //
        //        //return result; //JavaScript object
        //        return JSON.stringify(result); //JSON
    //
        //      };

        // window.history.forward(1);


}]);describe('TweetViewCtrl', function() {
```

```
        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('TweetViewCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('TweetViewCtrl',['
   $scope','$route','$routeParams','Tweet', function($scope,
   $route,$routeParams, Tweet){


        $scope.tweet = Tweet.get({id:$routeParams.idTweet});

        $scope.idDataset = $routeParams.idDataset;
        $scope.idProject = $routeParams.idProject;

        $scope.isNullOrEmpty = function (attr){
                return attr === "" || attr === null;
        }

        $scope.setVisibility = function(){

                $scope.tweet.isVisible = !$scope.tweet.
                   isVisible;

                Tweet.setVisibility({idTweet:$scope.tweet.id,
                    visible:$scope.tweet.isVisible},{});
        };
```

```
}]);describe('TweetListCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('TweetListCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('TweetListCtrl',['
   $scope','$routeParams', '$location','Dataset','Tweet', '
   Category', 'Project',function($scope,$routeParams,
   $location,Dataset,Tweet,Category,Project){

        $scope.project = Project.get({id:$routeParams.
           idProject});

        $scope.currentPage = 1;
        $scope.itensPerPage = 10;
        $scope.filter = "default";
        $scope.maxSize = 10;


        var buildParamsReq = function(){
                return {
                        idDataset:$routeParams.idDataset,
                        skip:($scope.currentPage -1)*$scope.
                           itensPerPage,
                        amount:$scope.itensPerPage,
                        filter:$scope.filter,
                        idCategory:$scope.idCategory,
```

```
                searchBy:$scope.searchBy
        };
};

$scope.search = function(){
        $scope.currentPage = 1;
        $scope.filter = "search";
        $scope.getPage();

};

$scope.getPage = function(){

        Tweet.listInAnInterval(buildParamsReq(),
           function(results){
                $scope.tweets = results.data;

                for (var i = 0; i < $scope.tweets.
                   length; i++) {
                        if($scope.tweets[i].
                           categories == null){
                                $scope.tweets[i].
                                   selTags = [];
                        }else{
                                $scope.tweets[i].
                                   selTags = $scope.
                                   tweets[i].
                                   categories;
                        }

                }
                $scope.count=results.count;
        });
};
$scope.getPage();

$scope.changeAmount = function(){
        $scope.currentPage = 1;
        $scope.getPage();
```

```
};

$scope.selectCategory=function(){
        $scope.currentPage = 1;
        $scope.idCategory = $scope.selectedCat.id;
        $scope.filter = 'byCategory';
        $scope.getPage();
};

$scope.hasTweets = function(){
        return $scope.tweets > 0;
};

$scope.toCSV = function(){

        Tweet.exportToCSV(buildParamsReq(),function(
           results){
                var blob = new Blob([results.values],
                    {type: "application/csv;charset=
                   utf-8"});
                saveAs(blob, 'export.csv');
        });

};




$scope.list3 = [{'title': 'Di logo'},{'title': '
   Pol tica'}];

$scope.list1 = [];
$scope.onDrop = function(event,ui,selCategories,
   _idTweet){

        var len =  selTags.length;
        var isNew = true;
        var last = selTags[len - 1];
```

```
                for (var i = 0; i < len - 1; i++) {
                        if(selTags[i].id === last.id){
                                selTags.splice(len - 1,1);
                                isNew = false;
                                break;
                        }
                }

                if(isNew){

                        Tweet.addCategory({idTweet:_idTweet,
                           idCategory:last.id},{});

                }
        };


}]);

angular.module('snaApp2').directive('tooltip', function(){
    return {
        restrict: 'A',
        link: function(scope, element, attrs){
            $(element).hover(function(){
                // on mouseenter
                $(element).tooltip('show');
            }, function(){
                // on mouseleave
                $(element).tooltip('hide');
            });
        }
    };
});

angular.module('snaApp2').controller('ViewConversationCtrl
   ',['$scope','$routeParams','Tweet','spinnerService',
   function($scope,$routeParams,Tweet,spinnerService){
```

```
Tweet.get({id:$routeParams.idTweet},function sucess(
   data){
        $scope.tweet = data;
});

$scope.loadConversation = function() {

        spinnerService.show('conversationSpinner');

        Tweet.viewConversation({idTweet: $routeParams
           .idTweet}, function sucess(data) {

                $scope.conversation = data;
                spinnerService.hide('
                   conversationSpinner');
        },function error(err){
                spinnerService.hide('
                   conversationSpinner');
                $scope.msg = "N o foi poss vel
                   recuperar o di logo. " +
                        "Prov veis motivos:\n\n" +
                        "1. O usu rio alterou a
                           visibilidade de sua conta
                           para privado.\n" +
                        "2. O tweet ou a conta do
                           usu rio foram exclu dos
                           ."
        });

};

$scope.idConv = $routeParams.idTweet;

$scope.hasMore = true;

$scope.getMore = function(){

        var root = $scope.conversation[0].id;
```

```
            var last = $scope.conversation[$scope.
               conversation.length - 1].id;

            Tweet.viewMoreConversation({idRootTweet:root,
               idLastTweet:last},function sucess(data){

                  if(data.length > 0){
                        for (var i = 0; i < data.
                           length; i++){
                              $scope.conversation.
                                 push(data[i]);
                        }
                  }else{
                        $scope.hasMore = false;
                  }

            });

};



$scope.export = function(){

            var arrayData = [];

            for (var i = 0; i < $scope.conversation.
               length; i++ ) {

                  var newObj = {};
                  newObj.text = $scope.conversation[i].
                     text;
                  newObj.username = $scope.conversation
                     [i].username;
                  newObj.id = $scope.conversation[i].id
                     ;
                  newObj.date = $scope.conversation[i].
                     date;
                  arrayData.push(newObj);
```

```
                }

                return arrayData;

        };




}]);describe('ViewConversationCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ViewConversationCtrl', {$scope:
         scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').factory('Tweet', ['$resource','
   BASE_URL', function($resource,BASE_URL){
        return $resource(BASE_URL + 'tweet/json/:id',{id:'
           @_id'},{

                update:{
                        method:"PUT"
                },

                importTweets:{
                        method:"POST",
```

```
            url:BASE_URL + "tweets/csv_to_json/:
               idDataset",
            params:{idDataset:'@_idDataset'},
            isArray:true
    },
    // listInAnInterval:{
    //      method:"GET",
    //      url:"http://localhost/projetos/sna/
       tweets/json/listInAnInterval/:idDataset/:
       skip/:amount",
    //      params:{idDataset:'@_idDataset',skip
       :'@_skip',amount:'@_amount'},
    // }
    listInAnInterval:{
            method:"GET",
            url:BASE_URL + "tweets/json/:
               idDataset",
            params:{idDataset:'@_idDataset'},
    },
    exportToCSV:{
            method:"GET",
            url:BASE_URL + "tweets/csv/:idDataset
               ",
            params:{idDataset:'@_idDataset'},
            transformResponse: function (data) {
               return {values: data} }
            // header:{
            //      'Content-Type':'application/
               csv',
            //      'Content-Disposition':'
               attachment; filename="category.csv
               "'
            // }
    },
    viewConversation:{
            method:"GET",
            url:BASE_URL + "/tweet/json/
               get_conversation/:idTweet",
            params:{idTweet:'@_idTweet'},
```

```
        isArray:true
},
viewMoreConversation:{
        method:"GET",
        url:BASE_URL + "/tweet/json/
           get_more_conversation/:idRootTweet
           /:idLastTweet",
        params:{idRootTweet:'@_idRootTweet',
           idLastTweet:'@_idLastTweet'},
        isArray:true
},
setVisibility:{
        method:"POST",
        url:BASE_URL + "tweet/json/
           set_visibility/:idTweet",
        params:{idTweet:'idTweet'},
},
addTag:{
        method:"POST",
        url:BASE_URL + "/tweet/json/add_tag/:
           idTweet/:idCategory",
        params:{idTweet:'_idTweet',idCategory
           :'_idCategory'},
},
removeTag:{
        method:"POST",
        url:BASE_URL + "tweet/json/remove_tag
           /:idTweet/:idCategory",
        // params:{idTweet:'_idTweet',
           idCategory:'_idCategory'},
},
addClass:{
        method:"POST",
        url:BASE_URL + "/tweet/json/add_class
           /:idTweet/:idCategory",
        params:{idTweet:'_idTweet',idCategory
           :'_idCategory'},
},
removeClass:{
```

```
                        method:"POST",
                        url:BASE_URL + "tweet/json/
                           remove_class/:idTweet/:idCategory
                           ",
                        // params:{idTweet:'_idTweet',
                           idCategory:'_idCategory'},
                },
                countClassesInDataset:{
                        method:"GET",
                        url:BASE_URL + "tweet/json/
                           count_classes_in_dataset/:
                           idProject/:idDataset",
                        isArray:true
                }

        });

}]);describe('tweetService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(tweetService) {

        //expect(tweetService.doSomething()).toEqual('
           something');

  }));

});describe('ErrorCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ErrorCtrl', {$scope: scope});
    }));
```

```
        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('ErrorCtrl',function(
   $scope){


});describe('FiltersCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('FiltersCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('FiltersCtrl',['
   $scope', 'Project','Tweet','FilterService','Session','
   StorageProject','FileSaver', 'Blob','spinnerService',
   function($scope,Project,Tweet,FilterService,Session,
   StorageProject,FileSaver, Blob,spinnerService){

    var idUser = Session.getCookieData().id;


    $scope.state = FilterService;
    $scope.state.hiddenQtt = 0;
```

```
$scope.filters = [{type:"byCategory", label: "Filtrar por
    Espa os de Possibilidade"},
    {type: "searchByTerm", label:"Pesquisar por termo"},
    {type:"default",label:"Sem filtro"},
    {type: "extractFragment",label: "Extrair fragmento"}
    ];

$scope.cutPoints = [{type:"BEGINNING", label:"In cio"},
   {type:'HALF', label:"Meio"}, {type: "END", label:"Fim
   "}];

Project.get({id:StorageProject.getProjectId()},function
   sucess(data){
    $scope.state.selectedProject = data;
});

$scope.selectedDatasets = [];

$scope.showFilters = function(){

    if (typeof $scope.selectedDatasets === "undefined"){
        return false;
    }

    return $scope.selectedDatasets.length > 0 ? true :
       false;
};

$scope.state.itensPerPage = 10;

var paramsDefault = function(){
    return {
        filter : $scope.state.filterSelected,
        idDataset: $scope.selectedDatasets,
        skip:($scope.state.currentPage -1)*$scope.state.
           itensPerPage,
        amount:$scope.state.itensPerPage
    }
};
```

```
$scope.configFilterByCategory = function(_idCategory){

    if (typeof _idCategory !== "undefined"){
        $scope.state.currentPage = 1;
        var params = paramsDefault();
        params["idCategory"] = _idCategory;

        $scope.state.lastConfig = params;
        $scope.getPage();
    }
};

$scope.configSearchByTerm = function(_term){
    if (typeof _term !== "undefined"){
        $scope.state.currentPage = 1;
        var params = paramsDefault();
        params["searchBy"] = _term;

        $scope.state.lastConfig = params;
        $scope.getPage();
    }

};

$scope.configFilterDefault = function(){

    $scope.state.currentPage = 1;
    if($scope.state.filterSelected=="default"){
        $scope.state.lastConfig = paramsDefault();
        $scope.getPage();
    }
};

$scope.configExtractFragment = function () {

    $scope.state.currentPage = 1;
    var params = paramsDefault();
    params["fragmentSize"] = $scope.state.fragmentSize;
```

```
    params["cutPoint"] = $scope.state.cutPoint;

    $scope.state.lastConfig = params;
    $scope.getPage();

};

// $scope.dateStringToTimeStamp = function(timestamp){
//
//     return Date.parse(timestamp);
// };

$scope.getPage = function(){

    spinnerService.show('filterSpinner');
    $scope.disablePagination = true;

    Tweet.listInAnInterval($scope.state.lastConfig,
       function sucess(results){
        $scope.state.tweets = results.data;


        for (var i = 0; i < $scope.state.tweets.length; i
           ++) {
            if($scope.state.tweets[i].tags.length == 0){
                $scope.state.tweets[i].selTags = [];

            }else{

                $scope.state.tweets[i].selTags = [];

                for(var j=0; j < $scope.state.tweets[i].
                   tags.length; j++){

                    for(var k=0; k < $scope.state.tweets[
                       i].tags[j].addedBy.length ;k++){

                        if($scope.state.tweets[i].tags[j
                           ].addedBy[k].idUser == idUser)
```

```
                                {
                                  $scope.state.tweets[i].
                                     selTags.push($scope.state.
                                     tweets[i].tags[j].category
                                     );
                            }
                        }

                    }


                }
            }
            if($scope.state.filterSelected != '
               extractFragment'){
                $scope.state.count=results.count;
            }else{
                $scope.state.count = $scope.state.
                   fragmentSize;
            }
            $scope.state.skipped = ($scope.state.currentPage
               -1)*$scope.state.itensPerPage;
            $scope.state.showTweets = true;
            $scope.disablePagination = false;
            spinnerService.hide('filterSpinner');

        },function error(err){
            console.log(err);
            $scope.disablePagination = false;
            spinnerService.hide('filterSpinner');
        });
    };

    $scope.changePage = function(){

        $scope.state.lastConfig["skip"] = ($scope.state.
           currentPage -1)*$scope.state.itensPerPage - $scope
           .state.hiddenQtt;
```

```
    $scope.getPage();


};

$scope.changeAmount = function(_qtt){
    console.log(_qtt);
    // $scope.state.currentPage = 1;
    $scope.state.lastConfig["skip"] = 0;
    $scope.state.lastConfig["amount"] = _qtt;
    $scope.getPage();
};

$scope.toCSV = function(){

    Tweet.exportToCSV($scope.state.lastConfig,function(
       results){

        var blob = new Blob([results.values], {type: "
           application/csv;charset=utf-8"});
        FileSaver.saveAs(blob, 'export.csv');
    });

};

$scope.setVisibility = function(tweet){

    tweet.isVisible = !tweet.isVisible;
    $scope.state.hiddenQtt +=1;

    Tweet.setVisibility({idTweet:tweet.id, visible:tweet.
       isVisible},{});
};

$scope.onDrop = function(event,ui,selTags,_idTweet){

    var len =  selTags.length;
    var isNew = true;
    var last = selTags[len - 1];
```

```
        for (var i = 0; i < len - 1; i++) {
            if(selTags[i].id === last.id){
                selTags.splice(len - 1,1);
                isNew = false;
                break;
            }
        }

        if(isNew){

            Tweet.addTag({idTweet:_idTweet, idCategory:last.
               id},{});

        }

    };

    $scope.removeCat = function (selTags, idCat, _idTweet){
        var len =  selTags.length;
        for (var i = 0; i < len; i++) {
            if(selTags[i].id === idCat){
                selTags.splice(i,1);
                break;
            }
        }

        Tweet.removeTag({idTweet:_idTweet, idCategory: idCat
           },{});
    }


}]);angular.module('snaApp2').factory('FilterService',['
   $rootScope',function($rootScope) {

        var service = {


    };
```

```
        function saveState() {
                sessionStorage.FilterService = angular.toJson
                   (service);
        console.log("Salvando" );
        }
        function restoreState() {
                service = angular.fromJson(sessionStorage.
                   FilterService);
        console.log("Recuperado");
        }
        $rootScope.$on("savestate", saveState);

        if (sessionStorage.FilterService){
                restoreState();
        }

        return service;
}]);describe('filterService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(filterService) {

        //expect(filterService.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').controller('ProjectUpdateCtrl
   ',['$scope','$timeout','$routeParams','$location','Project
   ',function($scope, $timeout, $routeParams, $location,
   Project){


$scope.project = Project.get({id:$routeParams.id});
$scope.projects = Project.query();
$scope.error = false;
$scope.msg = "";
```

```
$scope.update = function(){

        $scope.project.$update(function sucess(){
                        $scope.$parent.getProject();
                        $location.path('/projects');
                }, function error(res){

                        $timeout(function(){
                                $scope.error = true;
                                $scope.msg = res.data.error;
                        },0);

                });

};

}]);describe('ProjectUpdateCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ProjectUpdateCtrl', {$scope: scope
         });
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('ProjectSetCtrl',['
   $scope','$timeout', '$location','USER_ROLES', '$auth','
   Session','Project','StorageProject',function($scope,
```

```
    $timeout,$location,USER_ROLES,$auth,Session,Project,
    StorageProject){

     $scope.projects = Project.query();

     $scope.selectedProject = null;

     $scope.setProject = function () {

         StorageProject.setProjectId($scope.selectedProject.id
            );
         $scope.$parent.getProject();

     };

     $scope.isActive = function(){
         return $auth.isAuthenticated();
     };


}]);describe('SetCtrl', function() {

         beforeEach(module('snaApp2'));

         var scope,ctrl;

     beforeEach(inject(function($rootScope, $controller) {
       scope = $rootScope.$new();
       ctrl = $controller('SetCtrl', {$scope: scope});
     }));

         it('should ...', inject(function() {

                 expect(1).toEqual(1);

         }));

});
angular.module('snaApp2').controller('ProjectViewCtrl', ['
```

```
$scope','$route','$location','$routeParams','Project','
Category', 'modalService',function($scope,$route,$location
,$routeParams, Project, Category, modalService){

      $scope.project = Project.get({id:$routeParams.id},
         function sucess(){
               $scope.hasDatasets = function(){
                     return $scope.project.datasets.length
                         > 0;
               };

               $scope.hasCategories = function(){
                     return $scope.project.categories.
                        length > 0;
               };
      });

