

**UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

Isaac Luiz da Silva, Pedro Pacheco

**DESENVOLVIMENTO DE UM APLICATIVO DE  
AUXÍLIO DE TOMADA DE DECISÃO NA ESCOLHA DE  
GRADE DE HORÁRIOS UTILIZANDO IONIC**

Florianópolis

2018



Isaac Luiz da Silva, Pedro Pacheco

**DESENVOLVIMENTO DE UM APLICATIVO DE  
AUXÍLIO DE TOMADA DE DECISÃO NA ESCOLHA DE  
GRADE DE HORÁRIOS UTILIZANDO IONIC**

Trabalho de conclusão de curso submetido ao curso de Sistemas de Informação para a obtenção da graduação em Sistemas de Informação.

Orientador: Prof. Dr. Raul Sidnei Wazlawick

Florianópolis

2018

Ficha de identificação da obra elaborada pelo autor,  
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Silva, Pacheco, Isaac Luiz da Silva, Pedro Henrique  
Pacheco

Análise e Desenvolvimento de um aplicativo de  
auxílio de tomada de decisão na escolha de grade de  
horários utilizando Ionic e Jekyll / Isaac Luiz da  
Silva, Pedro Henrique Pacheco Silva, Pacheco ;  
orientador, Raul Sidnei Wazlawick Wazlawick, 2018.  
240 p.

Trabalho de Conclusão de Curso (graduação) -  
Universidade Federal de Santa Catarina, Centro  
Tecnológico, Graduação em Sistema de Informação,  
Florianópolis, 2018.

Inclui referências.

1. Sistema de Informação. 2. Ionic. 3. Jekyll. 4.  
Timetabling. 5. Aplicativo. I. Wazlawick, Raul  
Sidnei Wazlawick. II. Universidade Federal de Santa  
Catarina. Graduação em Sistema de Informação. III.  
Título.

Isaac Luiz da Silva, Pedro Pacheco

**DESENVOLVIMENTO DE UM APLICATIVO DE  
AUXÍLIO DE TOMADA DE DECISÃO NA ESCOLHA DE  
GRADE DE HORÁRIOS UTILIZANDO IONIC**

Este Trabalho de conclusão de curso foi julgado aprovado para a obtenção do Título de “graduação em Sistemas de Informação”, e aprovado em sua forma final pelo curso de Sistemas de Informação.

Florianópolis, 01 de julho 2018.

---

---

Prof. Dr. Raul Sidnei Wazlawick  
Orientador

**Banca Examinadora:**

---

Thaís Cardoso Lacerda

---

Prof. Dr. Frank Siqueira









*No meio da dificuldade encontra-se a oportunidade.*

Albert Einstein



## RESUMO

Existem diversos programas destinados à solução de problemas de *time-tabling* voltados para professores e instituições. Porém, nossos estudos não evidenciaram a existência de alguma ferramenta que seja mais voltada para as necessidades específicas dos estudantes. Nesse contexto, nosso trabalho terá como objetivo a criação de um aplicativo que possua uma boa usabilidade e que possui como funcionalidade principal auxiliar os alunos na escolha de disciplinas de modo que se possa realizar a completude da sua graduação no menor tempo possível. O aplicativo foi criado utilizando o Ionic Framework, que é um *framework* destinado à criação de aplicativos híbridos, com o qual é possível, com o mesmo código fonte, criar aplicativos para diversas plataformas. A documentação do aplicativo foi criada utilizando uma ferramenta chamada Jekyll, a qual cria páginas web estáticas utilizando *markdown*. Para resolver o problema de *timetabling* foi utilizada uma técnica conhecida por *constraint programming* (programação com restrições), a qual nos permite modelar diversas restrições, como horário, período e fazer a sugestão de possíveis matérias. Ao final do trabalho, fizemos todas as fases da análise de requisitos, desde a concepção de protótipos até a escrita da documentação propriamente dita, bem como o desenvolvimento do aplicativo para as plataformas *Android* e *iOS*, o servidor *Rest* baseado em **Spring**, testes unitários e de integração e testes de usabilidade.

**Palavras-chave:** grade de horários, *timetabling*, *constraint programming*, Ionic, Jekyll.



## ABSTRACT

There are several programs aimed at solving timetabling problems for teachers and institutions. However, our studies did not show the existence of any tool that is more focused on the specific needs of the students. In this context, our work will aim at the creation of an application that has a good usability and whose main function is to assist students in the choice of disciplines in order to complete the graduation in the shortest time possible. The application was created using the Ionic Framework, which is a framework for creating hybrid applications, with which it is possible, with the same source code, to create applications for various platforms. The application documentation was created using a tool called Jekyll, which creates static web pages using markdown. To solve the problem of timetabling we used a technique known as constraint programming, which allows us to model several constraints, such as time, period and suggestion of possible subjects. At the end of the work, we did all phases of requirements analysis, from prototype design to actual documentation writing, as well as application development for Android and iOS platforms, Spring-based Rest server, unit testing, and integration and usability testing.

**Keywords:** schedule grid, timetabling, constraint programming, Ionic, Jekyll.



## LISTA DE FIGURAS

|           |  |    |
|-----------|--|----|
| Figura 1  | Fluxo de obtenção dos arquivos no SVN .....    | 42 |
| Figura 2  | Restrições .....                               | 54 |
| Figura 3  | Teste Disciplinas Tarde .....                  | 58 |
| Figura 4  | Classe Teste .....                             | 59 |
| Figura 5  | Teste .....                                    | 60 |
| Figura 6  | Menu principal v1 .....                        | 68 |
| Figura 7  | Menu principal v2 .....                        | 69 |
| Figura 8  | Menu principal v3 .....                        | 70 |
| Figura 9  | Grade de horários v1 .....                     | 71 |
| Figura 10 | Grade de horários v2 .....                     | 72 |
| Figura 11 | Definir critérios .....                        | 73 |
| Figura 12 | Login .....                                    | 74 |
| Figura 13 | Menu principal .....                           | 74 |
| Figura 14 | Horário diário .....                           | 75 |
| Figura 15 | Horário semanal .....                          | 75 |
| Figura 16 | Geração de grade de horários - Passo 1 .....   | 76 |
| Figura 17 | Geração de grade de horários - Passo 2 .....   | 76 |
| Figura 18 | Geração de grade de horários - Passo 3 .....   | 77 |
| Figura 19 | Geração de grade de horários - Resultado ..... | 77 |
| Figura 20 | Estatísticas .....                             | 78 |
| Figura 21 | Formatura e próximos semestres .....           | 78 |





## LISTA DE TABELAS

|          |                                 |    |
|----------|---------------------------------|----|
| Tabela 1 | Comparação entre sistemas ..... | 39 |
|----------|---------------------------------|----|



