

UNIVERSIDADE FEDERAL DE SANTA CATARINA

INFRAESTRUTURA DE DADOS ABERTOS COM SUPORTE AO
LINKED DATA

FILIPE NUERNBERG SCHROEDER

Florianópolis - SC

2013 / 2

UNIVERSIDADE FEDERAL DE SANTA CATARINA CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

INFRAESTRUTURA DE DADOS ABERTOS COM SUPORTE AO LINKED

DATA

FILIPPE NUERNBERG SCHROEDER

Trabalho de conclusão de curso
apresentado como parte das atividades
para obtenção do título de Bacharel, do
curso de Sistemas de Informação da
Universidade Federal de Santa Catarina

Florianópolis - SC

2013 / 2

FILIPPE NUERNBERG SCHROEDER

INFRAESTRUTURA DE DADOS ABERTOS COM SUPORTE AO LINKED
DATA

Trabalho de conclusão de curso apresentado como parte dos requisitos para
obtenção do grau de Bacharel em Sistemas de Informação

Orientador: Prof. Dr. Fernando Ostuni Gauthier

Coorientador: Prof. Dr. José Leomar Todesco

Banca examinadora

Rafael de Moura Speroni

AGRADECIMENTOS

Aos familiares, pelo apoio incondicional e imprescindível para o meu desenvolvimento e estudos ao longo da vida.

Aos amigos e colegas, pelo incentivo e compreensão das consequências visíveis deste trabalho na convivência social.

Aos professores da universidade em geral e orientadores, pelo auxílio e principalmente paciência frente às dificuldades enfrentadas na conciliação de trabalho em tempo integral com os estudos.

SUMÁRIO

Lista de Figuras.....	7
Lista de Quadros.....	8
Lista de Tabelas.....	9
Glossário.....	10
Resumo.....	12
1 Introdução.....	13
1.1 Apresentação.....	13
1.2 Objetivos.....	14
1.3 Justificativa.....	14
1.4 Limitações do trabalho.....	15
1.5 Estrutura do trabalho.....	15
2 Referencial teórico.....	16
2.1 Introdução.....	16
2.2 Dados Abertos.....	16
2.3 Web Semântica.....	18
2.4 Linked Data.....	19
2.5 URI.....	21
2.6 RDF.....	23
2.7 SPARQL.....	26
3 Publicação de Linked Data.....	29
3.1 Introdução.....	29
3.2 Design e forma dos uris.....	29
3.3 Mecanismos de dereferenciação.....	32
3.4 Formatos de serialização de RDF.....	35
3.5 Vocabulários e Ontologias.....	39
3.6 Metadados.....	40
3.7 Resultado das Consultas SPARQL.....	43
3.8 Estatísticas do ambiente.....	43
3.9 Fluxo e Estrutura de publicação de linked data.....	44
3.10 Fluxo de publicação de dados.....	48

3.11	Arquitetura do ambiente.....	49
3.12	Elementos de operação de infraestrutura	51
4	Aplicação prática no laboratório de Engenharia do Conhecimento (EGC).....	56
4.1	Introdução.....	56
4.2	Sobre o EGC	56
4.3	Definições para a implantação.....	57
4.4	Fluxo de publicação de dados no ambiente EGC.....	59
4.5	Convenções de URIs Aplicadas	60
4.6	Arquitetura do ambiente.....	60
4.7	Implantação	62
4.8	Customizações da Interface	73
4.9	Otimizações e ajustes finais	76
5	Conclusão	78
5.1	Considerações finais.....	78
5.2	Sugestões para futuros trabalhos	78
6	Referências Bibliográficas	80
7	Anexos e Apêndices.....	82
	Anexo 1 - Instalação e configuração das ferramentas no ambiente EGC .	82
	Anexo 2 - Arquivos alterados para a customização visual EGC	94
	Anexo 3 - Artigo.....	103

LISTA DE FIGURAS

Figura 1 - Diagrama de distribuição de triplas por domínio em 2011

Figura 2 - Fluxo de acesso ao conteúdo

Figura 3 - Modelo de grafos visual RDF

Figura 4 - Fluxo de navegação por redirecionamento 303

Figura 5 - Fluxo de publicação de Linked Data

Figura 6 - Fluxo de publicação dados genérico

Figura 7 - Arquitetura padrão sugerida

Figura 8 - Esquema de pontos decisórios discutidos com membros do EGC

Figura 9 - Arquitetura aplicada para o ambiente EGC

Figura 10 - Interface de carregamento de triplas RDF no servidor Virtuoso

Figura 11 - Resposta gerada pelo Pubby para descrição de recurso em HTML

Figura 12 - Resposta HTML gerada pelo Pubby para descrição de recurso.

LISTA DE QUADROS

Quadro 1 - Modelo de afirmações RDF

Quadro 2 - Modelo de grafos RDF

Quadro 3 - Representação RDF de nodo em branco

Quadro 4 - Exemplo de formato de serialização RDF/XML

Quadro 5 - Exemplo de trecho de documento HTML anotado com RDFa

Quadro 6 - Exemplo de trecho em formato Turtle

Quadro 7 - Exemplo de trecho em formato N-Triplas

Quadro 8 - Exemplo de uso de licença Creative Commons CC-BY-SA

Quadro 9 - Esquema de pontos decisórios para a implantação de Linked Data

Quadro 10 - Esquema de pontos decisórios discutidos com membros do EGC

Quadro 11 - Exemplo de triplas RDF de descrição da entidade DBPedia

LISTA DE TABELAS

Tabela 1 - Representação de URIs por contexto

Tabela 2 - Representação de URIs por subdomínio

Tabela 3 - Representação de URIs por extensão de arquivo

Tabela 4 - Representação de URIs por contexto para o EGC

GLOSSÁRIO

Linked Data – Em português, dados ligados, consiste num método de publicação de dados estruturados para que possam ser interligados e, com isso, ser de maior utilidade, uma vez que os computadores conseguem auxiliar na leitura e interpretação do sentido dos dados.

Web Semântica – Proposta de extensão da web atual em que seres humanos e computadores possam trabalhar em cooperação, atribuindo significado para os dados de maneira que os computadores também sejam capazes de realizar inferências.

EGC – Engenharia do Conhecimento da Universidade Federal de Santa Catarina.

RDF – Do inglês Resource Description Framework, um modelo de dados para Linked Data.

XML - Do inglês, Extensible Markup Language, um simples, porém flexível formato de texto derivado do padrão SGML (ISO 8879). Padrão com amplo uso tanto na publicação de grandes quantidades de dados, como em troca de informações via diferentes ambientes computacionais.

Dataset – Conjunto de dados em formato Linked Data, usualmente agrupado em um campo de domínio (assunto).

URI – Do inglês, Universal Resource Identifier, é um conjunto de caracteres usado para identificar um recurso ou nome.

HTTP – Do inglês, Hypertext Transfer Protocol, é um protocolo de comunicação utilizado por sistemas de informação hipermedia distribuídos e colaborativos.

Serializar – A serialização é o processo de salvar um objeto em um meio de armazenamento (como um arquivo de computador ou um buffer de memória) ou transmiti-lo por uma conexão de rede, seja em forma binária ou em formato de texto como o XML.

Dereferenciar¹ - o ato de obter uma representação de uma descrição de uma entidade, através do seu URI.

¹ Deriva do termo *dereference* no inglês original. Não há uma definição formal do termo em língua portuguesa, apesar de ser utilizado no meio técnico há muitos anos. Dessa forma, é possível encontrar mais de uma forma de grafia em uso, sendo **derreferenciar** e **dereferenciar** as formas mais comuns.

RESUMO

O objetivo de compartilhar dados abertos é uma tendência mundial nas esferas governamentais, como exemplo do governo brasileiro através do portal *dados.gov.br*, e inclusive na iniciativa privada. O uso de Linked Data proposto pelo criador da Web, Tim Berners Lee, é uma alternativa baseada em experiências com web semântica e funciona como extensão à web tradicional. Com o uso de Linked data e o framework RDF é possível acessar os dados da web como uma enorme base global de dados. Este trabalho tem como desafio a definição de estratégias e práticas para a infraestrutura de software necessária em um ambiente de dados abertos com suporte ao Linked Data em um contexto geral e posteriormente utilizar tal informação em exemplo prático para o laboratório de Engenharia do Conhecimento da Universidade Federal de Santa Catarina.

Palavras-chave: linked data, infraestrutura, dados abertos, egc, rdf, dados lincados, web-semântica

1 INTRODUÇÃO

1.1 Apresentação

A disponibilização de dados abertos com uso das práticas do Linked Data (Dados Ligados, em tradução livre para o português) tem crescido notavelmente nos últimos anos, de acordo com mapeamentos feitos pelo projeto *Linking Open Data*². Através da prática do Linked Data, os dados interligados na web podem ser vistos como uma grande base mundial de dados, interpretados por máquinas e reutilizados de maneira a maximizar o potencial desse conjunto.

A capacidade de, através de uma consulta a esse ambiente, identificar dados de diferentes fontes e agregá-los de maneira dinâmica e possivelmente automatizada é um aspecto popular desse paradigma.

Inicialmente, o objetivo principal do projeto linked data era disponibilizar grandes quantidades de dados e identificar as melhores práticas para fazê-lo. Atualmente, há também o foco em outros fatores como a qualidade dos dados, usabilidade, confiabilidade da infraestrutura, desempenho, entre outros. Nesse contexto, o planejamento e uso apropriado da infraestrutura tornam-se importantes fatores na confiabilidade, desempenho e principalmente na futura viabilidade deste modelo em larga escala e crescente complexidade.

² <http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

1.2 Objetivos

Objetivo Geral

Avaliar os elementos necessários para a infraestrutura de dados abertos com suporte ao Linked Data e definir elementos e melhores práticas para este processo.

Objetivos Específicos

- Fundamentar a teoria base de Linked Data e dados abertos
- Avaliar o estado da arte na infraestrutura de ambientes de dados abertos ligados e identificar os elementos chave envolvidos na sua implantação e melhores práticas
- Aplicar os elementos-chave no contexto do laboratório de Engenharia do Conhecimento da UFSC
- Propor uma configuração de ferramentas e técnicas para implantação de uma infraestrutura de suporte a Dados Abertos Ligados

1.3 Justificativa

O movimento mundial de uso dos fundamentos do Linked Data para o engrandecimento semântico da web é recente e, apesar de apresentar um crescente número publicações e coleções importantes de dados como: Wikipedia, CiteSeer, SwissProt, Geonames, hoje disponíveis em RDF e consultáveis via Pontos de acesso SPARQL, ainda existem muitos desafios de pesquisa e espaço para crescimento. Como a análise da disponibilidade de ferramentas, e necessidade de definição de uma configuração das mesmas.

O departamento de Engenharia do Conhecimento da Universidade Federal de Santa Catarina através do Prof. Dr. Fernando Ostuni Gauthier, Prof. Dr. José Leomar Todesco e equipe, vêm desenvolvendo pesquisas nessa área. O Presente trabalho pretende contribuir na área de infraestrutura deste projeto, baseando-se nos estudos deste grupo e estado da arte na área.

1.4 Limitações do trabalho

O processo de alimentação dos dados abertos nos processos de coleta, tratamento e carregamento dos dados nos servidores não fará parte deste trabalho.

O presente trabalho não tem como foco a avaliação exaustiva dos softwares disponíveis para a infraestrutura de Linked Data.

1.5 Estrutura do trabalho

O texto se subdivide em seis capítulos:

No primeiro capítulo, tem-se uma introdução ao tema do trabalho, suas limitações e objetivos;

O segundo capítulo aborda conceitos-base de dados abertos e de linked data, bem como uma visão geral do seu processo de publicação;

No terceiro capítulo, são identificados os elementos que fazem parte da infraestrutura de dados abertos ligados;

O quarto capítulo contempla um caso prático dos elementos identificados nos capítulos anteriores;

Como encerramento do trabalho, temos as conclusões e considerações sobre o trabalho e sugestão de trabalhos futuros;

2 REFERENCIAL TEÓRICO

2.1 Introdução

Para que se faça uso de dados abertos com princípios do Linked Data, é preciso entender os fundamentos básicos, para então direcionar-se aos elementos necessários para o planejamento e construção de tal ambiente. Neste capítulo serão abordados conceitos e termos que servirão de ponto de partida para a aplicação.

2.2 Dados Abertos

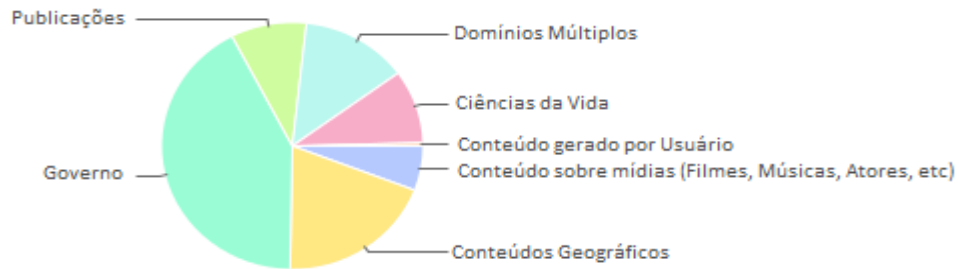
De acordo com a *Open Knowledge Foundation*³ (tradução disponível em <http://dados.gov.br/dados-abertos>): “dados são abertos quando qualquer pessoa pode livremente usá-los, reutilizá-los e redistribuí-los, estando sujeitos a, no máximo, a exigência de creditar a sua autoria e compartilhar pela mesma licença”.

Segundo (Lee & Galway 2011), há uma ênfase no uso de dados abertos por autoridades públicas, denominados de dados abertos governamentais, que se referem aos dados produzidos ou encomendados pelo governo. Ainda de acordo com Linking Open Data Cloud⁴, a área governamental é responsável pelo maior número de dados em linked data no formato de triplas atualmente. Como pode ser visto na figura 1:

³ Organização sem fins lucrativos que promove dados, conteúdos e conhecimento abertos, sua página na web encontra-se em: <http://opendefinition.org/>

⁴ Projeto de mapeamento de conjuntos de dados publicados na web em forma de Linked Data (datasets), acessível em: <http://www4.wiwiw.fu-berlin.de/lodcloud/state/>

Figura 1 - Diagrama de distribuição de triplas por domínio em 2011.



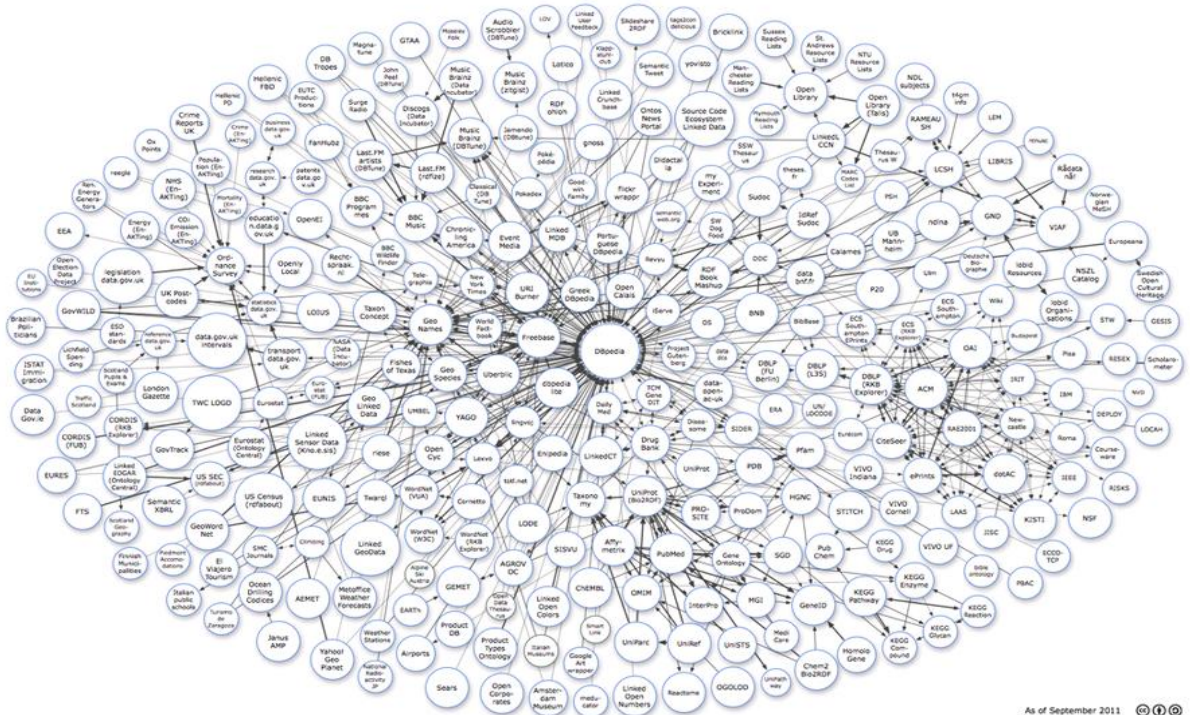
Fonte: <http://www4.wiwiw.fu-berlin.de/lodcloud/state>

Na figura 1, é possível perceber que os dados relacionados à área governamental representam quase a metade dos dados, o que sustenta o fato de que os governos estão investindo nesta prática e, portanto são valiosas fontes de dados.

O Projeto *Linking Open Data Cloud*⁵, inicialmente provia fotos da evolução dos conjuntos de dados no decorrer do tempo, com sua versão mais recente na figura 2:

Figura 2 - Os conjuntos de dados com suporte ao Linked Data (extraído em 19/09/2011).

⁵ <http://lod-cloud.net>



As of September 2011 ©

Fonte: Lod Cloud⁶

Mais recentemente, o projeto fornece um *website*⁷ com ferramentas de pesquisa, para melhor identificação e listagem de conjuntos de dados cadastrados, atualmente com um total de 8.912 conjuntos de dados.

2.3 Web Semântica

Tim Berners Lee, o inventor da World Wide Web foi quem conceituou o termo Web Semântica globalmente, e o definiu como: "(...) uma web de dados que podem ser processados direta e indiretamente por máquinas."

O termo *web de dados* definido anteriormente pode ser colocado em contraste com o termo *web de documentos* ou ainda *web tradicional* ou *web clássica*, que representa a web no seu primeiro momento, onde os dados não

⁶ Disponível em: <http://lod-cloud.net>

⁷ Disponível em: <http://datahub.io/dataset>

eram estruturados e nem projetados para o uso das máquinas na inferência de significado, tal papel caberia aos usuários humanos. Mais especificamente (Bizer, Heath, et al. 2009), definem:

Apesar dos benefícios indiscutíveis que a Web oferece, até recentemente os mesmos princípios que permitiram a Web de documentos florescer não foram aplicados aos dados. Tradicionalmente, os dados publicados na internet foram disponibilizados como grandes blocos de dados em formatos como CSV ou XML, ou marcados como tabelas HTML, sacrificando muito de sua estrutura e semântica. Na convencional Web hipertexto, a natureza da relação entre dois documentos ligados está implícita, como o formato dos dados, ou seja, HTML, não é suficientemente expressivo para permitir que entidades individuais descritas num documento particular sejam ligadas por ligações classificadas, às entidades afins.

Ainda, de forma abrangente, a organização W3C⁸ define a Web Semântica da seguinte maneira:

O termo “Web Semântica” refere-se à visão do W3C da Web dos Dados Ligados. A Web Semântica dá às pessoas a capacidade de criarem repositórios de dados na Web, construir vocabulários e escreverem regras para interoperarem com esses dados.

Logo, a Web Semântica pode ser aplicada através da prática de *Dados Ligados*, ou *Linked Data*, conceito que será definido na seção seguinte.

2.4 Linked Data

Para que a visão de Web Semântica seja realizada, a prática do Linked Data deve ser utilizada, como de acordo com a W3C:

(...)para fazer a Web de Dados uma realidade, é importante ter uma grande quantidade de dados na web disponível e um formato padronizado, alcançável e gerenciável por ferramentas da Web Semântica. Além disso, a Web Semântica precisa não só do acesso aos dados, mas também aos relacionamentos entre os dados (ao contrário de uma grande coleção de

⁸ Consórcio da World Wide Web, uma comunidade internacional onde organizações, pessoas e público em geral trabalham juntos para definir padrões para a web, seu sítio em português pode ser acessado em: <http://www.w3c.br>

datasets). Essa coleção de datasets inter-relacionados na web pode também ser chamada de Linked Data.

A idéia do Linked Data é utilizar a arquitetura já existente na web clássica, ou World Wide Web, na tarefa de compartilhar dados estruturados em escala global. Para entender tais princípios é necessário entender a arquitetura da web clássica de documentos.

A *web de documentos* é constituída de um pequeno conjunto de padrões: URIs (do inglês, *Uniform Resource Identifier*), como mecanismo de identificação único globalmente, o HTTP (do inglês, *Hypertext Markup Language*), como formato de conteúdo global. Além disso, a web é construída com a idéia de colocar hiperlinks entre documentos que podem residir em servidores web diferentes (Bizer & Heath 2011).

O Linked Data é construído diretamente na arquitetura da web clássica, e a aplica na tarefa de compartilhar dados em escopo global.

Para que os dados estejam no formato Linked Data, Berners-Lee (2006) definiu as seguintes regras:

1. Usar URIs como nomes para as coisas.
2. Usar HTTP URIs, assim pessoas podem consultar esses nomes.
3. Quando alguém consultar um URI, prover informações úteis, usando padrões (RDF, SPARQL).
4. Incluir links para outras URIs, então poderá se descobrir mais coisas.

As seções subseqüentes definirão os conceitos presentes nos princípios acima.

2.5 URI

Um URI (do inglês, *Uniform Resource Identifier*) é um conjunto de caracteres compacto, utilizado para identificar um recurso físico ou abstrato⁹.

Seu propósito é identificar itens na rede mundial de computadores (internet), de forma que possam ser acessados através de protocolos específicos, como o protocolo HTTP, como definida na segunda regra de Linked Data.

O padrão de um URI, de acordo com sua especificação formal, pode ser construído da seguinte forma:

```
<nome do esquema>://<parte hierárquica>[?<consulta>][#  
<fragmento>]
```

Onde o nome do esquema corresponde ao tipo de protocolo utilizado na comunicação, no caso HTTP para URIs HTTP, outros exemplos são FTP, HTTPS, WEBDAV, entre muitos outros.

A parte hierárquica compreende o nome do servidor cadastrado, também conhecido como domínio de internet e caminho a ser endereçado.

As partes de consulta e fragmento são variáveis específicas que visam recuperar parte do documento representado no endereço especificado à sua esquerda.

Como exemplo de URI HTTP, temos:

⁹ Definição disponível em: <http://www.ietf.org/rfc/rfc2396.txt>

`http://exemplo.org/dados/pessoas#filipe`

Onde os caracteres *http* representam o nome do esquema, *exemplo.org/dados/pessoas* a parte hierárquica, e *#filipe* o fragmento. As partes de consulta e fragmento não são obrigatórias.