      $scope.deleteCategory = function (_idCat) {

               var modalOptions = {
                     closeButtonText: 'Cancelar',
                     actionButtonText: 'Excluir Categoria
                        ',
                     headerText: 'Excluir ' + $scope.
                        project.categories.name + '?',
                     bodyText: 'Tem certeza que deseja
                        excluir esta categoria?'
               };

               modalService.showModal({}, modalOptions).then
                  (function (result) {
                     Category.delete({idProject: $scope.
                        project.id, id:_idCat}, function()
                        {
                           $route.reload();
                     });
               });
      };
```

```
        $scope.delete = function (_id,name) {

                var modalOptions = {
                        closeButtonText: 'Cancelar',
                        actionButtonText: 'Excluir Projeto',
                        headerText: 'Excluir ' + name + '?',
                        bodyText: 'Tem certeza que deseja
                           excluir este projeto?'
                };

                modalService.showModal({}, modalOptions).then
                   (function (result) {
                        Project.delete({id:_id}, function(){
                                $location.path('/set_project
                                   ');
                        });
                });
        };

}]);describe('ProjectViewCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ProjectViewCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('ProjectListCtrl', function() {

        beforeEach(module('snaApp2'));
```

```
        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ProjectListCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});
angular.module('snaApp2').controller('ProjectListCtrl', ['
   $scope', '$route','Project', 'modalService', function(
   $scope, $route,Project,modalService){

        $scope.projects = Project.query();

        $scope.hasProjects = function(){
                return $scope.projects.length > 0;
        };

        $scope.delete = function (_id,name) {

                var modalOptions = {
                        closeButtonText: 'Cancelar',
                        actionButtonText: 'Excluir Projeto',
                        headerText: 'Excluir ' + name + '?',
                        bodyText: 'Tem certeza que deseja
                           excluir este projeto?'
                };

                modalService.showModal({}, modalOptions).then
                   (function (result) {
                        Project.delete({id:_id}, function(){
                                $route.reload();
```

```
                        });
                });
        };

}]);describe('ProjectAddCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ProjectAddCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('ProjectAddCtrl',['
   $scope','$timeout','$location','Project',function($scope,
   $timeout,$location, Project){

        $scope.project = new Project();
        $scope.projects = Project.query();
        $scope.error = false;
        $scope.msg = "";

        $scope.add = function(){

                $scope.project.$save(function sucess(){
                        $location.path('/set_project');
                }, function error(res){

                        $timeout(function(){
                                $scope.error = true;
                                $scope.msg = res.data.error;
```

```
                        },0);

                });


        };


}]);angular.module('snaApp2').factory('Project', ['$resource
   ','BASE_URL', function($resource,BASE_URL){
        return $resource(BASE_URL + 'projects/json/:id',{id:'
           @_id'},{

                update:{
                        method:"PUT"
                }

        });

}]);describe('storageProject', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(storageProject) {

        //expect(storageProject.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').factory('StorageProject',['
   $cookies',function($cookies) {

        return {
                setProjectId: function(project_id) {


                        $cookies.put("project_id", project_id
                           );
```

```
                },
                getProjectId: function() {
                        return $cookies.get("project_id");

                },
                deleteProjectId: function() {

                        $cookies.remove("project_id");

                }
        }


}]);

describe('projectService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(projectService) {

        //expect(projectService.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').controller('IndexController',['
   $scope','$timeout', '$location','USER_ROLES', '$auth','
   Session','Project','StorageProject',function($scope,
   $timeout,$location,USER_ROLES,$auth,Session,Project,
   StorageProject){

    $scope.userRoles = USER_ROLES;
    $scope.isAuthorized = Session.isAuthorized;

    $scope.getUser = function(){
        $timeout(function(){
            $scope.userActive = Session.getCookieData();
```

```
        },0);
    };

    $scope.getUser();

    $scope.getProject = function(){

        var _id = StorageProject.getProjectId();

        Project.get({id:_id},function sucess(data){
            $scope.selectedProject = data;
        });

    };

    $scope.getProject();

    $scope.projectFixed = function(){
        return typeof $scope.selectedProject !== "undefined"
           && $scope.selectedProject !== null;
    }

    $scope.isActive = function(){
        return $auth.isAuthenticated();
    };

    $scope.logout = function(){
        $auth.logout();
        Session.clearCookieData();
        $scope.userActive = null;
        $location.path("#/login");
        StorageProject.deleteProjectId();
        $scope.selectedProject = null;

    };


}]);
```

```
angular.module('snaApp2').controller('TrainingSetCtrl',['
   $scope','Project','StorageProject','Tweet',function($scope
   ,Project,StorageProject,Tweet){


    Project.get({id:StorageProject.getProjectId()},function
       sucess(data){
        $scope.selectedProject = data;
        console.log(data);
        $scope.categories = data.categories;
    });

    $scope.itensPerPage = 10;
    $scope.selectedDataset;

    var paramsDefault = function(){
        return {
            filter : "default",
            idDataset: $scope.selectedDataset.id,
            skip:($scope.currentPage -1)*$scope.itensPerPage,
            amount:$scope.itensPerPage
        }
    };

    $scope.countClassesInDataset = [];

    $scope.getCountClassesInDataset = function(){

        Tweet.countClassesInDataset({idProject:$scope.
           selectedProject.id,idDataset:$scope.
           selectedDataset.id},function sucess(data){
            $scope.countClassesInDataset = data;

            console.log(data);
        });
```

```
};

$scope.getCount = function(idCat){

    for (var i = 0; i < $scope.countClassesInDataset.
       length; i++) {
        if($scope.countClassesInDataset[i].id === idCat){
            return $scope.countClassesInDataset[i].count;
        }
    }

}

$scope.defDataset = function(){

    $scope.currentPage = 1;
    $scope.lastConfig = paramsDefault();
    $scope.lastConfig["includeRepeated"] = $scope.
       includeRepeated;
    $scope.getCountClassesInDataset();
    $scope.getPage();

};


$scope.dateStringToTimeStamp = function(timestamp){

    return Date.parse(timestamp);
};

$scope.getPage = function(){

    Tweet.listInAnInterval($scope.lastConfig,function(
       results){
        $scope.tweets = results.data;

        for (var i = 0; i < $scope.tweets.length; i++) {
```

```
                $scope.tweets[i].selClasses = [];

                if($scope.tweets[i].classes.length > 0){
                    $scope.tweets[i].selClasses = [];

                    for(var j=0; j < $scope.tweets[i].classes
                       .length; j++){

                        $scope.tweets[i].selClasses.push(
                           $scope.tweets[i].classes[j].
                           category);

                    }


                }
            }

            $scope.count=results.count;
            $scope.showTweets = true;

        });
};

$scope.changePage = function(){

    $scope.lastConfig["skip"] = ($scope.currentPage -1)*
       $scope.itensPerPage;

    $scope.getPage();


};

$scope.changeAmount = function(_qtt){
    console.log(_qtt);
    $scope.currentPage = 1;
    $scope.lastConfig["skip"] = 0;
    $scope.lastConfig["amount"] = _qtt;
```

```
        $scope.getPage();
    };

    $scope.onDrop = function(event,ui,selClasses,_idTweet){

        var len =  selClasses.length;
        var isNew = true;
        var last = selClasses[len - 1];

        for (var i = 0; i < len - 1; i++) {
            if(selClasses[i].id === last.id){
                selClasses.splice(len - 1,1);
                isNew = false;
                break;
            }
        }

        if(isNew){

            Tweet.addClass({idTweet:_idTweet, idCategory:last
               .id},{},function sucess(){
                $scope.getCountClassesInDataset();});


        }

    };

    $scope.removeCat = function (selClasses, idCat, _idTweet)
       {
        var len =  selClasses.length;
        for (var i = 0; i < len; i++) {
            if(selClasses[i].id === idCat){
                selClasses.splice(i,1);
                break;
            }
        }

        Tweet.removeClass({idTweet:_idTweet, idCategory:
```

```
            idCat},{},function sucess(){
              $scope.getCountClassesInDataset();
          });

    }


}]);describe('TrainingSetCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('TrainingSetCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('UserViewCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('UserViewCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);
```

```
        }));

});angular.module('snaApp2').controller('UserViewCtrl',['
   $scope',function($scope){


        $scope.user = $scope.$parent.userActive;

}]);angular.module('snaApp2').controller('UpdateCtrl',
   function($scope){


});describe('UpdateCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('UpdateCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('UserUpdateCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('UserUpdateCtrl', {$scope: scope});
    }));
```

```
        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('UserUpdateCtrl',
   function($scope){


});angular.module('snaApp2').controller('UserSignupCtrl',['
   $scope','$location','User','$timeout', function($scope,
   $location, User,$timeout){

        $scope.user = new User();

        $scope.error = false;
        $scope.msg = "";

        $scope.signup = function(){
                $scope.user.$signup(function success(){
                        $location.path('/');
                }, function error(res){

                        $timeout(function(){
                                $scope.error = true;
                                $scope.msg = res.data.error;

                        },0);
                });
        };

}]);describe('UserSignupCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;
```

```
    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('UserSignupCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('UserLoginCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('UserLoginCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('UserLoginCtrl',['
   $scope','$rootScope','$localStorage', '$location','User','
   Session','$auth','$timeout',function($scope, $rootScope,
   $localStorage, $location,User,Session,$auth,$timeout){



    $scope.msg = "";
    $scope.error = false;

    $scope.login = function() {
```

```
        $auth.login($scope.user)
            .then(function(res) {
                $auth.setToken(res.token);

                var payload = $auth.getPayload();
                var user = payload["data"];

                Session.setCookieData(user.id,user.role,user.
                   name,user.email);

                $location.path('/set_project');

                $scope.$parent.getUser();


            })
            .catch(function(error) {
                $timeout(function(){
                    $scope.msg = error.data.message;
                    $scope.error = true;
                },0);

            });
    };




}]);describe('ChangePasswordCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('ChangePasswordCtrl', {$scope: scope
         });
    }));
```

```
        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('ChangePasswordCtrl
   ',['$scope','User','$location','Session',function($scope,
   User,$location,Session){



    var _id = Session.getCookieData().id;
    console.log(_id);
    // $scope.user = User.get({id:_id});


    $scope.user = new User();
    $scope.error = false;
    $scope.msg = "";

    $scope.changePw = function(){
        $scope.user.$changePassword({id:_id},function success
           (){
            $location.path('/');
        }, function error(res){

            $timeout(function(){
                $scope.error = true;
                $scope.msg = res.data.error;

            },0);
        });
    };

}]);describe('userService', function() {

  beforeEach(module('snaApp2'));
```

```
  it('should ...', inject(function(userService) {

        //expect(userService.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').factory('User', ['$resource','
   BASE_URL', function($resource,BASE_URL) {

        return $resource(BASE_URL + 'users/json/:id',{id:'
           @_id'},{

              listUsers:{
                      method:'GET',
                      url:BASE_URL + "users/json",
                      isArray:true
              },
              update:{
                      method:"PUT",
                      url: BASE_URL + "users/json/:id",
                      params:{id:'@id'}
              },
              login:{
                      method:"POST",
                      url: BASE_URL + "users/json/login"
              },
              signup:{
                      method:"POST",
                      url: BASE_URL + "users/json/signup"
              },
              changePassword:{
                      method:"PUT",
                      url: BASE_URL + "users/json/
                         change_password/:id",
                      params:{id:'@id'}

              }
```

```
        });
}]);angular.module('snaApp2').factory('Session',['$cookies','
   $auth',function($cookies,$auth) {

        return {
                setCookieData: function(id,role,name,email) {

            var sessionInfo = {
                id : id,
                name : name,
                email : email,
                role : role,
            };

                        $cookies.putObject("sessionInfo",
                           sessionInfo);

                },
                getCookieData: function() {
            return $cookies.getObject("sessionInfo");

                },
                clearCookieData: function() {

                        $cookies.remove("sessionInfo");

                },
                isAuthorized : function (authorizedRoles) {

            obj = $cookies.getObject("sessionInfo");


            if(typeof obj === "undefined"){
                return false;
            }

                        if (!angular.isArray(authorizedRoles)
                           ) {
                                authorizedRoles = [
```

```
                                        authorizedRoles];
                        }

                        return (($auth.isAuthenticated()) &&
                        authorizedRoles.indexOf(obj.role) !==
                            -1);
                }
        }


}]);

describe('sessionService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(sessionService) {

        //expect(sessionService.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').controller('HomeCtrl',['$scope
   ','$timeout', '$location','USER_ROLES', '$auth','Session
   ','Project','StorageProject',function($scope,$timeout,
   $location,USER_ROLES,$auth,Session,Project,StorageProject)
   {


        $scope.projects = Project.query();

        $scope.selectedProject = null;

        $scope.setProject = function () {

                StorageProject.setProjectId($scope.
                   selectedProject.id);
                $scope.$parent.getProject();
```

```
        };

        $scope.isActive = function(){
                return $auth.isAuthenticated();
        };

}]);




describe('HomeCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('HomeCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('QuestioningCtrl',['
   $scope','Project','Tweet','QuestioningService','Session','
   User','StorageProject','FileSaver', 'Blob','spinnerService
   ',function($scope,Project,Tweet,QuestioningService,Session
   ,User,StorageProject,FileSaver, Blob,spinnerService){


    $scope.questionings = [{name:"byCategoryAndUser", descr:
       "Tweets por categoria e pesquisador"},
    {name:"byCategoryAndQttTags", descr: "Tweets por
```

```
   categoria e quantidade de marca  es de pesquisadores
   "},
{name:"byDistinctCategories", descr: "Tweets por
   quantidade de categorias distintas marcadas"}];

var idUser = Session.getCookieData().id;

$scope.state = QuestioningService;

Project.get({id:StorageProject.getProjectId()},function
   sucess(data){
   $scope.state.selectedProject = data;
});

$scope.users = User.listUsers();

$scope.selectedDatasets = [];

$scope.showQuestioning = function(){

    if (typeof $scope.selectedDatasets === "undefined"){
        return false;
    }

    return $scope.selectedDatasets.length > 0 ? true :
       false;
};

$scope.state.itensPerPage = 10;

var paramsDefault = function(){
    return {
        filter : $scope.state.questSelected,
        idDataset: $scope.selectedDatasets,
        skip:($scope.state.currentPage -1)*$scope.state.
           itensPerPage,
        amount:$scope.state.itensPerPage
    }
};
```

```
$scope.configQuestByCategoryAndUser = function(
   _idCategory,_idUser){


    if (typeof _idCategory !== "undefined" && typeof
       _idUser !== "undefined"){
        $scope.state.currentPage = 1;
        var params = paramsDefault();
        params["idCategory"] = _idCategory;
        params["idUser"] = _idUser;

        $scope.state.lastConfig = params;
        $scope.getPage();
    }
};

$scope.configQuestByCategoryAndQttTags = function(
   _idCategory,qtt){

    if (typeof _idCategory !== "undefined" && typeof qtt
       !== "undefined"){
        $scope.state.currentPage = 1;
        var params = paramsDefault();
        params["idCategory"] = _idCategory;
        params["qtt"] = qtt;

        $scope.state.lastConfig = params;
        $scope.getPage();
    }

};

$scope.configQuestByDistinctCategories = function(qtt){

    if (qtt !== "undefined"){
        $scope.state.currentPage = 1;
        var params = paramsDefault();
        params["qtt"] = qtt;
```

```
        $scope.state.lastConfig = params;
        $scope.getPage();
    }

};

// $scope.dateStringToTimeStamp = function(timestamp){
//
//     return Date.parse(timestamp);
// };

$scope.getPage = function(){
    $scope.disablePagination = true;
    spinnerService.show('questSpinner');

    Tweet.listInAnInterval($scope.state.lastConfig,
       function(results){
        $scope.state.tweets = results.data;


        for (var i = 0; i < $scope.state.tweets.length; i
           ++) {
            $scope.state.tweets[i].selTags = [];

            if($scope.state.tweets[i].tags.length > 0){
                $scope.state.tweets[i].selTags = [];

                for(var j=0; j < $scope.state.tweets[i].
                   tags.length; j++){

                    $scope.state.tweets[i].selTags.push(
                       $scope.state.tweets[i].tags[j].
                       category);

                }


            }
```

```
            }

            $scope.state.count=results.count;
            $scope.state.skipped = ($scope.state.currentPage
               -1)*$scope.state.itensPerPage;
            $scope.state.showTweets = true;
            $scope.disablePagination = false;
            spinnerService.hide('questSpinner');

        },function error(err){
            console.log(err);
            $scope.disablePagination = false;
            spinnerService.hide('questSpinner');
        });
    };

    $scope.changePage = function(){

        $scope.state.lastConfig["skip"] = ($scope.state.
           currentPage -1)*$scope.state.itensPerPage;

        $scope.getPage();


    };

    $scope.changeAmount = function(_qtt){
        console.log(_qtt);
        $scope.state.currentPage = 1;
        $scope.state.lastConfig["skip"] = 0;
        $scope.state.lastConfig["amount"] = _qtt;
        $scope.getPage();
    };

    $scope.toCSV = function(){

        Tweet.exportToCSV($scope.state.lastConfig,function(
           results){
            var blob = new Blob([results.values], {type: "
```

```
            application/csv;charset=utf-8"});
        FileSaver.saveAs(blob, 'export.csv');
    });

};

$scope.onDrop = function(event,ui,selTags,_idTweet){

    var len =  selTags.length;
    var isNew = true;
    var last = selTags[len - 1];

    for (var i = 0; i < len - 1; i++) {
        if(selTags[i].id === last.id){
            selTags.splice(len - 1,1);
            isNew = false;
            break;
        }
    }

    if(isNew){

        Tweet.addTag({idTweet:_idTweet, idCategory:last.
           id},{});

    }

};

$scope.removeCat = function (selTags, idCat, _idTweet){
    var len =  selTags.length;
    for (var i = 0; i < len; i++) {
        if(selTags[i].id === idCat){
            selTags.splice(i,1);
            break;
        }
    }