## SUMÁRIO

|   |    |
|---|----|
| <b>1 OBJETIVOS</b> .....                  | 21 |
| 1.1 OBJETIVO GERAL .....                  | 21 |
| 1.2 OBJETIVOS ESPECÍFICOS .....           | 21 |
| 1.3 METODOLOGIA .....                     | 22 |
| <b>2 FUNDAMENTAÇÃO TEÓRICA</b> .....      | 27 |
| 2.1 <i>TIMETABLING</i> .....              | 27 |
| 2.2 <i>CONSTRAINT PROGRAMMING</i> .....   | 27 |
| 2.3 SPRING BOOT .....                     | 29 |
| 2.4 REST .....                            | 29 |
| 2.5 IONIC .....                           | 30 |
| 2.6 APACHE CORDOVA .....                  | 31 |
| 2.7 TYPESCRIPT .....                      | 31 |
| 2.8 INTERNACIONALIZAÇÃO .....             | 31 |
| 2.9 ANÁLISE DE REQUISITOS .....           | 32 |
| 2.10 USABILIDADE .....                    | 33 |
| 2.11 TESTES DE USABILIDADE .....          | 36 |
| <b>3 ESTADO DA ARTE</b> .....             | 37 |
| <b>4 ANÁLISE</b> .....                    | 41 |
| 4.1 O MODELO PADRÃO .....                 | 41 |
| 4.1.1 Histórico .....                     | 41 |
| 4.1.2 Troca de padrão .....               | 43 |
| 4.1.3 Implicações .....                   | 43 |
| 4.1.4 A documentação do TCC .....         | 44 |
| <b>5 DESENVOLVIMENTO</b> .....            | 45 |
| 5.1 IONIC .....                           | 45 |
| 5.1.1 Pré-requisitos .....                | 45 |
| 5.1.2 Iniciando o aplicativo .....        | 45 |
| 5.1.3 Executando o aplicativo .....       | 46 |
| 5.1.4 Estrutura do aplicativo .....       | 46 |
| 5.1.5 Arquivo inicial do aplicativo ..... | 47 |
| 5.1.6 Componentes .....                   | 47 |
| 5.1.6.1 Component .....                   | 47 |
| 5.1.6.2 Page .....                        | 48 |
| 5.1.6.3 Pipe .....                        | 49 |
| 5.1.6.4 Provider .....                    | 49 |
| 5.1.7 Deploy do aplicativo .....          | 50 |
| 5.2 INTERNACIONALIZAÇÃO .....             | 50 |

|   |     |
|---|-----|
| <b>5.3 REST API</b> .....                               | 51  |
| <b>5.3.1 Requisitos</b> .....                           | 51  |
| <b>5.3.2 Instalação</b> .....                           | 51  |
| <b>5.3.3 Rotas</b> .....                                | 52  |
| <b>5.3.4 Rotas criadas</b> .....                        | 53  |
| <b>5.3.5 Algoritmo para o cálculo do semestre</b> ..... | 53  |
| <b>5.3.6 Deploy da API Rest</b> .....                   | 54  |
| <b>6 TESTES</b> .....                                   | 57  |
| 6.1 PROCESSO DE TESTE .....                             | 57  |
| 6.2 TESTES DE INTEGRAÇÃO .....                          | 57  |
| <b>6.2.1 Testes do Algoritmo</b> .....                  | 58  |
| <b>6.2.2 Teste da API Rest</b> .....                    | 59  |
| <b>7 O SOFTWARE</b> .....                               | 61  |
| 7.1 VERSÕES PRELIMINARES .....                          | 61  |
| 7.2 RESULTADO FINAL .....                               | 61  |
| <b>7.2.1 Login</b> .....                                | 62  |
| <b>7.2.2 Menu principal</b> .....                       | 62  |
| <b>7.2.3 Grade de horários</b> .....                    | 62  |
| <b>7.2.4 Geração de horários</b> .....                  | 62  |
| <b>7.2.5 Resultado</b> .....                            | 63  |
| <b>7.2.6 Estatísticas</b> .....                         | 63  |
| 7.3 TESTES DE USABILIDADE .....                         | 64  |
| <b>7.3.1 Realização</b> .....                           | 64  |
| <b>7.3.2 Resultados</b> .....                           | 65  |
| <b>7.3.3 Principais alterações</b> .....                | 66  |
| <b>8 CONCLUSÃO</b> .....                                | 79  |
| 8.1 TRABALHOS FUTUROS .....                             | 79  |
| <b>9 DIREITOS AUTORAIS</b> .....                        | 81  |
| <b>REFERÊNCIAS</b> .....                                | 83  |
| <b>10 ANEXOS</b> .....                                  | 87  |
| 10.1 CÓDIGO FONTE APLICATIVO .....                      | 87  |
| 10.2 CÓDIGO FONTE DA API REST .....                     | 158 |
| 10.3 DOCUMENTAÇÃO .....                                 | 202 |
| 10.4 CASOS DE TESTE .....                               | 226 |
| 10.5 ARTIGO .....                                       | 237 |

## Introdução

Notoriamente, os alunos enfrentam problemas ao montar sua grade de horários através do Sistema Acadêmico de Graduação (CAGR), pois o mesmo possui uma usabilidade complexa ao primeiro acesso e não possibilita aos usuários uma montagem dinâmica de sua grade de horários, já que o resultado é exibido somente no final do processo, gerando um ciclo de tentativas e erros. Foi criada uma solução que auxilia os alunos a visualizarem sua grade e resolver choques de horários chamada Matrufsc (<https://pet.inf.ufsc.br/matrufsc/>). Porém, esta não auxilia o aluno em sua tomada de decisão no quesito "tempo de formatura". Ao mesmo tempo, há uma dificuldade para selecionar as matérias que lhe rendam menor tempo de formatura, que, caso sejam escolhidas de forma errônea, atrasam o tempo de formação e congestionam sua grade de horários. A existência de uma solução que auxilie os alunos a montar sua grade de horários de forma a otimizar seu tempo de formação a partir da melhor escolha de suas matérias seria útil para minimizar o esforço e possíveis problemas encontrados na utilização do CAGR.

O objetivo do aplicativo que propomos é ajudar os alunos a escolherem suas disciplinas para que sua formação ocorra no menor tempo possível, porém passando por todos os estágios de análise e desenvolvimento de um aplicativo e que ele possua uma interface fácil de se usar e que não exija uma grande curva de aprendizado dos usuários. A usabilidade foi avaliada a partir de testes de usabilidade que contarão com a participação de pessoas que farão parte do público alvo do *app*.

A partir do apresentado anteriormente, o objetivo deste projeto é a análise e desenvolvimento de um aplicativo mobile que auxilie os graduandos a escolherem suas próximas disciplinas de forma a concluir a graduação no menor tempo possível de acordo com requisitos definidos pelos mesmos.