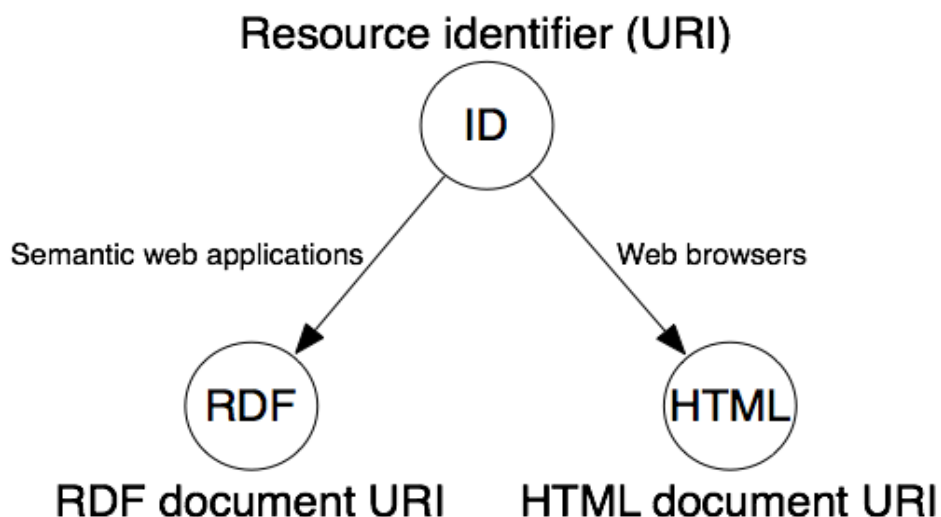
As entidades são identificadas por URIs que usam o esquema `http://`, essas entidades podem ser obtidas através da simples dereferenciação do URI sobre o protocolo HTTP. Dessa forma, o protocolo HTTP prove um mecanismo simples, porém universal para acessar recursos que podem ser serializados como um conjunto de bytes (como uma fotografia de um cachorro), ou acessar descrições de entidades que não podem ser enviadas pela rede, desta maneira (como um cachorro em si) (Bizer, Heath, et al. 2009).

Na Web Semântica, os URIs representam não apenas documentos, mas também objetos do mundo real como uma árvore ou pessoa e até conceitos abstratos, tais representações são chamadas *coisas* ou *objetos do mundo real*.

Portanto, quando do acesso a um URI, máquinas devem receber em formato RDF e as pessoas, uma representação legível, como HTML, através do protocolo de transferência padrão da web, o HTTP.

A figura 2 mostra as diferentes representações de um URI, no contexto de Linked Data.

Figura 2 - Fluxo de acesso ao conteúdo.



Fonte: <http://www.w3.org/TR/cooluris>

2.6 RDF

O RDF (do inglês, Resource Description Framework) é um modelo de dados utilizado para facilitar a troca de informações na web e faz parte de um dos padrões de representação de dados para o Linked Data, análogo ao papel do HTML na *web clássica*.

Dada a característica do Linked Data de ser um mecanismo de disponibilização de dados integrados entre si e entre outras fontes de dados, o modelo utilizado para este fim deve permitir simplificada integração e representação dos dados. Estas características podem ser observadas no modelo RDF, descrito a seguir.

O modelo RDF

De acordo com (Cyganiak & Wood 2013), os dados em RDF são representados por triplas ou afirmações, na forma: *sujeito, predicado, objeto*. Esta representação faz alusão à formação de frases, conforme definido no quadro 1:

Quadro 1 - Modelo de afirmações RDF.

José	nasceu em	Florianópolis
Sujeito	Predicado	Objeto

Fonte: Autor

Tal sequência de elementos também é denominada grafo RDF, por poder ser expressada na forma de grafo, com a relação entre sujeito, predicado e objeto, como no exemplo de duas triplas abaixo:

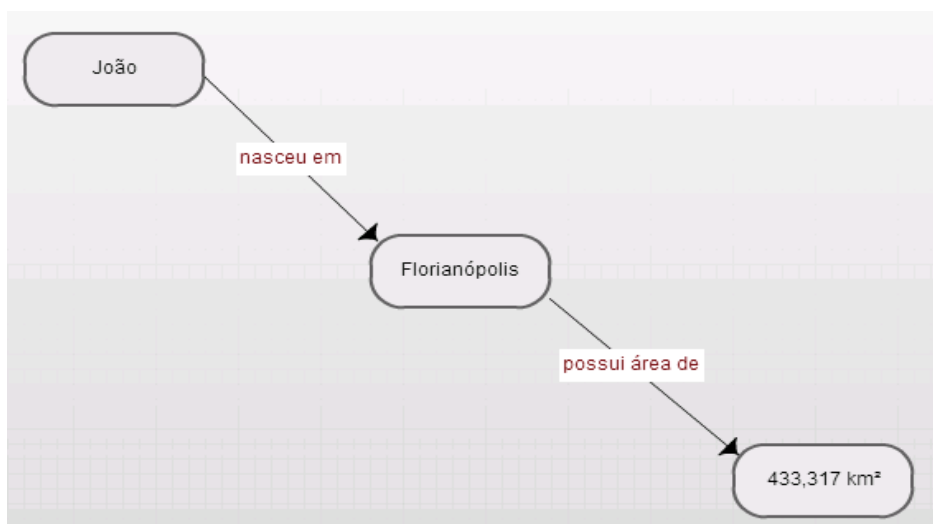
Quadro 2 - Modelo de grafos RDF.

João	-> nasceu em	-> Florianópolis
Florianópolis	-> possui área de	-> 433,317 km ²

Fonte: Autor

O exemplo pode ser visto de maneira gráfica na figura 3:

Figura 3 - Modelo de grafos visual RDF.



Fonte: Autor

Onde os vértices podem ser sujeitos ou objetos, também chamados de nodos, e as arestas são os predicados. Portanto a terminologia nodo RDF, refere-se ao conjunto de sujeitos e predicados presentes no trecho RDF. Note que o valor da área no exemplo é um tipo de valor chamado literal, ao passo que *João* e *Florianópolis* são tratados como entidades, identificáveis por URIs.

Nodo em branco

Em RDF pode ser necessário descrever um mesmo sujeito ou objeto múltiplas vezes em um único arquivo. Para tal, existe a abordagem de uso de um nodo em branco (também denominado *bnode*).

De acordo com (Klyne et al. 2004), um nodo em branco é um nodo que não é uma referencia URI ou literal. Na sintaxe abstrata RDF, um nodo em branco é um nodo único que não pode ser utilizado em um ou mais afirmativas (ou triplas) RDF, porém sem nome intrínseco.

O trecho do quadro 3 exemplifica um uso possível de nodo em branco:

Quadro 3 - Representação RDF de nodo em branco.

João	conhece	_:p1
_:p1	nasceu em	04/11/1984

Fonte: Autor

No caso acima, é possível representar os dados em triplas, mesmo quando não há informação completa sobre todos os sujeitos e objetos, no exemplo acima é possível inferir que uma determinada pessoa, que é conhecida de João também nasceu em 04/11/1984.

É imprescindível diferenciar o *modelo RDF* dos *formatos de representação* ou *serialização de RDF*, o primeiro é de fato o que a especificação e termo RDF se propõe, uma especificação de modelo de representação, o segundo se materializa no uso que se faz do modelo RDF na prática, através de formatos como o RDF/XML, N-Triplas, entre outros, chamados de formatos de serialização de RDF.

Para a publicação na web, dados em RDF podem ser serializados em diferentes formatos, os dois mais comuns são o RDF/XML e RDFa (Bizer & Heath 2011).

2.7 SPARQL

Para que seja possível que se obtenha de maneira automatizada, a semântica presente nos dados através do modelo RDF, uma linguagem de consulta foi criada, chamada SPARQL (do inglês, *SPARQL Protocol and RDF Query Language*). Tal linguagem também possui definição de protocolo e consta como recomendação pela instituição W3C¹⁰.

Para que estas funcionalidades fiquem disponíveis na web através do mecanismo de consulta SPARQL, é necessário que se tenha um ponto de acesso SPARQL (do inglês, *SPARQL Endpoint*), disponível.

De acordo com (DuCharme 2013, tradução livre):

O ponto de acesso SPARQL mais popular é a DBpedia¹¹, uma coleção de dados das caixas cinzas de informação de que é frequentemente vista no lado direito das páginas do site Wikipedia. Como muitos dos pontos de acesso SPARQL, a DBpedia inclui um formulário web onde é possível

¹⁰ <http://www.w3.org/2005/10/Process-20051014/tr.html#RecsW3C>

¹¹ <http://dbpedia.org>

submeter uma consulta e explorar os resultados, tornando muito fácil a exploração dos seus dados.(...)

Dentre as funcionalidades do SPARQL, temos a capacidade de consultar padrões de grafos opcionais e obrigatórios e suas conjunções e disjunções, além de prover testes de valores e limitações de consultas por grafo RDF fonte, os resultados de consultas SPARQL podem ser conjuntos de grafos RDF¹².

De acordo com sua especificação definida por (Harris et al. 2010), a linguagem SPARQL suporta quatro formas de operação: SELECT, CONSTRUCT, ASK e DESCRIBE, com os seguintes propósitos:

- SELECT - Retorna todos, ou um subconjunto, de todas as variáveis que correspondem à uma correspondência de resultado de uma consulta
- CONSTRUCT - Retorna um grafo construído através da substituição de variáveis em um conjunto de padrões de triplas
- ASK - Retorna um valor booleano indicando se uma consulta é correspondida ou não
- DESCRIBE - Retorna um grafo RDF que descreve os recursos encontrados, esse dado retornado não é determinado pela consulta SPARQL, onde o cliente precisaria conhecer a estrutura do RDF na fonte de dados, mas ao invés disso, é determinado pelo processador de consultas SPARQL (servidor). Um dos mecanismos possíveis para a determinação do conjunto

¹² <http://www.w3.org/TR/rdf-sparql-query/>

de informações a ser retornada para uma consulta DESCRIBE, é chamada de CBD (do inglês, *Concise Bounded Description*).

A especificação do mecanismo CBD proposta por (Stickler 2005), define as seguintes características sobre os dados a serem retornados em uma consulta:

Dado um nó inicial em um determinado grafo RDF (grafo inicial), um subgrafo deste grafo inicial, feita para compreender uma descrição delimitada concisa de recurso denotado pelo nó de partida, pode ser identificada como se segue:

1. Que inclua no subgrafo resultante todas as afirmações (triplas) no grafo inicial, onde o sujeito da afirmação é o nodo inicial;
2. Que recursivamente, para todas as afirmações identificadas no subgrafo do passo anterior, tendo um objeto como nodo em branco, que inclua no resultado todas as afirmações do grafo inicial onde o sujeito da afirmação é o nodo em branco em questão e que não tenha sido incluído nos resultados previamente.
3. Que recursivamente, para todas as afirmações identificadas no subgrafo nos passos anteriores, para todas as reificações de cada afirmação no grafo inicial, inclua a descrição delimitada concisa de recurso iniciando do nodo `rdf:Statement` node de cada reificação.

3 PUBLICAÇÃO DE LINKED DATA

3.1 Introdução

O presente capítulo tem como objetivo a definição dos processos e decisões na publicação de dados com suporte ao Linked Data.

O objetivo do Linked Data é facilitar o compartilhamento de dados estruturados pelas pessoas e gerados por computadores, como hoje é feito com os documentos, basicamente publicando os dados no modelo RDF e utilizando links RDF para interligar os dados de diferentes fontes. Para tal objetivo, é necessário definir como será a disponibilização do ambiente, incluindo formatos, padrões, tipos de softwares, entre outros fatores definidos por (Bizer et al. 2007; Bizer & Heath 2011).

As seções subseqüentes visam detalhar as questões associadas ao fluxo de publicação de de dados com suporte ao Linked Data.

3.2 Design e forma dos URIs

Um elemento importante antes mesmo da disponibilização dos dados é a definição do formato dos seus identificadores globais, os URIs. De acordo com (Bizer et al. 2007; Bizer & Heath 2011), os padrões comuns de URIs são os detalhados a seguir.

Separação via contexto

É o padrão utilizado pela DBpedia¹³, as URIs são separadas pelo contexto e tipo de dado retornado, no exemplo, uma representação do país Alemanha:

Tabela 1 - Representação de URIs por contexto.

URI	Dado retornado
http://exemplo.org/resource/alemanha	Conceito do mundo real
http://exemplo.org/data/alemanha	Representação para computadores, da coisa
http://exemplo.org/page/alemanha	Representação para seres humanos, da coisa

Fonte: Autor

Tal representação pode causar confusão ao desenvolvedor inexperiente ao Linked Data, já que as URIs com as representações são pouco diferentes da URI do conceito do mundo real (Bizer et al. 2007; Bizer & Heath 2011).

Separação via subdomínio

Neste padrão, o uso de subdomínios pode ajudar na separação dos servidores responsáveis para cada um dos tipos de representação. (exemplo: servidor web para as páginas HTML e id e base de dados RDF para os dados em RDF).

O mesmo exemplo de representação do país Alemanha, neste modo de separação, pode ser visto na tabela 2:

¹³ A DBpedia é um projeto que visa extrair informação estruturada do site <http://wikipedia.org> e disponibilizá-lo na forma de Linked Data.

Tabela 2 - Representação de URIs por subdomínio.

URI	Dado retornado
http://resource.exemplo.org/alemanha	Conceito do mundo real
http://data.exemplo.org/alemanha	Representação para computadores, da coisa
http://page.exemplo.org/alemanha	Representação para seres humanos, da coisa

Fonte: Autor

Separação via extensão de arquivo

Neste método, temos a extensão do arquivo determinando a URI, onde é possível saber, visualmente, o tipo de representação recebido. Na tabela 3 temos o mesmo exemplo das representações anteriores, com a separação via extensão de arquivo:

Tabela 3 - Representação de URIs por extensão de arquivo.

URI	Dado retornado
http://exemplo.org/alemanha	Conceito do mundo real
http://exemplo.org/alemanha.rdf	Representação para computadores, da coisa (RDF)
http://exemplo.org/alemanha.html	Representação para seres humanos, da coisa (HTML)

Fonte: Autor

As três abordagens descritas na tabela assumem que há separação clara entre o dado na representação RDF e HTML (como no uso de RDF/XML e HTML). Tal diferenciação não é observada quando o método de serialização de RDF é o RDFa, que é construído embutido na mesma entrada da

representação HTML, sendo assim necessário apenas um URI para as duas representações (RDF e HTML).

3.3 Mecanismos de dereferenciação

Para que um pedido por uma descrição de um recurso seja devidamente respondido, seja com RDF para máquinas ou formatos como o HTML para pessoas, uma estratégia de dereferenciação deve ser definida.

Conforme (Sauermann & Cyganiak 2008), as seguintes estratégias podem ser utilizadas, quando o dado RDF e HTML são disponibilizados separadamente:

URIs Hash

A estratégia de Hash é construída na possibilidade de que as URIs tem de possuir um fragmento especial, separado da base da URI por um símbolo hash (#). Esta parte especial é chamada de identificador de fragmento, como exemplificado na seção de URIs do capítulo 2 deste documento.

Quando um cliente deseja acessar um URI com Hash, o protocolo HTTP exige que a parte do fragmento Hash seja retirada antes da solicitação do URI ao servidor e, portanto não poderá ser acessada diretamente. O cliente acessará a URI sem o fragmento e fará o processamento necessário para dirigir-se ao fragmento especificado, sem intervenção ou processamento do servidor.

Dessa maneira, os URIs sem o fragmento podem ser utilizados para especificar os objetos do mundo real e conceitos abstratos, e os fragmentos

dentro do documento identificam as representações destes, seja em RDF ou HTML.

Um fator vantajoso para este método é a necessidade de apenas um único pedido HTTP para acessar o recurso desejado, portanto, a latência de rede é reduzida. No entanto, o pedido traz toda a informação disponível para a URI de um tipo de objeto do mundo real, podendo ser oneroso, uma vez que o objetivo pode ser buscar um fragmento do todo apenas, e o conjunto dos dados recuperados pode representar uma grande quantidade de dados.

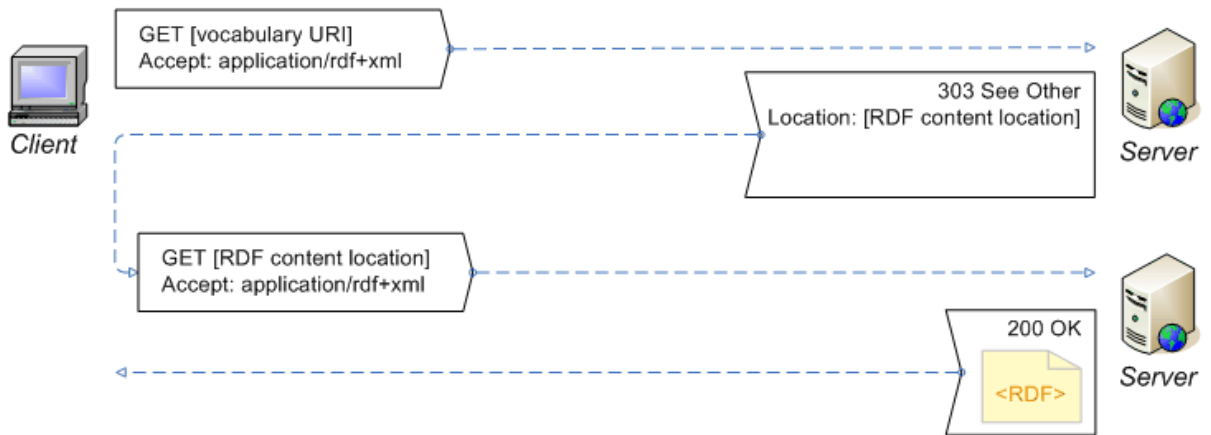
URIs 303

Nos URIs 303, o servidor responde ao cliente que acessa o objeto do mundo real com um código de resposta *303 See Other*, com o URI do documento web que descreve o objeto. Este processo é chamado de redirecionamento 303. Em um segundo passo, o cliente dereferencia esse novo URI e recebe o documento web descrevendo o objeto (Bizer & Heath 2011).

O tipo de representação, seja RDF ou HTML, será determinado conforme as preferências do cliente. Por exemplo, quando um navegador acessa um URI, explicita o formato dos dados que espera recebe, como: *application/rdf+HTML* ou *text/HTML*, nesse momento o servidor saberá direcionar o cliente para a representação desejada, no segundo passo descrito anteriormente.

Portanto, são realizados dois pedidos ao servidor, um para o objeto do mundo real, e outro para a descrição apropriada deste objeto, como exemplificado na figura 4:

Figura 4 - Fluxo de navegação por redirecionamento 303.



Fonte: (Bizer et al. 2007)

A vantagem principal do método 303 é o acesso apenas ao dado abstrato e em seguida ao dado pretendido, de maneira granular, não sendo necessária a transmissão de outros dados além dos solicitados. No entanto, são necessários dois pedidos HTTP do lado do cliente, um para o objeto do mundo real e outro para a representação desejada, aumentando a latência de rede.

É possível o uso híbrido de 303 e Hash, porém com impacto no desempenho e carga no servidor (Sauermann & Cyganiak 2008).

Em aplicações reais, é mais comum o uso de 303 em grandes conjuntos de dados (data sets), enquanto para pequenos vocabulários RDF e Ontologias OWL, é recomendável o uso de Hash (Sauermann & Cyganiak 2008; Bizer & Heath 2011).

3.4 Formatos de serialização de RDF

Em Ciência da Computação, no contexto de armazenamento e transmissão de dados, a serialização é o processo de salvar um objeto em um meio de armazenamento ou transmiti-lo por uma conexão de rede, seja em forma binária ou em formato de texto como o XML. Esta série de bytes pode ser usada para recriar um objeto com o mesmo estado interno que o original (Wikipedia n.d.).

O modelo de dados RDF não define por si só um formato de publicação e representação de dados, especifica um modelo de descrição de recursos através de triplas: *sujeito, predicado, objeto*. Portanto, para publicar Linked Data na web, é necessário definir um formato para a serialização dos dados em RDF.

De acordo com a instituição W3C, os formatos RDF/XML e RDFa já são considerados padrões (Bizer & Heath 2011), outros formatos detalhados a seguir também possuem status de formato recomendado pela mesma instituição.

RDF/XML

Utiliza o popular padrão de dados XML para comunicação e é amplamente utilizado e recomendado pela organização W3C como meio de serialização RDF (Manola & Miller 2004).

Apesar de ser o formato padrão recomendado, não é considerado o de maior facilidade de interpretação por seres humanos, portanto não recomendado onde seja necessária intervenção humana.

Um trecho de RDF/XML pode ser visto no quadro 4:

Quadro 4 - Exemplo de formato de serialização RDF/XML.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  <rdf:Description
rdf:about="http://en.wikipedia.org/wiki/Oxford">
  <dc:title>Oxford</dc:title>
  <dc:coverage>Oxfordshire</dc:coverage>
  <dc:publisher>Wikipedia</dc:publisher>
</rdf:Description>
</rdf:RDF>
```

Fonte: Autor

No quadro exposto, temos a descrição do sujeito: *http://en.wikipedia.org/wiki/Oxford*, com os predicados *title*, *coverage* e *publisher*, com os respectivos objetos de valor: *Oxford*, *Oxfordshire* e *Wikipedia*.

RDFa

O RDFa é um formato onde se insere as triplas RDF dentro dos documentos HTML, através da inserção dos dados com uso do HTML DOM (Document Object Model). Este formato é popular em situações onde é possível modificar os documentos HTML, porém há pouco controle sobre as mudanças na infraestrutura de publicação, como observado em diversos sistemas de gerência de conteúdo.

Um trecho de anotação RDFa pode ser visto no quadro 5:

Quadro 5 - Exemplo de trecho de documento HTML anotado com RDFa.

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/"  
  about="http://en.wikipedia.org/wiki/Oxford">  
  <span property="dc:title">Oxford</span>  
  <span property="dc:coverage">Oxfordshire</span>  
  <span property="dc:publisher">Wikipedia</span>  
</div>
```

Fonte: Autor

Neste quadro, temos a mesma descrição do quadro 4, para fins de comparação.

Turtle

Este formato é utilizado tipicamente para ler ou escrever RDF manualmente, pois possui suporte ao uso de *namespaces* e abreviaturas que facilitam o entendimento por seres humanos.

Um exemplo de Turtle pode ser visto no quadro 6:

Quadro 6 - Exemplo de trecho em formato Turtle.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
<http://en.wikipedia.org/wiki/Oxford>  
  dc:title "Oxford";  
  dc:coverage "Oxfordshire";  
  dc:publisher "Wikipedia".
```

Fonte: Autor

Neste quadro, temos a mesma descrição do quadro 4, para fins de comparação.

N-Triplas

É um subconjunto do formato turtle descrito no quadro 6, porém sem o suporte aos *namespaces* e abreviaturas. Como resultado, temos um alto grau de redundância e conseqüente aumento no tamanho dos arquivos. Tais características o tornam um formato vantajoso, já que permite que seus arquivos sejam interpretados linha por linha, facilitando no caso de arquivos maiores que não caberão na memória principal disponível e facilitando a compressão, reduzindo o uso de rede na transferência de arquivos.

Com as vantagens anteriormente citadas, este formato é considerado o padrão para troca de grandes quantidades de triplas, utilizados, por exemplo, em processos de backup e espelhamento de dados (Bizer & Heath 2011).

Um exemplo de N-Triplas pode ser visto no quadro 7:

Quadro 7 - Exemplo de trecho em formato N-Triplas.