    Tweet.removeTag({idTweet:_idTweet, idCategory: idCat
```

```
            },{});
    }

}]);

describe('QuestioningCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('QuestioningCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').factory('QuestioningService',['
   $rootScope',function($rootScope) {

        var service = {


        };

        function saveState() {
                sessionStorage.QuestioningService = angular.
                   toJson(service);
                console.log("Salvando" );
        }
        function restoreState() {
                service = angular.fromJson(sessionStorage.
                   QuestioningService);
                console.log("Recuperado");
```

```
        }
        $rootScope.$on("savestate", saveState);

        if (sessionStorage.QuestioningService){
                restoreState();
        }

        return service;
}]);describe('questioningService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(questioningService) {

        //expect(questioningService.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').controller('AdminViewUsersCtrl
   ',['$scope','Admin', function($scope, Admin){


        $scope.users = Admin.listUsers();

        $scope.roles = ["ADMINISTRATOR", "MODERATOR", "
           RESEARCHER"];

        $scope.enable = function(idUser, _enabled){
                Admin.enableUser({id:idUser, enabled:_enabled
                   });
        };

        $scope.setRole = function(idUser, _role){
                Admin.setRole({id:idUser, role:_role});
        };

}]);describe('AdminViewUsersCtrl', function() {
```

```
        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('AdminViewUsersCtrl', {$scope: scope
         });
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('adminService', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(adminService) {

        //expect(adminService.doSomething()).toEqual('
           something');

  }));

});angular.module('snaApp2').factory('Admin',['$resource','
   BASE_URL', function($resource,BASE_URL) {

        return $resource(BASE_URL+ 'users/json/:id',{id:'@id
           '},{


                listUsers:{
                        method:'GET',
                        url:BASE_URL + "users/json",
                        isArray:true
                },
```

```
                enableUser:{
                        method:'POST',
                        url:BASE_URL + 'users/json/enable/:id
                           ',
                        params:{enabled:'@enabled'}
                },
                setRole:{
                        method:'POST',
                        url:BASE_URL + 'users/json/set_role/:
                           id',
                        params:{role:'@role'}
                }

        });

}]);angular.module('snaApp2').controller('DatasetUpdateCtrl
   ',['$scope','$timeout','$routeParams','$location','Dataset
   ',function($scope, $timeout,$routeParams, $location,
   Dataset){


        $scope.dataset = Dataset.get({idProject: $routeParams
           .idProject, id:$routeParams.idDataset});
        $scope.idProject = $routeParams.idProject;
        $scope.idDataset = $routeParams.idDataset;
        $scope.error = false;
        $scope.msg = "";

        $scope.update = function(){

                $scope.dataset.$update({idProject:$scope.
                   idProject},function sucess(){
                        $location.path('/project/'+
                           $routeParams.idProject+"/datasets
                           ");
                }, function error(res){
                        $timeout(function(){
                                $scope.error = true;
                                $scope.msg = res.data.error;
```

```
                    },0);
                });

        };

}]);describe('DatasetUpdateCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('DatasetUpdateCtrl', {$scope: scope
         });
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('DatasetViewCtrl', ['
   $scope','$routeParams', '$location','Dataset','Tweet', '
   Category','modalService', function($scope,$routeParams,
   $location,Dataset,Tweet,Category,modalService){

        $scope.dataset = Dataset.get({idProject: $routeParams
           .idProject, id:$routeParams.idDataset});

        $scope.idProject = $routeParams.idProject;

        $scope.delete = function () {

                var modalOptions = {
                        closeButtonText: 'Cancelar',
                        actionButtonText: 'Excluir Dataset',
                        headerText: 'Excluir ' + $scope.
```

```
                        dataset.name + '?',
                    bodyText: 'Tem certeza que deseja
                        excluir este dataset?'
                };

                modalService.showModal({}, modalOptions).then
                    (function (result) {

                        Dataset.delete({idProject:
                            $routeParams.idProject, id:
                            $routeParams.idDataset},function()
                            {
                                $location.path("/project/"+
                                    $scope.idProject);
                        });

                });
        };

}]);describe('DatasetViewCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('DatasetViewCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('DatasetListCtrl',['
    $scope','StorageProject','Project','Dataset','modalService
    ','$route',function($scope,StorageProject,Project,Dataset,
```

```
   modalService,$route){


    $scope.projectId = StorageProject.getProjectId();

    Project.get({id:$scope.projectId},function sucess(data){
        $scope.datasets = data.datasets;
    });

    $scope.delete = function (dataset) {

        var modalOptions = {
            closeButtonText: 'Cancelar',
            actionButtonText: 'Excluir Dataset',
            headerText: 'Excluir ' + dataset.name + '?',
            bodyText: 'Tem certeza que deseja excluir este
               dataset?'
        };

        modalService.showModal({}, modalOptions).then(
           function (result) {

            Dataset.delete({idProject: $scope.projectId, id:
               dataset.id},function(){
              $route.reload();
            });

        });
    };
}]);describe('DatasetListCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;

    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('DatasetListCtrl', {$scope: scope});
    }));
```

```
        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});angular.module('snaApp2').controller('DatasetAddCtrl',['
   $scope','$timeout','$location','$routeParams','Dataset',
   function($scope,$timeout, $location, $routeParams, Dataset
   ){

        $scope.dataset = new Dataset();
        $scope.idProject = $routeParams.idProject;
        $scope.error = false;
        $scope.msg = "";

        $scope.add = function(){



                $scope.dataset.$saveDataset({idProject:
                   $routeParams.idProject},function sucess(){
                        $location.path("/project/"+
                           $routeParams.idProject+"/datasets
                           ");
                }, function error(res){
                        $timeout(function(){
                                $scope.error = true;
                                $scope.msg = res.data.error;
                        },0);
                });
        };

}]);describe('DatasetAddCtrl', function() {

        beforeEach(module('snaApp2'));

        var scope,ctrl;
```

```
    beforeEach(inject(function($rootScope, $controller) {
      scope = $rootScope.$new();
      ctrl = $controller('DatasetAddCtrl', {$scope: scope});
    }));

        it('should ...', inject(function() {

                expect(1).toEqual(1);

        }));

});describe('dataset', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(dataset) {

        //expect(dataset.doSomething()).toEqual('something');

  }));

});angular.module('snaApp2').factory('Dataset', ['$resource
   ','BASE_URL', function($resource,BASE_URL){
        return $resource(BASE_URL + 'datasets/json/:idProject
           /:id',{idProject:'@idProject', id:'@id'},{

                update:{
                        method:"PUT",
                        url:BASE_URL + "datasets/json/:
                           idProject",
                        params:{idProject:'@_idProject'}
                },
                saveDataset:{
                        method:"POST",
                        url: BASE_URL + "datasets/json/:
                           idProject",
                        params:{idProject:'@_idProject'}
                },
```

```
                filterByProject:{
                        method:"GET",
                        url: BASE_URL + "datasets/json/
                           filter_by_project/:idProject",
                        params:{idProject:'@_idProject'},
                        isArray:true
                }


        });

}]);describe('modal', function() {

  beforeEach(module('snaApp2'));

  it('should ...', inject(function(modal) {

        //expect(modal.doSomething()).toEqual('something');

  }));

});//http://weblogs.asp.net/dwahlin/building-an-angularjs-
   modal-service

angular.module('snaApp2').service('modalService', ['$uibModal
   ',
    function ($uibModal) {

        var modalDefaults = {
            backdrop: true,
            keyboard: true,
            modalFade: true,
            templateUrl: '/projetos/sonda/client/app/shared/
               modal-service/modal.html'
        };

        var modalOptions = {
            closeButtonText: 'Close',
            actionButtonText: 'OK',
```

```
    headerText: 'Proceed?',
    bodyText: 'Perform this action?'
};

this.showModal = function (customModalDefaults,
   customModalOptions) {
    if (!customModalDefaults) {
        customModalDefaults = {};}
    customModalDefaults.backdrop = 'static';
    return this.show(customModalDefaults,
       customModalOptions);
};

this.show = function (customModalDefaults,
   customModalOptions) {
    //Create temp objects to work with since we're in
        a singleton service
    var tempModalDefaults = {};
    var tempModalOptions = {};

    //Map angular-ui modal custom defaults to modal
       defaults defined in service
    angular.extend(tempModalDefaults, modalDefaults,
       customModalDefaults);

    //Map modal.html $scope custom properties to
       defaults defined in service
    angular.extend(tempModalOptions, modalOptions,
       customModalOptions);

    if (!tempModalDefaults.controller) {
        tempModalDefaults.controller = function (
           $scope, $uibModalInstance) {
            $scope.modalOptions = tempModalOptions;
            $scope.modalOptions.ok = function (result
               ) {
                $uibModalInstance.close(result);
            };
            $scope.modalOptions.close = function (
```

```
                    result) {
                      $uibModalInstance.dismiss('cancel');
                  };
              };
          }

          return $uibModal.open(tempModalDefaults).result;
      };

  }]);angular.module('snaApp2').directive('fileReader',
     function() {
        return {
          scope: {
            fileReader:"="
          },
          link: function(scope, element) {
            $(element).on('change', function(changeEvent) {
              var files = changeEvent.target.files;
              if (files.length) {
                var r = new FileReader();
                r.onload = function(e) {
                    var contents = e.target.result;
                    scope.$apply(function () {
                      scope.fileReader = contents;
                    });
                };

                r.readAsText(files[0]);
              }
            });
          }
        };
      });



describe('fileReader', function() {

  beforeEach(module('snaApp2'));
```

```
  var scope,compile;

  beforeEach(inject(function($rootScope,$compile) {
    scope = $rootScope.$new();
    compile = $compile;
  }));

  it('should ...', function() {

    /*
    To test your directive, you need to create some html that
        would use your directive,
    send that through compile() then compare the results.

    var element = compile('<div mydirective name="name">hi</
       div>')(scope);
    expect(element.text()).toBe('hello, world');
    */

  });
});angular.module('snaApp2').directive('compareTo', function
   () {
        return {
                require: "ngModel",
                scope: {
                        otherModelValue: "=compareTo"
                },
                link: function(scope, element, attributes,
                   ngModel) {

                        ngModel.$validators.compareTo =
                           function(modelValue) {
                                return modelValue == scope.
                                   otherModelValue;
                        };

                        scope.$watch("otherModelValue",
                           function() {
```

```
                                ngModel.$validate();
                        });
                }
        };
});


describe('compareTo', function() {

  beforeEach(module('snaApp2'));

  var scope,compile;

  beforeEach(inject(function($rootScope,$compile) {
    scope = $rootScope.$new();
    compile = $compile;
  }));

  it('should ...', function() {

    /*
    To test your directive, you need to create some html that
        would use your directive,
    send that through compile() then compare the results.

    var element = compile('<div mydirective name="name">hi</
       div>')(scope);
    expect(element.text()).toBe('hello, world');
    */

  });
});/*jslint node: true */
'use strict';

var pkg = require('./package.json');

//Using exclusion patterns slows down Grunt significantly
//instead of creating a set of patterns like '**/*.js' and
   '!**/node_modules/**'
```

```
//this method is used to create a set of inclusive patterns
   for all subdirectories
//skipping node_modules, bower_components, dist, and any .
   dirs
//This enables users to create any directory structure they
   desire.
var createFolderGlobs = function(fileTypePatterns) {
  fileTypePatterns = Array.isArray(fileTypePatterns) ?
     fileTypePatterns : [fileTypePatterns];
  var ignore = ['node_modules','bower_components','dist','
     temp'];
  var fs = require('fs');
  return fs.readdirSync(process.cwd())
          .map(function(file){
            if (ignore.indexOf(file) !== -1 ||
                file.indexOf('.') === 0 ||
                !fs.lstatSync(file).isDirectory()) {
              return null;
            } else {
              return fileTypePatterns.map(function(pattern) {
                return file + '/**/' + pattern;
              });
            }
          })
          .filter(function(patterns){
            return patterns;
          })
          .concat(fileTypePatterns);
};

module.exports = function (grunt) {

  // load all grunt tasks
  require('load-grunt-tasks')(grunt);

  grunt.loadNpmTasks('grunt-wiredep');
  grunt.loadNpmTasks('grunt-contrib-watch');

  // Project configuration.
```

```
grunt.initConfig({
  connect: {
    main: {
      options: {
        port: 9001
      }
    }
  },
  watch: {
    main: {
      options: {
          livereload: true,
          livereloadOnError: false,
          spawn: false
      },
      files: [createFolderGlobs(['*.js','*.less','*.html'])
         ,'!_SpecRunner.html','!.grunt'],
      tasks: [] //all the tasks are run dynamically during
         the watch event handler
    }
  },
  jshint: {
    main: {
      options: {
          jshintrc: '.jshintrc'
      },
      src: createFolderGlobs('*.js')
    }
  },
  clean: {
    before:{
      src:['dist','temp']
    },
    after: {
      src:['temp']
    }
  },
  less: {
    production: {
```

```
      options: {
      },
      files: {
        'temp/app.css': 'app.less'
      }
    }
  },
  ngtemplates: {
    main: {
      options: {
          module: pkg.name,
          htmlmin:'<%= htmlmin.main.options %>'
      },
      src: [createFolderGlobs('*.html'),'!index.html','!
         _SpecRunner.html'],
      dest: 'temp/templates.js'
    }
  },
  copy: {
    main: {
      files: [
        {src: ['img/**'], dest: 'dist/'},
        {src: ['bower_components/font-awesome/fonts/**'],
           dest: 'dist/',filter:'isFile',expand:true},
        {src: ['bower_components/bootstrap/fonts/**'], dest
           : 'dist/',filter:'isFile',expand:true}
        //{src: ['bower_components/angular-ui-utils/ui-
           utils-ieshiv.min.js'], dest: 'dist/'},
        //{src: ['bower_components/select2/*.png','
           bower_components/select2/*.gif'], dest:'dist/css
           /',flatten:true,expand:true},
        //{src: ['bower_components/angular-mocks/angular-
           mocks.js'], dest: 'dist/'}
      ]
    }
  },
  dom_munger:{
    read: {
      options: {
```

```
        read:[
          {selector:'script[data-concat!="false"]',
             attribute:'src',writeto:'appjs'},
          {selector:'link[rel="stylesheet"][data-concat!="
             false"]',attribute:'href',writeto:'appcss'}
        ]
      },
      src: 'index.html'
    },
    update: {
      options: {
        remove: ['script[data-remove!="false"]','link[data-
           remove!="false"]'],
        append: [
          {selector:'body',html:'<script src="app.full.min.
             js"></script>'},
          {selector:'head',html:'<link rel="stylesheet"
             href="app.full.min.css">'}
        ]
      },
      src:'index.html',
      dest: 'dist/index.html'
    }
  },
  cssmin: {
    main: {
      src:['temp/app.css','<%= dom_munger.data.appcss %>'],
      dest:'dist/app.full.min.css'
    }
  },
  concat: {
    main: {
      src: ['<%= dom_munger.data.appjs %>','<%= ngtemplates
         .main.dest %>'],
      dest: 'temp/app.full.js'
    }
  },
  ngAnnotate: {
    main: {
```

```
      src:'temp/app.full.js',
      dest: 'temp/app.full.js'
    }
  },
  uglify: {
    main: {
      src: 'temp/app.full.js',
      dest:'dist/app.full.min.js'
    }
  },
  htmlmin: {
    main: {
      options: {
        collapseBooleanAttributes: true,
        collapseWhitespace: true,
        removeAttributeQuotes: true,
        removeComments: true,
        removeEmptyAttributes: true,
        removeScriptTypeAttributes: true,
        removeStyleLinkTypeAttributes: true
      },
      files: {
        'dist/index.html': 'dist/index.html'
      }
    }
  },
  wiredep: {
        task: {
          src: ['index.html']
        }
      },
  //Imagemin has issues on Windows.
  //To enable imagemin:
  // - "npm install grunt-contrib-imagemin"
  // - Comment in this section
  // - Add the "imagemin" task after the "htmlmin" task in
     the build task alias
  // imagemin: {
  //   main:{
```

```
  //     files: [{
  //       expand: true, cwd:'dist/',
  //       src:['**/{*.png,*.jpg}'],
  //       dest: 'dist/'
  //     }]
  //   }
  // },
  karma: {
    options: {
      frameworks: ['jasmine'],
      files: [  //this files data is also updated in the
         watch handler, if updated change there too
        '<%= dom_munger.data.appjs %>',
        'bower_components/angular-mocks/angular-mocks.js',
        createFolderGlobs('*-spec.js')
      ],
      logLevel:'ERROR',
      reporters:['mocha'],
      autoWatch: false, //watching is handled by grunt-
         contrib-watch
      singleRun: true
    },
    all_tests: {
      browsers: ['PhantomJS','Chrome','Firefox']
    },
    during_watch: {
      browsers: ['PhantomJS']
    },
  }
});

grunt.registerTask('build',['jshint','clean:before','less
   ','dom_munger','ngtemplates','cssmin','concat','
   ngAnnotate','uglify','copy','htmlmin','clean:after']);
grunt.registerTask('serve', ['dom_munger:read','jshint','
   connect', 'watch']);
grunt.registerTask('test',['dom_munger:read','karma:
   all_tests']);
grunt.registerTask('default', ['wiredep']);
```

```
grunt.event.on('watch', function(action, filepath) {
  //https://github.com/gruntjs/grunt-contrib-watch/issues
     /156

  var tasksToRun = [];

  if (filepath.lastIndexOf('.js') !== -1 && filepath.
     lastIndexOf('.js') === filepath.length - 3) {

    //lint the changed js file
    grunt.config('jshint.main.src', filepath);
    tasksToRun.push('jshint');