# 1 OBJETIVOS

## 1.1 OBJETIVO GERAL

A partir do que foi elucidado, surgiu a ideia de desenvolver um aplicativo que auxilie os os graduandos a escolherem as disciplinas a serem cursadas em cada semestre de forma a minimizar seu tempo de formação. Para tanto, foi escolhido o nome Tantum, que em latim significa "Tudo em um só lugar". O nome faz alusão às diversas funcionalidades que estão contidas no *app* e às demais funções que podem ser inseridas no mesmo para que os graduandos possam utilizar apenas um aplicativo para ter acesso às principais informações que ele necessita na universidade e seu contexto, tais como seu horário de aulas, melhores disciplinas para cursar por semestre e suas estatísticas do curso.

Para o objetivo ser alcançado com sucesso e a maior quantidade de alunos ser alcançada, o aplicativo será desenvolvido para as duas plataformas *mobile* mais utilizadas (Android e iOS), aplicativos esses gerados a partir do *framework* Ionic e com o algoritmo baseado em *Constraint Programming*. Para guiar o desenvolvimento, será concebida uma documentação composta por requisitos e protótipos de tela. A documentação será armazenada no Github, escrita em *markdown* e gerada a partir da ferramenta Jekyll.

## 1.2 OBJETIVOS ESPECÍFICOS

Analisar, desenvolver e testar o aplicativo Tantum. a priori composto por:

- Algoritmo de montagem de grade de horários autônoma;
- Consulta aos horários das aulas do semestre;
- Consulta às estatísticas de conclusão do curso e próximos semestres de acordo com previsão realizada pelo algoritmo.

Escrever a documentação em arquivos *.md*, e apresentar através da ferramenta Jekyll, composta por:

- Requisitos funcionais;
- Requisitos não funcionais e

- Protótipos de telas.

### 1.3 METODOLOGIA

A pesquisa pode ser classificada como:

- Natureza aplicada, uma vez que obtendo seu resultado teremos um conjunto de parâmetros que serão aplicados no desenvolvimento do projeto;
- Objetivo exploratório, por meio de análise de entrevistas com possíveis usuários, testes de usabilidade e levantamento bibliográfico.

Para atingir os objetivos deste projeto será realizada a análise de requisitos, bem como o desenvolvimento do aplicativo assim como um experimento ao final do desenvolvimento com um conjunto de pessoas que possuam o perfil do público alvo do aplicativo, de forma a validar a usabilidade de interface do aplicativo.

Para o desenvolvimento deste projeto será utilizada a metodologia multi-método, a qual será subdividida em quatro etapas, as quais são:

**Etapla 1** - Fundamentação teórica: O início do projeto é dado pela pesquisa e análise dos conceitos básicos que nortearão o trabalho, de forma a conceder o arcabouço de conhecimento para os autores. A revisão foi focada em *Timetabling*, *Constraint programming*, *Ionic*, Análise de requisitos, Jekyll e Usabilidade. Essa etapa é composta pelas seguintes atividades:

1. *Timetabling*:
  - (a) Conceitos principais;
  - (b) Classes de problemas.
2. *Constraint programming*:
  - (a) Conceitos principais;
  - (b) Abordagens.
3. *Ionic*:
  - (a) Conceitos principais;
  - (b) Vantagens;



(c) Resultados.

4. *Análise de requisitos:*

- (a) Conceitos principais;
- (b) Prototipação incremental.

5. *Usabilidade:*

- (a) Conceitos principais;
- (b) Critérios de avaliação por Nielsen e Shneiderman.

**Etapa 2** - Revisão de trabalhos correlatos: O foco desta etapa é realizar uma pesquisa sobre trabalhos correlatos e similares de forma a compreender o estado da arte e o que já foi desenvolvido nesta área. A revisão levou em consideração algumas das palavras chave do trabalho, as quais foram:

- Grade de horários;
- Timetabling;
- Constraint programming;

A pesquisa foi realizada majoritariamente no repositório *Scholar Google*, o qual trouxe resultados que utilizavam os mesmos conceitos, porém não abordavam a mesma temática do nosso trabalho (decisão de grades de horários de alunos).

**Etapa 3** - Análise, implementação e teste: O foco desta etapa é a realização de três tarefas principais e suas respectivas subtarefas:

1. **Análise:** O objetivo da tarefa de análise é a produção de protótipos de tela, os quais auxiliarão na elucidação dos requisitos (funcionais e não funcionais) gerando como artefato de saída a documentação do projeto. Para alcançar tais objetivos, a tarefa foi dividida em duas fases:
  - (a) Prototipação: Nesta fase haverá a concepção dos protótipos de tela do projeto, sendo estes gerados e logo depois validados com possíveis usuários e designers de modo a atestar a usabilidade e facilidade do aplicativo. Após uma rodada de validação, os protótipos serão alterados, aplicando as alterações julgadas coerentes e depois passando novamente pela rodada de avaliação, até que se alcance um ponto ótimo de usabilidade e facilidade de uso do projeto.

- (b) Documentação de requisitos: A partir dos protótipos concebidos na fase anterior e entrevistas informais com os usuários serão elucidados os requisitos do aplicativo. A documentação será escrita em arquivos de texto (formato .md) que serão armazenados em nuvem e submetidos a controle de versão através do Github (DABBISH et al., 2012). Para visualização da documentação de forma mais clara será utilizada a ferramenta Jekyll (<https://jekyllrb.com/>), a qual gera páginas *Web* a partir dos arquivos .md e um *template* de montagem de páginas. Esse conjunto de requisitos será utilizado na etapa de implementação de forma a guiar o desenvolvimento das funcionalidades do projeto e na elaboração do manual de utilização.
2. **Implementação:** O objetivo da tarefa de implementação é a produção do artefato principal do projeto: o aplicativo. O desenvolvimento será baseado a partir dos requisitos e protótipos concebidos na etapa 1, este também interferindo nos requisitos definidos, uma vez que alguns problemas e abordagens são encontradas somente na fase de efetiva codificação das regras já apontadas. Para alcançar tais objetivos, a etapa foi dividida em três fases:
- (a) Desenvolvimento do algoritmo de montagem de grade de horários: Nesta fase será desenvolvida a parte mais complexa do projeto: O algoritmo de montagem de grade de horário. Algoritmo esse que se baseará nas matérias que o aluno já cursou para sugerir as próximas disciplinas que o mesmo deve cursar no próximo semestre, respeitando o número máximo de créditos do curso e as matérias que ainda lhe restam cursar. Essa fase ocorrerá em paralelo com a concepção e avaliação cíclica dos protótipos, já que o desenvolvimento do algoritmo é independente da escolha de layout do aplicativo e demais decisões tomadas na fase de prototipação.
- (b) Desenvolvimento do servidor *rest*: Para não sobrecarregar o aplicativo e evitar que o mesmo possua um elevado consumo de recursos e bateria do aparelho, a equipe decidiu por concentrar todo o processamento em um servidor remoto, deixando o *app* somente para visualização e solicitação de informações para o servidor. Nesta fase ocorrerá o desenvolvimento do servidor *rest* que conterà o algoritmo e fará o gerenciamento de carga de acessos.