```
<http://en.wikipedia.org/wiki/Oxford>  
<http://purl.org/dc/elements/1.1/title> "Oxford".  
  
<http://en.wikipedia.org/wiki/Oxford>  
<http://purl.org/dc/elements/1.1/coverage> "Oxfordshire".  
  
<http://en.wikipedia.org/wiki/Oxford>  
<http://purl.org/dc/elements/1.1/publisher> "Wikipedia".
```

Fonte: Autor

Para o quadro 7, temos a mesma descrição do quadro 4, para fins de comparação.

RDF/JSON

Este formato é altamente desejável, principalmente pelo fato de que várias linguagens de programação provêm suporte nativo ao JSON, o que implicaria em uso direto de RDF com JSON sem a necessidade de instalação adicional de bibliotecas específicas. No presente período não há um padrão consolidado, inclusive a instituição W3C possui mais de um rascunho para padrão^{14 15}.

3.5 Vocabulários e Ontologias

De acordo com (Bizer & Heath 2011), temos:

O padrão RDF provê um modelo de dados abstrato e genérico para descrever recursos usando triplas do tipo sujeito, predicado, objeto. No entanto, não provê termos de domínio específico, para descrever classes de coisas no mundo e como se relacionam-se entre si (Bizer & Heath 2011) Essa função é servida pelas taxonomias, vocabulários e ontologias, expressadas em SKOS (do inglês, Simple Knowledge Organization System), RDFS (do inglês RDF Vocabulary Description Language, ou RDF Schema) e OWL (do inglês, Web Ontology Language).

Dentre os tipos citados, temos o padrão SKOS usualmente utilizado para as taxonomias ou conceitos de hierarquias apenas, enquanto os padrões RDFS e OWL são utilizados para determinar conceitos do mundo real em classes, com suas propriedades (Bizer & Heath 2011).

O reuso de vocabulários pré-existentes é estimulado, nos casos onde é necessária a criação de vocabulários próprios, as ontologias e/ou esquema

¹⁴ <http://json-ld.org/spec/latest/json-ld>

¹⁵ <https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-json/index.html>

RDF (RDFS) devem ser utilizados, de acordo com as melhores práticas de Linked Data (Bizer & Heath 2011).

3.6 Metadados

As descrições das entidades também devem conter metadados que possibilitem ao cliente entender mais sobre a origem e o conjunto de dados do qual cada descrição faz parte. Os conjuntos de dados também são chamados de *datasets*. Até o momento da elaboração deste trabalho, duas alternativas para a descrição da estrutura dos dados são utilizadas:

Semantic Sitemaps

Uso do conhecido e estabelecido protocolo sitemaps com extensões semânticas. Faz-se uso de um arquivo XML localizado na raiz do servidor web, com marcações sobre a estrutura semântica do conjunto de dados. Tem como objetivo reduzir custos de tempo e recursos computacionais, uma vez que os clientes, através da descoberta de detalhes sobre os conjuntos de dados, nas definições de *sitemap*, não percorreriam todos os dados em uma estratégia de *crawling* e sim realizariam, por exemplo, um download de partes dos dados, caso haja este suporte (Cyganiak et al. n.d.).

Descrições Void

De acordo com sua especificação, Void é um vocabulário em formato RDF Schema para expressar metadados sobre conjuntos de dados RDF. Tem como objetivo ser uma ponte entre os publicadores e usuários de dados RDF, com aplicações variando desde a descoberta de dados à catalogação e arquivamento de conjuntos de dados (Alexander et al. 2011).

VoID é o padrão reconhecido para a descrição de datasets, incorpora funcionalidades do método anterior e acrescenta outras, tudo através de RDF, diferentemente do método anterior.

De acordo com (Bizer & Heath 2011), a descrição void é o padrão recomendado.

Licença

Outro metadado importante é a licença ao qual a descrição da entidade está submetida. É importante salientar que não é possível licenciar o objeto do mundo real ou fato, pois o mesmo não pode ser licenciado. Apesar de não ser obrigatória, a ausência de licença não garante o direito aos dados e pode desestimular o seu reuso, portanto todos os dados linkados publicados na web devem incluir declarações de licença (Bizer & Heath 2011).

Um exemplo de possível uso de namespace e atribuição da URI do Creative Commons, para a licença CC-BY-SA podem ser vistos no quadro 8, em formato *Turtle*:

Quadro 8 - Exemplo de uso de licença Creative Commons CC-BY-SA.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix sioc: <http://rdfs.org/sioc/ns#> .
@prefix cc: <http://creativecommons.org/ns#> .

<http://biglynx.co.uk/blog/making-pacific-sharks>
  rdf:type sioc:Post ;
  dcterms:title "Making Pacific Sharks " ;
  dcterms:date "2010-11-10T16:34:15Z " ^^ xsd:dateTime ;
```

```

sioc:has_container <http://biglynx.co.uk/blog/> ;
sioc:has_creator <http://biglynx.co.uk/people/matt-briggs> ;
foaf:primaryTopic <http://biglynx.co.uk/productions/pacific-sharks> ;
sioc:topic <http://biglynx.co.uk/productions/pacific-sharks> ;
sioc:topic <http://biglynx.co.uk/locations/pacific-ocean> ;
sioc:topic <http://biglynx.co.uk/species/shark> ;
sioc:content "Day one of the expedition was a shocker - monumental
swell, tropical storms, and not a shark in sight. The Pacific was up to
its old
tricks again. I wasn't holding out hope of filming in the following 48
hours , when the unexpected happened ..." ;
foaf:isPrimaryTopicOf
<http://biglynx.co.uk/blog/making-pacific-sharks.rdf > ;
foaf:isPrimaryTopicOf
<http://biglynx.co.uk/blog/making-pacific-sharks.html > ;
cc:license <http://creativecommons.org/licenses/by-sa/3.0/> .
<http://biglynx.co.uk/blog/making-pacific-sharks.rdf>
rdf:type foaf:Document ;
dcterms:title "Making Pacific Sharks (RDF version )" ;
foaf:primaryTopic<http://biglynx.co.uk/blog/making-pacific-sharks> .

<http://biglynx.co.uk/blog/making-pacific-sharks.html>
rdf:type foaf:Document ;
dcterms:title "Making Pacific Sharks (HTML version )" ;
foaf:primaryTopic<http://biglynx.co.uk/blog/making-pacific-sharks> .

```

Fonte: (Bizer & Heath 2011)

É importante notar que, nesse caso a licença foi aplicada à uma entrada de blog, não aos documentos que o representam, pois os documentos que representam a entrada de blog contém um misto de material passível e não-passível de licenciamento (conteúdo da entrada e data de publicação, respectivamente).

Termos de licença também podem ser adicionados ao nível de um amplo conjunto de dados, dentro da especificação Void (Alexander et al. 2009).

3.7 Resultado das Consultas SPARQL

De acordo com recomendação da W3C até o presente momento deste trabalho, existem definições de resultados de consultas SPARQL nos formatos XML, CSV e TSV, JSON, sendo o primeiro formato o primeiro a ter especificação definida. Os formatos suportados para a representação dos resultados de consultas SPARQL variam de acordo com as ferramentas utilizadas.

Recentemente, com as especificações SPARQL 1.1, temos adição de funcionalidades como consultas federadas, que provê suporte às consultas em diferentes pontos de acesso SPARQL, à escolha da aplicação e/ou usuário.

3.8 Estatísticas do ambiente

Uso

O projeto RDFStats (Langegger & Wöß 2009) consiste em um gerador de estatísticas de fontes RDF, como endpoints SPARQL e documentos RDF. Possui em suas funcionalidades a capacidade de geração de diferentes itens estatísticos como quantidade de instâncias por classe, histogramas por classe, propriedade e tipo de valor, dois geradores de estatísticas para documentos RDF (arquivos locais e recursos na web) e endpoints SPARQL, API para acesso à estatísticas e diversas funções de estimação, entre outras.

Conteúdo

Apesar de a atual especificação SPARQL não contemplar amplas consultas sobre as bases para determinar dados estatísticos, existem iniciativas que tem como objetivo a coleta e geração de dados estatísticos, nessa área temos

iniciativas como o projeto *make-void*¹⁶ e *LODStats*, que se propõe como alternativa mais performática quando comparada com abordagens existentes, especialmente em grandes conjuntos de dados com muitos milhões de triplas (Demter et al. n.d.).

As abordagens citadas utilizam o vocabulário *VOID* (Alexander et al. 2011) para prover dados sobre as características de um conjunto de dados (também chamado tecnicamente de *dataset*), como: número total de triplas, número total de entidades, número total de classes distintas, número total de propriedades distintas, lista de todas as classes usadas no conjunto de dados, entre outras estatísticas.

3.9 Fluxo e Estrutura de publicação de linked data

A publicação de *Linked data* requer a adoção dos princípios básicos expostos anteriormente na definição teórica de *Linked Data*. A conformidade com os padrões e melhores práticas que sustentam esses princípios é o que permite a interoperabilidade e reuso de *Linked Data* na *Web*.

No entanto, a conformidade com os princípios do *Linked Data* não implica o abandono dos sistemas de gerenciamento de dados existentes e aplicações de negócio porém simplesmente a adição de camada técnica de fusão para conectar esses na *Web* de dados.

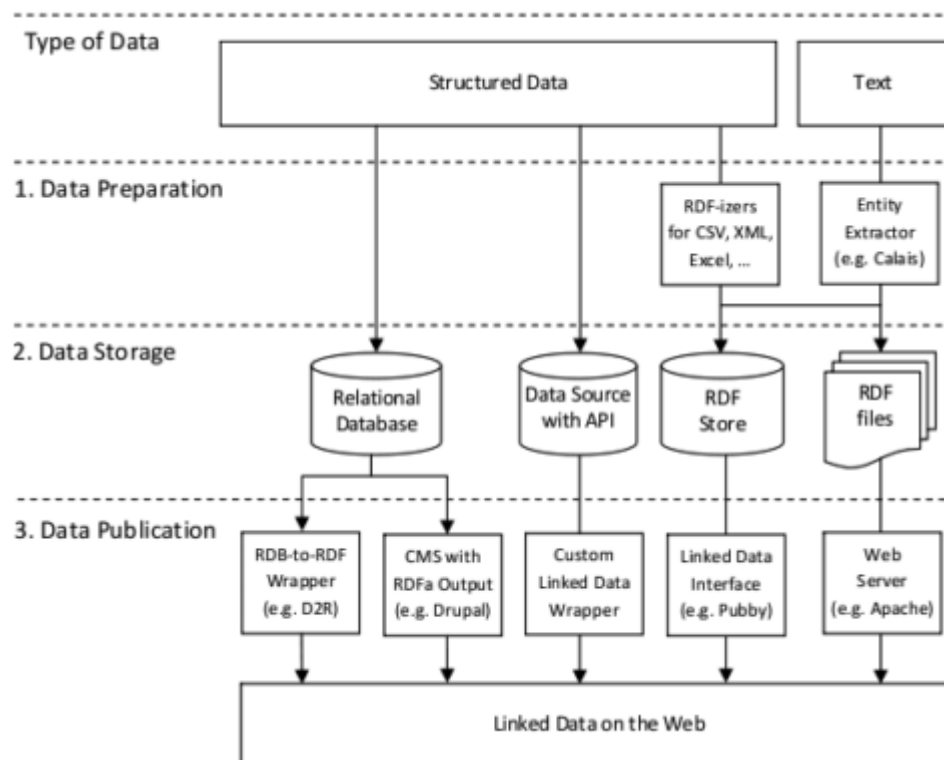
Enquanto existe um grande número de sistemas que podem ser conectados na *web* de dados, os mecanismos para tal reduzem-se à um pequeno número

¹⁶ <https://github.com/cygri/make-void>

de padrões de publicação de Linked Data. Nesta seção veremos mais detalhes sobre esses padrões.

Na figura 5, temos os padrões de publicação mais comuns de Linked Data, de acordo com (Bizer & Heath 2011).

Figura 5 - Fluxo de publicação de Linked Data.



Fonte: (Christian Bizer & Heath 2011)

Dados de entrada

Os dados de entrada são a parte principal da decisão sobre o fluxo de publicação a ser seguido, e estão associados às fontes de dados planejadas para futuro uso no formato Linked Data. Na figura 5, temos a divisão entre dados estruturados e textuais, proposto por (Bizer & Heath 2011).

No caso de dados textuais, associa-se o uso de ferramentas que adicionam anotações aos documentos, com os URIs de entidades referenciadas nos documentos. Ferramentas como o Calais¹⁷, Ontos¹⁸ e DBPediaSpotlight¹⁹, uma vez que esse tipo de dado não possui estrutura definida.

Para os dados estruturados, o processo de preparação dos dados se faz necessário, seja via construção de *wrappers* customizados ou automatizados (através de arquivos de mapeamento das estruturas equivalentes entre as tabelas e modelo RDF), ou ainda no uso dos chamados *RDF-izers*, que convertem os dados que estão em formatos como CSV, Excel, entre outros, para posterior armazenamento em arquivos estáticos ou em bancos de triplas RDF, também chamados de *Triplestores* ou *RDF Stores*.

Para que o fluxo seja determinado para cada ambiente de Linked Data, algumas considerações adicionais se fazem necessárias, detalhadas a seguir.

Volume dos dados

De acordo com (Bizer & Heath 2011), se o volume for de apenas algumas centenas de triplas RDF, é provavelmente desejável publicar na forma de arquivos estáticos RDF, tal processo pode requerer mais esforço manual na gerência dos dados se os mesmos dados devem ser mantidos em múltiplos locais e/ou formatos, porém evita os esforços de padrões mais tecnicamente complexos.

¹⁷ <http://www.opencalais.com>

¹⁸ http://www.ontos.com/o_eng/index.php?cs=1

¹⁹ <http://wiki.dbpedia.org/spotlight>

Uma recomendação para evitar uso indesejado tráfego de rede, seria a separação de cada entidade em um único arquivo, seja estático ou inserido em Servidor RDF, tal medida evita a transmissão de dados que poderão não ser utilizados.

Portanto, se a quantidade de dados superar centenas de triplas, poderá ser desejável o uso de alternativas como armazenamento em bases de dados RDF especializadas.

Frequência de atualização dos dados

Em se tratando de baixo volume de dados, se os dados sofrem pouca ou nenhuma mudança, pode ser apropriado o uso de arquivos estáticos, no entanto se mudanças ocorrem com frequência, pode ser mais apropriado o uso de bancos de dados RDF. No caso em que os dados de entrada estejam num banco de dados relacional, podem ser armazenados neste banco e com o uso de *wrappers* para mapeamento relacional para RDF.

De maneira prática, (Bizer & Heath 2011) especifica objetivos dos tipos principais de representação RDF:

- Arquivos estáticos em RDF – Pequena quantidade de dados com baixa taxa de atualização, requiere trabalho manual para modificação de dados.
- Dados em Bancos de dados para RDF (RDF stores) – Para quantidades maiores de dados, com atualização constante. Alterações semi ou totalmente automatizadas.

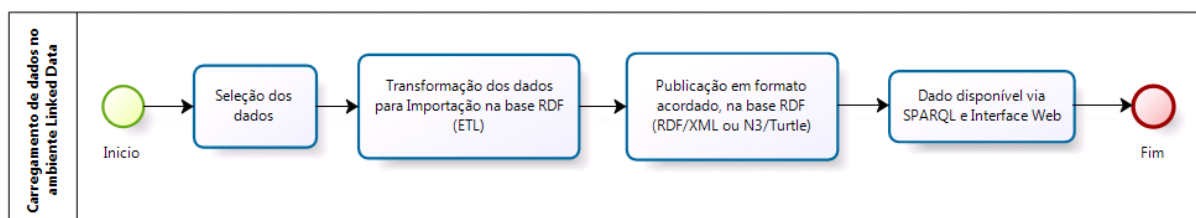
- Dados em Bancos de dados relacionais legados e uso de *wrappers* - Quando já existem dados úteis nos sistemas de bancos de dados relacionais e não planeja-se migrá-los, é possível definir mapeamentos entre a estrutura relacional e o RDF, afim de traduzir em tempo de execução de pesquisa, os dados de uma fonte relacional para RDF.

É possível o uso híbrido de soluções, como o uso de arquivos estáticos em RDF/XML para vocabulários e ontologias e bancos de dados RDF para os dados em RDF, assim como o mecanismo de dereferenciação pode ser diferente para cada um deles, hash para vocabulários e ontologias e 303 para os dados, como comentado na seção *Mecanismos de dereferenciação*, deste trabalho.

3.10 Fluxo de publicação de dados

Apesar de não ser escopo deste trabalho, o fluxo de publicação de dados também é um ponto decisório importante na implantação de um ambiente Linked Data, a figura 6 exemplifica um processo genérico considerado, para a elaboração posterior da infraestrutura:

Figura 6 - Fluxo de publicação dados genérico



Fonte:Autor

O fluxo da figura considera um modelo simplificado para fins didáticos e tem como premissa que todos os dados serão transformados antes dos

carregamento no ambiente Linked Data, preferencialmente no formato padrão RDF/XML, portanto neste momento não faz parte das tarefas da Infraestrutura de Linked Data, transformar os dados dos publicadores ou prover métodos semi-automatizados para tal.

3.11 Arquitetura do ambiente

Arquitetura Inicial

Em alguns casos, onde é possível utilizar servidores RDF com múltiplas funções, é possível construir uma arquitetura de ambiente somente composto por um este único elemento, como, por exemplo, o *Openlink Virtuoso Open Source*, tal como a arquitetura atual do projeto DBPedia²⁰. No entanto, a arquitetura possui alguns pontos de atenção, pelos seguintes fatores:

- Desacoplamento - Dificuldade no desacoplamento dos elementos do sistema quando a arquitetura é baseada em um único produto, nesse caso seria necessário a construção de todo o ambiente novamente, no caso da necessidade da troca do servidor RDF
- Customização dificultada - No caso do produto *Openlink Virtuoso Open Source*, o processo de customização mostrou-se complexo, exigindo um domínio de linguagem específica do produto (Virtuoso Server Pages), através da programação do arquivo *description.vsp* e criação de um conjunto de artefatos para a representação das consultas de dados da base RDF em formato HTML

²⁰ <http://dbpedia.org/Architecture>

- Robustez - O argumento favorável de maior robustez do ambiente com uma única camada especializada e redução nos encaminhamentos de mensagens é válido em implementações de grande porte, a própria *DBPedia* não considerou este ponto no início da sua arquitetura²¹, apenas com o crescimento dos acessos passou a ser um ponto predominante na arquitetura de software.

De acordo com os pontos citados, uma arquitetura padrão será considerada para projetos onde não se caracteriza um grande número de acessos, de maneira que a maioria dos pontos acima fossem endereçados.

Arquitetura Padrão Sugerida

Para suprir as deficiências de uma arquitetura única com apenas uma camada, a arquitetura aplicada sugerida consiste em três camadas:

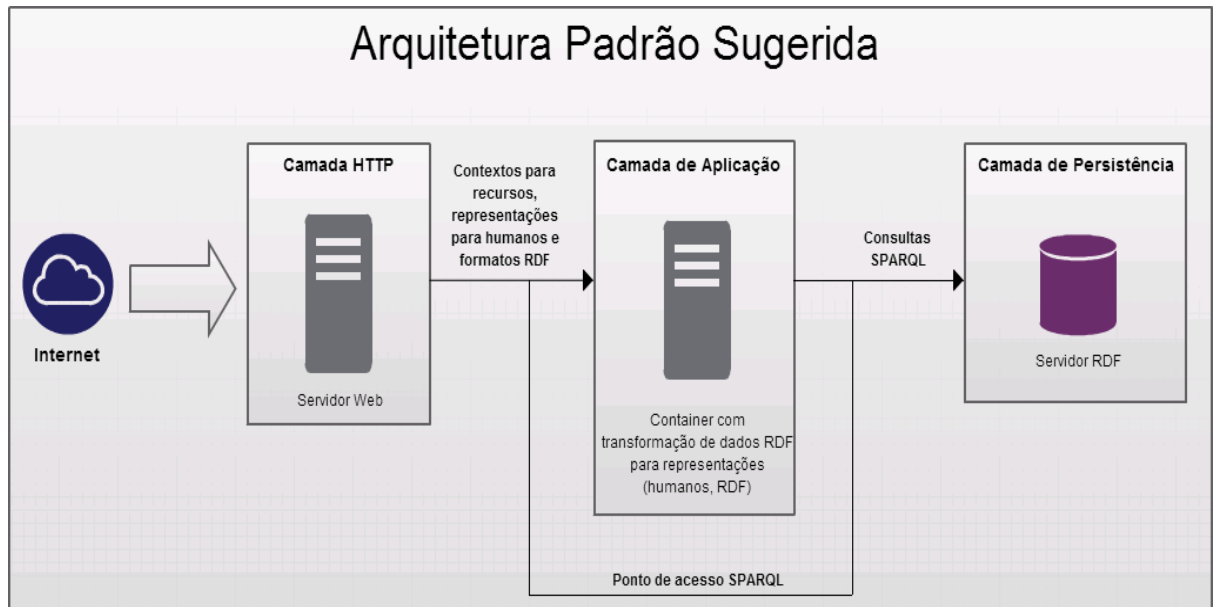
- Camada HTTP - Composta por servidor Web HTTP (*Apache HTTP Server*, por exemplo) e responsável pela camada de interação inicial de todas as requisições
- Camada de Aplicação - Composta por container responsável pela transformação das URIs e apresentação do conteúdo recebido do do servidor RDF (*Apache Tomcat*, hospedando a aplicação *Pubby*, por exemplo)
- Camada de Persistência - Composta pelo servidor de banco de dados RDF (como *Open Link Virtuoso Open Source*, *Sesame*, entre outros) e

²¹ Disponível em: <http://dbpedia.org/Architecture>

responsável por hospedar e retornar as consultas sobre as triplas de dados, ontologias e vocabulários

Na figura 7 temos o diagrama da arquitetura padrão sugerida:

Figura 7: Arquitetura padrão sugerida



Fonte: Autor

No diagrama da figura 7, observa-se as informações em alto-nível, onde temos uma ponte direta entre a camada HTTP e a Camada de Persistência apenas para o acesso ao *endpoint* SPARQL, e os outros contextos devem ser definidos na etapa de design das URIs direcionados primeiramente à camada de Aplicação, onde a camada de aplicação realiza a conversão dos pedidos para o servidor RDF, via consultas SPARQL.

3.12 Elementos de operação de infraestrutura

Otimizações desejáveis

De maneira geral, bancos de dados possuem meios de controle de requisições e validade por tempo (*deadlocks* e *timeouts*), otimização de consultas e de uso de memória. (Elmasri & Navathe 2004, tradução nossa)

O uso de servidores web como o apache HTTP Server também suportam otimizações diversas²² como uso de estratégias de *caching*, alocação de múltiplos processos, entre outros.

Como exemplo, no caso do uso da ferramenta OpenLink Virtuoso, é possível aplicar diversas configurações a fim de tornar o ambiente mais robusto. Desde configurações gerais tanto ao processamento de requisições, quanto específicas ao endpoint SPARQL^{23 24}.