    //find the appropriate unit test for the changed file
    var spec = filepath;
    if (filepath.lastIndexOf('-spec.js') === -1 || filepath
       .lastIndexOf('-spec.js') !== filepath.length - 8) {
      spec = filepath.substring(0,filepath.length - 3) + '-
         spec.js';
    }

    //if the spec exists then lets run it
    if (grunt.file.exists(spec)) {
      var files = [].concat(grunt.config('dom_munger.data.
         appjs'));
      files.push('bower_components/angular-mocks/angular-
         mocks.js');
      files.push(spec);
      grunt.config('karma.options.files', files);
      tasksToRun.push('karma:during_watch');
    }
  }

  //if index.html changed, we need to reread the <script>
     tags so our next run of karma
  //will have the correct environment
  if (filepath === 'index.html') {
    tasksToRun.push('dom_munger:read');
```

```
    }

    grunt.config('watch.main.tasks',tasksToRun);

  });
};
//Experimental

/*jslint node: true */
var gulp = require('gulp');
var concat = require('gulp-concat');
var uglify = require('gulp-uglify');
var imagemin = require('gulp-imagemin');
var less = require('gulp-less');
var gCheerio = require('gulp-cheerio');
var ngHtml2js = require("gulp-ng-html2js");
var ngmin = require('gulp-ngmin');
var htmlmin = require('gulp-htmlmin');
var cssmin = require('gulp-cssmin');
var packagejson = require('./package.json');
var streamqueue = require('streamqueue');
var rimraf = require('rimraf');
var rename = require('gulp-rename');
var jshint = require('gulp-jshint');
var jasmine = require('gulp-jasmine');
var stylish = require('jshint-stylish');
var domSrc = require('gulp-dom-src');

var htmlminOptions = {
    collapseBooleanAttributes: true,
    collapseWhitespace: true,
    removeAttributeQuotes: true,
    removeComments: true,
    removeEmptyAttributes: true,
    // removeRedundantAttributes: true,
    removeScriptTypeAttributes: true,
    removeStyleLinkTypeAttributes: true
};
```

```
gulp.task('clean', function() {
    rimraf.sync('dist');
});

gulp.task('css', ['clean'], function() {
    return gulp.src('app.less')
        .pipe(less())
        .pipe(cssmin({keepSpecialComments: 0}))
        .pipe(rename('app.full.min.css'))
        .pipe(gulp.dest('dist/'));
});

gulp.task('js', ['clean'], function() {

    var templateStream = gulp.src(['!node_modules/**','!index
       .html','!_SpecRunner.html','!.grunt/**','!dist/**','!
       bower_components/**','**/*.html'])
        .pipe(htmlmin(htmlminOptions))
        .pipe(ngHtml2js({
            moduleName: packagejson.name
        }));

    var jsStream = domSrc({file:'index.html',selector:'script
       [data-build!="exclude"]',attribute:'src'});

    var combined = streamqueue({ objectMode: true });

    combined.queue(jsStream);
    combined.queue(templateStream);

    return combined.done()
        .pipe(concat('app.full.min.js'))
        .pipe(ngmin())
        .pipe(uglify())
        .pipe(gulp.dest('dist/'));


    /*
        Should be able to add to an existing stream easier,
```

```
            like:
        gulp.src([... partials html ...])
          .pipe(htmlmin())
          .pipe(ngHtml2js())
          .pipe(domSrc(... js from script tags ...))  <-- add
              new files to existing stream
          .pipe(concat())
          .pipe(ngmin())
          .pipe(uglify())
          .pipe(gulp.dest());

        https://github.com/wearefractal/vinyl-fs/issues/9
    */
});

gulp.task('indexHtml', ['clean'], function() {
    return gulp.src('index.html')
        .pipe(gCheerio(function ($) {
            $('script[data-remove!="exclude"]').remove();
            $('link').remove();
            $('body').append('<script src="app.full.min.js
               "></script>');
            $('head').append('<link rel="stylesheet" href="
               app.full.min.css">');
        }))
        .pipe(htmlmin(htmlminOptions))
        .pipe(gulp.dest('dist/'));
});

gulp.task('images', ['clean'], function(){
    return gulp.src('img/**')
        .pipe(imagemin())
        .pipe(gulp.dest('dist/'));
});

gulp.task('fonts', ['clean'], function(){
    return gulp.src('bower_components/font-awesome/fonts/**')
        .pipe(gulp.dest('dist/bower_components/font-awesome/
           fonts/'));
```

```
});

gulp.task('jshint', function(){
    gulp.src(['!node_modules/**','!.grunt/**','!dist/**','!
       bower_components/**','**/*.js'])
        .pipe(jshint())
        .pipe(jshint.reporter(stylish));
});

gulp.task('build', ['clean', 'css', 'js', 'indexHtml', '
   images', 'fonts']);

/*

-specifying clean dependency on each task is ugly
https://github.com/robrich/orchestrator/issues/26

-gulp-jasmine needs a phantomjs option
https://github.com/sindresorhus/gulp-jasmine/issues/2

*/

/*
    "gulp-dom-src": "~0.1.0",
    "gulp-concat": "~2.1.7",
    "gulp-uglify": "~0.2.1",
    "gulp-cssmin": "~0.1.3",
    "gulp-imagemin": "~0.1.5",
    "gulp-less": "~1.2.2",
    "gulp-cheerio": "~0.2.0",
    "gulp-rename": "~1.2.0",
    "gulp-ng-html2js": "~0.1.6",
    "gulp-ngmin": "~0.1.2",
    "gulp-htmlmin": "~0.1.2",
    "gulp-jshint": "~1.5.0",
    "gulp-jasmine": "~0.2.0",
    "jshint-stylish": "~0.1.5",
    "rimraf": "~2.2.6",
    "streamqueue": "0.0.5",
```

```
    "gulp": "~3.5.5"
*/

<div class="form-group">
<label for="name">Nome</label>
<input type="text" class="form-control" id="name" ng-model="
   category.name" required>
</div>

<div class="form-group">
    <label for="type">Tipo</label><br>
    <select required ng-model="category.type" ng-options="key
       as value for (key, value) in categoriesTypes" id="
       type" >
        <option></option>
    </select>
</div>

<div class="form-group" ng-if="category.type ==’
   SPACES_OF_POSSIBILITY’">
    <label for="keywords">Palavras-chave (insira separadas
       por v rgula)</label><br>
    <textarea name="keywords" id="keywords" cols="80" ng-
       model="category.keywords" rows="10" required ></
       textarea>
</div>

<div ng-if="category.type ==’RELEVANT_PROCESS’">
<div class="form-group">
<label for="description">Descri  o</label><br>
<textarea name="description" id="description" cols="80" ng-
   model="category.description" rows="10" required ></
   textarea>
</div>

<label for="color">Cor</label><br>
<input type="color" value="{{category.color}}" id="color"
   name="color" ng-model="category.color" required>

</div>
```

```
<br><br>
<button type="submit" class="btn btn-default">Enviar</button>

<h1 class="page-header">
    Categoria <small>edi   o</small>
</h1>


<div  ng-controller="CategoryUpdateCtrl">


        <div ng-if="error" class="alert alert-danger">
          {{msg}}
        </div>

        <form ng-submit="update()" role="form">

                <div ng-include src="'app/components/category
                   /templates/form.html'"></div>

        </form>



</div>



<h1 class="page-header">
        Categoria <small>detalhes</small>
</h1>


<div ng-controller="CategoryViewCtrl">

        <p><strong>Nome: </strong>{{category.name}}</p>
        <p><strong>Tipo: </strong>{{categoriesTypes[category.
           type]}}</p>
```

```
        <p ng-if="category.type == 'RELEVANT_PROCESS'"><
           strong>Cor: </strong><input type="color" value="{{
           category.color}}" placeholder=""></p>
        <p ng-if="category.type == 'SPACES_OF_POSSIBILITY'"><
           strong>Palavras-chave:</strong>
                <br><span ng-repeat="kw in category.keywords"
                   >{{kw}}{{$last ? '' : ', '}} </span></p>

        <p ng-if="category.type == 'RELEVANT_PROCESS'"><
           strong>Descri  o:</strong> {{category.
           description}}</p>

        <a href="#/project/{{idProject}}/category/{{category.
           id}}/update"><button type="button" class="btn btn-
           default">Editar</button></a>
        <button type="button" class="btn btn-default" ng-
           click="delete()" ng-show="isAuthorized([userRoles.
           admin,userRoles.moderator])">Excluir</button>

</div>




<h1 class="page-header">
    Categorias
</h1>

<div  ng-controller="CategoryListCtrl">

        <br>

        <h4>Espa os de possibilidade</h4><br>

        <table class="table">
                <tbody>
                <tr ng-repeat="category in categories |
                   filter : {type: 'SPACES_OF_POSSIBILITY'}"
```

```
                >
                    <td><a href="#/project/{{projectId}}/
                        category/{{category.id}}">{{
                        category.name}}</a></td>
                    <td><button type="button" class="btn␣
                        btn-danger" ng-show="isAuthorized
                        ([userRoles.admin,userRoles.
                        moderator])" ng-click="delete(
                        category)">Excluir</button></td>
            </tr>

            </tbody>
        </table>

        <h4>Fatores e circunst ncias</h4><br>

        <table class="table">
            <tbody>
            <tr ng-repeat="category␣in␣categories␣|␣
                filter␣:␣{type:␣'RELEVANT_PROCESS'␣}">
                    <td><a href="#/project/{{projectId}}/
                        category/{{category.id}}">{{
                        category.name}}</a></td>
                    <td><button type="button" class="btn␣
                        btn-danger" ng-show="isAuthorized
                        ([userRoles.admin,userRoles.
                        moderator])" ng-click="delete(
                        category)">Excluir</button></td>
            </tr>

            </tbody>
        </table>

        <a href="#/project/{{projectId}}/category/add"><
            button type="button" class="btn␣btn-default">
            Adicionar categoria</button></a>

</div>
```

```
<h1 class="page-header">
        Categoria <small>cadastro</small>
</h1>


<div ng-controller="CategoryAddCtrl">

        <div ng-if="error" class="alert␣alert-danger">
          {{msg}}
        </div>


        <form ng-submit="add()" role="form">

                <div ng-include src="'app/components/category
                   /templates/form.html'"></div>

        </form>



</div>



<div class="form-group">
    <label for="name">Nome</label>
    <input type="text" class="form-control" id="name" ng-
       model="matrix.name" required>
</div>


<div class="form-group">

    <label>Datasets</label>

    <div ng-repeat="dataset␣in␣project.datasets">
```

```
        <label >
            <input type="checkbox" checklist-model="matrix.
               datasetsIds" checklist-value="dataset.id"> {{
               dataset.name}}
        </label>
    </div>


</div>

<div class="form-group">

    <label>Linhas</label>

    <div ng-repeat="category␣in␣project.categories␣|␣filter␣:
       ␣{type:'RELEVANT_PROCESS'}">
        <label>
            <input type="checkbox"  checklist-model="matrix.
               rowsIds" checklist-value="category.id"> {{
               category.name}}
        </label> <br>
    </div>

</div>

<div class="form-group">

    <label>Colunas</label>

    <div ng-repeat="category␣in␣project.categories␣|␣filter␣:
       ␣{type:'RELEVANT_PROCESS'}">
    <label >
        <input type="checkbox"  checklist-model="matrix.
           columnsIds" checklist-value="category.id"> {{
           category.name}}
    </label> <br>
    </div>

</div>
```

```
<div class="form-group">

    <label for="relation">Rela  o</label>
    <select class="form-control" id="relation" ng-model="
       matrix.relation">
        <option value="and">AND</option>
        <option value="or">OR</option>
        <option value="not">NOT</option>
    </select>

</div>

<button type="submit" class="btn btn-default">Submit</button>
<h1 class="page-header">
    Matriz <small>edi  o</small>
</h1>

<div class="col-md-12" ng-controller="MatrixUpdateCtrl" class
   ="col-md-6">

    <div ng-if="error" class="alert alert-danger">
        {{msg}}
    </div>

    <form ng-submit="update()" role="form">

        <div ng-include src="'app/components/matrix/templates
           /form.html'"></div>

    </form>

</div>
<h1 class="page-header">
    Matriz <small>detalhes</small>
</h1>

<div class="col-md-12" ng-controller="MatrixViewCtrl">
```

```
<p><strong>Nome: </strong>{{matrix.name}}</p>
<p><strong>Rela  o: </strong>{{matrix.relation}}</p>
<p><strong>Datasets: </strong><span ng-repeat="dataset␣in
   ␣matrix.datasetsIds">{{retrieveNames("datasets",
   dataset)}}{{$last ? '' : ', '}} </span></p>
<p><strong>Linhas: </strong><span ng-repeat="row␣in␣
   matrix.rowsIds">{{retrieveNames("categories",row)}}{{
   $last ? '' : ', '}} </span></p>
<p><strong>Colunas: </strong><span ng-repeat="col␣in␣
   matrix.columnsIds">{{retrieveNames("categories",col)
   }}{{$last ? '' : ', '}} </span></p>


<table class="table␣table-bordered">
    <thead>
    <tr>
        <th ></th>
        <th ng-repeat="col␣in␣matrix.columnsIds" style="
           text-align:center">{{retrieveNames("categories
           ",col)}}</th>
    </tr>
    </thead>
    <tbody>
        <tr ng-repeat="row␣in␣matrix.rowsIds" ng-init="
           rowIndex␣=␣$index">
            <td align="center"><strong>{{retrieveNames("
               categories",row)}}</strong></td>
            <td align="center" ng-repeat="cel␣in␣
               retrieveRows(rowIndex)" ng-init="colIndex␣
               =␣$index"><a href="" ng-click="
               configRequest(rowIndex,colIndex)">{{cel.
               symbol}}</a></td>
        </tr>
    </tbody>
</table>


<a href="#/project/{{idProject}}/matrix/{{id}}/update"><
   button class="btn␣btn-default">Editar matriz</button><
```

```
    /a>
<button type="button" class="btn␣btn-default" ng-click="
    delete()" ng-show="isAuthorized([userRoles.admin,
    userRoles.moderator])">Excluir</button>

<div class="container-fluid" ng-show="showTweets">
     <div class="row">

         <div class="col-sm-12">
             <div class="container">
                 <div class="col-md-2"> <h3>Resultados</h3
                    ><spinner name="matrixSpinner"><i
                    class="fa␣fa-spinner␣fa-spin␣fa-2x␣fa-
                    fw"></i></spinner></div>
             </div>
             <div class="col-sm-8">

                 <div class="col-sm-4">
                     <p>Itens por p gina</p>
                 </div>
                 <div class="col-sm-3">

                     <select class="form-control␣input-sm"
                         ng-model="itensPerPage" ng-click=
                        "changeAmount(itensPerPage)" ng-
                        options="qtt␣for␣qtt␣in␣
                        [10,20,30,50,100]" class="ng-
                        pristine␣ng-untouched␣ng-valid␣ng-
                        not-empty">
                          <option value="" ng-if="false"></
                             option>
                     </select>

                 </div>
             </div>

             <div class="col-sm-12">

                 <table class="table␣table-hover">
```

```
<thead>
<tr>
    <th>#</th>
    <th>Texto</th>
    <th>Autor</th>
    <th>Publicado em</th>
    <th></th>
</tr>
</thead>
<tbody>
<tr ng-repeat="tweet in tweets">
    <td>{{skipped + $index + 1}}</td>
    <td><a href="#/project/{{
       selectedProject.id}}/dataset
       /{{tweet.idDataset}}/tweet/{{
       tweet.id}}">{{tweet.text}}</a>
       <div class="thumbnail" data-
       drop="true" ng-model="tweet.
       selTags" data-jqyoui-options="
       " jqyoui-droppable="{multiple:
       true, onDrop:'onDrop(tweet.
       selTags, tweet.id)'}">

        <span ng-repeat="item in
           tweet.selTags track by
           $index" class="ng-scope"><
           span class="label label-
           default" ng-show="item.
           name" data-jqyoui-options=
           "{revert: 'invalid', 
           helper:'clone'}" ng-style=
           "{'background-color':'{{
           item.color}}'}" ng-model="
           tweet.selTags" jqyoui-
           draggable="{index: 0,
           animate:true}">{{item.name
           }}</span> </span></div></
           td>
    <td> {{tweet.fromUser}}</td>
```

```
                    <td>{{tweet.unixTimestamp * 1000
                       | date:"dd/MM/yyyy␣HH:mm:ss"}}
                       </td>
                    <td>

                        <a  href="#/project/{{
                           selectedProject.id}}/
                           dataset/{{tweet.idDataset
                           }}/tweet/{{tweet.id}}/view
                           -conversation"><i class="
                           fa␣fa-comments-o" aria-
                           hidden="true"></i> Ver
                           conversa  o</a></li>



                    </td>
                </tr>

                </tbody>
            </table>

        </div>

        <div class="col-sm-12">

            <uib-pagination ng-disabled="
               disablePagination" boundary-links="
               true" total-items="count" ng-click="
               changePage()" ng-model="currentPage"
               class="pagination-sm" previous-text="&
               lsaquo;" next-text="&rsaquo;" first-
               text="&laquo;" last-text="&raquo;" max
               -size="10" items-per-page="
               itensPerPage"></uib-pagination>
            <p>Total : {{count}}</p>
            <button type="button" class="btn␣btn-
               default" ng-click="toCSV()">Exportar
               para CSV</button>
```

```
                </div>


            </div>
        </div>


    </div>


</div>
<h1 class="page-header">
    Matrizes
</h1>


<div class="col-md-12" ng-controller="MatrixListCtrl">

    <table class="table">
        <thead>
        <tr>
            <th>Nome</th>
            <th ng-show="isAuthorized([userRoles.admin,
               userRoles.moderator])">Op   es</th>
        </tr>
        </thead>
        <tbody>
        <tr ng-repeat="matrix␣in␣matrices">
            <td><a href="#/project/{{projectId}}/matrix/{{
               matrix.id}}">{{matrix.name}}</a></td>
            <td ng-show="isAuthorized([userRoles.admin,
               userRoles.moderator])"><button type="button"
               class="btn␣btn-danger" ng-click="delete(matrix
               )">Excluir</button>
            </td>
        </tr>
```