- (c) Desenvolvimento do aplicativo: O objetivo desta fase é a implementação do *app* a partir dos requisitos e protótipos concebidos na etapa 1. Para alcançar esse objetivo será utilizado o *framework* de desenvolvimento Ionic, o qual gera aplicativos para os dois sistemas operacionais para celulares mais utilizados atualmente (Android e iOS) a partir de código escrito em linguagem Javascript.
3. **Teste:** A partir do aplicativo desenvolvido, serão realizados testes exploratórios e dedicados a contextos controlados para atestar a qualidade e confiabilidade do artefato.

**Etapa 4 - Estudo de usabilidade:** O estudo de usabilidade será realizado em todas as etapas e tarefas anteriores do projeto, de forma que o conceito de boa usabilidade do aplicativo seja construído antes mesmo deste ser implementado. Para alcançar tal objetivo a etapa foi dividida em duas fases:

1. **Validação na concepção dos protótipos:** Durante a concepção dos protótipos serão chamados possíveis usuários do sistema de forma a atestar sua facilidade de uso e entendimento dos termos que foram utilizados no aplicativo através de testes de usabilidade informais e comentários sobre suas opiniões. Essas avaliações serão cíclicas, até que se alcance um ponto ótimo de usabilidade e facilidade de uso do aplicativo.
2. **Validação após o término da implementação:** Após a implementação, um outro grupo de possíveis usuários será chamado para utilizar diretamente o *app* em um dispositivo móvel, umas vez que este já se encontrará implementado. A partir da aplicação de testes de usabilidade moderados pelos próprios autores, será oferecido um roteiro de teste e serão coletadas informações através do questionário SUS (*System Usability Scale*), de forma a atestar formalmente a qualidade da usabilidade e satisfação do usuário.



## 2 FUNDAMENTAÇÃO TEÓRICA

O problema da definição de horários que serão cursados nos semestres seguintes não é novo e, a cada nova etapa de matrícula, um grande número de estudantes passa por esta situação: "Quais matérias eu devo cursar no próximo semestre para me formar no menor tempo possível?". Além do fator de formação em menor tempo há ainda o fator de possíveis choques de horários que podem ocorrer dependendo das matérias que forem escolhidas pelo aluno, choques esses que demandam um grande esforço para serem solucionados gerando perda de tempo e, dependendo do caso, decisões erradas.

### 2.1 TIMETABLING

Para o problema de alocação de horários, há uma classe de problemas conhecida como *timetabling*, a qual trata de alocações de recursos em um determinado espaço de tempo. Falando especificamente da área acadêmica, Schaerf (1995) classifica este problema em três grupos:

- *School timetabling*: Classe que limita os problemas ao contexto de escolas que já possuem turmas fechadas e imutáveis;
- *Course timetabling*: Classe que limita os problemas ao contexto de cursos superiores, os quais possuem um maior conjunto de combinações, pois consideram uma grande gama de alunos, professores e salas de aula, gerando uma maior complexidade neste grupo do que nos outros dois;
- *Examination timetabling*: Classe que limita os problemas ao contexto de realização de avaliações para determinado conjunto de alunos.

### 2.2 CONSTRAINT PROGRAMMING

Para resolver o problema de *timetabling*, utilizaremos um conceito chamado de *Constraint Programming* (Programação com restrições), o qual é um paradigma de programação que permite-se formular algumas regras sobre uma sequência de variáveis para uma determinada solução, e a mesma só será solucionada quando todas as restrições forem cumpridas (JAFFAR; LASSEZ, 1987).

Normalmente as variáveis de programação com restrições são:

- Domínios booleanos;
- Domínios Inteiros;
- Domínios Reais;
- Domínios Lineares;
- Domínios Finitos;
- Domínios Mistos (vários dos anteriores).

*Constraint programming* pode ser expresso sob a forma de *Constraint Logic Programming*, o qual embute as restrições em um programa baseado em lógica.

Esse conceito teve início em 1987 com Jaffar e Lassez que estenderam um conjunto de classes de restrições que foram introduzidos em Prolog II e assim tendo as primeiras implementações de *Constraint Logic Programming* no Prolog III.

Programação por restrições é um paradigma de programação que pertence ao campo de estudo da Inteligência Artificial Simbólica, que possui representação de problemas em alto nível (simbólicos), como lógica e busca.

Uma definição de Inteligência Artificial pode ser como fazer uma máquina comportar-se de tal forma que seja chamada de inteligente, uma abordagem voltada para a IA fraca. Em contrapartida a IA forte tem como base a criação de máquinas que tenham autoconsciência e que possam pensar; e não somente simular raciocínios (RUSSELL; NORVIG, 2009).

Existem duas abordagens na qual a Inteligência Artificial foi categorizada, a IA Simbólica e a IA Conexionista. A IA simbólica está relacionada com a forma como o ser humano raciocina, e popularizou-se com os sistemas especialistas e com linguagens como Prolog.

Já a IA Conexionista é baseada na simulação dos componentes do cérebro (modelagem da inteligência humana) e o principal exemplo são as redes neurais e os algoritmos genéticos (RUSSELL; NORVIG, 2009).

Nesse trabalho utilizamos uma biblioteca Java *open-source* chamada Choco Solver para fazer a computação das restrições. As restrições são as opções que o usuário tem para escolher as matérias, sendo elas:

- Período das matérias (Matutino, Vespertino e Noturno);

- Horários das aulas, para não haver choque;
- Limite máximo da carga horária semestral;
- Pré requisitos cursados;
- Matérias equivalentes.

A biblioteca Choco Solver possui as principais classes para resolver as *constraints*: *Model* e *Solver*.

A classe *Model* na qual são adicionadas as restrições, e a classe *Solver* a qual possui o método `solver.solve()`, que indica se as restrições foram cumpridas ou não.

Para se adicionar restrições, um dos possíveis métodos é: `model.addClauseTrue`, o qual adiciona uma restrição do tipo booleana. Essa restrição é aceita somente se seu valor for *true*. O método que cria essa restrição também é da classe *Model*, sendo este: *boolVar*

Em adição ao uso de *Constraint Logic Programming* foi criado um algoritmo para listar as disciplinas com relação a suas prioridades, que são definidas pela quantidades de matérias na cadeia de pré requisitos, no qual em cima dessa lista são organizados os possíveis semestres dos alunos, aplicando as devidas restrições.

## 2.3 SPRING BOOT

Neste trabalho a forma escolhida para desenvolver o algoritmo foi a criação uma API Rest, a qual será consumida pelo aplicativo, assim realizando o processamento no *back-end* e não sobrecarregando o aplicativo. Também com a criação desta API é possível re-utilizar o algoritmo em outras aplicações, não somente no aplicativo apresentado neste trabalho.