Controle e Versionamento de ontologias e vocabulários

No caso de armazenamento em arquivos estáticos, diversos softwares podem ser utilizados e até utilitários tradicionais de controle de mudanças de linha de comando como o *git*²⁵ e *subversion*²⁶, os softwares Neologism²⁷ e DOME²⁸ apresentam-se como soluções especializadas para o controle, versionamento e autoria de ontologias.

No caso de versionamento de Ontologias em RDF, observa-se um campo ainda em desenvolvimento, com iniciativas práticas como o SemVersion²⁹ e

²² <http://httpd.apache.org/docs/2.4/misc/perf-tuning.html>

²³ <http://www.openlinksw.com/dataspace/dav/wiki/Main/VirtConfigScale>

²⁴ <http://www.openlinksw.com/dataspace/dav/wiki/Main/VirtSPARQLEndpointProtection>

²⁵ <http://git-scm.com>

²⁶ <http://subversion.apache.org>

²⁷ <http://neologism.deri.ie>

²⁸ <http://dome.sourceforge.net>

²⁹ <http://semanticweb.org/wiki/SemVersion>

OntoView³⁰, sendo o primeiro com última versão entregue em 2008 e com status "abandonado em 2012" e o último com *website* fora do ar.

Estratégias de backup de dados

Quando do uso de um sistema de banco de dados RDF (RDF Store), práticas comuns às bases de dados relacionais também são frequentemente suportadas, no caso os backups totais e incrementais (Elmasri & Navathe 2004, tradução nossa).

Quando há uso de arquivos locais no sistema, como arquivos HTML e RDF/XML, abordagens comuns ao backup de arquivos podem ser aplicadas, seja o backup físico ou lógico dos dados, onde o backup lógico seria a cópia periódica de dados de um meio de armazenamento à outro, ou cópias no mesmo meio de armazenamento e o backup físico sendo o armazenamento dos blocos do disco onde os dados residem para outro meio de armazenamento (Hutchinson et al. 1999, tradução nossa).

Mecanismos de extração de porções/dumps dos dados

Uma maneira genérica de obter partes dos dados de um ambiente é fazendo o uso do mecanismo padrão de consulta de Linked Data, o SPARQL, onde o escopo é definido de acordo com as características dos arquivos RDF. Além desse, existem mecanismos específicos dependendo da ferramenta utilizada, como o suporte a 'dump' de triplas em formato TTL pelo produto Virtuoso Open Source Edition³¹ pela linha de comando através do comando `dump_graphs`³².

³⁰ <http://ontoview.org>

³¹ <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main>

³² <http://www.openlinksw.com/dataspace/dav/wiki/Main/VirtDumpLoadRdfGraphs>

Quadro simplificado de Elementos para Infraestrutura de Linked Data

O quadro apresentado a seguir provê uma síntese dos pontos importantes detalhados nas seções anteriores, objetivando a praticidade nas decisões básicas.

Quadro 9 - Esquema de pontos decisórios para a implantação de Linked Data.

(continua)

Elemento	Alternativas	Recomendações
Volume de dados esperados	Alto (Mais de 500 triplas)	Para volumes maiores de triplas ou dinamicidades mais altas, a recomendação é utilizar os servidores RDF, caso contrário, é possível o uso de arquivos estáticos.
	Baixo (Abaixo de 500 triplas)	
Frequência de atualização dos dados	Alta	
	Baixa	
Design das URIs	Separação por contexto	A separação por extensão de arquivo é uma possibilidade quando há armazenamento em arquivos estáticos, as separações por sub-domínio e contexto são opções similares em termos de alcance, a separação por contexto já possui implementação simplificada em ferramentas como o Pubby ³³
	Separação por subdomínio	
	Separação por extensão de arquivo	
Mecanismos de Dereferenciação	Hash	O mecanismo de Hash é utilizado com mais frequência quando cada URI representa um pequeno volume dados armazenado. Para maiores volumes de dados armazenados, é recomendável o mecanismo 303, ou ainda o mecanismo Híbrido, porém com custo na performance e complexidade de configuração.
	URIs 303	
	Híbrido	
Licença e Metadados	Não incluída	Apesar de ser uma prática recomendada, não há obrigatoriedade na publicação de condições de licença sobre os dados armazenados.
	Incluída manualmente	
	Incluída automaticamente	

³³ <http://wifo5-03.informatik.uni-mannheim.de/pubby/>

Quadro 9 - Esquema de pontos decisórios para a implantação de Linked Data.

(continuação)

Elemento	Alternativas	Recomendações
Armazenamento e controle de Ontologias e Vocabulários	Armazenadas em servidores RDF	Para armazenamento de Ontologias e Vocabulários como arquivos estáticos, é possível uma estratégia de versionamento com ferramentas de controle de arquivos.
	Armazenadas em arquivos estáticos	
Estatísticas sobre os dados de Uso e Conteúdo	Uso	Não existe exigência, mas é uma abordagem crucial para entender como está o uso do ambiente e informar aos usuários (sejam programas ou pessoas), sobre os detalhes e quantidades de dados em cada conjunto de dados (dataset).
	Conteúdo	
	Ambos	
Operação da Infraestrutura	Otimizações	Cada item deve ser identificado e executado e é dependente da arquitetura definida.
	Backup/Restore	
Formatos de serialização pretendidos	RDF/XML	A escolha da quantidade de formatos de serialização suportados é um fator relacionado às ferramentas necessárias para sua realização.
	N3/Turtle	
	CSV	
	JSON	
	Outros	

Fonte: Autor

4 APLICAÇÃO PRÁTICA NO LABORATÓRIO DE ENGENHARIA DO CONHECIMENTO (EGC)

4.1 Introdução

Uma vez fundamentados os pontos principais na definição da infraestrutura de Linked Data, foi então explorado um exemplo prático de processo desde a definição das opções possíveis até as melhorias e ajuste fino para um ambiente real, no caso um exemplo de uso para o laboratório EGC.

O exercício prático consistiu em preparar um ambiente de laboratório local para posterior validação e implantação pelo laboratório EGC.

4.2 Sobre o EGC

O Departamento de Engenharia do Conhecimento (dEGC) da Universidade Federal de Santa Catarina, em consonância com os objetivos da Universidade de produzir, sistematizar e socializar o saber científico e tecnológico, ampliando e aprofundando a formação do ser humano para o exercício profissional e a reflexão crítica, ocupa-se do desenvolvimento de atividades integradas de ensino, pesquisa e extensão na área multidisciplinar relacionada à engenharia, gestão e disseminação do conhecimento e áreas correlatas.

Em sua área de atuação, o dEGC objetiva contribuir significativamente para:

- A formação, nos níveis de graduação e pós-graduação, de recursos humanos qualificados, criativos e críticos, na perspectiva da construção de uma sociedade justa e democrática;

- O desenvolvimento de atividades multidisciplinares, transversais e multidimensionais de modo a possibilitar que os egressos da UFSC (graduandos e pós-graduandos) agreguem às suas competências técnicas específicas, uma visão sistêmica sobre o processo de codificação, gestão e disseminação do conhecimento (tácito ou explícito), de forma a tornarem-se profissionais mais capacitados na construção uma sociedade que vise o bem comum da sociedade;
- A implementação de modelos, métodos e técnicas para a promoção do desenvolvimento em codificação, gestão e disseminação dos conhecimentos (explícitos e tácitos) em organizações, públicas e privadas, e na sociedade em geral;
- O avanço do conhecimento científico e tecnológico;
- A difusão de conhecimento para o setor produtivo regional e nacional.

4.3 Definições para a implantação

O objetivo principal da implantação no laboratório EGC foi um modelo robusto o suficiente, porém simplificado para a manutenção e extensão posterior.

Para melhor definição da arquitetura, o quadro proposto na seção anterior foi utilizado para discussão dos elementos, com as escolhas marcadas na cor cinza:

Quadro 10 - Esquema de pontos decisórios discutidos com membros do EGC.

(continua)

Elemento	Alternativas	Considerações EGC
Volume de dados esperados	Alto (Mais de 500 triplas)	É desejável o suporte a grandes volumes de dados, além de preparação caso a frequência de atualização destes dados for alta.
	Baixo (Abaixo de 500 triplas)	
Frequência de atualização dos dados	Alta	
	Baixa	
Design das URIs	Separação por contexto	A separação por contexto foi decidida, em parte por já ser o método utilizado pela DBPedia, e além da simplicidade maior de configuração da infra-estrutura nesse modelo
	Separação por subdomínio	
	Separação por extensão de arquivo	
Mecanismos de Dereferenciação	Hash	O mecanismo de URIs 303 será utilizado inicialmente, pelos ganhos em organização e transmissão dos dados, que é compensatório pela intenção de suportar grandes volumes de dados, sem a necessidade do envio de dados extras, que ocorre no modo Hash
	URIs 303	
	Híbrido	
Licença e Metadados	Não incluída	Não fará parte do escopo deste trabalho
	Incluída manualmente	
	Incluída automaticamente	
Armazenamento e controle de Ontologias e Vocabulários	Armazenadas em servidores RDF	Como o armazenamento de Ontologias será centralizado no servidor RDF, não foram aplicadas estratégias de versionamento
	Armazenadas em arquivos estáticos	
Estatísticas sobre os dados de Uso e Conteúdo	Uso	Ambos os meios são desejáveis, porém não farão parte do escopo deste trabalho.
	Conteúdo	
	Ambos	
Operação da Infraestrutura	Otimizações	Todos os dois fatores devem ser considerados, além de customização básica visual HTML.
	Backup/Restore	

Quadro 10 - Esquema de pontos decisórios discutidos com membros do EGC.

(continuação)

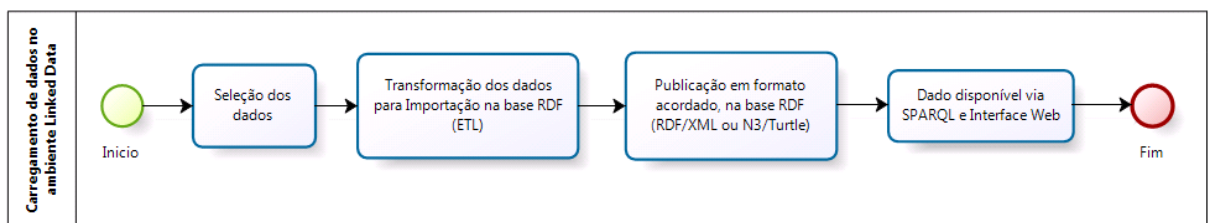
Elemento	Alternativas	Considerações EGC
Formatos de serialização pretendidos	RDF/XML	Após a definição da arquitetura, os formatos suportados serão: RDF/XML, N3/Turtle e CSV,
	N3/Turtle	
	CSV	
	JSON	
	Outros	

Fonte: Autor

4.4 Fluxo de publicação de dados no ambiente EGC

No caso prático do EGC, o Fluxo de publicação genérico detalhado na seção 3.10 deste trabalho, foi realizado, a figura 8 repete o fluxo, para fins didáticos:

Figura 8 - Fluxo de publicação dados no ambiente EGC.



Fonte:Autor

O fluxo da figura considera como premissa que todos os dados serão transformados antes dos carregamento no ambiente Linked Data, preferencialmente no formato padrão RDF/XML, portanto neste momento não faz parte das tarefas da Infraestrutura de Linked Data, transformar os dados dos publicadores ou prover métodos semi-automatizados para tal.

4.5 Convenções de URIs Aplicadas

O nome de rede (conhecido tecnicamente como *hostname*) escolhido para o exercício foi *lodkem.egc.ufsc.br*, sendo este o nome que estará disponível aos usuários do ambiente. Uma vez definido o design e mecanismo de dereferenciação das URIs, as convenções de nome foram definidas, conforme detalhados na tabela 4:

Tabela 4 - Representação de URIs por contexto para o EGC.

URI	Dado retornado
http://lodkem.egc.ufsc.br/resource/coisa	Conceito do mundo real
http://lodkem.egc.ufsc.br/data/coisa	Representação para computadores, da coisa
http://lodkem.egc.ufsc.br/page/coisa	Representação para seres humanos, da coisa
http://lodkem.egc.ufsc.br/onto/coisa	Representação de Ontologias e Vocabulários
http://lodkem.egc.ufsc.br/property/coisa	Representação de propriedades de Ontologias e Vocabulários
http://lodkem.egc.ufsc.br/class/coisa	Representação de classes de Ontologias e Vocabulários

Fonte: Autor

4.6 Arquitetura do ambiente

Arquitetura Aplicada

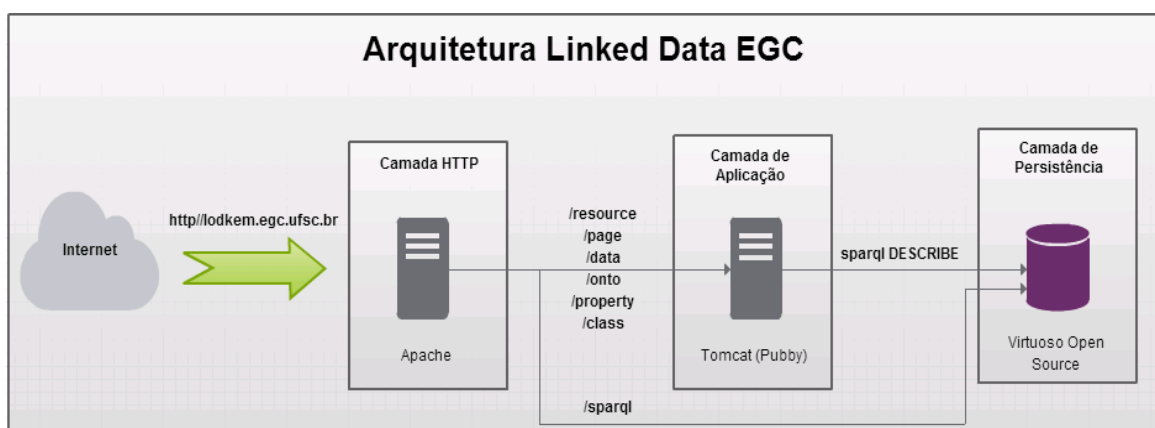
Para suprir as deficiências de uma arquitetura única com apenas uma camada, e considerando que no caso do laboratório EGC, não haverá

expectativa de elevado número de acessos em primeiro momento, a arquitetura aplicada foi testada e completada e consiste em três camadas:

- Camada HTTP - Composta pelo servidor Web HTTP *Apache HTTP Server* e responsável pela camada de interação inicial de todas as requisições, configurado para responder na porta 80
- Camada de Aplicação - Composta pelo container *Apache Tomcat*, hospedando a aplicação *Pubby* e responsável pela transformação das URIs e apresentação do conteúdo recebido do Virtuoso (negociação de conteúdo), configurado para responder na porta 8080
- Camada de Persistência - Composta pelo servidor de banco de dados RDF *Open Link Virtuoso Open Source* e responsável por hospedar e retornar as consultas sobre as triplas de dados, ontologias e vocabulários, configurado para responder na porta 8890

Na figura 9 temos o diagrama da arquitetura aplicada:

Figura 9: Arquitetura aplicada para o ambiente EGC.



Fonte: Autor

No diagrama da figura 9, observa-se as informações adicionais dos contextos e redirecionamentos em alto-nível, onde temos uma ponte direta entre a camada HTTP e a Camada de Persistência apenas para o acesso ao *endpoint* SPARQL, e os outros contextos definidos para o design das URIs direcionados primeiramente à camada de Aplicação, onde o aplicativo Pubby realiza a conversão dos pedidos para o Virtuoso, via consultas SPARQL do tipo DESCRIBE.

4.7 Implantação

Todas as camadas foram implementadas em um único servidor para fins de simplificação, sendo possível tanto a separação quanto o aumento de servidores em cada camada, conforme a necessidade, com a observação de que a camada de persistência com o produto Open Link Virtuoso Open Source só suporta mais de um servidor agrupado fora da versão open source, exigindo custos em seu licenciamento.

A construção do ambiente de laboratório foi realizada em uma estação pessoal do autor deste trabalho, dentro de uma máquina virtual, em ambiente de virtualização VMWare Player, versão 6.0.0, com 2 GB de memória disponível e aproximadamente 2 TB de disco rígido virtual.

O Sistema operacional escolhido para a implantação foi o GNU/Linux Ubuntu Server 12.04.3 LTS x86_64 (64 Bits), além de ser não implicar em custos, possui suporte à instalação da maioria dos softwares envolvidos na solução via gerenciador de pacotes nativo. A instalação de cada componente será omitida nas seções seguintes, estando disponível no Apêndice 1 - Instalação e configuração das ferramentas no ambiente EGC, ao final deste trabalho.

De maneira sintética, os seguintes programas foram utilizados:

- **Apache HTTP Server 2.2.22** - Servidor Web
- **Apache Tomcat 7.0.42** - Container de Servlets Java para o programa Pubby
- **Java 6 SE 1.6.0_25** - Plataforma Java Standard Edition requerida pelo Apache Tomcat
- **Pubby 0.3.3** - Aplicação Java para publicação de Linked Data via SPARQL
- **Open Link Virtuoso Open Source 6.1** - Aplicação tanto como base RDF com endpoint SPARQL, quanto para a própria publicação em si (não utilizada no ambiente Linked Data EGC)

Configuração da camada HTTP

Além da instalação propriamente dita, houve a necessidade de configuração do servidor web para que envie as requisições para os serviços desejados, função denominada de *proxy*. E para os contextos */onto*, */class/* e */property*, desejáveis para o uso de ontologias e vocabulários, será necessário um processo de reescrita das URIs para compatibilidade com o Pubby. Essas duas funções serão o foco principal da camada HTTP.

Para que seja possível realizar a função de proxy e reescrita, se fez necessária a habilitação dos módulos, através dos comandos abaixo:

```
fschroeder@lodkem:~$ sudo a2enmod proxy_http proxy rewrite headers
```

O seguinte comando se faz necessário para a aplicação das configurações do passo anterior:

```
fschroeder@lodkem:~$ sudo service apache2 restart
```

Após as configurações abaixo, o servidor Web já poderá ser configurado para realizar as funções desejadas. As configurações são realizadas através da edição do arquivo *default*, localizado no diretório */etc/apache2/sites-available*, na instalação via repositório de pacotes do Ubuntu.

Para acesso através da camada HTTP ao *endpoint* SPARQL do Virtuoso, as seguintes entradas foram adicionadas:

```
ProxyPass /sparql http://lodkem.egc.ufsc.br:8890/sparql
ProxyPassReverse /sparql http://lodkem.egc.ufsc.br:8890/sparql
```

Com a configuração acima completada, é possível acessar a interface sparql do servidor Virtuoso diretamente.

Para envio das requisições entre as camadas HTTP e de Aplicação as seguintes entradas foram adicionadas:

```
ProxyPass /resource http://lodkem.egc.ufsc.br:8080/resource
ProxyPassReverse /resource
http://lodkem.egc.ufsc.br:8080/resource
```

```
ProxyPass /page http://lodkem.egc.ufsc.br:8080/page
ProxyPassReverse /page http://lodkem.egc.ufsc.br:8080/page
```

```
ProxyPass /data http://lodkem.egc.ufsc.br:8080/data
ProxyPassReverse /data http://lodkem.egc.ufsc.br:8080/data
```


As regras acima encaminham o acesso direto externo ao *endpoint* SPARQL do servidor Virtuoso representado pelo contexto `/sparql` e permitem o encaminhamento de requisições entre as camadas HTTP e Aplicações, para o caso das URIs `/resource`, `/data` e `/page`. Neste momento, já é possível o acesso aos recursos armazenados no Virtuoso, com exceção das ontologias e vocabulários.

Há também uma configuração necessária para o encaminhamento de requisições para os arquivos de estilo da aplicação Pubby, acessíveis pelo contexto `/static`, listada abaixo:

```
ProxyPass /static http://lodkem.egc.ufsc.br:8080/static
ProxyPassReverse /static http://lodkem.egc.ufsc.br:8080/static
```

Com a adição da configuração acima, a exibição da versão HTML dos recursos é exibida de maneira apropriada, com os arquivos de folha de estilo do Pubby acessíveis.

Além dessas configurações, houve a necessidade do tratamento das URIs `/onto`, `/class`, e `/property`, tais contextos são necessários no acesso às ontologias e vocabulários hospedados no ambiente, além da capacidade de acesso direto às classes e propriedades das ontologias, de maneira similar ao que observado no projeto DBPedia (Bizer, Lehmann, et al. 2009).

Durante a implementação de tais contextos, percebeu-se que o aplicativo Pubby não seria capaz de realizar a negociação de conteúdo da maneira esperada para todos os conjuntos de URIs definidos. Existe a exigência de que, na mesma instância do aplicativo, todos os recursos representáveis sejam

iniciados por contextos únicos, no caso */page* para a forma HTML e */data* para todos os recursos representáveis por RDF/XML ou N3/Turtle.

Para evitar os custos de manutenção, desempenho e organização de se possuir duas instâncias do Pubby executando em paralelo, foram configurados conjuntos de reescritas de URI para que o servidor HTTP realize a negociação do conteúdo e redirecionamento de URIs e encaminhe diretamente à camada de aplicação (Pubby).

Na prática, ao invés de deixar o redirecionamento de um pedido exemplificado pela URI *http://lodkem.egc.ufsc.br/onto/exemplo*, ser redirecionado no Pubby para:

http://lodkem.egc.ufsc.br/page/onto/exemplo (HTML), ou

http://lodkem.egc.ufsc.br/data/onto/exemplo (RDF/XML), o servidor Apache

HTTP Server fará esse redirecionamento no modo *proxy*, que manterá a URI de requisição inalterada em todas as solicitações, sem a adição do prefixo */page* ou */data*, alcançando os objetivos do design de URIs de maneira completa. O mesmo processo foi configurado para os contextos */class* e */property*.

Para a implementação da negociação de conteúdo no servidor Apache HTTP Server, se faz necessário a checagem de um atributo enviado no cabeçalho das solicitações HTTP, chamado *Accept*. Este atributo padrão HTTP indica um ou mais formatos desejáveis pelo solicitante, para a resposta de uma solicitação. No caso de um navegador HTML, é comum o envio do atributo *Accept: text/html*, já para o recebimento de RDF/XML, temos o atributo como

Accept: application/rdf+xml e para recebimento de N3/Turtle, temos o atributo

Accept: application/x-turtle.

Se nenhum dos formatos for recebido, ou não for nenhum dos citados anteriormente, a regra foi implementada para que a forma HTML seja retornada.

As regras definidas para a reescrita do contexto */onto*, que foram repetidas de maneira análoga para os contextos */class* e */property*, podem ser vistas abaixo:

#Redirecionamento do contexto */onto*, para */page/onto* ou */data/onto*

```
RewriteCond %{HTTP:Accept} text/html [NC]
RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8080/page/onto/$1 [P]

RewriteCond %{HTTP:Accept} application/rdf+xml [NC]
RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8080/data/onto/$1 [P]

RewriteCond %{HTTP:Accept} application/x-turtle [NC]
RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8080/data/onto/$1?output=ttl [P]

RewriteCond %{HTTP:Accept}
!(application/rdf+xml|application/x-turtle|text/html) [NC]
RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8080/page/onto/$1 [P]
```

A configuração da camada HTTP está concluída, as configurações da camada de Aplicações será detalhada na seção seguinte

Configuração da camada de Aplicações

A camada de aplicações do ambiente restringe-se ao uso do programa Pubby, que possui como pré-requisito os programas Java SE 6 e Apache Tomcat.