```
        </tbody>
    </table>

    <a  href="#/project/{{projectId}}/matrix/add"><button
       type="button" class="btn␣btn-default">Cadastrar matriz
       </button></a>


</div>


<h1 class="page-header">
    Matriz <small>cadastro</small>
</h1>


<div ng-controller="MatrixAddCtrl" class="col-md-6">

    <div ng-if="error" class="alert␣alert-danger">
        {{msg}}
    </div>

    <form ng-submit="add()" role="form">

        <div ng-include src="'app/components/matrix/templates
           /form.html'"></div>

    </form>

</div>
<div ng-controller="ImportTweetCtrl" ng-init="tab=1">

    <h1 class="page-header">
        Tweets <small>importa  o</small>
    </h1>

    <div class="container">

        <div ng-if="error" class="alert␣alert-danger">
```

```
    {{msg}}
</div>


<ul class="nav␣nav-tabs">
    <li ng-class="{'active':tab==1}"><a data-toggle="
       tab" href="" ng-click="tab=1">Upload do
       arquivo</a></li>
    <li ng-class="{'active':tab==2,␣'disabled':tab
       ==1}"><a data-toggle="tab" href="" >
       Configura  o</a></li>
</ul>

<div class="tab-content">


    <div ng-show="tab==1" class="tab-pane␣fade␣in␣
       active">
        <br><br>
        <p>Selecione um arquivo no formato CSV</p>
        <input type="file" accept=".csv" file-reader=
           "fileContent" />

        <br><br>

        <button type="submit" class="btn␣btn-primary"
            ng-click="tab=2" ng-disabled="fileContent
           ==undefined">Avan ar</button>
    </div>


    <div ng-show="tab==2"  class="tab-pane␣fade␣in␣
       active">
        <form ng-submit="importTweets()" role="form">

            <br>

          <div class="container">
                  <div class="col-md-2">
```

```
                <p>Separador de campo:</p>
            </div>
            <div class="col-md-1">

                <select class="form-control␣
                   input-sm" ng-model="
                   delimiter" ng-click="
                   getHeader()" ng-options="del
                   ␣for␣del␣in␣delimiters"
                   class="ng-pristine␣ng-
                   untouched␣ng-valid␣ng-not-
                   empty">
                    <option value="" ng-if="
                       false"></option>
                </select>

            </div>
        </div>

    <br>

    <div class="container">
        <div class="col-md-2">
            <p>Delimitador de texto:</p>
        </div>
        <div class="col-md-1">
            <select class="form-control␣input
               -sm" ng-model="enclosure" ng-
               options="encl␣for␣encl␣in␣
               enclosures" class="ng-pristine
               ␣ng-untouched␣ng-valid␣ng-not-
               empty">
                <option value="" ng-if="false
                   "></option>
            </select>

        </div>
    </div>
```

```
<br>

<div ng-show="delimiter!=undefined␣&&␣
   enclosure␣!=␣undefined">

<div class="container">
    <div class="col-md-2">
        <p>Campo id</p>
    </div>
    <div class="col-md-4">
        <select class="form-control␣input
           -sm" ng-model="id" ng-options=
           "attr␣for␣attr␣in␣header"
           class="ng-pristine␣ng-
           untouched␣ng-valid␣ng-not-
           empty">
            <option value="" ng-if="false
               "></option>
        </select>
    </div>
</div>

<br>

<div class="container">
    <div class="col-md-2">
        <p>Campo texto</p>
    </div>
    <div class="col-md-4">
        <select class="form-control␣input
           -sm" ng-model="text" ng-
           options="attr␣for␣attr␣in␣
           header" class="ng-pristine␣ng-
           untouched␣ng-valid␣ng-not-
           empty">
            <option value="" ng-if="false
               "></option>
        </select>
    </div>
```

```
</div>

<br>

<div class="container">
    <div class="col-md-2">
        <p>Campo autor</p>
    </div>
    <div class="col-md-4">
        <select class="form-control␣input
           -sm" ng-model="fromUser" ng-
           options="attr␣for␣attr␣in␣
           header" class="ng-pristine␣ng-
           untouched␣ng-valid␣ng-not-
           empty">
            <option value="" ng-if="false
               "></option>
        </select>
    </div>
</div>

<br>

<div class="container">
    <div class="col-md-2">
        <p>Campo data</p>
    </div>
    <div class="col-md-4">
        <select class="form-control␣input
           -sm" ng-model="createdAt" ng-
           options="attr␣for␣attr␣in␣
           header" class="ng-pristine␣ng-
           untouched␣ng-valid␣ng-not-
           empty">
            <option value="" ng-if="false
               "></option>
        </select>
    </div>
</div>
```

```
                    <br>

                    <div class="container">
                        <div class="col-md-1"><button class="
                            btn␣btn-primary" type="submit"
                            class="btn␣btn-default" ng-disabled
                            ="importEnabled()">Enviar</button>
                            </div>
                        <div class="col-md-1"><spinner name="
                            importSpinner"><i class="fa␣fa-
                            spinner␣fa-spin␣fa-3x␣fa-fw"></i></
                            spinner></div>

                    </div>

                    </div>


                </form>
            </div>

        </div>
    </div>

</div>



<h1 class="page-header">
        Tweet <small>detalhes</small>
</h1>


<div ng-controller="TweetViewCtrl">

        <h4><a href="https://twitter.com/{{tweet.fromUser}}/
            status/{{tweet.idTweet}}" target="_blank"   title=
            "">Ver no Twitter</a></h4>
```

```
        <p><strong>Texto:</strong><br>{{tweet.text}}</p>
        <p><strong>Publicado por:</strong><br>{{tweet.
           fromUser}}</p>
        <p><strong>Data de publica  o:</strong><br>{{tweet.
           unixTimestamp * 1000 | date:"dd/MM/yyyy␣HH:mm:ss"
           }}</p>

        <div ng-repeat="(key,value)␣in␣tweet.otherAttributes"
           >

              <p ng-if="!isNullOrEmpty(value)"><strong>{{
                 key}}:</strong><br>{{value}}</p>

        </div>

        <a href="#/project/{{idProject}}/dataset/{{idDataset
           }}/tweet/{{tweet.id}}/view-conversation"><button
           type="button" class="btn␣btn-default">Ver
           conversa  o</button></a>
        <button ng-click="setVisibility()" type="button"
           class="btn␣btn-default"><span ng-if="!tweet.
           isVisible">Desocultar tweet</span><span ng-if="
           tweet.isVisible">Ocultar tweet</span></button>

        </div>

</div>
```

```
<div ng-controller="TweetListCtrl" >

        <h1 class="page-header">
              Tweets <small>list</small>
        </h1>
```

```
<div class="container-fluid">
        <div class="row">
                <div class="col-sm-2">



                <span ng-repeat='cat in project.
                   categories' class="label␣label-
                   default" style="background-color
                   :{{cat.color}};" title="{{cat.
                   description}}" data-toggle="
                   tooltip" data-placement="bottom"
                   tooltip  ng-show="cat.name" rel="
                   tooltip" data-drag="true" data-
                   jqyoui-options="{revert:␣'invalid
                   ',␣helper:␣'clone'}" ng-model="cat
                   " jqyoui-draggable="{index:␣{{
                   $index}},␣placeholder:␣'keep'}">{{
                   cat.name}}</span>


                </div>
                <div class="col-sm-10">

                        <p>Tweets per page</p>
                        <select ng-model="
                           itensPerPage" ng-options="
                           o␣as␣o␣for␣o␣in␣
                           [10,20,30,50,100]" ng-
                           click="changeAmount()">
                        </select>
                        <p>Filter by Category</p>


                        <select ng-model="selectedCat
                           " ng-options="category.
                           name␣for␣category␣in␣
```

```
   project.categories" ng-
   click="selectCategory()">
</select>

<br><br>
<form ng-submit="search()"
   class="form-inline" role="
   form">

         <div class="form-
            group">
               <input type="
                  text"
                  class="
                  form-
                  control"
                  ng-model="
                  searchBy"
                  placeholder
                  ="Search"
                  required>
         </div>

         <button type="submit"
             class="btn␣btn-
            primary">Submit</
            button>
</form>

<br><br>

<table class="table␣table-
   hover">
         <thead>
                  <tr>
                           <th>
                              Text
                              </
                              th
```

```
                                    >
                                <th>
                                    Published

                                    by
                                    </
                                    th
                                    >
                                <th>
                                    Created

                                    at
                                    </
                                    th
                                    >
                                <!--
                                    <
                                    th
                                    >
                                    Class
                                    </
                                    th
                                    >
                                    --
                                    >
                        </tr>
                </thead>
                <tbody>
                        <tr ng-repeat
                            ="tweet␣in
                            ␣tweets">

                                <td><
                                    a
                                    href
                                    ="
                                    #/
                                    project
                                    /{{
```

```
project
.
id
}}/
dataset
/{{
dataset
.
id
}}/
tweet
/{{
tweet
.
id
}}
">
{{
tweet
.
text
}}
</
a>
<
div

class
="
thumbnail
"
data
-
drop
="
true
"
ng
-
```

```
model
='
tweet
.
selCategories
'
data
-
jqyoui
-
options

jqyoui
-
droppable
="
{
multiple
:
true
,␣
onDrop
:'
onDrop
(
tweet
.
selCategories
,␣
tweet
.
id
)
'}
">
      <
         span

         class
```

```
=
"
lab
␣
lab
-
def
"

ng
-
rep
=
"
ite
␣
in
␣
twe
.
sel
␣
tra
␣
by
␣
$in
"

ng
-
sho
=
"
ite
.
nam
"
```

```
style
=
"
backgr
-
color
:{{
item
.
color
}};
"

data
-
jqyoui
-
option
=
"
{
revert
:
␣
'
invali
',
␣
helper
:
␣
'
clone
'}
"

ng
-
model
```

```
=
"
twe
.
sel
"

jqy
-
dra
=
"
{
inc
:
␣
{{
$in
}}
ani
:
tru
}
"
>
{{
ite
.
nam
}}
<
/
spa
>
<
/
td
>
<
```

```
    /
    td
    >

<
    td
    >
    {{
    tweet
    .
    fromUs
    }}
    <
    /
    td
    >

<
    td
    >
    {{
    tweet
    .
    create
    }}
    <
    /
    td
    >

<
    !
    --

    <
    td
    >
    {{
    tweet
```

```
                                                    .
                                                    cla
                                                    }}
                                                    <
                                                    /
                                                    td
                                                    >

                                                    --
                                                    >
                                    </tr>

                            </tbody>
                    </table>

                    <pagination page="
                       currentPage" ng-
                       click="getPage()"
                       boundary-links="
                       true" total-items=
                       "count" items-per-
                       page="itensPerPage
                       " max-size="
                       maxSize" previous-
                       text="&lsaquo;"
                       next-text="&rsaquo
                       ;" first-text="&
                       laquo;" last-text=
                       "&raquo;"></
                       pagination>

                    <p>Total: {{count}}</
                       p>
                    <button type="button"
                        class="btn␣btn-
                       default" ng-click=
                       "toCSV()">Export
```

```
                                                    to CSV</button>


                                        </div>
                                </div>
                        </div>


        </div>


<div class="col-md-12" ng-controller="ViewConversationCtrl">
<h1 class="page-header">
        Tweet <small>ver conversa  o</small>
</h1>

        <spinner name="conversationSpinner" on-loaded="
           loadConversation()"><i class="fa␣fa-spinner␣fa-
           spin␣fa-2x␣fa-fw"></i></spinner>

        <div class="alert␣alert-danger" ng-if="msg" style="
           white-space:␣pre">{{msg}}</div>


<div ng-if="conversation">


        <blockquote ng-repeat="conv␣in␣conversation" style="
           border-left-width:␣10px;" ng-style="{'border-left-
           color':␣(conv.id␣!=␣tweet.idTweet)␣?␣'#EEEEEE'␣:␣
           '#286090'}"><p lang="pt" dir="ltr">{{conv.text}}</
```

```
            p>
                    {{conv.name}} ({{conv.username}}) <a href
                    ="https://twitter.com/{{conv.username}}/
                    status/{{conv.id}}">{{conv.date}}</a></
                    blockquote>



        <button type="button" class="btn btn-primary" ng-
            click="getMore()" ng-show="hasMore">Buscar mais
            tweets</button>

        <button type="button" class="btn btn-primary" ng-csv=
            "export()" csv-header="['text', 'username', 'id',
            'time']" field-separator="|" filename="
            conversation_{{idConv}}.csv">Exportar para CSV</
            button>

</div>

</div>
<div class="col-md-12" ng-controller="ErrorCtrl">

<h4>Voc  n o tem autoriza  o para acessar este conte do.
    </h4>


</div>
<div class="col-md-12 ng-scope" ng-controller="FiltersCtrl"
    ng-init="collapse=1">

     <h1 class="page-header">
         Filtros
     </h1>

     <div class="panel-group" id="accordion">

         <div class="panel panel-default">
             <div class="panel-heading">
```

```
        <h4 class="panel-title">
            <a data-toggle="collapse" data-parent="#
               accordion" href="" ng-click="collapse
               =1" class="collapsed">Selecionar
               datasets</a>
        </h4>
    </div>
    <div id="collapse2" class="panel-collapse␣
       collapse␣in␣ng-hide" ng-show="collapse==1"
       style="">
        <div class="panel-body">

            <div ng-repeat="dataset␣in␣state.
               selectedProject.datasets">
                <label>
                    <input type="checkbox" checklist-
                       model="selectedDatasets"
                       checklist-value="dataset.id">
                       {{dataset.name}}
                </label> <br>
            </div>

        </div>
    </div>
    <div class="panel␣panel-default">
        <div class="panel-heading">
            <h4 class="panel-title">
                <a data-toggle="collapse" data-parent
                   ="#accordion" href="" ng-click="
                   collapse=2">Selecionar filtro</a>
            </h4>
        </div>
        <div id="collapse3" class="panel-collapse␣
           collapse␣in" ng-show="collapse==2" style="
           ">
            <div class="panel-body">
                <div ng-show="showFilters()" class=""
                   >
```

```
<div class="form-group␣col-xs-5">
<select class="form-control"
   required ng-model="state.
   filterSelected" ng-click="
   configFilterDefault()">

    <option ng-repeat="filter␣in␣
       filters" value="{{filter.
       type}}" class="ng-binding␣
       ng-scope">{{filter.label}}
       </option>
</select>

    <div ng-if="state.
       filterSelected==’
       searchByTerm’">

        <br>
        <form class="form-inline"
            role="form">

            <div class="form-
               group">
                <input type="text
                   " class="form-
                   control" ng-
                   model="state.
                   searchBy"
                   placeholder="
                   Search"
                   required>
            </div>

            <button type="submit"
                ng-click="
               configSearchByTerm
               (state.searchBy)"
               class="btn␣btn-
               primary">Enviar</
```

```
            button>
      </form>
</div>


<div ng-if="state.
   filterSelected==’
   byCategory’">
      <br>
      <p>Selecionar categoria</
         p>
      <div class="form-group␣
         col-xs-10">
         <select class="form-
            control" required
            ng-model="state.
            selCategory" ng-
            options="category.
            name␣for␣category␣
            in␣state.
            selectedProject.
            categories␣|filter
            ␣:␣{type:␣’
            SPACES_OF_POSSIBILITY
            ’␣}␣" ng-click="
            configFilterByCategory
            (state.selCategory
            .id)">
            <option value=""
               ng-if="false">
               </option>
         </select>
      </div>
</div>

<div ng-if="state.
   filterSelected==’
   extractFragment’">
      <br>
```

```
<p>Selecione o tamanho do
    fragmento</p>
<input type="number" min=
   "1" ng-init="state.
   fragmentSize = 500" ng
   -model="state.
   fragmentSize">
<br>
<br>
<p>Indique o ponto do
   dataset para extrair o
    fragmento</p>
<div class="form-group
   col-md-6">

    <select class="form-
       control" required
       ng-model="state.
       cutPoint">

        <option ng-repeat
           ="cp in
           cutPoints"
           value="{{cp.
           type}}" class=
           "ng-binding ng
           -scope">{{cp.
           label}}</
           option>
    </select>


    <!--<select class="
       form-control"
       required ng-model=
       "state.point" ng-
       options="label for
       label in ['
       In cio ','Meio','
```

```
                                        Fim']">-->
                                          <!--<option value
                                             ="" ng-if="
                                             false"></
                                             option>-->
                                       <!--</select>-->
                                       <br><br>
                                       <button type="submit"
                                           ng-click="
                                          configExtractFragment
                                          ()" class="btn btn
                                          -primary">Enviar</
                                          button>
                                   </div>

                                </div>
                            </div>


                        </div>
                        <div ng-hide="showFilters()" class="
                           ng-hide">
                            <p>Selecione um ou mais datasets<
                               /p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <div class="container-fluid" ng-show="state.
       showTweets">
        <div class="row">
            <div class="col-sm-2">

                <h3>Categorias</h3>

                <div ng-repeat="cat in state.
```

```
                selectedProject.categories␣|filter
                ␣:␣{type:␣'RELEVANT_PROCESS'␣}"> <
                span  class="label␣label-primary"
                ng-style="{'background-color':'{{
                cat.color}}'}" ng-show="cat.name"
                data-drag="true" data-jqyoui-
                options="{revert:␣'invalid',␣
                helper:␣'clone'}" ng-model="cat"
                jqyoui-draggable="{index:␣{{$index
                }},␣placeholder:␣'keep'}" uib-
                popover="{{cat.description}}"
                popover-title="{{cat.name}}">{{cat
                .name}}</span><br>
            </div>