Para isso foi criado uma aplicação com *Spring Boot*, que é um *framework* para a plataforma Java. Sua utilização torna o desenvolvimento da aplicação mais simples, pois com servidores embarcados não existe a necessidade do *deploy* de um arquivo WAR (SPRINGBOOT, 2018).

## 2.4 REST

*Representational State Transfer* (REST) é um estilo de arquitetura baseado no protocolo HTTP. Essa arquitetura se baseia no conceito

de cliente-servidor, no qual são obtidos recursos através de endpoints utilizando os métodos HTTP. Ela também é *stateless*, sendo que não se guarda o estado da transação, logo tudo o que é preciso para uma transação deve estar contido no seu escopo (FIELDING, 2000).

Os protocolos HTTP mais utilizados nesse estilo são: *GET*, *PUT*, *PATCH*, *POST* e *DELETE*; de modo que é possível criar recursos, editá-los, deleta-los e acessá-los por esses métodos.

## 2.5 IONIC

Nos últimos anos o uso de *smartphones* se intensificou devido ao barateamento dos mesmos e à popularização de seu uso através de diversas funções desenvolvidas como aplicativos. A partir dessa visão, decidimos desenvolver nossa ferramenta na forma de aplicativo de modo a facilitar aos estudantes o acesso a essa ferramenta de auxílio.

Existem três formas de se fazer um aplicativo:

1. A primeira é criar um aplicativo nativo, utilizando os *SDKs* e linguagens nativas de cada plataforma. Isso resulta na melhor performance, porém é custoso criar para várias plataformas;
2. A segunda é criar um *site mobile*, porém ele não tem acesso *offline* nem a recursos nativos. Porém é necessário fazer somente uma vez e roda em múltiplas plataformas;
3. A terceira é o aplicativo híbrido que junta o melhor dos anteriores, é a união da portabilidade, permitindo o desenvolvimento uma vez para todas as plataformas com o acesso a recursos nativos de cada plataforma (HUYNH; GHIMIRE; TRUONG, 2017).

As vantagens de se desenvolver aplicativos *mobile* híbridos é a capacidade dos desenvolvedores utilizarem tecnologias web como HTML5, CSS e JavaScript. Também é possível ter acesso a *APIs* do dispositivo, através do Apache Cordova, Ionic Framework, etc.

Assim o resultado é um aplicativo que se parece com um aplicativo nativo, porém é feito para múltiplas plataformas utilizando linguagens para *web*, sem o uso de Java, Objective C e C#.

Para atingirmos tal objetivo será utilizado o Ionic Framework, sendo o *framework* líder no desenvolvimento de aplicativos híbridos. (HUYNH; GHIMIRE; TRUONG, 2017). O Ionic é desenvolvido utilizando outros *frameworks* e tecnologias, como **AngularJS**, que é um *fra-*



*mework* JavaScript *open-source* desenvolvido pela Google para criação de páginas *web*.

Um aplicativo usando Ionic pode ser dividido em três componentes:

1. O *site* criado em HTML;
2. O estilo nativo que é provido pelo próprio *framework* e
3. O empacotador nativo para a plataforma como o Apache Cordova.

O Ionic gera aplicativos para os dois sistemas operacionais mais utilizados em *smartphones*: Android e iOS.

## 2.6 APACHE CORDOVA

Como o Ionic, é um *framework* desenvolvido para a criação de aplicativos móveis híbridos. O Ionic mescla o funcionamento do Apache Cordova e do AngularJS para facilitar o desenvolvimento, de modo que é possível utilizar todas as ferramentas do AngularJS para desenvolver o aplicativo, sendo o Apache Cordova responsável pela geração do aplicativo e também por funcionalidades específicas de dispositivos móveis, como o acesso a câmera, *bluetooth*, etc.

## 2.7 TYPESCRIPT

É uma linguagem *open-source* desenvolvida pela Microsoft. É um superconjunto de JavaScript e foi criada com a finalidade de facilitar o desenvolvimento de grandes aplicações (TYPESCRIPT, 2018). Seu código é transpilado para JavaScript.

## 2.8 INTERNACIONALIZAÇÃO

Nas últimas décadas houve um avanço na internacionalização de software, principalmente de empresas multinacionais, que desejam vender seus produtos em diversos mercados (ESSELINK, 2000).

É importante notar que internacionalização não inclui somente interface, mas diversas outras informações que também precisam ser internacionalizadas, como código fonte, ferramentas utilizadas, codificação do texto, entre outras.

Neste trabalho utilizamos técnicas de internacionalização de modo que alunos de outras nacionalidades também possam utilizar o aplicativo sem nenhuma complicação.

## 2.9 ANÁLISE DE REQUISITOS

A análise de requisitos deve ser uma das primeiras etapas do desenvolvimento de software, pois é seu papel levantar e detalhar os comportamentos das funcionalidades de forma a guiar a implementação e teste das mesmas. Independente de qual processo de desenvolvimento seja escolhido (Clássico, Scrum, XP...) há quatro atividades que são comuns a todos eles (SOMMERVILLE, 2011):

1. Especificação de software: Nesta atividade, os profissionais responsáveis por levantar os requisitos conversam com o cliente para definir as características da aplicação que será desenvolvida. A análise de requisitos está contida nessa atividade;
2. Projeto e implementação de software: A partir do levantamento realizado na atividade anterior é iniciada o desenvolvimento do software em si, concepção de diagramas (caso necessário) e a implementação do software em alguma linguagem de programação;
3. Validação de software: A partir da implementação realizada na atividade anterior e os requisitos que foram definidos na atividade inicial, é iniciada a validação do software de modo a garantir que as funcionalidades foram desenvolvidas corretamente de acordo com os requisitos definidos.
4. Evolução de software: Ao final das atividades anteriores é interessante que o software evolua para que continue sendo interessante e útil ao cliente e usuários. Dependendo do tipo de contrato firmado entre as partes, o projeto pode ser finalizado na etapa anterior, sem necessidade de evolução do software.

Para realizar o levantamento de requisitos desse projeto será utilizada a técnica de prototipação incremental (IBM, 2002), a qual possui como prerrogativa a utilização de protótipos de tela para nortear o desenvolvimento das funcionalidades. Essa técnica possui prós e contras:

### **Prós:**

- Requisitos são elucidados ao decorrer do projeto;

- O sistema acaba ficando mais intuitivo e focado nas reais necessidades de uso, uma vez que o usuário colabora com sua evolução e pode opinar sobre a utilização num contexto visual real;
- O envolvimento e satisfação do usuário final também são aumentados, visto que ele pode ver o fruto de suas considerações sendo alterado e rediscutido;
- Como maior característica positiva da prototipação incremental podemos destacar a rapidez em testar o ambiente de desenvolvimento, sua interface com o usuário e performance esperada direcionados para uma funcionalidade específica por vez.