Após a instalação dos programas, será necessário configurar o arquivo de configurações do Pubby, em formato Turtle, chamado *config.ttl*, para realizar as ações desejadas. As configurações principais são detalhadas no trecho abaixo:

```
conf:dataset [  
  
  conf:sparqlEndpoint <http://lodkem.egc.ufsc.br:8890/sparql>;  
  
  conf:datasetBase <http://lodkem.egc.ufsc.br/resource/>;  
  
  conf:webResourcePrefix "resource/";  
  
  conf:fixUnescapedCharacters "(),'!$&*+;=@";  
  
  ];
```

O trecho acima representa a primeira parte das regras, onde tem-se a definição do endereço do servidor Virtuoso na configuração *conf:sparqlEndpoint*, a URI utilizada para identificar os conceitos do mundo real, no caso */resource/*, é definida em conjunto pelas configurações *conf:datasetBase* e *conf:webResourcePrefix*.

De maneira complementar, as configurações referentes aos vocabulários e ontologias estão também presentes no arquivo de configuração como se fossem um novo *dataset*, no trecho detalhado abaixo:

```
conf:dataset [  
  conf:sparqlEndpoint <http://lodkem.egc.ufsc.br:8890/sparql>;
```

```
conf:sparqlDefaultGraph <http://lodkem.egc.ufsc.br>;
conf:datasetBase <http://lodkem.egc.ufsc.br/>;
conf:datasetURIPattern "(class|property|onto)/.*";
conf:fixUnescapedCharacters "(),!$&*+;=@";
];
```

O trecho acima representa a primeira parte das regras, onde tem-se a mesma definição do endereço do servidor Virtuoso na configuração *conf:sparqlEndpoint*, a URI utilizada para identificar os conceitos do mundo real é definida em conjunto pelas configurações *conf:datasetBase* e *conf:datasetURIPattern*, onde apenas as URIs *http://lodkem.egc.ufsc.br/class*, *http://lodkem.egc.ufsc.br/property* e *http://lodkem.egc.ufsc.br/onto* farão parte do conjunto de regras e não ficará abaixo do outro conjunto de dados definido, que tem como base a URI *http://lodkem.egc.ufsc.br/resource*.

Uma configuração esteve presente nos dois trechos citados acima, chamada *conf:fixUnescapedCharacters*, e se faz necessária para evitar problemas de codificação de caracteres em uma arquitetura onde há um servidor mediando as requisições, através da função *proxy* Apache HTTP Server, que procede no caso em questão.

Configuração da camada de Persistência

A simplicidade na preparação desta camada foi facilitada pela arquitetura aplicada, sendo que para a persistência dos dados foi necessário o uso do servidor Virtuoso nas funções de armazenamento e carregamento das triplas, apenas. Sua instalação e configuração não necessitou alterações, uma vez que os pacotes disponíveis no repositório do sistema operacional já o preparam para uso e operação.

O pacote *conductor* do produto é responsável pela interface via navegador para diversas das operações, como carregamento de RDF, alterações nos arquivos de configuração, entre outras.

O Virtuoso permite diversos ajustes para melhor desempenho e controle em ambientes de produção, através da edição de parâmetros presentes no arquivo *virtuoso.ini*. Tais referências serão abordadas na seção de otimizações deste trabalho.

Após a instalação e configuração preparada, neste momento foi possível a importação de triplas para testes, no quadro 11 observa-se o trecho importado, em formato RDF/XML:

Quadro 11 - Exemplo de triplas RDF de descrição da entidade DBPedia.

(continua)

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:dbpedia-owl="http://dbpedia.org/ontology/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:ns5="http://www.w3.org/ns/prov#">
  <rdf:Description
    rdf:about="http://lodkem.egc.ufsc.br/resource/DBPedia">
    <owl:sameAs rdf:resource="http://dbpedia.org/resource/DBPedia"/>
    <dbpedia-owl:wikiPageID
      rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">15428045</dbpedia-owl:wikiPageID>
    <dbpedia-owl:wikiPageRevisionID
      rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">257988581</dbpedia-owl:wikiPageRevisionID>
    <rdfs:label xml:lang="en">DBPedia</rdfs:label>
    <dbpedia-owl:wikiPageRedirects
      rdf:resource="http://dbpedia.org/resource/DBpedia"/>
```

Quadro 11 - Exemplo de triplas RDF de descrição da entidade DBPedia.

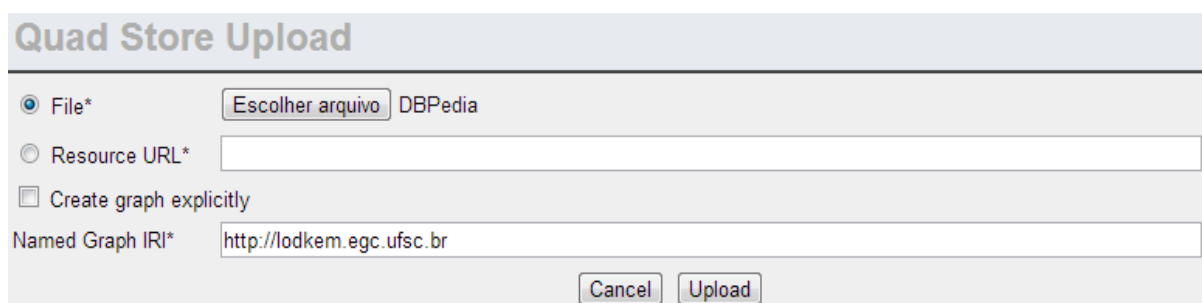
(continuação)

```
<foaf:isPrimaryTopicOf
rdf:resource="http://en.wikipedia.org/wiki/DBPedia" />
  <ns5:wasDerivedFrom
rdf:resource="http://en.wikipedia.org/wiki/DBPedia?oldid=257988581" />
  </rdf:Description>
</rdf:RDF>
```

Fonte:Autor

Provido do trecho abaixo, executou-se a importação através da função *Quad Store Upload* do Virtuoso, através da interface do pacote *conductor*, visto na figura 10:

Figura 10 - Interface de carregamento de triplas RDF no servidor Virtuoso OpenSource.



Fonte:Autor

Após a importação, será então possível acessar o recurso recém importado, definido como *http://lodkem.egc.ufsc.br/resource/DBPedia*. Para a visualização de resposta para seres humanos, em HTML, o teste foi realizado acessando a URI do recurso em um navegador, com o resultado presente na figura 11:

Figura 11 - Resposta gerada pelo Pubby para descrição de recurso em HTML.

← → ↻ lodkem.egc.ufsc.br/page/DBPedia ☆

DBPedia at lodKEM - EGC
<http://lodkem.egc.ufsc.br/resource/DBPedia>

Property	Value
?isPrimaryTopicOf	▪ < http://en.wikipedia.org/wiki/DBPedia >
?label	▪ DBPedia (en)
?sameAs	▪ < http://dbpedia.org/resource/DBPedia >
?wasDerivedFrom	▪ < http://en.wikipedia.org/wiki/DBPedia?oldid=257988581 >
?wikiPageID	▪ 15428045 ()
?wikiPageRedirects	▪ < http://dbpedia.org/resource/DBpedia >
?wikiPageRevisionID	▪ 257988581 ()

Metadata

Anon_0

< http://www.w3.org/1999/02/22-rdf-syntax-ns#type >	< http://purl.org/net/provenance/ns#DataItem >
< http://www.w3.org/1999/02/22-rdf-syntax-ns#type >	< http://www.w3.org/2004/03/trix/rdfg-1/Graph >
< http://xmlns.com/foaf/0.1/primaryTopic >	< http://lodkem.egc.ufsc.br/resource/DBPedia >
< http://xmlns.com/foaf/0.1/topic >	Anon_0
< http://www.ontologydesignpatterns.org/cp/owl/informationrealization.owl#realizes >	< http://lodkem.egc.ufsc.br/data/DBPedia >
< http://purl.org/net/provenance/ns#createdBy >	Anon_1 (more)

[expand all](#)

This page shows information obtained from the SPARQL endpoint at <http://lodkem.egc.ufsc.br:8890/sparql>.
[As Turtle](#) | [As RDF/XML](#) | [Browse in Disco](#) | [Browse in Tabulator](#) | [Browse in OpenLink Browser](#)

Fonte:Autor

Cabe notar que na figura 11 temos uma apresentação visual genérica gerada pelo Pubby, pontos de interrogação denotando os prefixos não mapeados, uma seção de metadados abaixo das triplas, além do conteúdo estar apresentado na língua inglesa.

É necessário também realizar testes para determinar se o ambiente também é capaz de realizar a negociação dos conteúdos em formatos apropriados para o processamento de computadores, no exemplo abaixo, temos a execução do comando:

```
fschroeder@lodkem:~$ curl -L -iH Accept:application/rdf+xml
http://lodkem.egc.ufsc.br/resource/DBPedia
```


O seguinte trecho inicial pode ser observado, indicando a estratégia de redirecionamento de URIs 303:

```
HTTP/1.1 303 See Other
Date: Tue, 29 Oct 2013 02:30:47 GMT
Server: Apache-Coyote/1.1
Vary: Accept,User-Agent,Accept-Encoding
Location: http://lodkem.egc.ufsc.br/data/DBPedia
Content-Type: text/plain
Content-Length: 91
```

A conexão procede para a nova URI <http://lodkem.egc.ufsc.br/data/DBPedia>, indicando sucesso na negociação do conteúdo e posterior entrega do conteúdo, sendo apenas a resposta HTTP mantida, pra efeitos de simplificação e legibilidade:

```
HTTP/1.1 200 OK
Date: Tue, 29 Oct 2013 02:30:47 GMT
Server: Apache-Coyote/1.1
Vary: Accept
Content-Type: application/rdf+xml
Transfer-Encoding: chunked
```

4.8 Customizações da Interface

Com a infraestrutura funcional, realizou-se um trabalho de customização da resposta HTML, afim de refletir de maneira básica as cores e logo do EGC para este projeto. Tal processo de apresentação da versão HTML dos dados é realizado pelo programa *Pubby*, que possui um conjunto de arquivos de estilo e estrutura, listados abaixo:

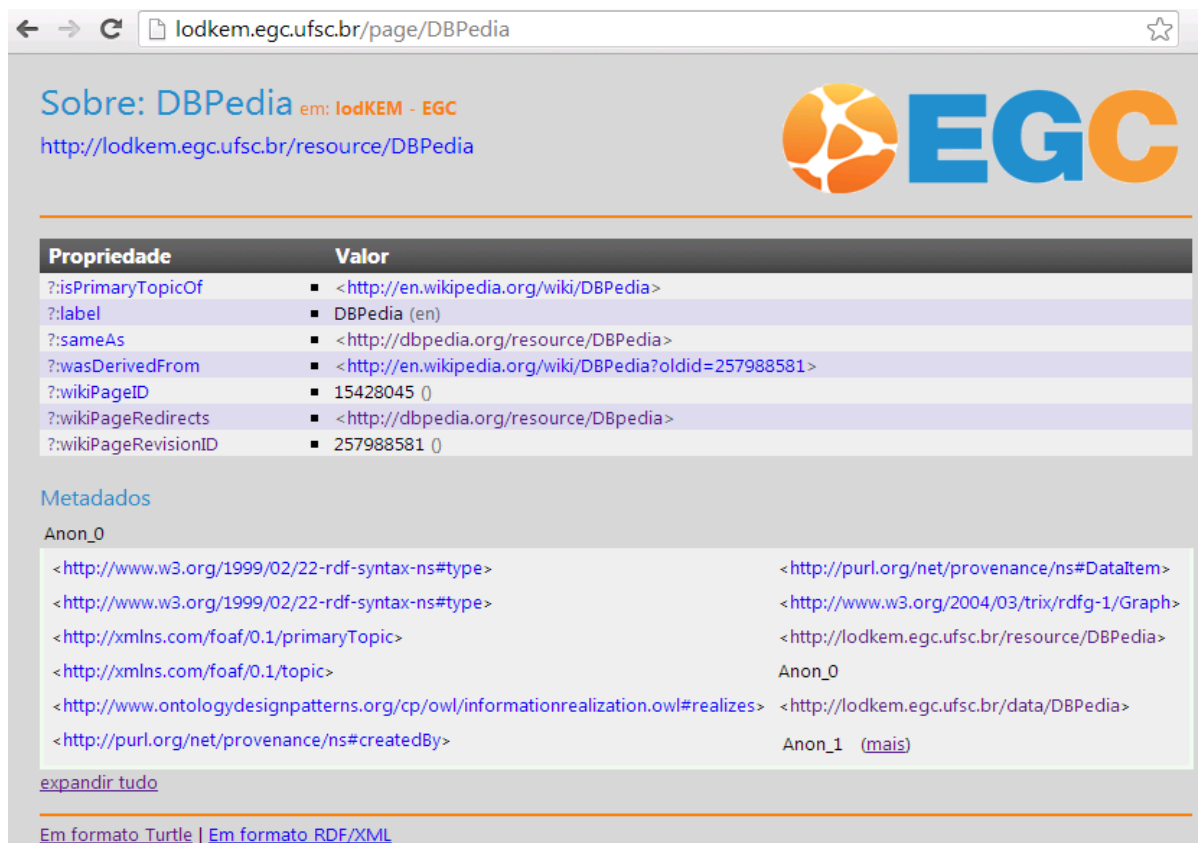
- static/style.css - Arquivo responsável pelo estilo CSS da página de resposta HTML, contendo informações como posicionamento dos blocos, cores, e fontes
- WEB-INF/templates/404.vm - Arquivo correspondente à página responsável quando não houver resultado válido para a consulta do usuário
- WEB-INF/templates/header.vm - Arquivo correspondente ao código dinâmico HTML gerado na parte superior da tela.
- WEB-INF/templates/page.vm - Arquivo correspondente ao código dinâmico HTML gerado no corpo central da tela
- WEB-INF/templates/proptable.vm - Arquivo correspondente ao código dinâmico HTML gerado na tabela com as propriedades e valores dos recursos, na parte central da tela
- WEB-INF/templates/metadatatable.vm - Arquivo correspondente ao código dinâmico HTML gerado na parte da descrição de proveniência dos dados (metadados), situado na parte inferior da tela, acima do rodapé
- WEB-INF/templates/footer.vm - Arquivo correspondente ao código dinâmico HTML gerado no rodapé da tela
- WEB-INF/templates/footer.vm - Arquivo correspondente à renderização dos arquivos anteriores na ordem correta na página

As alterações nos arquivos tiveram como objetivo incorporar as cores e logo do EGC, bem como a tradução dos textos presentes para português e simplificação das opções de formatos para download.

Foram utilizadas referências de imagem, fontes e estilo do atual *website* do departamento EGC³⁴ como referência para as customizações. Para efeito de simplificação, os detalhes da customização dos arquivos poderão ser consultados no Apêndice 2 - Arquivos alterados para a customização visual EGC, ao final deste trabalho.

Na figura 12 temos o resultado final das customizações da resposta HTML:

Figura 12 - Resposta HTML gerada pelo Pubby para descrição de recurso.



Sobre: DBPedia em: lodKEM - EGC
<http://lodkem.egc.ufsc.br/resource/DBPedia>

Propriedade	Valor
?:isPrimaryTopicOf	▪ < http://en.wikipedia.org/wiki/DBPedia >
?:label	▪ DBPedia (en)
?:sameAs	▪ < http://dbpedia.org/resource/DBPedia >
?:wasDerivedFrom	▪ < http://en.wikipedia.org/wiki/DBPedia?oldid=257988581 >
?:wikiPageID	▪ 15428045 ()
?:wikiPageRedirects	▪ < http://dbpedia.org/resource/DBpedia >
?:wikiPageRevisionID	▪ 257988581 ()

Metadados

Anon_0

< http://www.w3.org/1999/02/22-rdf-syntax-ns#type >	< http://purl.org/net/provenance/ns#DataItem >
< http://www.w3.org/1999/02/22-rdf-syntax-ns#type >	< http://www.w3.org/2004/03/trix/rdfg-1/Graph >
< http://xmlns.com/foaf/0.1/primaryTopic >	< http://lodkem.egc.ufsc.br/resource/DBPedia >
< http://xmlns.com/foaf/0.1/topic >	Anon_0
< http://www.ontologydesignpatterns.org/cp/owl/informationrealization.owl#realizes >	< http://lodkem.egc.ufsc.br/data/DBPedia >
< http://purl.org/net/provenance/ns#createdBy >	Anon_1 (mais)

[expandir tudo](#)

[Em formato Turtle](#) | [Em formato RDF/XML](#)

Fonte: Autor

³⁴ <http://egc.ufsc.br>

4.9 Otimizações e ajustes finais

Apesar dos dados já estarem disponíveis após a conclusão das etapas anteriores, foram necessários ajustes para melhor funcionamento do ambiente, em especial do aplicativo Pubby. A versão utilizada e mais recente do programa disponibilizada publicamente, não atendeu a duas questões:

1. Os prefixos definidos nos arquivos de configuração não eram apresentados nos resultados HTML, resultando no retorno do caracter '?' para cada prefixo, mesmo mapeado corretamente.
2. As consultas SPARQL do tipo DESCRIBE, realizadas pelo Pubby no ambiente Virtuoso não faziam uso de uma consulta DESCRIBE do tipo CBD (do inglês, Concise Bounded Description), retornando mais resultados que o desejado.

Para a resolução destas questões, entrou-se em contato com a equipe de desenvolvimento do Pubby e obteve-se uma versão ajustada do software, com os ajustes implementados da maneira esperada.

Além da extração dos novos conteúdos que fazem parte da instalação, houve a necessidade dos seguintes ajustes, no arquivo de configuração *config.ttl*, do Pubby:

- Adição do parametro *conf:queryPrefix "DEFINE sql:describe-mode \"CBD\""*; - Para que o Pubby execute as consultas no modo CBD suportado pelo Virtuoso
- Adição propriamente dita dos prefixos necessários no começo do arquivo, para verificação na interface. A lista completa de prefixos está

disponível no Apêndice 1 - Instalação e configuração das ferramentas no ambiente EGC, ao final deste trabalho

Além dos passos executados anteriormente, uma melhoria dependente da infraestrutura de produção do EGC, seria o ajuste dos atributos do Virtuoso, de acordo com as recomendações divulgadas em sua documentação³⁵.

³⁵ <http://www.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtConfigScale>

5 CONCLUSÃO

5.1 Considerações finais

O enaltecimento dos benefícios inerentes ao modelo Linked Data em dados abertos trouxe interesse e diversas iniciativas em âmbito global. Iniciativas em áreas distintas como o *Geonames*³⁶ na área geográfica, *data.gov*³⁷ do governo dos Estados Unidos, *dados.gov.br*³⁸ do governo Brasileiro, entre tantos outros, estão disponíveis atualmente.

Neste momento onde os desafios se voltam para a implantação propriamente dita e melhores práticas sobre como fazê-la, o presente trabalho se inseriu e alcançou os objetivos esperados, desde a definição dos elementos e fluxos possíveis de publicação presentes nas seções teóricas deste trabalho, até a aplicação propriamente dita em um cenário real do laboratório EGC, onde foram superados desafios de arquitetura e deficiências nas ferramentas atuais para, de fato, disponibilizar os dados abertos com suporte ao Linked Data da maneira apropriada.

5.2 Sugestões para futuros trabalhos

Ao curso da elaboração deste trabalho foi observado em diversas ocasiões pontos abertos ou em pesquisa no momento e oportunidades de continuidade

³⁶ <http://www.geonames.org>

³⁷ <http://www.data.gov>

³⁸ <http://dados.gov.br>

para trabalhos afins. De maneira sintética, eis uma lista dos tópicos específicos do domínio deste trabalho, onde é possível vislumbrar novos trabalhos:

- Trabalho comparativo de performance entre diversos servidores RDF, afim de ponderar as características de cada uma das opções estudadas e permitir melhor identificação das ferramentas ao propósito à que se destinam
- Estudo sobre as práticas para o controle e versionamento de triplas RDF, buscando determinar estratégias de gestão de conteúdo nas bases RDF, ao longo do tempo
- Implementação de estatísticas sobre os dados acessados em um ambiente Linked Data, em todos os formatos disponibilizados, com intuito de verificar técnicas de controle definição de padrões e tendências de acesso
- Elaboração de estudo sobre fluxo de ETL e carregamento de dados em servidores RDF, com intuito de avaliar estratégias e práticas que possam ser integradas com a infraestrutura do ambiente

6 REFERÊNCIAS BIBLIOGRÁFICAS

- Alexander, K. et al., 2009. Describing Linked Datasets On the Design and Usage of void , the “ Vocabulary Of Interlinked Datasets C. Bizer et al., eds. *Design*, 19. Available at:
http://events.linkeddata.org/ldow2009/papers/ldow2009_paper17.pdf.
- Alexander, K. (Talis) et al., 2011. Describing Linked Datasets with the Void Vocabulary. *W3C Interest Group Note 03 March 2011*. Available at:
<http://www.w3.org/TR/void> [Accessed September 17, 2013].
- Bizer, C., Lehmann, J., et al., 2009. DBpedia - A Crystallization Point for the Web of Data.
- Bizer, C., Cyganiak, R. & Heath, T., 2007. How to Publish Linked Data on the Web. Available at: <http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial>.
- Bizer, C. & Heath, T., 2011. *Linked Data: Evolving the Web into a Global Data Space*,
- Bizer, C., Heath, T. & Berners-Lee, T., 2009. Linked Data - The Story So Far.
- Cyganiak, R. et al., Semantic Sitemaps : Efficient and Flexible Access to Datasets on the Semantic Web.
- Cyganiak, R. & Wood, D., 2013. RDF 1.1 Concepts and Abstract Syntax. *W3C Last Call Working Draft 23 July 2013*. Available at: <http://www.w3.org/TR/rdf11-concepts> [Accessed July 30, 2013].
- Demter, J., Martin, M. & Lehmann, J., LODStats – An Extensible Framework for High-performance Dataset Analytics.
- DuCharme, B., 2013. *Learning SPARQL, Second Edition Querying and Updating with SPARQL 1.1*,
- Elmasri, R. & Navathe, S.B., 2004. *Fundamentals Of Database systems, Fourth Edition*,
- Harris, S., Seaborne, A. & Prud'hommeaux, E., 2010. SPARQL Query Language 1.1. *W3C Working Draft 26 January 2010*. Available at:
<http://www.w3.org/TR/2010/WD-sparql11-query-20100126> [Accessed August 3, 2013].
- Hutchinson, N.C. et al., 1999. Logical vs . Physical File System Backup Logical vs . Physical File System Backup. , p.12.
- Klyne, G., Carroll J., J. & McBride, B., 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C Recommendation 10 February*

2004. Available at: <http://www.w3.org/TR/rdf-concepts/> [Accessed July 27, 2013].

Langegger, A. & Wöß, W., 2009. RDFStats – An Extensible RDF Statistics Generator and Library.

Lee, D. & Galway, N.U.I., 2011. Open Data Overview.

Manola, F. & Miller, E., 2004. RDF Primer. Available at: <http://www.w3.org/TR/rdf-primer>.

Sauermann, L. & Cyganiak, R., 2008. Cool URIs for the Semantic Web. Available at: <http://www.w3.org/TR/cooluris>.

Stickler, P. (Nokia), 2005. CBD - Concise Bounded Description. *W3C Member Submission 3 June 2005*. Available at: <http://www.w3.org/Submission/CBD> [Accessed August 5, 2013].]

T. Berners-Lee, "Linked Data," W3C Design Issues, (2006), [Online]. Available: <http://www.w3.org/DesignIssues/LinkedData.html>.

Wikipedia contributors, "Serialization," Wikipedia, The Free Encyclopedia, <http://en.wikipedia.org/w/index.php?title=Serialization&oldid=583764952> [accessed June 8, 2013].

7 ANEXOS E APÊNDICES

Anexo 1 - Instalação e configuração das ferramentas no ambiente EGC

Apache HTTP Server

1. Instalação e configuração do Apache HTTP Server, através do comando:

```
fschroeder@lodkem:~$ sudo apt-get install apache2
```

2. Habilitação dos módulos necessários no apache:

```
fschroeder@lodkem:~$ sudo a2enmod proxy_http proxy rewrite headers
```

3. Reinicialização do apache, com os módulos carregados:

```
fschroeder@lodkem:~$ sudo service apache2 restart
```

4. Modificação do arquivo `/etc/apache2/sites-available/default`, com resultado final abaixo:

```
<VirtualHost *:80>
    ServerName lodkem.egc.ufsc.br
    ServerAdmin admin@lodkem.egc.ufsc.br
    DocumentRoot /var/www
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>
    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
```

```

    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews
+SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    # Possible values include: debug, info, notice, warn,
error, crit,
    # alert, emerg.
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    <IfModule mod_proxy.c>
# Disable forward proxy requests
ProxyRequests Off
# Allow requests from selected hosts or domains
<Proxy *>
    Order Allow,Deny
    Allow from all
</Proxy>
# Configure reverse proxy requests for RequisiteWeb
<IfModule mod_rewrite.c>
    RewriteEngine On
    #Este trecho de redirecionamento se faz necessario
por conta de limitacao do Pubby em se definir multiplos esquemas
de URIs que estejam fora dos contextos /page, /resource, /data
    #Redirecionamento do contexto /onto, para /page/onto
ou /data/onto
    RewriteCond %{HTTP:Accept} text/html [NC]
    RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8079/page/onto/$1 [P]
    RewriteCond %{HTTP:Accept} application/rdf\+xml [NC]
    RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8079/data/onto/$1 [P]
    RewriteCond %{HTTP:Accept} application/x\-turtle
[NC]
    RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8079/data/onto/$1?output=ttl [P]

```

```

RewriteCond %{HTTP:Accept}
!(application/rdf+xml|application/x-turtle|text/html) [NC]
RewriteRule ^/onto/(.*)
http://lodkem.egc.ufsc.br:8079/page/onto/$1 [P]
#Redirecionamento do contexto /class, para
/page/class ou /data/class
RewriteCond %{HTTP:Accept} text/html [NC]
RewriteRule ^/class/(.*)
http://lodkem.egc.ufsc.br:8079/page/class/$1 [P]
RewriteCond %{HTTP:Accept} application/rdf+xml [NC]
RewriteRule ^/class/(.*)
http://lodkem.egc.ufsc.br:8079/data/class/$1 [P]
RewriteCond %{HTTP:Accept} application/x-turtle
[NC]
RewriteRule ^/class/(.*)
http://lodkem.egc.ufsc.br:8079/data/class/$1?output=ttl [P]
RewriteCond %{HTTP:Accept}
!(application/rdf+xml|application/x-turtle|text/html) [NC]
RewriteRule ^/class/(.*)
http://lodkem.egc.ufsc.br:8079/page/class/$1 [P]
#Redirecionamento do contexto /property, para
/page/property ou /data/property
RewriteCond %{HTTP:Accept} text/html [NC]
RewriteRule ^/property/(.*)
http://lodkem.egc.ufsc.br:8079/page/property/$1 [P]
RewriteCond %{HTTP:Accept} application/rdf+xml [NC]
RewriteRule ^/property/(.*)
http://lodkem.egc.ufsc.br:8079/data/property/$1 [P]
RewriteCond %{HTTP:Accept} application/x-turtle
[NC]
RewriteRule ^/property/(.*)
http://lodkem.egc.ufsc.br:8079/data/property/$1?output=ttl [P]
RewriteCond %{HTTP:Accept}
!(application/rdf+xml|application/x-turtle|text/html) [NC]
RewriteRule ^/property/(.*)
http://lodkem.egc.ufsc.br:8079/page/property/$1 [P]
</IfModule>
#Configuracoes de proxy para as camadas de aplicacao
e persistencia
ProxyPass /sparql
http://lodkem.egc.ufsc.br:8890/sparql
ProxyPassReverse /sparql
http://lodkem.egc.ufsc.br:8890/sparql

```

```

        ProxyPass /resource
http://lodkem.egc.ufsc.br:8079/resource
        ProxyPassReverse /resource
http://lodkem.egc.ufsc.br:8079/resource
        ProxyPass /page http://lodkem.egc.ufsc.br:8079/page
        ProxyPassReverse /page
http://lodkem.egc.ufsc.br:8079/page
        ProxyPass /data http://lodkem.egc.ufsc.br:8079/data
        ProxyPassReverse /data
http://lodkem.egc.ufsc.br:8079/data
        ProxyPass /static
http://lodkem.egc.ufsc.br:8079/static
        ProxyPassReverse /static
http://lodkem.egc.ufsc.br:8079/static
    </IfModule>
    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Order deny,allow
        Deny from all
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>
</VirtualHost>

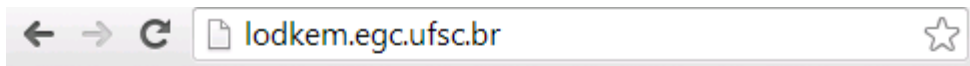
```

5. Nova reinicialização do Apache HTTP Server, para aplicação das configurações:

```
fschroeder@lodkem:~$ sudo service apache2 restart
```

6. Teste de inicialização, através do acesso de navegador, no endereço

<http://lodkem.egc.ufsc.br>.



It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Apache Tomcat e Java SE

1. Criação de diretório no servidor, para hospedagem dos instaladores

```
fschroeder@lodkem:~$ sudo mkdir /opt/instaladores
```

2. Download do aplicativo na versão 7.0.42, sendo o aplicativo do tipo *Core*, opção *tar.gz*, no momento presente através do seguinte comando:

```
fschroeder@lodkem:~$ sudo wget
http://archive.apache.org/dist/tomcat/tomcat-
7/v7.0.42/bin/apache-tomcat-7.0.42.tar.gz -O
/opt/instaladores/apache-tomcat-7.0.42.tar.gz
```

3. Extração do arquivo salvo no servidor, através dos seguintes comandos:

```
fschroeder@lodkem:~$ cd /opt/instaladores/
fschroeder@lodkem:~$ sudo tar -xvzf /opt/instaladores/apache-
tomcat-7.0.42.tar.gz
fschroeder@lodkem:~$ sudo mv apache-tomcat-7.0.42 ../
```

4. Download do Java SE 6 na pasta de instaladores do servidor, através do seguinte comando:

```
fschroeder@lodkem:~$ sudo wget
http://download.oracle.com/otn-pub/java/jdk/6u34-b04/jre-6u34-
linux-
x64.bin?AuthParam=1383308270_ca2c12f10f0a551805eb30b414f2dc8b -O
/opt/instaladores/jre-6u34-linux-x64.bin
```

5. Execução e extração em diretório, do Java SE 6:

```

fschroeder@lodkem:~$ cd /opt/instaladores
fschroeder@lodkem:~$ sudo ./jre-6u34-linux-x64.bin
fschroeder@lodkem:~$ sudo mv /opt/instaladores$ sudo mv
jre1.6.0_25/ /opt/

```

6. Criação de arquivo de serviço para subida e parada do Tomcat, no sistema operacional, através da criação do arquivo `/etc/init.d/tomcat`, com permissões de execução, listado abaixo:

```

#!/bin/sh
#
# Startup script for Tomcat
#
# chkconfig: 345 82 20
# description: Tomcat is a servlet runner

JAVA_HOME=/opt/jre1.6.0_25/
TOMCAT_HOME=/opt/apache-tomcat-7.0.42
XMLFILE=$TOMCAT_HOME/conf/server.xml
JAVA_OPTS="-server      -Xms1536m      -Xmx1536m      -Xmn384m      -
XX:+UseParallelGC"
CATALINA_OPTS=""
export JAVA_HOME TOMCAT_HOME JAVA_OPTS CATALINA_OPTS

# See how we were called.
case "$1" in
  start)
    cd $TOMCAT_HOME
    ./bin/startup.sh -config $XMLFILE
    ;;
  stop)
    cd $TOMCAT_HOME
    ./bin/shutdown.sh -config $XMLFILE
    ;;
  restart)
    $0 stop

```

```

sleep 3
$0 start
;;
*)
echo "Usage: $0 {start|stop|restart}"
exit 1
esac

exit 0

```

7. Teste de inicialização do serviço:

```
fschroeder@lodkem:~$ sudo /etc/init.d/tomcat start
```

8. Verificação do serviço tomcat, através do acesso de navegador, no endereço <http://lodkem.egc.ufsc.br:8080>:

Pubby

1. Download do aplicativo final na pasta de instaladores, conforme comando abaixo:


```
fschroeder@lodkem:~$ sudo wget
http://boris.villazon.terrazas.name/data/pubby-0.3.3-
lodkem.ufsc.br.zip -O /opt/instaladores/pubby-0.3.3-
lodkem.ufsc.br.zip
```

2. Extração dos conteúdos do aplicativo, através dos comandos:

```
fschroeder@lodkem:~$ sudo unzip /opt/instaladores/pubby-0.3.3-
lodkem.ufsc.br.zip

fschroeder@lodkem:~$ sudo mv pubby-0.3.3-lodkem.ufsc.br
/opt/instaladores/

fschroeder@lodkem:~$ sudo cd /opt/instaladores/pubby-0.3.3-
lodkem.ufsc.br
```

3. Renomeação do diretório de aplicação-padrão do Tomcat e colocação da aplicação Pubby:

```
fschroeder@lodkem:~$ sudo mv /opt/apache-tomcat-
7.0.42/webapps/ROOT /opt/apache-tomcat-
7.0.42/webapps/ROOT_Original

fschroeder@lodkem:~$ sudo mkdir /opt/apache-tomcat-
7.0.42/webapps/ROOT

fschroeder@lodkem:~$ sudo cp -pR /opt/instaladores/pubby-0.3.3-
lodkem.ufsc.br/* /opt/apache-tomcat-7.0.42/webapps/ROOT
```

4. Alteração do arquivo de configuração do Pubby, no caso localizado em /opt/apache-tomcat-7.0.42/webapps/ROOT/WEB-INF/config.ttl, para que fique como abaixo :

```
# Pubby Example Configuration
#
# This configuration connects to the DBpedia SPARQL endpoint and
# re-publishes on your local machine, with dereferenceable
# localhost URIs.
#
# This assumes you already have a servlet container running
# on your machine at http://localhost:8080/ .
#
# Install Pubby as the root webapp of your servlet container,
# and make sure the config-file parameter in Pubby's web.xml
```

```

# points to this configuration file.
#
# Then browse to http://localhost:8080/ .

# Prefix declarations to be used in RDF output
@prefix conf:
<http://richard.cyganiak.de/2007/pubby/config.rdf#> .
@prefix meta: <http://example.org/metadata#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix p: <http://dbpedia.org/property/> .
@prefix mpeg7-ext:
<http://webenemasuno.linkeddata.es/ontology/MPEG7/> .
@prefix mpeg7: <http://metadata.net/mpeg7/mpeg7.owl#> .
@prefix yago: <http://localhost:8080/class/yago/> .
@prefix units: <http://dbpedia.org/units/> .
@prefix geonames: <http://www.geonames.org/ontology#> .
@prefix prv: <http://purl.org/net/provenance/ns#> .
@prefix prvTypes: <http://purl.org/net/provenance/types#> .
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix ir:
<http://www.ontologydesignpatterns.org/cp/owl/informationrealization.owl#> .
@prefix opmopviajero:
<http://webenemasuno.linkeddata.es/ontology/OPMO/> .
@prefix opmo: <http://openprovenance.org/model/opmo#> .
@prefix scovo: <http://purl.org/NET/scovo#> .
@prefix sioc: <http://rdfs.org/sioc/ns#> .
@prefix aemetprop:
<http://aemet.linkeddata.es/ontology/property/> .

```

```

@prefix geonto: <http://geo.linkeddata.es/ontology/> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix otalex: <http://otalex.linkeddata.es/resource/> .
@prefix
                                phenomenontology:
<http://phenomenontology.linkeddata.es/ontology/> .
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix otaprop: <http://otalex.linkeddata.es/property/> .
@prefix          sdmx-dimension:          <http://purl.org/linked-
data/sdmx/2009/dimension#> .

```

```

# Server configuration section

```

```

<> a conf:Configuration;
    # Project name for display in page titles
    conf:projectName "LodKEM - EGC";
    # Homepage with description of the project for the link in
the page header
    conf:projectHomepage <http://lodkem.egc.ufsc.br/>;
    # The Pubby root, where the webapp is running inside the
servlet container.
    conf:webBase <http://lodkem.egc.ufsc.br/>;
    # URL of an RDF file whose prefix mapping is to be used by
the
    # server; defaults to <>, which is *this* file.
    conf:usePrefixesFrom <>;
    # If labels and descriptions are available in multiple
languages,
    # prefer this one.
    conf:defaultLanguage "en";
    # When the homepage of the server is accessed, this resource
will
    # be shown.
    conf:indexResource
<http://lodkem.egc.ufsc.br/resource/DBpedia>;

```

```

# Dataset configuration section #1 (for DBpedia resources)

```

```

#

```

```

# URIs in the SPARQL endpoint: http://dbpedia.org/resource/*

```

```

# URIs on the Web:          http://localhost:8080/resource/*

```

```

conf:dataset [
    # SPARQL endpoint URL of the dataset
    conf:sparqlEndpoint
<http://lodkem.egc.ufsc.br:8890/sparql>;
    # Default graph name to query (not necessary for most
endpoints)
    #
                                conf:sparqlDefaultGraph
<http://webenemasuno.linkeddata.es/dataset/viajero/IPTCS>;
    # Common URI prefix of all resource URIs in the SPARQL
dataset
    conf:datasetBase <http://lodkem.egc.ufsc.br/resource/>;
    # Will be appended to the conf:webBase to form the
public
    # resource URIs; if not present, defaults to ""
    conf:webResourcePrefix "resource/";
    #conf:datasetURIPattern "resource/.*";
    # Fixes an issue with the server running behind an
Apache proxy;
    # can be ignored otherwise
    conf:fixUnescapedCharacters "(),!$&*+;=@";

    # include metadata
    #conf:metadataTemplate "metadata.ttl";

    # configure your metadata here
    # Use properties with the meta: prefix where the
property name
    # corresponds to the placeholder URIs in metadata.ttl
that begin
    # with about:metadata:metadata:
    # Examples for such properties are:
    #meta:pubbyUser
<http://aemet.linkeddata.es/resource/Organizaci%C3%B3n/UPM>;
    #meta:pubbyOperator <http://aemet.linkeddata.es/>;
    #meta:endpointUser <http://aemet.linkeddata.es/sparql>;
    #meta:endpointOperator <URI of the service provider who
operates the SPARQL endpoint>;
    #meta:graph
<http://webenemasuno.linkeddata.es/dataset/ngce>;
    #conf:addSameAsStatements "true";
    conf:queryPrefix "DEFINE sql:describe-mode \"CBD\"";

```

```

];

# Dataset configuration section #2 (for DBpedia classes and
properties)
#
# URIs in the SPARQL endpoint: http://dbpedia.org/class/*
#                               http://dbpedia.org/property/*
# URIs on the Web:               http://localhost:8080/class/*
#                               http://localhost:8080/property/*

  conf:dataset [
    conf:sparqlEndpoint
<http://lodkem.egc.ufsc.br:8890/sparql>;
    conf:sparqlDefaultGraph <http://lodkem.egc.ufsc.br>;
    conf:datasetBase <http://lodkem.egc.ufsc.br/>;
    # Dataset URIs are mapped only if the part after the
    # conf:webBase matches this regular expression
    conf:datasetURIPattern "(class|property|onto)/.*";
    conf:fixUnescapedCharacters "(),!$&*+;=@";
  ];
.

```

5. Reinicialização do aplicativo Tomcat, para aplicação das mudanças:

```
fschroeder@lodkem:~$ sudo /etc/init.d/tomcat restart
```

OpenLink Virtuoso OpenSource

1. Instalação padrão do aplicativo via repositório de pacotes ubuntu:

```
fschroeder@lodkem:~$ sudo apt-get install virtuoso-opensource
```

2. Acesso ao pacote de gerenciamento *conductor*, através do endereço <http://lodkem.egc.ufsc.br:8090/conductor>, para verificação:



Anexo 2 - Arquivos alterados para a customização visual EGC

/opt/apache-tomcat-7.0.42/webapps/ROOT/static/style.css:

```
html { margin: 0; padding: 0; }
body { font-family: "Segoe UI", Arial, Helvetica, sans-serif;
font-size: 80%; margin: 0; padding: 1.2em 2em; }
#rdficon { float: right; position: relative; top: -28px; }
#header { border-bottom: 2px solid #F78210; margin: 0 0 1.2em;
padding: 0 0 3em; }
#footer { border-top: 2px solid #F78210; margin: 1.2em 0 0;
padding: 0.3em 0 0; }
#homelink { display: inline; }
#homelink, #homelink a { color: #F78210; }
#homelink a { font-weight: bold; text-decoration: none; }
```

```

#homelink a:hover { color: red; text-decoration: underline; }
h1 { display: inline; font-weight: normal; font-size: 200%;
margin: 0; text-align: left; }
h2 { font-weight: normal; font-size: 124%; margin: 1.2em 0
0.2em; }
.page-resource-uri { font-size: 124%; margin: 0.2em 0; }
.page-resource-uri a { color: blue; text-decoration: none; }
.page-resource-uri a:hover { color: red; text-decoration:
underline; }
a.sparql-uri { color: black; text-decoration: none; }
a.sparql-uri:hover { color: red; text-decoration: underline; }
img { border: none; }
#projectLogo { float: right; }
table.description { border-collapse: collapse; clear: left;
font-size: 100%; margin: 0 0 1em; width: 100%; }
table.description th { height:25px; background:
url(/static/mainnavBg.png) repeat-x top #1a191a; text-align:
left; color:#fff; font-size: 15px;}
table.description td, table.description th { line-height: 1.2em;
padding: 0.2em 0.4em; vertical-align: top; }
table.description ul { margin: 0; padding-left: 0em; }
table.description li { list-style-type: square; }
.uri { white-space: nowrap; }
.uri a, a.uri { text-decoration: none; }
.unbound { color: #888; }
table.description a small, .metadata-table a small { font-size:
100%; color: #55a; }
table.description small, .metadata-table a small { font-size:
100%; color: #666; }
table.description .property { white-space: nowrap; }
h1, h2 { color: #208ccc; }
body { background: #d6d6d6; }
table.description .odd td { background: #efefef; }
table.description .even td { background: #DCDCF1; }
.image { background: white; float: left; margin: 0 1.5em 1.5em
0; padding: 2px; }
a.expander { text-decoration: none; }
.metadata-label {
    font-size: 100%;
    background: #f0fcf0;

```

```

        padding: 3px;
    }
    .metadata-table {
        font-size: 100%;
        border-left: 3px solid #f0fcf0;
        border-bottom: 3px solid #f0fcf0;
        border-right: 3px solid #f0fcf0;
        background: #efefef;
        border-top: 0px solid none;
        margin: 0px;
    }
    .metadata-table td {
        padding: 3px;
    }
}

```

/opt/apache-tomcat-7.0.42/webapps/ROOT/WEB-INF/templates/page.vm:

```

#parse("header.vm")

#if ($image)
    <div class="image"></div>
#end

#if ($comment)
    <p>$comment</p>
#end

#if (!$properties.isEmpty())
#parse("proptable.vm")
#else
    <p>No further information is available.</p>
#end

#if ($metadata)
    <a name="meta"></a>
    <h2>Metadados</h2>
    <div id="metadata-tables">

```



```

        #parse("metadatatable.vm")
    </div>
    <a href="#meta" onclick="showAllMetadata('metadatatables')">expandir tudo</a>
#end

```

```
#parse("footer.vm")
```

/opt/apache-tomcat-7.0.42/webapps/ROOT/WEB-INF/templates/header.vm:

```

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>${title} | ${project_name}</title>
    #if ($rdf_link)
      <link rel="alternate" type="application/rdf+xml" href="$rdf_link" title="RDF" />
    #end
    <link rel="stylesheet" type="text/css" href="${server_base}static/style.css" />
    <script type="text/javascript" src="${server_base}static/script.js"></script>
  </head>
  <body onLoad="init();" >
    <div id="header">
      <div>
        <h1 id="title">Sobre: ${title}</h1>
        <div id="homelink">em: <a href="$project_link">${project_name}</a></div>
        <div id="projectLogo"></div>
      </div>
      #if ($uri)
        <div class="page-resource-uri"><a href="$uri">${uri}</a></div>
      #end
    </div>

    #if ($back_uri)

```

```

        <p><a href="$back_uri">Back to $back_label</a></p>
#end

/opt/apache-tomcat-7.0.42/webapps/ROOT/WEB-INF/templates/proptable.vm:

    <table class="description">
        <tr><th width="25%">Propriedade</th><th>Valor</th></tr>
#foreach ($property in $properties)
#if ($velocityCount % 2 == 0)
#set ($odd_even = "even")
#else
#set ($odd_even = "odd")
#end
        <tr class="$odd_even">
            <td class="property">
#if ($property.isInverse())
                <small>is</small>
#end
#if ($property.hasPrefix())
                <a
                    class="uri"
                    href="$property.URI"
                    title="$property.URI"><small>$property.Prefix:</small>$property.
                    LocalName</a>
#else
                <a
                    class="uri"
                    href="$property.URI"
                    title="$property.URI"><small>?:</small>$property.LocalName</a>
#end
#if ($property.isInverse())
                <small>of</small>
#end
            </td>
            <td>
                <ul>
#foreach ($value in $property.Values)
                    <li>
#if ($value.Node.isURI())
#if ($value.hasPrefix())
                            <a
                                class="uri"
                                href="$value.Node.URI"
                                title="$value.Node.URI"><small>$value.Prefix:</small>$value.Loca
                                lName</a>

```