        </div>
        <div class="col-sm-10">
            <div class="container">
                <div class="col-md-2"> <h3>Resultados
                    </h3><spinner name="filterSpinner"
                    ><i class="fa␣fa-spinner␣fa-spin␣
                    fa-2x␣fa-fw"></i></spinner></div>
            </div>


            <div class="col-sm-8">

                <div class="col-sm-4">
                    <p>Itens por p gina</p>
                </div>
                <div class="col-sm-3">

            <select class="form-control␣input-sm" ng-
               model="state.itensPerPage" ng-click="
               changeAmount(state.itensPerPage)" ng-
               options="qtt␣for␣qtt␣in␣
               [10,20,30,50,100]" class="ng-pristine␣
               ng-untouched␣ng-valid␣ng-not-empty">
                    <option value="" ng-if="false">></
```

```
                option>
            </select>

            </div>
            </div>

        <div class="col-sm-12␣div-table-content">

        <table class="table␣table-hover">
            <thead>
            <tr>
                <th>#</th>
                <th>Texto</th>
                <th>Autor</th>
                <th>Publicado em</th>
                <th></th>
            </tr>
            </thead>
            <tbody>
            <tr ng-repeat="tweet␣in␣state.tweets"
               >
                <td>{{state.skipped + $index +
                   1}}</td>
                <td><a href="#/project/{{state.
                   selectedProject.id}}/dataset
                   /{{tweet.idDataset}}/tweet/{{
                   tweet.id}}">{{tweet.text}}</a>
                   <div class="thumbnail" data-
                   drop="true" ng-model="tweet.
                   selTags" data-jqyoui-options="
                   " jqyoui-droppable="{multiple:
                   true,␣onDrop:'onDrop(tweet.
                   selTags,␣tweet.id)'}">

                    <span ng-repeat="item␣in␣
                       tweet.selTags␣track␣by␣
                       $index" class="ng-scope"><
                       span class="label␣label-
                       default" ng-show="item.
```

```
            name" data-jqyoui-options=
            "{revert: ' invalid ' , 
            helper: ' clone ' }" ng-style=
            "{ ' background-color ' : ' {{
            item.color}} ' }" ng-model="
            tweet.selTags" jqyoui-
            draggable="{index: 0,
            animate:true}">{{item.name
            }}</span><a class="close2"
             href="" ng-click="
            removeCat(tweet.selTags,
            item.id, tweet.id)">    </
            a></span></div></td>
<td> {{tweet.fromUser}}</td>
<td>{{tweet.unixTimestamp * 1000
   | date:"dd/MM/yyyy HH:mm:ss"}}
   </td>
<td>

    <div class="dropdown">
        <a type="button" data-
           toggle="dropdown"><i
           class="fa fa-cog fa-lg
           " aria-hidden="true"><
           /i></a>
        <ul class="dropdown-menu 
           dropdown-menu-right">
            <li ng-click="
               setVisibility(
               tweet)">
                <a ng-if="tweet.
                   isVisible"><i
                   class="fa fa-
                   eye-slash"
                   aria-hidden="
                   true"></i>
                   Hide tweet</a>
                <a ng-if="!tweet.
                   isVisible"><i
```

```
                                        class="fa␣fa-
                                        eye" aria-
                                        hidden="true">
                                        </i> Unhide
                                        tweet</a>
                                </li>
                                <li ><a  href="#/
                                   project/{{state.
                                   selectedProject.id
                                   }}/dataset/{{tweet
                                   .idDataset}}/tweet
                                   /{{tweet.id}}/view
                                   -conversation"><i
                                   class="fa␣fa-
                                   comments-o" aria-
                                   hidden="true"></i>
                                    View Conversation
                                   </a></li>
                            </ul>
                        </div>


                    </td>
                </tr>

            </tbody>
            </table>
                </div>

                <div class="col-sm-12">


            <uib-pagination ng-disabled="
               disablePagination" boundary-links="true"
               total-items="state.count" ng-click="
               changePage()" ng-model="state.currentPage"
                class="pagination-sm" previous-text="&
               lsaquo;" next-text="&rsaquo;" first-text="
               &laquo;" last-text="&raquo;" max-size="10"
```

```
                    items-per-page="state.itensPerPage"></uib
                   -pagination>
                    <p>Total : {{state.count}}</p>
                  <button type="button" class="btn btn-default
                    " ng-click="toCSV()">Exportar para CSV</
                    button>

                    </div>

            </div>
        </div>


</div>
</div><div class="form-group">
<label for="name">Nome</label>
<input type="text" class="form-control" id="name" ng-model="
   project.name" required>
</div>

<div class="form-group">
<label for="description">Descri  o</label><br>
<textarea name="description" id="description" cols="80" ng-
   model="project.description" rows="10" required ></textarea
   >
</div>
<button type="submit" class="btn btn-default">Enviar</button>


<h1 class="page-header">
    Projeto <small>edi  o</small>
</h1>

<ol class="breadcrumb">
        <li>
        <i class="fa fa-tasks"></i>  <a href="#/projects">
           Projects</a>
        </li>
        <li class="active">
```

```
                <i class="fa␣fa-pencil-square-o"></i> Update
        </li>
</ol>

<div ng-controller="ProjectUpdateCtrl">

<div ng-if="error" class="alert␣alert-danger">
  {{msg}}
</div>

        <form ng-submit="update()" role="form">

                <div ng-include src="'app/components/project/
                   templates/form.html'"></div>

        </form>



</div>



<div class="col-md-12" ng-controller="ProjectSetCtrl">

    <div class="row">
        <div class="col-md-3">
            <h4>Escolha um projeto para trabalhar</h4>
            <select ng-attr-size="{{projects.length}}" class=
               "form-control"  ng-options="project.name␣for␣
               project␣in␣projects" ng-model="selectedProject
               " >
                <option value="" ng-if="false"></option>
            </select>
            <br>
            <button type="button" ng-click="setProject()"
               class="btn␣btn-default">OK</button>

            <br><br>
```

```
        <a ng-show="isAuthorized([userRoles.admin,
           userRoles.moderator])" href="#/project/add"><
           button type="button" class="btn btn-default">
           Adicionar projeto</button></a>

    </div>

    <div class="col-md-9" ng-if="selectedProject!=null">

        <h4>Detalhes</h4><br>

        <p><strong>Nome: </strong>{{selectedProject.name
           }}</p>

        <p><strong>Descri  o: </strong>{{
           selectedProject.description}}</p>
        <br>
        <div>
            <h4>Datasets</h4>

            <table class="table">
                <tbody>
                <tr ng-repeat="dataset in selectedProject
                   .datasets">
                    <td>{{dataset.name}}</td>
                </tr>

                </tbody>
            </table>

        </div>

        <div >
            <h4>Categorias</h4><br>

            <p><strong>Espa os de possibilidade</strong>
               </p><br>
```

```
                <table class="table">
                    <tbody>
                    <tr ng-repeat="category␣in␣
                       selectedProject.categories␣|␣filter␣:␣
                       {type:␣'SPACES_OF_POSSIBILITY'␣}">
                        <td>{{category.name}}</td>
                    </tr>

                    </tbody>
                </table>

                <br><p><strong>Fatores e circunst ncias</
                   strong></p><br>

                <table class="table">
                    <tbody>
                    <tr ng-repeat="category␣in␣
                       selectedProject.categories␣|␣filter␣:␣
                       {type:␣'RELEVANT_PROCESS'␣}">
                        <td>{{category.name}}</td>
                    </tr>

                    </tbody>
                </table>

            </div>

        </div>
    </div>


</div>
<h1 class="page-header">
        Projeto <small>detalhes</small>
</h1>


<div ng-controller="ProjectViewCtrl">
```

```
<p><strong>Nome: </strong>{{project.name}}</p>

<p><strong>Descri  o: </strong>{{project.
   description}}</p>

<a href="#/project/{{project.id}}/update" ng-show="
   isAuthorized([userRoles.admin,userRoles.moderator
   ])"><button type="button" class="btn btn-default">
   Editar</button></a>

<button ng-show="isAuthorized([userRoles.admin,
   userRoles.moderator])" type="button" class="btn
   btn-danger" ng-click="delete(project.id,project.
   name)">Excluir</button>

<div ng-show="hasDatasets()">
        <h3>Datasets</h3>

        <table class="table">
                <tbody>
                        <tr ng-repeat="dataset in
                           project.datasets">
                                <td><a href="#/
                                   project/{{project.
                                   id}}/dataset/{{
                                   dataset.id}}">{{
                                   dataset.name}}</a>
                                   </td>
                        </tr>

                </tbody>
        </table>

</div>


<a href="#/project/{{project.id}}/dataset/add" ng-
   show="isAuthorized([userRoles.admin,userRoles.
   moderator])"><button type="button" class="btn btn-
```

```
    default">Adicionar Dataset</button></a>

<div ng-show="hasCategories()">
        <h3>Categorias</h3><br>

        <h4>Espa os de possibilidade</h4><br>

        <table class="table">
            <tbody>
                <tr ng-repeat="category␣in␣
                    project.categories␣|␣
                    filter␣:␣{type:␣'
                    SPACES_OF_POSSIBILITY'␣}">
                    <td><a href="#/
                        project/{{project.
                        id}}/category/{{
                        category.id}}">{{
                        category.name}}</a
                        ></td>
                    <td><button type="
                        button" class="btn
                        ␣btn-danger" ng-
                        show="isAuthorized
                        ([userRoles.admin,
                        userRoles.
                        moderator])" ng-
                        click="
                        deleteCategory(
                        category.id)">
                        Excluir</button></
                        td>
                </tr>

            </tbody>
        </table>

        <br><h4>Fatores e circunst ncias</h4><br>

        <table class="table">
```

```
                        <tbody>
                        <tr ng-repeat="category␣in␣project.
                           categories␣|␣filter␣:␣{type:␣'
                           RELEVANT_PROCESS'␣}">
                              <td><a href="#/project/{{
                                 project.id}}/category/{{
                                 category.id}}">{{category.
                                 name}}</a></td>
                              <td><button type="button"
                                 class="btn␣btn-danger" ng-
                                 show="isAuthorized([
                                 userRoles.admin,userRoles.
                                 moderator])" ng-click="
                                 deleteCategory(category.id
                                 )">Excluir</button></td>
                        </tr>

                        </tbody>
                  </table>

         </div>


         <a href="#/project/{{project.id}}/category/add"><
            button type="button" class="btn␣btn-default">
            Adicionar categoria</button></a>

</div>




<h1 class="page-header">
    Projetos
</h1>
```

```
<div  ng-controller="ProjectListCtrl">

                <table ng-show="hasProjects()" class="table">
                    <thead>
                      <tr>
                        <th>Nome</th>
                        <th>Op  es</th>
                      </tr>
                    </thead>
                    <tbody>
                      <tr ng-repeat="project in projects">
                        <td><a href="#/project/{{project.id}}
                           ">{{project.name}}</a></td>
                        <td ><a href="#/project/{{project.id
                           }}"><button type="button" class="
                           btn btn-primary">View</button></a>
                                        <button ng-show="
                                           isAuthorized([
                                           userRoles.admin,
                                           userRoles.
                                           moderator])" type=
                                           "button" class="
                                           btn btn-danger" ng
                                           -click="delete(
                                           project.id,project
                                           .name)">Delete</
                                           button>
                        </td>
                      </tr>

                    </tbody>
                  </table>

                  <a ng-show="isAuthorized([userRoles.admin,
                     userRoles.moderator])" href="#/project/
                     add"><button type="button" class="btn
                     btn-default">Add Project</button></a>
```

```
</div>



<h1 class="page-header">
        Projeto <small>cadastro</small>
</h1>



<div ng-controller="ProjectAddCtrl">

        <div ng-if="error" class="alert␣alert-danger">
          {{msg}}
        </div>

        <form ng-submit="add()" role="form">

                <div ng-include src="'app/components/project/
                   templates/form.html'"></div>

        </form>

</div>



<h1 class="page-header">
    Conjunto de treinamento
</h1>

<div class="col-md-12" ng-controller="TrainingSetCtrl" ng-
   init="collapse=1">

    <div class="panel-group" id="accordion">

        <div class="panel␣panel-default">
            <div class="panel-heading">
```

```
                <h4 class="panel-title">
                    <a data-toggle="collapse" data-parent="#
                       accordion" href="" ng-click="collapse
                       =1" class="collapsed">Selecionar
                       datasets</a>
                </h4>
            </div>
            <div id="collapse2" class="panel-collapse␣
               collapse␣in␣ng-hide" ng-show="collapse==1"
               style="">
                <div class="panel-body">

                    <select ng-options="dataset.name␣for␣
                       dataset␣in␣selectedProject.datasets"
                       ng-model="selectedDataset" ></select>

                    <br><br>
                    <label for="includeRepeated"><input type=
                       "checkbox" ng-model="includeRepeated"
                       id="includeRepeated"> Incluir
                       repetidos</label>

                    <br><br>
                    <button class="btn␣btn-default" ng-click=
                       "defDataset()">OK</button>

                </div>
            </div>

        </div>
    </div>


    <div class="container-fluid" ng-show="showTweets">
        <div class="row">

            <div class="col-sm-2">
                <h3>Categorias</h3>
```

```
        <div ng-repeat="cat␣in␣selectedProject.
           categories␣|filter␣:␣{type:␣'
           SPACES_OF_POSSIBILITY'␣}"> <span  class="
           label␣label-primary" ng-style="{'
           background-color':'{{cat.color}}'}" ng-
           show="cat.name" data-drag="true" data-
           jqyoui-options="{revert:␣'invalid',␣helper
           :␣'clone'}" ng-model="cat" jqyoui-
           draggable="{index:␣{{$index}},␣placeholder
           :␣'keep'}" uib-popover="{{cat.description
           }}" popover-title="{{cat.name}}">{{cat.
           name}}</span> <span class="badge">{{
           getCount(cat.id)}}</span><br>
        </div>
    </div>

    <div class="col-sm-10">
        <h3>Resultados</h3>
        <div class="col-sm-8">

            <div class="col-sm-4">
                <p>Itens por p gina</p>
            </div>
            <div class="col-sm-3">

                <select class="form-control␣input-sm"
                    ng-model="itensPerPage" ng-click=
                   "changeAmount(itensPerPage)" ng-
                   options="qtt␣for␣qtt␣in␣
                   [10,20,30,50,100]" class="ng-
                   pristine␣ng-untouched␣ng-valid␣ng-
                   not-empty">
                     <option value="" ng-if="false"></
                        option>
                </select>
            </div>
        </div>

        <div class="col-sm-12␣div-table-content">
```

```
<table class="table table-hover">
    <thead>
    <tr>
        <th>#</th>
        <th>Texto</th>
        <th>Autor</th>
        <th>Publicado em</th>

    </tr>
    </thead>
    <tbody>
    <tr ng-repeat="tweet in tweets">
        <td>{{((itensPerPage * (
           currentPage-1)) + $index) +
           1}}</td>
        <td><a href="#/project/{{
           selectedProject.id}}/dataset
           /{{tweet.idDataset}}/tweet/{{
           tweet.id}}">{{tweet.text}}</a>
           <div class="thumbnail" data-
           drop="true" ng-model="tweet.
           selClasses" data-jqyoui-
           options="" jqyoui-droppable="{
           multiple:true, onDrop:'onDrop(
           tweet.selClasses, tweet.id)'}"
           >

            <span ng-repeat="item in
               tweet.selClasses track by
               $index" class="ng-scope"><
               span class="label label-
               default" ng-show="item.
               name" data-jqyoui-options=
               "{revert: 'invalid', 
               helper:'clone'}" ng-style=
               "{'background-color':'{{
               item.color}}'}" ng-model="
               tweet.selClasses" jqyoui-
```

```
                                    draggable="{index:␣0,
                                    animate:true}">{{item.name
                                    }}</span><a class="close2"
                                     href="" ng-click="
                                    removeCat(tweet.selClasses
                                    ,item.id,␣tweet.id)">    <
                                    /a></span></div></td>
                                <td> {{tweet.fromUser}}</td>
                                <td>{{dateStringToTimeStamp(tweet
                                   .createdAt) | date:"dd/MM/yyyy
                                   ␣HH:mm:ss"}}</td>

                            </tr>

                            </tbody>
                        </table>
                    </div>

                    <div class="col-sm-12">


                        <uib-pagination boundary-links="true"
                           total-items="count" ng-click="
                           changePage()" ng-model="currentPage"
                           class="pagination-sm" previous-text="&
                           lsaquo;" next-text="&rsaquo;" first-
                           text="&laquo;" last-text="&raquo;" max
                           -size="10" items-per-page="
                           itensPerPage"></uib-pagination>
                        <p>Total : {{count}}</p>

                    </div>

                </div>
            </div>


        </div>
```

```
</div>
<div class="form-group">
<label for="name">Name</label>
<input type="text" class="form-control" id="name" ng-model="
   user.name" required>
<label for="email">E-mail</label>
<input type="email" class="form-control" id="name" ng-model="
   user.email" required>
<label for="password">Password</label>
<input type="password" class="form-control" id="name" ng-
   model="user.password" required>
</div>
<button type="submit" class="btn btn-default">Submit</button>
<h1 class="page-header">
        User <small>profile</small>
</h1>


<div class="col-md-12" ng-controller="UserViewCtrl">

<label>Nome:</label>
<p>{{user.name}}</p>
<label>E-mail</label>
<p>{{user.email}}</p>
<label>Fun   o</label>
<p>{{user.role}}</p>

        <a href="#/change_password"><button type="button"
           class="btn btn-default">Alterar senha</button></a>