**Contras:**

- Normalmente a abordagem é iniciada precocemente, sem a definição completa do contexto do problema, levando a protótipos que podem não abordar o problema como um todo;
- Um risco eminente é a falha do projeto, já que a cada iteração todos os protótipos podem ser alterados dando prioridade a uma funcionalidade irrisória em detrimento das funcionalidades realmente mais importantes;
- Como maior característica negativa da prototipação incremental podemos destacar o fato de lidarmos com a expectativa do cliente: Este método pode passar a falsa impressão que as funcionalidades podem ser alteradas facilmente como ocorria com os protótipos, gerando choques de interesses entre as partes. Outro fator importante é a impaciência que pode ser gerada no cliente pela demora no desenvolvimento do software a partir da versão demo que foi apresentada.

## 2.10 USABILIDADE

Um ponto muito importante e que foi tomado como uma das prioridades deste projeto é a usabilidade do aplicativo, já que o foco é a facilidade de uso e o rápido entendimento do usuário de como usufruir de todas as funcionalidades do app. Mesmo não possuindo um conceito único e consensual (PASCHOARELLI; SILVA, 2006) a usabilidade pode ser entendida como a escala de facilidade e simplicidade de alcance de objetivos desejados pelo usuário final de uma aplicação (KOOHANG, 2004).

Algumas abordagens foram propostas de forma a guiar o estudo de usabilidade e o desenvolvimento das aplicações. Segundo (CYBIS; BETIOL; FAUST, 2015), foram destacadas duas delas para o contexto do trabalho, as quais seguem detalhadas:

1. Heurísticas de Nielsen: Jakob Nielsen avalia a usabilidade de um sistema a partir de dez heurísticas, as quais foram expostas no seu artigo (NIELSEN, 1994). As heurísticas são:
  - (a) **Visibilidade do status do sistema:** Uma das responsabilidades do sistema é manter o usuário ciente do que está acontecendo no sistema;
  - (b) **Compatibilidade entre o sistema e o mundo real:** O sistema deve se manter o mais fiel à realidade possível, mantendo processos, linguagens e fluxos iguais aos da vida real;
  - (c) **Controle e liberdade para o usuário:** O usuário deve possuir liberdade de navegar e realizar ações livremente pelo sistema, exceto as ações que vão de encontro a regras pré-definidas ou que possam causar prejuízo à consistência dos dados;
  - (d) **Consistência e padronização:** É de extrema importância que a consistência e padrões de design sejam consistentes no sistema como um todo, de modo que as funções sejam análogas em qualquer ponto do sistema;
  - (e) **Prevenção de erros:** O sistema deve evitar que o usuário cometa erros, não apenas corrigí-lo caso erre;
  - (f) **Reconhecimento em vez de memorização:** O sistema deve auxiliar o usuário a reconhecer padrões e assim se guiar pelas funcionalidades ao invés de memorização de um caminho feliz e não intuitivo;
  - (g) **Eficiência e flexibilidade de uso:** O sistema deve ser simples e entendível independentemente do grau de instrução e familiaridade do usuário;
  - (h) **Estética e design minimalista:** Manter o layout o mais limpo possível, de forma a não gerar confusão no usuário;
  - (i) **Reconhecimento, diagnóstico e recuperação de erros:** O usuário deve ser capaz de reconhecer seu erro, através de mensagens claras e objetivas do que causou o imprevisto e ser capaz de se recuperar dos mesmos por conta própria;

- (j) **Ajuda e documentação:** O sistema pode possuir ajudas de contexto que auxiliem o usuário a realizar tarefas específicas, porém o interessante é que o mesmo não as possua por possuir uma interface simples.
2. Regras de ouro de Shneiderman: Ben Shneiderman utiliza oito regras de ouro para guiar o desenvolvimento e avaliação de aplicações (SHNEIDERMAN, 2010). As regras são:
- (a) **Consistência:** As ações do sistema, sequências de passos, mensagens de erros e todos elementos de design devem ser consistentes entre si, se repetindo em situações semelhantes;
  - (b) **Usabilidade universal:** Manter a simplicidade de uso e o entendimento geral do sistema, porém adequar a aplicação dependendo do grau de conhecimento do usuário, faixas etárias entre outros;
  - (c) **Feedback informativo:** Para cada ação do usuário deve haver um feedback do sistema informando a conclusão da tarefa ou um passo do usuário. Os feedbacks devem mudar de tipo e tamanho dependendo da importância da ação realizada;
  - (d) **Diálogos que indiquem o fim de uma ação:** Fluxos de passos devem possuir feedbacks que indiquem o fim de um conjunto de passos ou transição para um novo conjunto de passos, de forma a despertar a satisfação e o conhecimento do usuário sobre a completude da ação;
  - (e) **Evite erros:** O sistema deve ser projetado de forma a evitar que o usuário não cometa erros e, caso um imprevisto seja causado, auxiliar o usuário a consertá-lo o mais rápido possível;
  - (f) **Permitir a fácil reversão de ações:** Quanto for possível, possibilitar a reversão de ações realizadas no sistema, de forma a permitir que o usuário se sinta mais à vontade em usar o sistema ciente que poderá desfazer algum erro posteriormente;
  - (g) **Suportar o controle do usuário:** O usuário deve se sentir no controle de interface, obtendo informações de forma simples e objetiva e sem encontrar surpresas no comportamento já previamente conhecido;

- (h) **Reduzir a carga de memória de curta duração:** O sistema deve fazer o possível para evitar que o usuário precise se recordar do que foi inserido ou visualizado em outra tela do sistema para continuar com o fluxo de passos. A recomendação é evitar a criação de interfaces em que os usuários precisem memorizar dados para serem utilizados posteriormente.

Esses métodos de avaliação auxiliarão a equipe a modelar os protótipos e posteriormente avaliar as interfaces.

## 2.11 TESTES DE USABILIDADE

O teste de usabilidade pode ser definido como de uma técnica que tem por objetivo avaliar um determinado produto ou serviço, este podendo ser realizado de forma pessoal ou virtual, utilizando usuários representativos do grupo de pessoas que usará o objeto de avaliação. O teste é composto por um roteiro de tarefas, as quais o usuário deve realizar sem auxílio do avaliador de forma a medir diversos tipos de variáveis, como Eficácia, Eficiência, Satisfação do usuário e Tempo médio de utilização.

Os testes de usabilidade podem ser separados em dois grupos de acordo com a sua forma de realização:

- **Moderado:** É o tipo de teste mais tradicional e mais realizado, podendo acontecer pessoalmente ou remotamente. Envolve um avaliador que guia o usuário pelo roteiro do teste, fazendo perguntas e atribuindo tarefas que ele precisa realizar usando o produto ou a interface que está sendo testada;
- **Não-moderado:** Esse tipo de teste pode ser realizado online pelo próprio usuário através de ferramentas que o guiam pelas tarefas. Os usuários são encorajados a falarem em voz alta o que estão pensando e por que estão clicando em cada item na tela para que, posteriormente, alguém possa avaliar o resultado.