```

#else
            <small>&lt;</small><a
                                class="uri"
href="$value.Node.URI">$value.Node.URI</a><small>&gt;</small>
#end
#elseif ($value.Node.isBlank())
            <span
class="blank">_: $value.Node.BlankNodeLabel</span>
#elseif ($value.Node.isLiteral())
            <span
class="literal">$value.Node.LiteralLexicalForm
#if ($value.DatatypeLabel)
            <small> ($value.DatatypeLabel)</small>
#end
#if ($value.Node.LiteralLanguage != "")
            <small> ($value.Node.LiteralLanguage)</small>
#end
            </span>
#end
        </li>
#end
#if ($property.BlankNodeCount > 0)
#if ($property.BlankNodeCount == 1)
#set ($text = "1 anonymous resource")
#elseif ($property.BlankNodeCount > 1)
#set ($text = "$property.BlankNodeCount anonymous resources")
#end
#if ($property.PathPageURL)
        <li>[<a href="$property.PathPageURL">$text</a>]</li>
#else
        <li>[$text]</li>
#end
#end
    </ul>
</td>
</tr>
#end
</table>

```

/opt/apache-tomcat-7.0.42/webapps/ROOT/WEB-

INF/templates/metadatatable.vm:

```
#set($bnidCount = 0)
#set($idadd = 0)

#macro (prettyNode $node)
    #set ($foundSth = false)

    #if ($node.asNode().isURI())
        #foreach ($entry in $prefixes)
            #if
($node.getURI().startsWith($entry.getValue()))
                <a                                class="uri"
href="$node.getURI()"
title="$node.getURI()"><small>$entry.getKey()</small>:$node.getU
RI().substring($entry.getValue().length())</a>
                    #set ($foundSth = true)
            #end
        #end
    #end

    #elseif ($node.asNode().isBlank())
        #if (!$blankNodesMap.containsKey($node))
            #set ($bnid = "Anon_{$bnidCount}")
            #set ($test =
$blankNodesMap.put($node,$bnid))
            #set ($bnidCount = $bnidCount + 1)
        #end
        #set ($bnidP = $blankNodesMap.get($node))
        $bnidP
        #set ($foundSth = true)
    #elseif ($node.asNode().isLiteral())

    $node.asNode().getLiteralValue().toString().replace("<", "&lt;").
replace(">", "&gt;")
        #set ($foundSth = true)
    #end
#end
```

```

        #if (!$foundSth)
            <small>&lt;</small><a class="uri"
href="$node.toString()">$node.toString()</a><small>&gt;</small>
        #end
    #end

#macro( listProperties $name $metadataPropList $style)
    #set ($idadd = $idadd + 1)
    <div class="metadata-label">#prettyNode ($name) &nbsp;

    #if ($metadataPropList.size() > 0 && $style != "block")
        <a name="anchor_$idadd"></a>
        (<a href="#anchor_$idadd"
onclick="document.getElementById('ele_$idadd').style.display =
'block';">mais</a>)
    #end
    </div>

    <table id="ele_$idadd" class="metadata-table"
style="display: $style">
        #foreach ($statement in $metadataPropList)
            <tr>
                <td valign="top" class="metadata-
property">#prettyNode ($statement.getPredicate())</td>
                <td class="metadata-object">
                    #set ($object =
$statement.getObject())
                    #if ($object.isResource() &&
$object.listProperties().toList().size() > 0 &&
!$statement.getSubject().equals($statement.getObject()))
                        #listProperties ($object
$object.listProperties().toList() 'none')
                    #else
                        #prettyNode
($statement.getObject())
                    #end
                </td>
            </tr>
        #end
    </table>

```

```
#end

## Replaced the commented line by the following one because the
## RDF graph we want to talk about is a specific representation
## of the data identified by the $rdf_link URI.
##                                     Olaf, May 28, 2010
## #listProperties ($rdf_link $metadata 'block')
## #listProperties ('' $metadata 'block')
#listProperties ($metadata $metadata.listProperties().toList()
'block')
```

/opt/apache-tomcat-7.0.42/webapps/ROOT/WEB-INF/templates/footer.vm:

```
#set ($has_text = false)
    <div id="footer">

#if ($rdf_link)
    <a href="$rdf_link?output=ttl">Em formato Turtle</a> |
    <a href="$rdf_link?output=xml">Em formato RDF/XML</a>
#end
    </div>
</body>
</html>
```

Anexo 3 - Artigo

Infraestrutura de dados abertos com suporte ao Linked Data

Filipe N. Schroeder¹

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

fns@inf.ufsc.br

Abstract. *The practice of sharing open data is a global trend on governments, such as the Brazilian's through dados.gov.br portal, and also on the private initiative. The Linked Data usage encouraged by the Web's creator, Tim Berners Lee, is an alternative based on semantic web experiences and works as an extension to the traditional web. This paper focuses on the definition of strategies and actual practices for the software infrastructure required in an environment of open data with Linked Data support, in a general context and posterior application on a real environment to the laboratory of Knowledge Engineering from the Universidade Federal de Santa Catarina.*

Resumo. *A prática de compartilhar dados abertos é uma tendência mundial nas esferas governamentais, como exemplo do governo brasileiro através do portal dados.gov.br, e inclusive na iniciativa privada. O uso de Linked Data proposto pelo criador da Web, Tim Berners Lee, é uma alternativa baseada em experiências com web semântica e funciona como extensão à web tradicional. Esse artigo tem como ênfase a definição de estratégias e práticas para a infraestrutura de software necessária em um ambiente de dados abertos com suporte ao Linked Data em um contexto geral, e posterior aplicação prática para o laboratório de Engenharia do Conhecimento da Universidade Federal de Santa Catarina.*

1. Introdução

A disponibilização de dados abertos com uso das práticas do Linked Data (Dados Ligados, em tradução livre para o português) tem crescido notavelmente nos últimos anos, de acordo com mapeamentos feitos pelo projeto Linking Open Data³⁹. Através da prática do Linked Data, os dados interligados na web podem ser vistos como uma grande base mundial de dados, interpretados por máquinas e reutilizados de maneira a maximizar o potencial desse conjunto.

A capacidade de, através de uma consulta a esse ambiente, identificar dados de diferentes fontes e agregá-los de maneira dinâmica e possivelmente automatizada é um aspecto popular desse paradigma.

Inicialmente, o objetivo principal do projeto linked data era disponibilizar grandes quantidades de dados e identificar as melhores práticas para fazê-lo. Atualmente, há também o foco em outros fatores como a qualidade dos dados, usabilidade, confiabilidade da infraestrutura, desempenho, entre outros. Nesse contexto, o planejamento e uso apropriado da

³⁹ Disponível em:
<http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

infraestrutura tornam-se importantes fatores na confiabilidade, desempenho e principalmente na futura viabilidade deste modelo em larga escala e crescente complexidade.

Para que se faça uso de dados abertos com princípios do Linked Data, é preciso entender os fundamentos básicos, para então direcionar-se aos elementos necessários para o planejamento e construção de tal ambiente. Neste capítulo serão abordados conceitos e termos que servirão de ponto de partida para a aplicação.

2. Dados Abertos

De acordo com a Open Knowledge Foundation (tradução disponível em <http://dados.gov.br/dados-abertos>): “dados são abertos quando qualquer pessoa pode livremente usá-los, reutilizá-los e redistribuí-los, estando sujeitos a, no máximo, a exigência de creditar a sua autoria e compartilhar pela mesma licença”.

Segundo (Lee & Galway 2011), há uma ênfase no uso de dados abertos por autoridades públicas, denominados de dados abertos governamentais, que se referem aos dados produzidos ou encomendados pelo governo. Ainda de acordo com Linking Open Data Cloud, a área governamental é responsável pelo maior número de dados em linked data no formato de triplas atualmente. Como pode ser visto na figura 1:

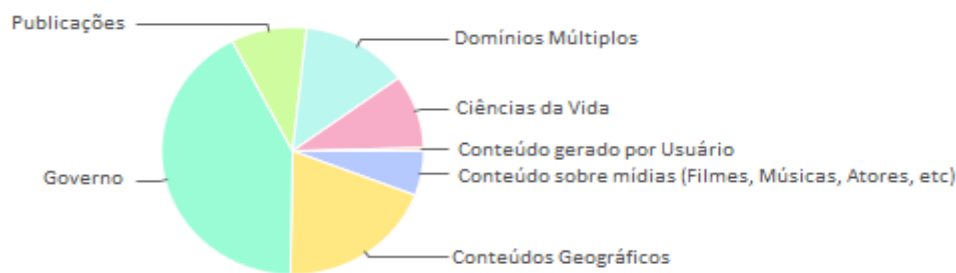


Figura 1. Diagrama de distribuição de triplas por domínio em 2011. Fonte: <http://www4.wiwiss.fu-berlin.de/locloud/state>

Na figura 1, é possível perceber que os dados relacionados à área governamental representam quase a metade dos dados, o que sustenta o fato de que os governos estão investindo nesta prática e, portanto são valiosas fontes de dados.

3. Web Semântica

Tim Berners Lee, o inventor da World Wide Web foi quem conceituou o termo Web Semântica globalmente, e o definiu como: "(...) uma web de dados que podem ser processados direta e indiretamente por máquinas."

O termo web de dados definido anteriormente pode ser colocado em contraste com o termo web de documentos ou ainda web tradicional ou web clássica, que representa a web no seu primeiro momento, onde os dados não eram estruturados e nem projetados para o uso das máquinas na inferência de significado, tal papel caberia aos usuários humanos.

4. Linked Data

A idéia do Linked Data é utilizar a arquitetura já existente na web clássica, ou World Wide Web, na tarefa de compartilhar dados estruturados em escala global. Para entender tais princípios é necessário entender a arquitetura da web clássica de documentos.

A web de documentos é constituída de um pequeno conjunto de padrões: URIs (do inglês, Uniform Resource Identifier), como mecanismo de identificação único globalmente, o HTTP (do inglês, Hypertext Markup Language), como formato de conteúdo global. Além disso, a web é construída com a idéia de colocar hiperlinks entre documentos que podem residir em servidores web diferentes (Bizer & Heath 2011).

O Linked Data é construído diretamente na arquitetura da web clássica, e a aplica na tarefa de compartilhar dados em escopo global.

Para que os dados estejam no formato Linked Data, Berners-Lee (2006) definiu as seguintes regras:

1. Usar URIs como nomes para as coisas.
2. Usar HTTP URIs, assim pessoas podem consultar esses nomes.
3. Quando alguém consultar um URI, prover informações úteis, usando padrões (RDF, SPARQL).
4. Incluir links para outras URIs, então poderá se descobrir mais coisas.

As seções subseqüentes definirão os conceitos presentes nos princípios acima.

4.1. URI

Um URI (do inglês, Uniform Resource Identifier) é um conjunto de caracteres compacto, utilizado para identificar um recurso físico ou abstrato .

Seu propósito é identificar itens na rede mundial de computadores (internet), de forma que possam ser acessados através de protocolos específicos, como o protocolo HTTP, como definida na segunda regra de Linked Data.

Na Web Semântica, os URIs representam não apenas documentos, mas também objetos do mundo real como uma árvore ou pessoa e até conceitos abstratos, tais representações são chamadas coisas ou objetos do mundo real.

Portanto, quando do acesso a um URI, máquinas devem receber em formato RDF e as pessoas, uma representação legível, como HTML, através do protocolo de transferência padrão da web, o HTTP.

A figura 2 mostra as diferentes representações de um URI, no contexto de Linked Data.

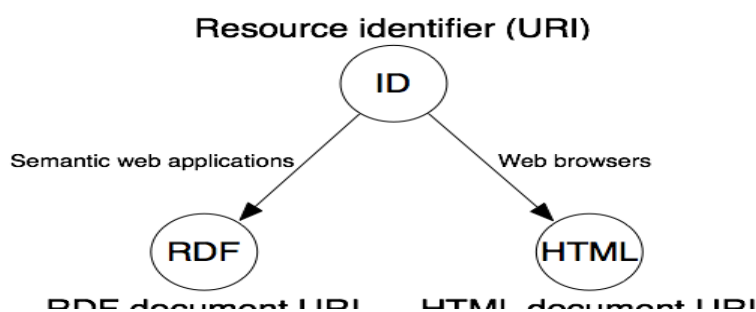


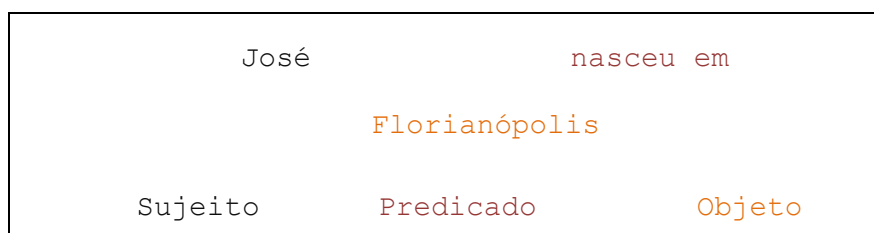
Figura 2. Fluxo de acesso ao conteúdo. Fonte: <http://www.w3.org/TR/cooluris>

4.2. RDF

O RDF (do inglês, Resource Description Framework) é um modelo de dados utilizado para facilitar a troca de informações na web e faz parte de um dos padrões de representação de dados para o Linked Data, análogo ao papel do HTML na web clássica.

De acordo com (Cyganiak & Wood 2013), os dados em RDF são representados por triplas ou afirmações, na forma: sujeito, predicado, objeto. Esta representação faz alusão à formação de frases, conforme definido no quadro 1:

Quadro 1. Modelo de afirmações RDF. Fonte: Autor



É imprescindível diferenciar o modelo RDF dos formatos de representação ou serialização de RDF, o primeiro é de fato o que a especificação e termo RDF se propõe, uma especificação de modelo de representação, o segundo se materializa no uso que se faz do modelo RDF na prática, através de formatos como o RDF/XML, N-Triples, entre outros, chamados de formatos de serialização de RDF.

Para a publicação na web, dados em RDF podem ser serializados em diferentes formatos, os dois mais comuns são o RDF/XML e RDFa (Bizer & Heath 2011).

4.3. SPARQL

Para que seja possível que se obtenha de maneira automatizada, a semântica presente nos dados através do modelo RDF, uma linguagem de consulta foi criada, chamada SPARQL (do inglês, SPARQL Protocol and RDF Query Language). Tal linguagem também possui definição de protocolo e consta como recomendação pela instituição W3C⁴⁰.

Dentre as funcionalidades do SPARQL, temos a capacidade de consultar padrões de grafos opcionais e obrigatórios e suas conjunções e disjunções, além de prover testes de

⁴⁰ Disponível em: <http://www.w3.org/2005/10/Process-20051014/tr.html#RecsW3C>

valores e limitações de consultas por grafo RDF fonte, os resultados de consultas SPARQL podem ser conjuntos de grafos RDF .

De acordo com sua especificação definida por (Harris et al. 2010), a linguagem SPARQL suporta quatro formas de operação: SELECT, CONSTRUCT, ASK e DESCRIBE, com os seguintes propósitos:

- SELECT - Retorna todos, ou um subconjunto, de todas as variáveis que correspondem à uma correspondência de resultado de uma consulta
- CONSTRUCT - Retorna um grafo construído através da substituição de variáveis em um conjunto de padrões de triplas
- ASK - Retorna um valor booleano indicando se uma consulta é correspondida ou não
- DESCRIBE - Retorna um grafo RDF que descreve os recursos encontrados, esse dado retornado não é determinado pela consulta SPARQL, onde o cliente precisaria conhecer a estrutura do RDF na fonte de dados, mas ao invés disso, é determinado pelo processador de consultas SPARQL (servidor).

5. Publicação de Linked Data

O objetivo do Linked Data é facilitar o compartilhamento de dados estruturados pelas pessoas e gerados por computadores, como hoje é feito com os documentos, basicamente publicando os dados no modelo RDF e utilizando links RDF para interligar os dados de diferentes fontes. Para tal objetivo, é necessário definir como será a disponibilização do ambiente, incluindo formatos, padrões, tipos de softwares, entre outros fatores definidos por (Bizer et al. 2007; Bizer & Heath 2011).

5.1 Design e forma dos URIs

Um elemento importante antes mesmo da disponibilização dos dados é a definição do formato dos seus identificadores globais, os URIs. De acordo com (Bizer et al. 2007; Bizer & Heath 2011), os padrões comuns de URIs são os detalhados a seguir.

Separação via contexto

É o padrão utilizado pela DBpedia , as URIs são separadas pelo contexto e tipo de dado retornado, no exemplo, uma representação do país Alemanha:

Tabela 1. Representação de URIs por contexto. Fonte: Autor

URI	Dado retornado
http://resource.exemplo.org/alemanha	Conceito do mundo real
http://data.exemplo.org/alemanha	Representação para computadores, da coisa
http://page.exemplo.org/alemanha	Representação para seres humanos, da coisa

Tal representação pode causar confusão ao desenvolvedor inexperiente ao Linked Data, já que as URIs com as representações são pouco diferentes da URI do conceito do mundo real (Bizer et al. 2007; Bizer & Heath 2011).

Separação via subdomínio

Neste padrão, o uso de subdomínios pode ajudar na separação dos servidores responsáveis para cada um dos tipos de representação. (exemplo: servidor web para as páginas HTML e id e base de dados RDF para os dados em RDF).

O mesmo exemplo de representação do país Alemanha, neste modo de separação, pode ser visto na tabela 2:

Tabela 2. Representação de URIs por subdomínio. Fonte: Autor

URI	Dado retornado
http://exemplo.org/alemanha	Conceito do mundo real
http://exemplo.org/alemanha.rdf	Representação para computadores, da coisa (RDF)
http://exemplo.org/alemanha.html	Representação para seres humanos, da coisa (HTML)

Separação via extensão de arquivo

Neste método, temos a extensão do arquivo determinando a URI, onde é possível saber, visualmente, o tipo de representação recebido. Na tabela 3 temos o mesmo exemplo das representações anteriores, com a separação via extensão de arquivo:

Tabela 3. Representação de URIs por extensão de arquivo. Fonte: Autor

URI	Dado retornado
http://exemplo.org/alemanha	Conceito do mundo real
http://exemplo.org/alemanha.rdf	Representação para computadores, da coisa (RDF)
http://exemplo.org/alemanha.html	Representação para seres humanos, da coisa (HTML)

5.2 Mecanismos de Dereferenciação

Para que um pedido por uma descrição de um recurso seja devidamente respondido, seja com RDF para máquinas ou formatos como o HTML para pessoas, uma estratégia de dereferenciação deve ser definida.

Conforme (Sauermann & Cyganiak 2008), as seguintes estratégias podem ser utilizadas, quando o dado RDF e HTML são disponibilizados separadamente:

URIs Hash

A estratégia de Hash é construída na possibilidade de que as URIs tem de possuir um fragmento especial, separado da base da URI por um símbolo hash (#).

Quando um cliente deseja acessar um URI com Hash, o protocolo HTTP exige que a parte do fragmento Hash seja retirada antes da solicitação do URI ao servidor e, portanto não poderá ser acessada diretamente. O cliente acessará a URI sem o fragmento e fará o processamento necessário para dirigir-se ao fragmento especificado, sem intervenção ou processamento do servidor.

URIs 303

Nos URIs 303, o servidor responde ao cliente que acessa o objeto do mundo real com um código de resposta *303 See Other*, com o URI do documento web que descreve o objeto. Este processo é chamado de redirecionamento 303. Em um segundo passo, o cliente dereferencia esse novo URI e recebe o documento web descrevendo o objeto (Bizer & Heath 2011).

É possível o uso híbrido de 303 e Hash, porém com impacto no desempenho e carga no servidor (Sauermann & Cyganiak 2008).

Em aplicações reais, é mais comum o uso de 303 em grandes conjuntos de dados (data sets), enquanto para pequenos vocabulários RDF e Ontologias OWL, é recomendável o uso de Hash (Sauermann & Cyganiak 2008; Bizer & Heath 2011).

5.3 Formatos de serialização de RDF

O modelo de dados RDF não define por si só um formato de publicação e representação de dados, especifica um modelo de descrição de recursos através de triplas: sujeito, predicado, objeto. Portanto, para publicar Linked Data na web, é necessário definir um formato para a serialização dos dados em RDF.

De acordo com a instituição W3C, os formatos RDF/XML e RDFa já são considerados padrões (Bizer & Heath 2011), outros formatos detalhados a seguir também possuem status de formato recomendado pela mesma instituição. Além desses formatos, temos como alternativas populares: Turtle, N-Triplas e RDF/JSON.

5.4 Vocabulários e Ontologias

Dentre os tipos disponíveis, temos o padrão SKOS usualmente utilizado para as taxonomias ou conceitos de hierarquias apenas, enquanto os padrões RDFS e OWL são utilizados para

determinar conceitos do mundo real em classes, com suas propriedades (Bizer & Heath 2011).

O reuso de vocabulários pré-existentes é estimulado, nos casos onde é necessária a criação de vocabulários próprios, as ontologias e/ou esquema RDF (RDFS) devem ser utilizados, de acordo com as melhores práticas de Linked Data (Bizer & Heath 2011).

5.5 Metadados

As descrições das entidades também devem conter metadados que possibilitem ao cliente entender mais sobre a origem e o conjunto de dados do qual cada descrição faz parte. Os conjuntos de dados também são chamados de *datasets*. Até o momento da elaboração deste trabalho, duas alternativas para a descrição da estrutura dos dados são utilizadas:

1. Semantic Sitemaps - Uso do conhecido e estabelecido protocolo sitemaps com extensões semânticas. Faz-se uso de um arquivo XML localizado na raiz do servidor web, com marcações sobre a estrutura semântica do conjunto de dados.
2. Descrições Void - De acordo com sua especificação, Void é um vocabulário em formato RDF Schema para expressar metadados sobre conjuntos de dados RDF. Void é o padrão reconhecido para a descrição de datasets, incorpora funcionalidades do método anterior e acrescenta outras, tudo através de RDF, diferentemente do método anterior.

Licença

Outro metadado importante é a licença ao qual a descrição da entidade está submetida. É importante salientar que não é possível licenciar o objeto do mundo real ou fato, pois o mesmo não pode ser licenciado. Apesar de não ser obrigatória, a ausência de licença não garante o direito aos dados e pode desestimular o seu reuso, portanto todos os dados linkados publicados na web devem incluir declarações de licença (Bizer & Heath 2011).

Termos de licença também podem ser adicionados ao nível de um amplo conjunto de dados, dentro da especificação Void (Alexander et al. 2009).

5.6 Resultado das Consultas SPARQL

De acordo com recomendação da W3C até o presente momento deste trabalho, existem definições de resultados de consultas SPARQL nos formatos XML, CSV e TSV, JSON, sendo o primeiro formato o primeiro a ter especificação definida. Os formatos suportados para a representação dos resultados de consultas SPARQL variam de acordo com as ferramentas utilizadas.

5.7 Estatísticas do ambiente

A análise de estatísticas do ambiente auxilia na governança e controle de um ambiente Linked Data, com dois pontos principais:

Uso

O projeto RDFStats (Langegger & Wöß 2009) consiste em um gerador de estatísticas de fontes RDF, como endpoints SPARQL e documentos RDF. Possui em suas funcionalidades a capacidade de geração de diferentes itens estatísticos como quantidade de instâncias por classe, histogramas por classe, propriedade e tipo de valor, dois geradores de

estatísticas para documentos RDF (arquivos locais e recursos na web) e endpoints SPARQL, API para acesso à estatísticas e diversas funções de estimação, entre outras.

Conteúdo

Apesar de a atual especificação SPARQL não contemplar amplas consultas sobre as bases para determinar dados estatísticos, existem iniciativas que tem como objetivo a coleta e geração de dados estatísticos, nessa área temos iniciativas como o projeto make-void e LODStats, que se propõe como alternativa mais performática quando comparada com abordagens existentes, especialmente em grandes conjuntos de dados com muitos milhões de triplas (Demter et al. n.d.)

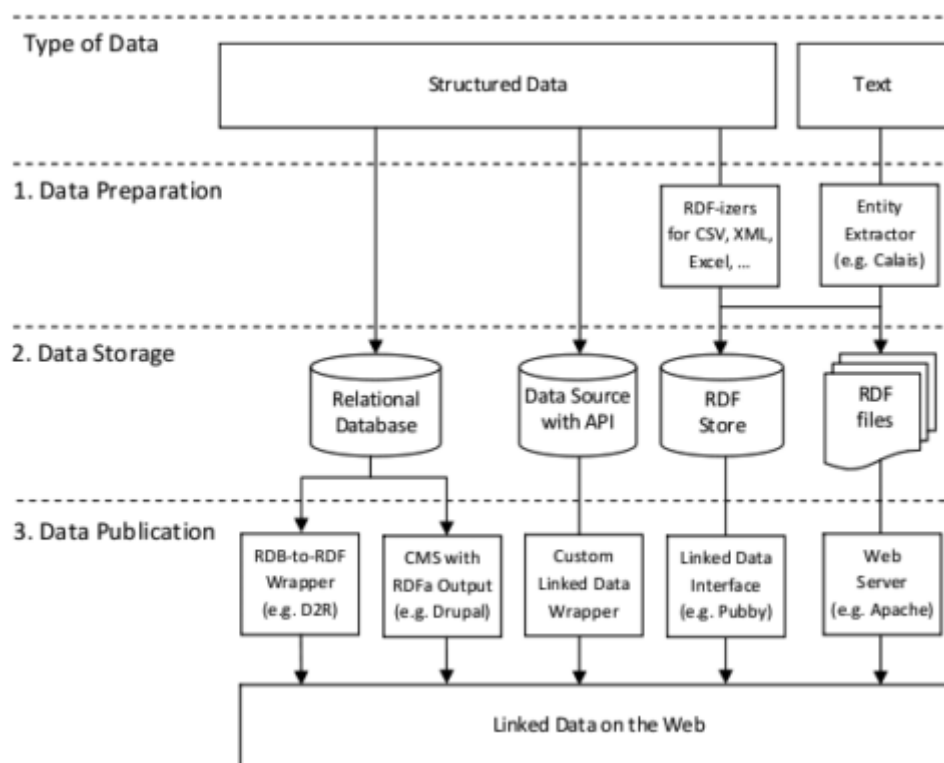
5.8 Fluxo e Estrutura de publicação de Linked Data

A publicação de Linked data requer a adoção dos princípios básicos expostos anteriormente na definição teórica de Linked Data. A conformidade com os padrões e melhores práticas que sustentam esses princípios é o que permite a interoperabilidade e reuso de Linked Data na Web.

Enquanto existe um grande número de sistemas que podem ser conectados na web de dados, os mecanismos para tal reduzem-se à um pequeno número de padrões de publicação de Linked Data. Nesta seção veremos mais detalhes sobre esses padrões.

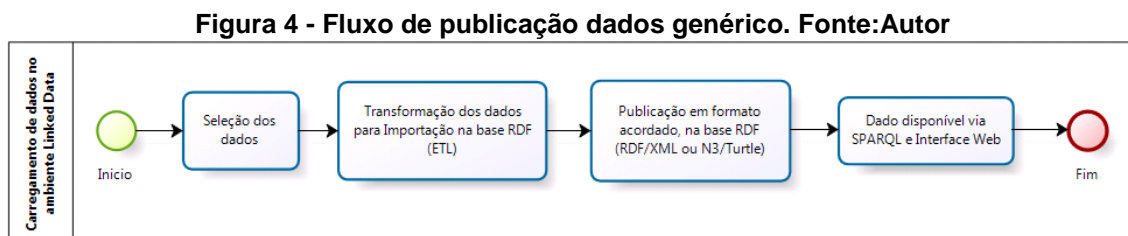
Na figura 3, temos os padrões de publicação mais comuns de Linked Data, de acordo com (Bizer & Heath 2011).

Figura 3. Fluxo de publicação de Linked Data. Fonte: (Christian Bizer & Heath 2011)



5.9 Fluxo de publicação de dados

O fluxo de publicação de dados também é um ponto decisório importante na implantação de um ambiente Linked Data, a figura 4 exemplifica um processo genérico considerado, para a elaboração posterior da infraestrutura:



O fluxo da figura considera um modelo simplificado para fins didáticos e tem como premissa que todos os dados serão transformados antes dos carregamento no ambiente Linked Data, preferencialmente no formato padrão RDF/XML, portanto neste momento não faz parte das tarefas da Infraestrutura de Linked Data, transformar os dados dos publicadores ou prover métodos semi-automatizados para tal.

5.10 Arquitetura do ambiente

O fluxo de publicação de dados também é um ponto decisório importante na implantação de um ambiente Linked Data, a figura 4 exemplifica um processo genérico considerado, para a elaboração posterior da infraestrutura:

Arquitetura Inicial

Em alguns casos, onde é possível utilizar servidores RDF com múltiplas funções, é possível construir uma arquitetura de ambiente somente composto por um este único elemento, como, por exemplo, o Openlink Virtuoso Open Source, tal como a arquitetura atual do projeto DBPedia . No entanto, a arquitetura possui alguns pontos de atenção, pelos seguintes fatores:

- Desacoplamento - Dificuldade no desacoplamento dos elementos do sistema quando a arquitetura é baseada em um único produto, nesse caso seria necessário a construção de todo o ambiente novamente, no caso da necessidade da troca do servidor RDF
- Customização dificultada - No caso do produto Openlink Virtuoso Open Source, o processo de customização mostrou-se complexo, exigindo um domínio de linguagem específica do produto (Virtuoso Server Pages), através da programação do arquivo description.vsp e criação de um conjunto de artefatos para a representação das consultas de dados da base RDF em formato HTML
- Robustez - O argumento favorável de maior robustez do ambiente com uma única camada especializada e redução nos encaminhamentos de mensagens é válido em implementações de grande porte, a própria DBPedia não considerou este ponto no início da sua arquitetura , apenas com o crescimento dos acessos passou a ser um ponto predominante na arquitetura de software.

De acordo com os pontos citados, uma arquitetura padrão será considerada para projetos onde não se caracteriza um grande número de acessos, de maneira que a maioria dos pontos acima fossem endereçados.

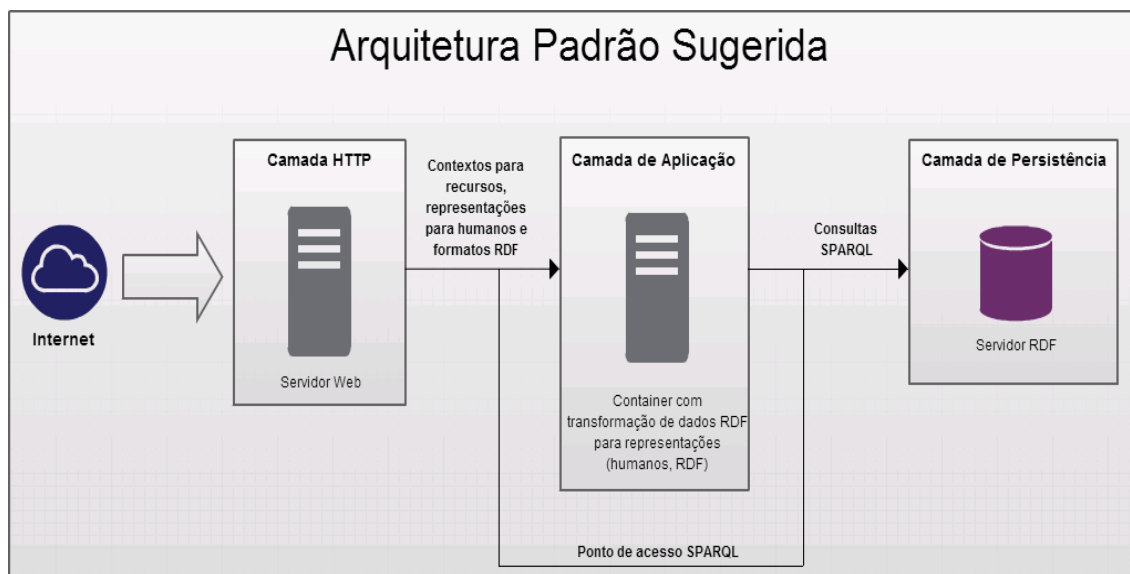
Arquitetura Padrão Sugerida

Para suprir as deficiências de uma arquitetura única com apenas uma camada, a arquitetura aplicada sugerida consiste em três camadas:

- Camada HTTP - Composta por servidor Web HTTP (Apache HTTP Server, por exemplo) e responsável pela camada de interação inicial de todas as requisições
- Camada de Aplicação - Composta por container responsável pela transformação das URIs e apresentação do conteúdo recebido do do servidor RDF (Apache Tomcat, hospedando a aplicação Pubby, por exemplo)
- Camada de Persistência - Composta pelo servidor de banco de dados RDF (como Open Link Virtuoso Open Source, Sesame, entre outros) e responsável por hospedar e retornar as consultas sobre as triplas de dados, ontologias e vocabulários

Na figura 5 temos o diagrama da arquitetura padrão sugerida:

Figura 5: Arquitetura padrão sugerida. Fonte: Autor



5.11 Elementos de operação de infraestrutura

Outros elementos serão necessários quando da operação do ambiente propriamente dito, esses são:

- Otimizações desejáveis - Ajuste fino de parametros e configurações para melhor desempenho possível dos produtos em cada camada da arquitetura
- Controle e Versionamento de ontologias e vocabulários - Controle das alterações nas ontologias e vocabulários, onde as opções de versionamento em arquivos estáticos possuem alternativas como o *git*⁴¹ e *subversion*⁴², os softwares Neologism⁴³ e DOME⁴⁴ apresentam-se como soluções especializadas para o controle, versionamento e autoria de ontologias.

⁴¹ <http://git-scm.com>

⁴² <http://subversion.apache.org>

⁴³ <http://neologism.deri.ie>

⁴⁴ <http://dome.sourceforge.net>

- Estratégias de backup de dados - Quando do uso de um sistema de banco de dados RDF (RDF Store), práticas comuns às bases de dados relacionais também são frequentemente suportadas, no caso os backups totais e incrementais (Elmasri & Navathe 2004, tradução nossa). Quando há uso de arquivos locais no sistema, como arquivos HTML e RDF/XML, abordagens comuns ao backup de arquivos podem ser aplicadas, seja o backup físico ou lógico dos dados, onde o backup lógico seria a cópia periódica de dados de um meio de armazenamento à outro, ou cópias no mesmo meio de armazenamento e o backup físico sendo o armazenamento dos blocos do disco onde os dados residem para outro meio de armazenamento (Hutchinson et al. 1999, tradução nossa).
- Mecanismos de extração de porções/dumps dos dados - Uma maneira genérica de obter partes dos dados de um ambiente é fazendo o uso do mecanismo padrão de consulta de Linked Data, o SPARQL, onde o escopo é definido de acordo com as características dos arquivos RDF. Além desse, existem mecanismos específicos dependendo da ferramenta utilizada.

6. Aplicação prática no laboratório de Engenharia do Conhecimento (EGC)

O objetivo principal da implantação no laboratório EGC foi um modelo robusto o suficiente, porém simplificado para a manutenção e extensão posterior.

Para melhor definição da arquitetura, o quadro proposto abaixo foi utilizado para discussão dos elementos, com as escolhas marcadas na cor cinza:

Quadro 2. Esquema de pontos decisórios discutidos com membros do EGC.

(continua)

Elemento	Alternativas	Considerações EGC
Volume de dados esperados	Alto (Mais de 500 triplas)	É desejável o suporte a grandes volumes de dados, além de preparação caso a frequência de atualização destes dados for alta.
	Baixo (Abaixo de 500 triplas)	
Frequência de atualização dos dados	Alta	
	Baixa	
Design das URIs	Separação por contexto	A separação por contexto foi decidida, em parte por já ser o método utilizado pela DBPedia, e além da simplicidade maior de configuração da infra-estrutura nesse modelo
	Separação por subdomínio	
	Separação por extensão de arquivo	
Mecanismos de Dereferenciação	Hash	O mecanismo de URIs 303 será utilizado inicialmente, pelos ganhos em organização e transmissão dos dados, que é compensatório pela intenção de suportar grandes volumes de dados, sem a necessidade do envio de dados extras, que ocorre no modo Hash
	URIs 303	
	Híbrido	

Quadro 2. Esquema de pontos decisórios discutidos com membros do EGC.

(continuação)

Elemento	Alternativas	Considerações EGC
Licença e Metadados	Não incluída	Não fará parte do escopo deste trabalho
	Incluída manualmente	
	Incluída automaticamente	
Armazenamento e controle de Ontologias e Vocabulários	Armazenadas em servidores RDF	Como o armazenamento de Ontologias será centralizado no servidor RDF, não foram aplicadas estratégias de versionamento
	Armazenadas em arquivos estáticos	
Estatísticas sobre os dados de Uso e Conteúdo	Uso	Ambos os meios são desejáveis, porém não farão parte do escopo deste trabalho.
	Conteúdo	
	Ambos	
Operação da Infraestrutura	Otimizações	Todos os dois fatores devem ser considerados, além de customização básica visual HTML.
	Backup/Restore	
Formatos de serialização pretendidos	RDF/XML	Após a definição da arquitetura, os formatos suportados serão: RDF/XML, N3/Turtle e CSV
	N3/Turtle	
	CSV	
	JSON	
	Outros	

6.1 Fluxo de publicação de dados no ambiente EGC e Convenções de URIs Aplicadas

No caso prático do EGC, o Fluxo de publicação genérico detalhado na seção 5.9 deste artigo. As convenções de URIs ficaram definidas como na tabela 4 a seguir:

Tabela 4. Representação de URIs por contexto para o EGC. Fonte: Autor

URI	Dado retornado
http://lodkem.egc.ufsc.br/resource/coisa	Conceito do mundo real
http://lodkem.egc.ufsc.br/data/coisa	Representação para computadores, da coisa
http://lodkem.egc.ufsc.br/page/coisa	Representação para seres humanos, da coisa
http://lodkem.egc.ufsc.br/onto/coisa	Representação de Ontologias e Vocabulários
http://lodkem.egc.ufsc.br/property/coisa	Representação de propriedades de Ontologias e Vocabulários
http://lodkem.egc.ufsc.br/class/coisa	Representação de classes de Ontologias e Vocabulários

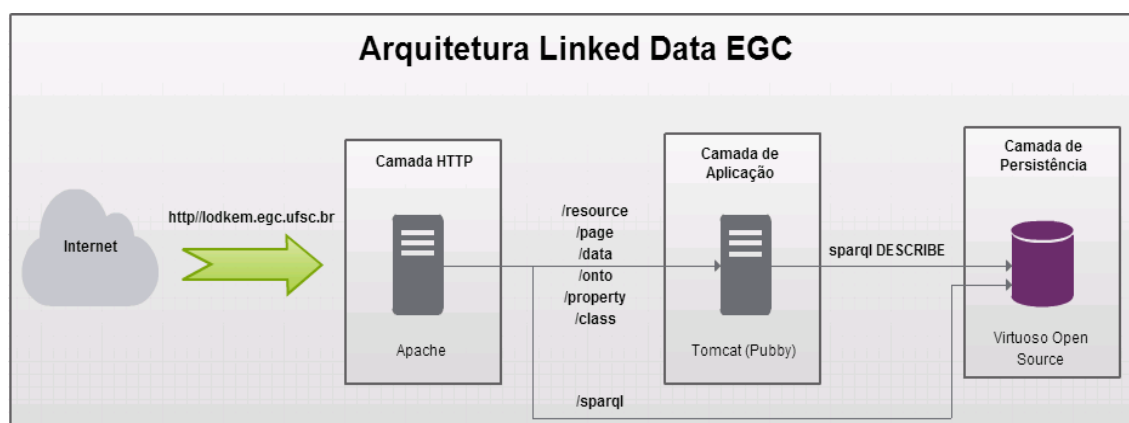
6.2 Arquitetura do ambiente

Para suprir as deficiências de uma arquitetura única com apenas uma camada, e considerando que no caso do laboratório EGC, não haverá expectativa de elevado número de acessos em primeiro momento, a arquitetura aplicada foi testada e completada e consiste em três camadas:

- Camada HTTP - Composta pelo servidor Web HTTP *Apache HTTP Server* e responsável pela camada de interação inicial de todas as requisições, configurado para responder na porta 80
- Camada de Aplicação - Composta pelo container *Apache Tomcat*, hospedando a aplicação *Pubby* e responsável pela transformação das URIs e apresentação do conteúdo recebido do *Virtuoso* (negociação de conteúdo), configurado para responder na porta 8080
- Camada de Persistência - Composta pelo servidor de banco de dados RDF *Open Link Virtuoso Open Source* e responsável por hospedar e retornar as consultas sobre as triplas de dados, ontologias e vocabulários, configurado para responder na porta 8890

Na figura 6 temos o diagrama da arquitetura aplicada:

Figura 6. Arquitetura aplicada para o ambiente EGC. Fonte: Autor



No diagrama da figura 6, observa-se as informações adicionais dos contextos e redirecionamentos em alto-nível, onde temos uma ponte direta entre a camada HTTP e a Camada de Persistência apenas para o acesso ao endpoint SPARQL, e os outros contextos definidos para o design das URIs direcionados primeiramente à camada de Aplicação, onde o aplicativo Pubby realiza a conversão dos pedidos para o Virtuoso, via consultas SPARQL do tipo DESCRIBE.

6.3 Implantação

Todas as camadas foram implementadas em um único servidor para fins de simplificação, sendo possível tanto a separação quanto o aumento de servidores em cada camada, conforme a necessidade, com a observação de que a camada de persistência com o produto Open Link Virtuoso Open Source só suporta mais de um servidor agrupado fora da versão open source, exigindo custos em seu licenciamento.

De maneira sintética, os seguintes programas foram utilizados:

- **Apache HTTP Server 2.2.22** - Servidor Web
- **Apache Tomcat 7.0.42** - Container de Servlets Java para o programa Pubby

- **Java 6 SE 1.6.0_25** - Plataforma Java Standard Edition requerida pelo Apache Tomcat
- **Pubby 0.3.3** - Aplicação Java para publicação de Linked Data via SPARQL
- **Open Link Virtuoso Open Source 6.1** - Aplicação tanto como base RDF com endpoint SPARQL, quanto para a própria publicação em si (não utilizada no ambiente Linked Data EGC)

6.4 Resultados

Após a implantação e posterior customização da interface para a resposta HTML dos dados, os resultados podem ser vistos na figura 7:

Figura 7. Resposta HTML gerada pelo Pubby para descrição de recurso. Fonte: Autor

Propriedade	Valor
?:isPrimaryTopicOf	▪ <http://en.wikipedia.org/wiki/DBPedia>
?:label	▪ DBPedia (en)
?:sameAs	▪ <http://dbpedia.org/resource/DBPedia>
?:wasDerivedFrom	▪ <http://en.wikipedia.org/wiki/DBPedia?oldid=257988581>
?:wikiPageID	▪ 15428045 ()
?:wikiPageRedirects	▪ <http://dbpedia.org/resource/DBpedia>
?:wikiPageRevisionID	▪ 257988581 ()

Metadados

Anon_0

<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://purl.org/net/provenance/ns#DataItem>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.w3.org/2004/03/trix/rdfg-1/Graph>
<http://xmlns.com/foaf/0.1/primaryTopic>	<http://lodkem.egc.ufsc.br/resource/DBPedia>
<http://xmlns.com/foaf/0.1/topic>	Anon_0
<http://www.ontologydesignpatterns.org/cp/owl/informationrealization.owl#realizes>	<http://lodkem.egc.ufsc.br/data/DBPedia>
<http://purl.org/net/provenance/ns#createdBy>	Anon_1 (mais)

[expandir tudo](#)

[Em formato Turtle](#) | [Em formato RDF/XML](#)

7 Conclusão

O enaltecimento dos benefícios inerentes ao modelo Linked Data em dados abertos trouxe interesse e diversas iniciativas em âmbito global. Iniciativas em áreas distintas como o *Geonames* na área geográfica, *data.gov*, do governo dos Estados Unidos, *dados.gov.br*, do governo Brasileiro, entre tantos outros, estão disponíveis atualmente.

Neste momento onde os desafios se voltam para a implantação propriamente dita e melhores práticas sobre como fazê-la, onde o presente artigo se inseriu e alcançou os objetivos esperados, desde a definição dos elementos e fluxos possíveis de publicação presentes nas seções teóricas deste trabalho, até a aplicação propriamente dita em um cenário real do laboratório EGC, onde foram superados desafios de arquitetura e deficiências nas ferramentas atuais para, de fato, disponibilizar os dados abertos com suporte ao Linked Data da maneira apropriada.

8 Referências

- Alexander, K. et al., (2009). Describing Linked Datasets On the Design and Usage of void , the “ Vocabulary Of Interlinked Datasets C. Bizer et al., eds. Design, 19. [Online]. Available: http://events.linkedata.org/ldow2009/papers/ldow2009_paper17.pdf.
- Bizer, C., Cyganiak, R. & Heath, T., (2007). How to Publish Linked Data on the Web. [Online]. Available: <http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial>.
- Bizer, C. & Heath, T., (2011). Linked Data: Evolving the Web into a Global Data Space,
- Cyganiak, R. & Wood, D., (2013). RDF 1.1 Concepts and Abstract Syntax. W3C Last Call Working Draft 23 July 2013. [Online]. Available: <http://www.w3.org/TR/rdf11-concepts> [Accessed July 30, 2013].
- Demter, J., Martin, M. & Lehmann, J., LODStats – An Extensible Framework for High-performance Dataset Analytics.
- Elmasri, R. & Navathe, S.B., (2004). Fundamentals Of Database systems, Fourth Edition,
- Hutchinson, N.C. et al., (1999). Logical vs . Physical File System Backup Logical vs . Physical File System Backup. , p.12.
- Klyne, G., Carroll J., J. & McBride, B., (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004. [Online]. Available: <http://www.w3.org/TR/rdf-concepts>
- Langegger, A. & Wöß, W., (2009). RDFStats – An Extensible RDF Statistics Generator and Library.
- Lee, D. & Galway, N.U.I., (2011). Open Data Overview.
- Sauermaun, L. & Cyganiak, R., (2008). Cool URIs for the Semantic Web. [Online]. Available: <http://www.w3.org/TR/cooluris>.
- T. Berners-Lee, “Linked Data,” W3C Design Issues, (2006), [Online]. Available: <http://www.w3.org/DesignIssues/LinkedData.html>.