</div>
<div class="col-md-12" ng-controller="UpdateCtrl">

    <div ng-if="error" class="alert alert-danger">
        {{msg}}
    </div>

    <form ng-submit="signup()" role="form" name="userForm"
```

```
    novalidate>

     <div class="form-group">
         <label for="name">Name</label>
         <input type="text" name="name" class="form-
            control"  id="name" ng-model="user.name"
            required>

         <div ng-messages="userForm.name.$error">

             <p ng-message="required">Your name is
                required.</p>
         </div>


         <label for="email">E-mail</label>
         <input type="email" class="form-control" name="
            email" id="email" ng-model="user.email"
            required>


         <div ng-messages="userForm.email.$error">

             <p ng-message="required">Your email is
                required.</p>
             <p ng-message="email">Your email is invalid.<
                /p>
         </div>



     </div>
     <button type="submit" ng-class="userForm.$valid ? '
        btn btn-default active' : 'btn btn-default
        disabled'">Submit</button>


</form>
```

```
</div>
<div class="col-md-12" ng-controller="UserUpdateCtrl">


</div>
<h1 class="page-header">
        User <small>sign up</small>
</h1>



<div ng-controller="UserSignupCtrl">

        <div ng-if="error" class="alert␣alert-danger">
                {{msg}}
        </div>

        <form ng-submit="signup()" role="form" name="userForm
           " novalidate>

                <div class="col-md-4␣form-group">
                        <label for="name">Nome</label>
                        <input type="text" name="name" class=
                           "form-control"  id="name" ng-model
                           ="user.name" required>

                        <div ng-messages="userForm.name.
                           $error">

                                <p ng-message="required">Nome
                                        campo obrigat rio.</p
                                   >
                        </div>


                        <label for="email">E-mail</label>
                        <input type="email" class="form-
                           control" name="email" id="email"
                           ng-model="user.email" required>
```

```
<div ng-messages="userForm.email.
   $error">

        <p ng-message="required">E-
           mail    campo obrigat rio
           .</p>
        <p ng-message="email">E-mail
           inv lido.</p>
</div>


<label for="password">Senha</label>
<input type="password" class="form-
   control" name="password" id="
   password" ng-model="user.password"
    required>

<div ng-messages="userForm.password.
   $error">

        <p ng-message="required">A
           senha    campo
           obrigat rio.</p>
</div>

<label for="confirm-password">
   Confirmar senha</label>
<input type="password" class="form-
   control" name="confirmPassword" id
   ="confirm-password" ng-model="
   confirmPassword" compare-to="user.
   password" required>

<div ng-messages="userForm.
   confirmPassword.$error">
        <p ng-message="required">>A
           confirma  o da senha
```

```
                                    campo obrigat rio.</p>
                                <div ng-message="compareTo">
                                    Senhas n o coincidem</div
                                    >
                            </div>

                            <br>

                            <button type="submit" ng-class="
                                userForm.$valid␣?␣'btn␣btn-default
                                ␣active'␣:␣␣'btn␣btn-default␣
                                disabled'">Submit</button>
                        </div>



        </form>

</div>



<h1 class="page-header">
        User <small>login</small>
</h1>

<div ng-controller="UserLoginCtrl">

        <div ng-if="error" class="alert␣alert-danger">
                {{msg}}
        </div>

        <form ng-submit="login()" role="form">

                <div class="col-md-4␣form-group">
                <label for="email">E-mail</label>
                <input type="email" class="form-control" id="
                    email" ng-model="user.email" required>
                <label for="password">Senha</label>
```

```
                <input type="password" class="form-control"
                   id="password" ng-model="user.password"
                   required>
                        <br>
                        <button type="submit" class="btn␣btn-
                           default">Enviar</button>
                </div>


        </form>

        <!--<div align="center"><button class="btn␣btn-danger
           ␣btn-lg" ng-click="authenticate()"><i class="fa␣fa
           -google-plus"></i> Sign in with Google</button></
           div>-->

</div>



<div class="col-md-12" ng-controller="ChangePasswordCtrl">

    <h1 class="page-header">
        User <small>change password</small>
    </h1>


    <div ng-if="error" class="alert␣alert-danger">
        {{msg}}
    </div>

    <form ng-submit="changePw()" role="form" name="
       changePwForm" novalidate>

        <div class="form-group">

            <label for="password">Senha</label>
            <input type="password" class="form-control" name=
               "password" id="password" ng-model="user.
```

```
                password" required>

            <div ng-messages="changePwForm.password.$error">

                <p ng-message="required">A senha    campo
                    obrigat rio.</p>
            </div>

            <label for="confirm-password">Confirmar senha</
                label>
            <input type="password" class="form-control" name=
                "confirmPassword" id="confirm-password" ng-
                model="confirmPassword" compare-to="user.
                password" required>

            <div ng-messages="changePwForm.confirmPassword.
                $error">
                <p ng-message="required">A confirma  o da
                    senha    campo obrigat rio.</p>
                <div ng-message="compareTo">Senhas  n  o
                    coincidem</div>
            </div>
        </div>
        <button type="submit" ng-class="changePwForm.$valid␣?
            ␣'btn␣btn-default␣active'␣:␣␣'btn␣btn-default␣
            disabled'">Enviar</button>


    </form>

</div>
<div class="col-md-12" ng-controller="HomeCtrl">


<div ng-show="isActive()">

<h3>Bem vindo, {{userActive.name}}</h3>
<br>
```

```
    <!--<div class="row">-->
<!--<div class="col-md-3">-->
    <!--<h4>Choice a project to work</h4>-->
    <!--<select ng-attr-size="{{projects.length}}" class="
       form-control"  ng-options="project.name␣for␣project␣in
       ␣projects" ng-model="selectedProject" >-->
        <!--<option value="" ng-if="false"></option>-->
    <!--</select>-->
    <!--<br>-->
    <!--<button type="button" ng-click="setProject()" class="
       btn␣btn-default">OK</button>-->
<!--</div>-->

    <!--<div class="col-md-9" ng-if="selectedProject!=null">
       -->

        <!--<h4>Project Details</h4><br>-->

        <!--<p><strong>Name: </strong>{{selectedProject.name
           }}</p>-->

        <!--<p><strong>Description: </strong>{{
           selectedProject.description}}</p>-->
        <!--<br>-->
        <!--<div>-->
            <!--<h4>Datasets</h4>-->

            <!--<table class="table">-->
                <!--<tbody>-->
                <!--<tr ng-repeat="dataset␣in␣selectedProject
                   .datasets">-->
                    <!--<td>{{dataset.name}}</td>-->
                <!--</tr>-->

                <!--</tbody>-->
            <!--</table>-->

        <!--</div>-->
```

```
        <!--<div >-->
            <!--<h4>Categories</h4><br>-->

            <!--<p><strong>Spaces of possibility</strong></p>
               <br>-->

            <!--<table class="table">-->
                <!--<tbody>-->
                <!--<tr ng-repeat="category␣in␣
                   selectedProject.categories␣|␣filter␣:␣{
                   type:␣'SPACES_OF_POSSIBILITY'␣}">-->
                    <!--<td>{{category.name}}</td>-->
                <!--</tr>-->

                <!--</tbody>-->
            <!--</table>-->

            <!--<br><p><strong>Relevant process</strong></p><
               br>-->

            <!--<table class="table">-->
                <!--<tbody>-->
                <!--<tr ng-repeat="category␣in␣
                   selectedProject.categories␣|␣filter␣:␣{
                   type:␣'RELEVANT_PROCESS'␣}">-->
                    <!--<td>{{category.name}}</td>-->
                    <!--</tr>-->

                <!--</tbody>-->
            <!--</table>-->

        <!--</div>-->

    <!--</div>-->
    <!--</div>-->
</div>

<div ng-hide="isActive()">
```

```
<h4>Identifique-se no sistema para ter acesso  s
   funcionalidades</h4>

</div>


</div>

<div class="col-md-12 ng-scope" ng-controller="
   QuestioningCtrl" ng-init="collapse=1">

    <h1 class="page-header">
        Perguntas pr  -definidas
    </h1>

    <div class="panel-group" id="accordion">
        <div class="panel panel-default">
            <div class="panel-heading">
                <h4 class="panel-title">
                    <a data-toggle="collapse" data-parent="#
                       accordion" href="" ng-click="collapse
                       =1" class="collapsed">Selecionar
                       datasets</a>
                </h4>
            </div>
            <div id="collapse2" class="panel-collapse
               collapse in ng-hide" ng-show="collapse==1"
               style="">
                <div class="panel-body">

                    <div ng-repeat="dataset in state.
                       selectedProject.datasets">
                        <label>
                            <input type="checkbox" checklist-
                               model="selectedDatasets"
                               checklist-value="dataset.id">
                               {{dataset.name}}
                        </label>
                    </div>
```

```
    </div>
</div>
<div class="panel panel-default">
    <div class="panel-heading">
        <h4 class="panel-title">
            <a data-toggle="collapse" data-parent
               ="#accordion" href="" ng-click="
               collapse=2">Selecionar
               questionamento</a>
        </h4>
    </div>
    <div id="collapse3" class="panel-collapse
       collapse in" ng-show="collapse==2" style="
       ">
        <div class="panel-body">
            <div ng-show="showQuestioning()"
               class="col-xs-5">

                <select class="form-control"
                   required ng-model="state.
                   questSelected" ng-click="
                   configFilterDefault()">
                    <option ng-repeat="quest in
                       questionings" value="{{
                       quest.name}}" class="ng-
                       binding ng-scope">{{quest.
                       descr}}</option>
                </select>

                <div ng-if="state.questSelected
                   =='byCategoryAndUser'">

                    <br>
                    <p>Selecione uma categoria</p
                       >
                    <select class="form-control"
                       required ng-model="state.
                       selCategory" ng-options="
```

```
                category.name␣for␣category
                ␣in␣state.selectedProject.
                categories␣|filter␣:␣{type
                :␣'RELEVANT_PROCESS'␣}␣" >
                  <option value="" ng-if="
                     false"></option>
              </select>
              <br>
              <p>Seleciona um pesquisador</
                 p>
              <select class="form-control"
                 required ng-model="state.
                 selUser" ng-options="user.
                 name␣for␣user␣in␣users␣">
                  <option value="" ng-if="
                     false"></option>
              </select>
              <br>
              <form class="form-inline"
                 role="form">
                  <button type="submit" ng-
                     click="
                     configQuestByCategoryAndUser
                     (state.selCategory.id,
                     state.selUser.id)"
                     class="btn␣btn-primary
                     ">Submit</button>
              </form>

          </div>


          <div ng-if="state.questSelected
             =='byCategoryAndQttTags'">
              <br>

              <p>Selecione a quantidade
                 m nima de marca  es</p>
              <input type="number" min="1"
```

```
                ng-init="state.qttTags = 1
                " ng-model="state.qttTags"
                >

            <br><br>
            <p>Selecione a categoria</p>
            <select class="form-control"
                required ng-model="state.
                selCategory" ng-options="
                category.name for category
                in state.selectedProject.
                categories | filter : {type
                : 'RELEVANT_PROCESS' }" >
                 <option></option>
            </select>
            <br><br>
            <form class="form-inline"
                role="form">
                 <button type="submit" ng-
                    click="
                    configQuestByCategoryAndQttTags
                    (state.selCategory.id,
                    state.qttTags)" class=
                    "btn btn-primary">
                    Submit</button>
            </form>

        </div>

        <div ng-if="state.questSelected
            =='byDistinctCategories'">
            <br>
            <p>Selecione a quantidade
                m nima de categorias
                distintas por tweet</p>
            <input type="number" min="1"
                ng-init="state.qttCat = 1"
                 ng-model="state.qttCat">
```

```
                                <br><br>
                                <form class="form-inline"
                                   role="form">
                                    <button type="submit" ng-
                                       click="
                                       configQuestByDistinctCategories
                                       (state.qttCat)" class=
                                       "btn␣btn-primary">
                                       Submit</button>
                                </form>
                            </div>

                        </div>
                        <div ng-hide="showQuestioning()"
                           class="ng-hide">
                            <p>Selecione pelo menos um
                               dataset</p>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <div class="container-fluid" ng-show="state.showTweets">
            <div class="row">

                <div class="col-sm-12">
                    <div class="container">
                        <div class="col-md-2"> <h3>Resultados</h3
                           ><spinner name="questSpinner"><i class
                           ="fa␣fa-spinner␣fa-spin␣fa-2x␣fa-fw"><
                           /i></spinner></div>
                    </div>
                    <div class="col-sm-8">

                        <div class="col-sm-4">
                            <p>Tweets por p gina</p>
                        </div>
```

```
        <div class="col-sm-3">

            <select class="form-control␣input-sm"
                ng-model="state.itensPerPage" ng-
               click="changeAmount(state.
               itensPerPage)" ng-options="qtt␣for
               ␣qtt␣in␣[10,20,30,50,100]" class="
               ng-pristine␣ng-untouched␣ng-valid␣
               ng-not-empty">
                <option value="" ng-if="false"></
                   option>
            </select>

        </div>
    </div>

    <div class="col-sm-12␣div-table-content">

        <table class="table␣table-hover">
            <thead>
            <tr>
                <th>#</th>
                <th>Texto</th>
                <th>Autor</th>
                <th>Publicado em</th>
                <th></th>
            </tr>
            </thead>
            <tbody>
            <tr ng-repeat="tweet␣in␣state.tweets"
               >
                <td>{{state.skipped + $index +
                   1}}</td>
                <td><a href="#/project/{{state.
                   selectedProject.id}}/dataset
                   /{{tweet.idDataset}}/tweet/{{
                   tweet.id}}">{{tweet.text}}</a>
                   <div class="thumbnail" data-
                   drop="true" ng-model="tweet.
```

```
    selTags" data-jqyoui-options="
    " jqyoui-droppable="{multiple:
    true, onDrop:'onDrop(tweet.
    selTags, tweet.id)'}">

      <span ng-repeat="item in
        tweet.selTags track by
        $index" class="ng-scope"><
        span class="label label-
        default" ng-show="item.
        name" data-jqyoui-options=
        "{revert: 'invalid',
        helper:'clone'}" ng-style=
        "{'background-color':'{{
        item.color}}'}" ng-model="
        tweet.selTags" jqyoui-
        draggable="{index: 0,
        animate:true}">{{item.name
        }}</span><a class="close2"
         href="" ng-click="
        removeCat(tweet.selTags,
        item.id, tweet.id)">    </
        a></span></div></td>
<td> {{tweet.fromUser}}</td>
<td>{{tweet.unixTimestamp * 1000
    | date:"dd/MM/yyyy HH:mm:ss"}}
    </td>
<td>

      <a  href="#/project/{{state.
        selectedProject.id}}/
        dataset/{{tweet.idDataset
        }}/tweet/{{tweet.id}}/view
        -conversation"><i class="
        fa fa-comments-o" aria-
        hidden="true"></i> Ver
        di logo</a></li>
```

```
                        </td>
                    </tr>

                    </tbody>
                </table>

            </div>

            <div class="col-sm-12">

                <uib-pagination ng-disabled="
                    disablePagination" boundary-links="
                    true" total-items="state.count" ng-
                    click="changePage()" ng-model="state.
                    currentPage" class="pagination-sm"
                    previous-text="&lsaquo;" next-text="&
                    rsaquo;" first-text="&laquo;" last-
                    text="&raquo;" max-size="10" items-per
                    -page="state.itensPerPage"></uib-
                    pagination>
                <p>Total : {{state.count}}</p>
                <button type="button" class="btn␣btn-
                    default" ng-click="toCSV()">Exportar
                    para CSV</button>

            </div>



        </div>
    </div>


</div>
</div><h1 class="page-header">
        Administra  o <small>Listar usu rios</small>
</h1>
```

```
<div class="col-md-12" ng-controller="AdminViewUsersCtrl">


        <table class="table table-hover">
                <thead>
                        <tr>
                                <th>Nome</th>
                                <th>E-mail</th>
                                <th>Fun   o</th>
                                <th>Autorizado</th>
                        </tr>
                </thead>
                <tbody>
                        <tr ng-repeat="user in users">
                                <td>{{user.name}}</td>
                                <td>{{user.email}}</td>
                                <td>
                                        <select
                                        ng-options="option
                                           for option in
                                           roles"
                                        ng-model="user.role"
                                           ng-change="setRole
                                           (user.id,user.role
                                           )"></select>
                                </td>
                                <td><input type="checkbox" ng
                                   -model="user.enabled" ng-
                                   change="enable(user.id,
                                   user.enabled)"></td>
                        </tr>
                </tbody>

        </table>


</div>
<div class="form-group">
```

```
<label for="name">Nome</label>
<input type="text" class="form-control" id="name" ng-model="
   dataset.name" required>
</div>

<div class="form-group">
<label for="description">Descri  o</label><br>
<textarea name="description" id="description" cols="80" ng-
   model="dataset.description" rows="10" required ></textarea
   >
</div>



<button type="submit" class="btn btn-default">Submit</button>

<div ng-controller="DatasetUpdateCtrl">

        <h1 class="page-header">
                Dataset <small>edi  o</small>
        </h1>


        <div ng-if="error" class="alert alert-danger">
          {{msg}}
        </div>

        <form ng-submit="update()" role="form">


                <div ng-include src="'app/components/dataset/
                   templates/form.html'"></div>

        </form>

</div>
```

```
<div ng-controller="DatasetViewCtrl">

        <h1 class="page-header">
                Dataset <small>detalhes</small>
        </h1>


        <p><strong>Nome: </strong>{{dataset.name}}</p>

        <p><strong>Descri  o: </strong>{{dataset.
           description}}</p>

        <a href="#/project/{{idProject}}/dataset/{{dataset.id
           }}/tweets/import" ng-show="isAuthorized([userRoles
           .admin,userRoles.moderator])"><button ng-show="!
           dataset.hasTweets" type="button" class="btn btn-
           default">Importar Tweets</button></a>