### 3 ESTADO DA ARTE

Observamos que diversos trabalhos acadêmicos tratam da alocação de horários para disciplinas de um determinado curso a partir dos horários em que os professores ministrantes estão disponíveis, exemplos são (FREITAS et al., 2007), (VIEIRA; MACEDO, 2011), (JÚNIOR; OLIVEIRA et al., 2000), (SILVA; SILVA, 2010), (CISCON; ALVARENGA, 2005). Outros trabalhos focam na alocação de salas para as aulas, por exemplo (SOUZA; COSTA; GUIMARÃES, 2002). A partir de pesquisas em repositórios de trabalhos acadêmicos e artigos científicos - principalmente Google Scholar e IEEE - utilizando as seguintes palavras-chave:

- Grade de horários;
- Timetabling.

Por esse motivo, resolvemos utilizar como estado da arte os sistemas que hoje auxiliam os graduandos da UFSC a montarem sua grade de horários semestral. A seguir detalharemos as três principais ferramentas normalmente utilizadas para a realização da tarefa:

**Ferramenta 1:** Matrufsc2 (<https://matrufsc2.appspot.com/>)

O Matrufsc 2 é o sistema mais utilizado pelo alunos para realizar a simulação de sua grade de horários devido a sua interface amigável, a funcionalidade de salvar seus horários e garantir que o processo possa ser concluído posteriormente. Isso sem contar a grande quantidade de disciplinas que podem ser selecionadas, essas mantendo seus horários e professores ministrantes atualizados a cada troca de semestre. Ele é mantido por alunos do curso de Ciências da Computação da própria universidade.

A ferramenta foi lançada no primeiro semestre de 2015 e é derivada de outros sistemas anteriores que possuíam a mesma proposta, como o CAPIM (Combinador Automático de Possibilidades Interativo de Matrícula) este baseado no GRAMA (GRAdo de MAtrícula), este último tendo suas atividades encerradas em 2012.

**Ferramenta 2:** SisAcad (<https://admin.inf.ufsc.br/>)

O SisAcad (Sistema Acadêmico) foi concebido a partir do interesse de disponibilizar aos graduandos uma forma simplificada de apresentar as principais informações para nortear e situar o estudante sobre o grau de completude do curso e os pontos que ainda restam ser concluídos para que possa ocorrer a tão sonhada formatura.

O sistema está disponível somente para alunos dos cursos de

Ciências da Computação e Sistemas de Informação e não foram encontradas informações sobre os criadores bem como a data de lançamento da ferramenta.

**Ferramenta 3:** CAGR (<http://cagr.sistemas.ufsc.br/>)

O CAGR (Sistema Acadêmico de Graduação) é o sistema responsável por gerenciar todas as informações do aluno no que se refere a administração escolar. Alguns exemplos de funcionalidades são:

- Calendários acadêmicos;
- Currículos dos cursos;
- Cadastro de turmas;
- Cadastro do Aluno;
- Espelho de matrícula;
- Geração de atestado de matrícula, Histórico Escolar e controle curricular;
- Solicitação de matrícula a cada início de semestre letivo.

Sobre a funcionalidade de solicitação de matrícula, o CAGR conta com a listagem de disciplinas sempre atualizadas, porém peca na questão de usabilidade e design, com informações confusas e sem nenhum auxílio para a resolução de conflitos de horários de matérias. Outro ponto importante é o fato do CAGR ser o único meio de oficializar a matrícula do aluno junto a UFSC, já que nenhuma ferramenta o faz por si só.

De forma a comparar as características específicas e guiar o desenvolvimento do projeto proposto, foi concebida uma tabela com as funcionalidades principais de cada sistema apresentado e o sistema proposto:

De acordo com o apresentado nas descrições dos sistemas e na tabela abaixo, chegamos à conclusão que nenhuma das aplicações oferecidas atualmente na UFSC atende as necessidades que enfrentamos a cada nova etapa de matrícula, assim gerando a motivação para a execução do presente projeto.



Tabela 1 – Comparação entre sistemas

| <b>Característica</b>                    | <b>Tantum</b> | <b>Matrufsc</b> | <b>Sisacad</b> | <b>CAGR</b> |
|--|---------------|-----------------|----------------|-------------|
| Resolução de conflitos                   | <b>X</b>      | <b>X</b>        |                |             |
| Escolha de disciplinas                   | <b>X</b>      | <b>X</b>        |                | <b>X</b>    |
| Montagem autônoma                        | <b>X</b>      |                 | <b>X</b>       |             |
| Escolha de turno de estudo               | <b>X</b>      |                 |                |             |
| Apresentação de disciplinas equivalentes | <b>X</b>      |                 |                |             |
| Avaliação do histórico do aluno          |               |                 | <b>X</b>       |             |
| Avaliação de desempenho do aluno         |               |                 | <b>X</b>       |             |
| Provável semestre de formatura           | <b>X</b>      |                 | <b>X</b>       | <b>X</b>    |
| Matrícula definitiva                     |               |                 |                | <b>X</b>    |
| Múltiplos planos de matrícula            |               | <b>X</b>        |                | <b>X</b>    |



## 4 ANÁLISE

O padrão de escrita da documentação utilizado neste projeto foi baseado no padrão adotado no Laboratório Bridge, local de trabalho dos autores deste trabalho. No laboratório, foi definido um padrão de documentação que tem como objetivo a padronização dos documentos gerados pelos diferentes analistas do projeto, além de guiar a escrita de documentações dos diferentes módulos.

### 4.1 O MODELO PADRÃO

#### 4.1.1 Histórico

Antes da utilização das ferramentas Jekyll (JEKYLL, 2018) e Github (GITHUB, 2018), era utilizado o EA (Enterprise Architect) (EA, 2018) para a confecção da documentação bem como sua disponibilização e o SVN (SVN, 2018) para o controle de versão. Nesse momento havia um padrão proposto para a organização de cada pasta de cada módulo:

- Requisitos Funcionais;
- Requisitos não funcionais;
- Casos de uso;
- Regras de negócio ou Regras gerais.