        <a href="#/project/{{idProject}}/dataset/{{dataset.id
           }}/update" ng-show="isAuthorized([userRoles.admin,
           userRoles.moderator])"><button type="button" class
           ="btn btn-default">Editar</button></a>

        <button type="button" class="btn btn-default" ng-
           click="delete()" ng-show="isAuthorized([userRoles.
           admin,userRoles.moderator])">Excluir</button>


</div>

<!-- include tweet list -->




<h1 class="page-header">
```

```
    Datasets
</h1>

<div class="col-md-12" ng-controller="DatasetListCtrl">

    <table class="table">
        <thead>
        <tr>
            <th>Nome</th>
            <th ng-show="isAuthorized([userRoles.admin,
               userRoles.moderator])">Op  es</th>
        </tr>
        </thead>
        <tbody>
        <tr ng-repeat="dataset in datasets">
            <td><a href="#/project/{{projectId}}/dataset/{{
               dataset.id}}">{{dataset.name}}</a></td>
            <td ng-show="isAuthorized([userRoles.admin,
               userRoles.moderator])"><button type="button"
               class="btn btn-danger" ng-click="delete(
               dataset)">Excluir</button>
            </td>
        </tr>

        </tbody>
    </table>

    <a href="#/project/{{projectId}}/dataset/add" ng-show="
       isAuthorized([userRoles.admin,userRoles.moderator])"><
       button type="button" class="btn btn-default">Adicionar
        dataset</button></a>


</div>
<div ng-controller="DatasetAddCtrl">

<h1 class="page-header">
        Dataset <small>cadastro</small>
</h1>
```

```
        <div ng-if="error" class="alert␣alert-danger">
          {{msg}}
        </div>

        <form ng-submit="add()" role="form">

                <div ng-include src="'app/components/dataset/
                   templates/form.html'"></div>

        </form>

</div>



<div class="modal-header">
  <h3>{{modalOptions.headerText}}</h3>
</div>
<div class="modal-body">
  <p>{{modalOptions.bodyText}}</p>
</div>
<div class="modal-footer">
  <button type="button" class="btn"
          data-ng-click="modalOptions.close()">{{modalOptions
             .closeButtonText}}</button>
  <button class="btn␣btn-primary"
          data-ng-click="modalOptions.ok();">{{modalOptions.
             actionButtonText}}</button>
</div><!DOCTYPE html>
<html lang="en" ng-app="snaApp2">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,␣
       initial-scale=1">
    <meta name="description" content="">
```

```
<meta name="author" content="">

<title>SoNDA</title>

<!-- Bootstrap Core CSS -->
<link href="bower_components/bootstrap/dist/css/bootstrap
   .min.css" rel="stylesheet">

<!-- Custom CSS -->
<link href="assets/css/sb-admin.css" rel="stylesheet">

<!-- Custom Fonts -->
<link href="bower_components/font-awesome/css/font-
   awesome.min.css" rel="stylesheet" type="text/css">

<!-- HTML5 Shim and Respond.js IE8 support of HTML5
   elements and media queries -->
<!-- WARNING: Respond.js doesn't work if you view the
   page via file:// -->
<!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/libs/html5shiv
       /3.7.0/html5shiv.js"></script>
    <script src="https://oss.maxcdn.com/libs/respond.js
       /1.4.2/respond.min.js"></script>
    <![endif]-->

</head>

<body ng-controller="IndexController">

    <div id="wrapper">

        <!-- Navigation -->
        <nav class="navbar navbar-inverse navbar-fixed-
           top" role="navigation">
            <!-- Brand and toggle get grouped for better
               mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-
```

```
        toggle" data-toggle="collapse" data-
        target=".navbar-ex1-collapse">
          <span class="sr-only">Toggle
             navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#/"><i
        class="fa␣fa-home␣fa-1␣␣x"></i> SoNDA<
        /a>
</div>
<!-- Top Menu Items -->
<ul class="nav␣navbar-right␣top-nav">
      <li class="dropdown" ng-hide="isActive()"
        >

          <a href="#/signup" ><i class="fa␣fa-
             user-plus" aria-hidden="true"></i>
              Increver-se </a>
      </li>

  <li class="dropdown" ng-hide="isActive()">

      <a href="#/login" ><i class="fa␣fa-sign-
        in" aria-hidden="true"></i> Entrar </a
        >
</li>

      <li class="dropdown" ng-if="isActive()␣&&
        ␣projectFixed()">

          <a href="#/project/{{selectedProject.
             id}}" ><i class="fa␣fa-check" aria
             -hidden="true"></i> Projeto atual:
              {{selectedProject.name}} </a>
      </li>

<li class="dropdown" ng-show="isActive()">
```

```
            <a href="" class="dropdown-toggle" data-
               toggle="dropdown"><i class="fa␣fa-user
               "></i> {{userActive.name}} <b class="
               caret"></b></a>
            <ul class="dropdown-menu">
                <li>
                    <a href="#/user/profile"><i class
                       ="fa␣fa-fw␣fa-user"></i>
                       Perfil</a>
                </li>
                <li class="divider"></li>
                <li>
                    <a href="" ng-click="logout()"><i
                        class="fa␣fa-sign-out" aria-
                       hidden="true"></i> Sair</a>
                </li>
            </ul>
        </li>
    </ul>
    <!-- Sidebar Menu Items - These collapse to the
       responsive navigation menu on small screens --
       >
    <div  class="collapse␣navbar-collapse␣navbar-ex1-
       collapse">
        <ul class="nav␣navbar-nav␣side-nav">
            <li ng-if="isActive()">
                <a href="#/set_project"><i class="fa␣
                   fa-fw␣fa-tasks"></i> Projetos</a>
            </li>
            <li ng-if="isActive()␣&&␣projectFixed()">
                <a href="#/project/{{selectedProject.
                   id}}/categories"><i class="fa␣fa-
                   tags" aria-hidden="true"></i>
                   Categorias</a>
            </li>
            <li ng-if="isActive()␣&&␣projectFixed()">
                <a href="#/project/{{selectedProject.
                   id}}/datasets"><i class="fa␣fa-
                   database" aria-hidden="true"></i>
```

```
        Datasets</a>
</li>
<li ng-if="isActive() && projectFixed()">
    <a href="#/project/{{selectedProject.
       id}}/training_set"><i class="fa fa
       -database" aria-hidden="true"></i>
        Conjunto de treinamento</a>
</li>
<li ng-if="isActive() && projectFixed()">
    <a href="#/filters"><i class="fa fa-
       fw fa-filter"></i> Filtros</a>
</li>
<!--<li ng-if="isActive() && projectFixed
   ()">-->
    <!--<a href="#/questioning"><i class=
       "fa fa-question-circle" aria-
       hidden="true"></i> Questioning</a>
       -->
<!--</li>-->

<li ng-if="isActive() && projectFixed()">
    <a  href="javascript:;" data-toggle="
       collapse" data-target="#quest"><i
       class="fa fa-fw fa-arrows-v"></i>
       Questionamento <i class="fa fa-fw
       fa-caret-down"></i></a>
    <ul id="quest" class="collapse">
        <li>
            <a href="#/
               predefined_questions"><i
               class="fa fa-question-
               circle" aria-hidden="true"
               ></i> Perguntas pre-
               definidas</a>
        </li>
        <li>
            <a href="#/project/{{
               selectedProject.id}}/
               matrices"><i class="fa fa-
```

```
                                table" aria-hidden="true">
                                </i> Matrizes</a>
                            </li>
                        </ul>
                    </li>

                    <li ng-show="isAuthorized(userRoles.admin
                       )">
                    <a  href="javascript:;" data-toggle="
                       collapse" data-target="#admin"><i
                       class="fa fa-fw fa-arrows-v"></i>
                       Admin <i class="fa fa-fw fa-caret-down
                       "></i></a>
                        <ul id="admin" class="collapse">
                            <li>
                                <a href="#/admin/users">
                                   Listar usuarios</a>
                            </li>
                        </ul>
                    </li>
<!--                     <li>
                        <a href="javascript:;" data-toggle="
                           collapse" data-target="#demo"><i
                           class="fa fa-fw fa-arrows-v"></i>
                           Projects <i class="fa fa-fw fa-
                           caret-down"></i></a>
                        <ul id="demo" class="collapse">
                            <li>
                                <a href="#">Dropdown Item</a>
                            </li>
                            <li>
                                <a href="#">Dropdown Item</a>
                            </li>
                        </ul>
                    </li> -->
                </ul>
            </div>
            <!-- /.navbar-collapse -->
        </nav>
```

```
        <div id="page-wrapper">

            <div class="container-fluid">

                <!-- Page Heading -->
                <div class="row" >
                    <div class="col-lg-12" ng-view>

                    </div>
                </div>
                <!-- /.row -->

            </div>
            <!-- /.container-fluid -->

        </div>
        <!-- /#page-wrapper -->

    </div>
    <!-- /#wrapper -->

    <!-- jQuery -->
    <!-- // <script src="js/jquery.js"></script> -->

    <!-- Bootstrap Core JavaScript -->
    <!-- // <script src="js/bootstrap.min.js"></script> -->

    <!-- Livereload script for development only (stripped
       during dist build) -->
    <script src="http://localhost:35729/livereload.js" data-
       concat="false"></script>

    <!-- JS from Bower Components -->
    <script src="bower_components/less.js/dist/less-1.6.2.js"
        data-concat="false"></script>
    <script src="bower_components/jquery/dist/jquery.js"></
       script>
    <script src="bower_components/jquery-ui/jquery-ui.js"></
```

```
    script>
<script src="bower_components/bootstrap/dist/js/bootstrap
    .js"></script>
<script src="bower_components/underscore/underscore.js"><
    /script>
<script src="bower_components/moment/moment.js"></script>
<script src="bower_components/angular/angular.js"></
    script>
<script src="bower_components/angular-route/angular-route
    .js"></script>
<script src="bower_components/angular-animate/angular-
    animate.js"></script>
<script src="bower_components/angular-touch/angular-touch
    .js"></script>
<script src="bower_components/angular-cookies/angular-
    cookies.js"></script>
<script src="bower_components/angular-resource/angular-
    resource.js"></script>
<script src="bower_components/angular-bootstrap/ui-
    bootstrap-tpls.js"></script>
<script src="bower_components/ng-csv/build/ng-csv.js"></
    script>
<script src="bower_components/angular-ui-utils/ui-utils.
    js"></script>
<script src="bower_components/angular-sanitize/angular-
    sanitize.js"></script>
<script src="bower_components/ngstorage/ngStorage.js"></
    script>
<script src="bower_components/satellizer/satellizer.js"><
    /script>
<script src="bower_components/angular-dragdrop/src/
    angular-dragdrop.min.js"></script>
     <script src="bower_components/angular-messages/
        angular-messages.js"></script>
     <script src="bower_components/checklist-model/
        checklist-model.js"></script>
     <script src="bower_components/angular-file-saver/dist
        /angular-file-saver.bundle.js"></script>
     <script src="bower_components/angular-spinners/dist/
```

```
        angular-spinners.min.js"></script>

    <!-- Add New Bower Component JS Above -->

<!-- Main App JS -->
<script src="app.js"></script>
<script src="app/components/project/controllers/add/
   project-add.js"></script>
<script src="app/components/project/controllers/view/
   project-view.js"></script>
<script src="app/components/project/controllers/update/
   project-update.js"></script>
<script src="app/components/project/controllers/list/
   project-list.js"></script>
<script src="app/components/project/services/project-
   service.js"></script>
<script src="app/components/category/controllers/list/
   category-list.js"></script>
<script src="app/components/category/controllers/add/
   category-add.js"></script>
<script src="app/components/category/controllers/view/
   category-view.js"></script>
<script src="app/components/category/controllers/update/
   category-update.js"></script>
<script src="app/components/category/services/category-
   service.js"></script>
<script src="app/components/dataset/services/dataset-
   service.js"></script>
<script src="app/components/dataset/controllers/add/
   dataset-add.js"></script>
<script src="app/components/dataset/controllers/update/
   dataset-update.js"></script>
<script src="app/components/dataset/controllers/view/
   dataset-view.js"></script>
<script src="app/components/tweet/services/tweet-service.
   js"></script>
<script src="app/components/tweet/controllers/import/
```

```
    tweet-import.js"></script>
<script src="app/shared/directives/fileReader/fileReader.
    js"></script>
<script src="app/components/tweet/controllers/list/tweet-
    list.js"></script>
<script src="app/components/tweet/controllers/view/tweet-
    view.js"></script>
<script src="app/components/user/controllers/signup/user-
    signup.js"></script>
<script src="app/components/user/controllers/user-login/
    user-login.js"></script>
<script src="app/components/user/controllers/user-view/
    user-view.js"></script>
<script src="app/components/user/controllers/user-update/
    user-update.js"></script>
<script src="app/components/admin/controllers/view-users/
    admin-view-users.js"></script>
<script src="app/components/admin/services/admin-service.
    js"></script>
<script src="app/components/user/services/user-service.js
    "></script>
<script src="app/components/user/services/session-service
    /session-service.js"></script>
<script src="app/components/home/home.js"></script>
<script src="app/components/tweet/controllers/view-
    conversation/view-conversation.js"></script>
<script src="app/components/error/error.js"></script>
<script src="app/shared/modal-service/modal.js"></script>
<script src="app/components/filters/controllers/filters/
    filters.js"></script>
<script src="app/components/filters/services/filter-
    service/filter-service.js"></script>
<script src="app/components/questioning/controllers/
    questioning/questioning.js"></script>
<script src="app/components/questioning/services/
    questioning-service/questioning-service.js"></script>
<script src="app/shared/compareTo/compareTo.js"></script>
<script src="app/components/user/controllers/change-
    password/change-password.js"></script>
```

```
    <script src="app/components/user/controllers/update/
       update.js"></script>
    <script src="app/components/matrix/controllers/add/matrix
       -add.js"></script>
    <script src="app/components/matrix/controllers/view/
       matrix-view.js"></script>
    <script src="app/components/matrix/controllers/update/
       matrix-update.js"></script>
    <script src="app/components/matrix/controllers/list/
       matrix-list.js"></script>
    <script src="app/components/matrix/services/matrix-
       service.js"></script>
    <script src="app/components/project/services/storage-
       project/storage-project.js"></script>
    <script src="app/components/project/controllers/set/
       project-set.js"></script>
        <script src="app/components/index/index-controller.js
           "></script>
    <script src="app/components/dataset/controllers/list/
       dataset-list.js"></script>
    <script src="app/components/training-set/training-set.js"
       ></script>
    <!-- Add New Component JS Above -->

</body>

</html>


</body>
</html>

-->

{
  "generator-cg-angular": {
    "uirouter": false,
    "partialDirectory": "partial/",
    "modalDirectory": "partial/",
    "directiveDirectory": "directive/",
```

```
    "filterDirectory": "filter/",
    "serviceDirectory": "service/",
    "inject": {
      "js": {
        "file": "index.html",
        "marker": "<!-- Add New Component JS Above -->",
        "template": "<script src=\"<%= filename %>\"></script
           >"
      },
      "less": {
        "relativeToModule": true,
        "file": "<%= module %>.less",
        "marker": "/* Add Component LESS Above */",
        "template": "@import \"<%= filename %>\";"
      }
    }
  }
}
//DEPENDENCIAS GRUNT
{
  "name": "snaApp2",
  "version": "0.0.0",
  "devDependencies": {
    "grunt": "~0.4",
    "grunt-angular-templates": "~0.5",
    "grunt-browser-output": "0.1.0",
    "grunt-contrib-clean": "~0.5",
    "grunt-contrib-concat": "~0.3",
    "grunt-contrib-connect": "~0.6",
    "grunt-contrib-copy": "~0.5",
    "grunt-contrib-cssmin": "~0.7",
    "grunt-contrib-htmlmin": "~0.1",
    "grunt-contrib-jshint": "~0.9",
    "grunt-contrib-less": "~0.8",
    "grunt-contrib-uglify": "~0.2",
    "grunt-contrib-watch": "~0.6",
    "grunt-dom-munger": "~3.4",
    "grunt-karma": "~0.8.3",
    "grunt-ng-annotate": "~0.5",
```

```
    "grunt-wiredep": "^2.0.0",
    "karma": "~0.12.6",
    "karma-chrome-launcher": "~0.1.3",
    "karma-firefox-launcher": "~0.1.3",
    "karma-jasmine": "~0.1.5",
    "karma-mocha-reporter": "~0.2.5",
    "karma-phantomjs-launcher": "~0.1.4",
    "load-grunt-tasks": "~0.2"
  }
}
//DEPENDENCIAS BOWER
{
  "name": "snaapp2",
  "version": "0.0.0",
  "main": "index.html",
  "ignore": [
    "**/.*",
    "node_modules",
    "bower_components"
  ],
  "dependencies": {
    "jquery": "^2.2.2",
    "underscore": "~1.5",
    "bootstrap": "^3.3.6",
    "angular": "^1.5.3",
    "angular-route": "~1.2",
    "angular-animate": "^1.5.3",
    "angular-resource": "~1.2",
    "angular-cookies": "^1.5.3",
    "angular-mocks": "~1.2",
    "angular-ui-utils": "~0.1",
    "angular-bootstrap": "^1.2.5",
    "moment": "~2.5",
    "less.js": "~1.6",
    "font-awesome": "^4.6.3",
    "angular-dragdrop": "^1.0.13",
    "jquery-ui": "^1.11.4",
    "angular-touch": "^1.5.3",
    "ng-csv": "^0.3.6",
```

```
    "satellizer": "^0.14.0",
    "ngstorage": "^0.3.10",
    "angular-messages": "ng-messages#*",
    "checklist-model": "^0.10.0",
    "angular-file-saver": "^1.1.2",
    "angular-spinner": "^0.8.1"
  }
}
```