Dentro de cada pasta haviam os documentos específicos, os quais seguiam a seguinte forma de organização:

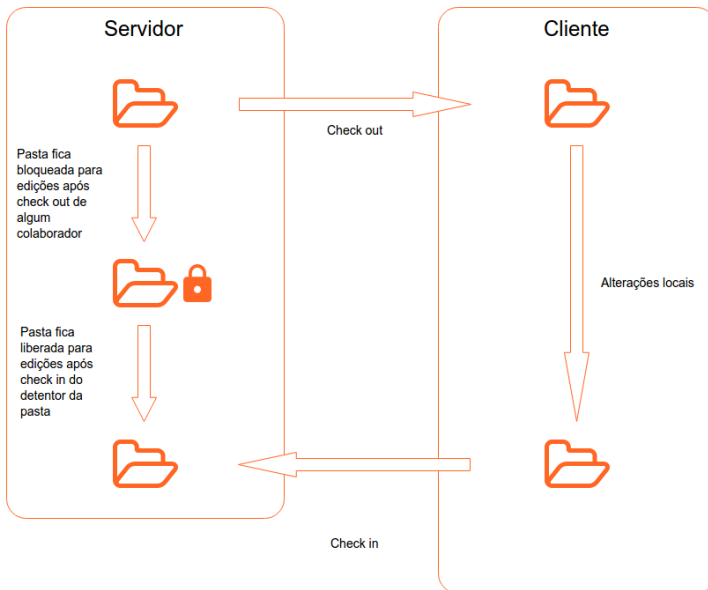
- Campos da tela;
- Regras que são aplicadas;
- Protótipos de telas.

Nessa época o laboratório executava um processo de desenvolvimento praticamente em forma de cascata, com a equipe de análise separada do desenvolvimento, não somente de forma física, mas da forma que uma determinada tarefa só poderia ser implementada uma vez que

algum artefato de análise fosse concluído. Nesse contexto, a comunicação entre analistas, desenvolvedores e testers era dada de uma forma mais burocrática e menos fluida do que encontramos hoje, utilizando o método ágil.

Neste momento, o controle de versão utilizado era o SVN, o qual possui o seguinte fluxo de obtenção dos arquivos:

Figura 1 – Fluxo de obtenção dos arquivos no SVN



Com esse comportamento, dois analistas não conseguiam alterar documentos de uma mesma pasta ao mesmo tempo, contribuindo ainda mais para a complicação do fluxo de trabalho. Sem contar as inúmeras vezes que um dos analistas esquecia de liberar sua pasta de trabalho, fazendo com o que o próximo profissional tivesse que aguardar a liberação por parte do dono.

Outro problema comum na época era a disponibilização da documentação no mesmo momento em que o código-fonte correspondente era integrado, devido a dificuldade de sincronização entre os dois. Não eram raras as vezes que itens eram integrados sem possuírem documentação ou documentação de itens que ainda não estavam integrados.

### 4.1.2 Troca de padrão

A partir das condições citadas anteriormente foi pensado em uma nova forma de armazenamento dos artefatos da documentação de forma a minimizar as desvantagens do método anterior. Após vários estudos, o método escolhido foi a forma já utilizada pelo Github, a qual consiste na escrita de arquivos no formato markdown (VOEGLER; BORNSCHEIN; WEBER, 2014), usando o github (que já era utilizado desde o início do projeto para armazenar e versionar o código) para o armazenamento dos arquivos e o Jekyll para a interface de disponibilização.

A criação do guia de documentação utilizado atualmente foi elaborado a partir do momento que a tecnologia de escrita e disponibilização da documentação foi alterado do Enterprise Architect para a escrita em arquivos markdown e disponibilização através do Jekyll.

O padrão atual é composto por seções que são específicas para cada tipo de requisito que o analista irá escrever, de forma a padronizar a escrita dos documentos que são produzidos por diferentes analistas nas diferentes equipes ágeis do projeto bem como guiar a própria escrita e lembrança das regras envolvidas.

### 4.1.3 Implicações

Após a troca de padrão (tanto ferramental quanto escrita), houveram ganhos surpreendentes na organização e distribuição dos artefatos de documentação do laboratório, praticamente excluindo o efeito da espera de um analista por outro para alterações nos documentos bem como a sincronização da documentação com o código. Hoje, a documentação é totalmente fiel às funcionalidades implementadas, aumentando a confiabilidade dos artefatos e sua utilização.

Um ponto importante a se destacar foi a curva de aprendizado dos analistas que não conheciam o github e sua utilização. Num primeiro momento foi disponibilizado um treinamento para que os analistas pudessem aprender como utilizar a ferramenta e resolver possíveis conflitos nos arquivos. Essa curva de aprendizado também compreendeu o Jekyll e seu funcionamento.

#### 4.1.4 A documentação do TCC

A documentação do aplicativo foi baseada no padrão de documentação utilizado no laboratório Bridge.

A documentação foi organizada de acordo com o contexto de cada documento, assumindo a seguinte estrutura ao final do processo:

- Geral: Composto por componentes e telas que são utilizados mais de uma vez no aplicativo;
- Login: Composto pelas telas que formam o Login do usuário bem como a tela de “Sobre” do sistema;
- Menu principal: Composto pelas telas que podem ser acessadas através do Menu principal, englobando os módulos do aplicativo.

Cada documento, por sua vez, possui as seguintes seções:

- Campos: Seção responsável pela elucidação de quais campos devem estar contidos na tela, bem como seus nomes. A ordem dos campos nessa seção deve ser a mesma da tela.
- Regras: Seção responsável por enumerar as regras que a tela e os campos deverão seguir. Por padrão, a primeira regra apresenta o mapeamento dos campos e como estes devem vir preenchidos quando a tela for aberta;
- Internacionalização: Seção responsável por apresentar como cada label da tela deverá se comportar em cada língua;
- Dicionário de dados: Seção responsável por apresentar a composição de informações de cada campo, tamanho mínimo e máximo de entradas, seus tipos e configurações;
- Protótipos de telas: Seção responsável por apresentar os protótipos de tela.

## 5 DESENVOLVIMENTO

### 5.1 IONIC

#### 5.1.1 Pré-requisitos

Para desenvolver com Ionic, é necessário que algumas alguns pontos já estejam configurados no computador como o *node js*. Apesar do Ionic não utilizar o *node js* diretamente, ele necessita do *node package manager* (npm).

Também é necessário o *Apache Cordova*, que é a base no qual o Ionic é desenvolvido, e por fim o próprio Ionic. Para fazer a instalação do NodeJs, basta fazer o download pelo site (NODEJS, 2018).

Após a instalação do NodeJs, deve-se executar:

```
npm install -g cordova ionic
```

Isso irá instalar a versão mais atual do Cordova e do Ionic.

#### 5.1.2 Iniciando o aplicativo

O Ionic Framework possui diversos templates prontos para serem usados, como por exemplo um aplicativo com menu lateral ou um aplicativo vazio. Esses templates servem para ajudar quem está iniciando a compreender como funcionam as principais formas de design de um aplicativo. Para este trabalho foi utilizado um template vazio e para isso deve-se executar:

```
ionic start tantum blank
```

Esse comando irá criar uma pasta com o nome do aplicativo que foi passado no comando, neste caso *tantum*. Para acessar a pasta e adicionar as plataformas necessárias para o desenvolvimento deve-se executar os seguintes comandos:

```
cd tantum
```

Para adicionar a plataforma de iOS é necessário que você esteja

no macOS. Para isso deve se executar o comando:

```
ionic platform add ios
```

Se você deseja publicar o aplicativo para Android, deve ser executado o comando:

```
ionic platform add android
```

### 5.1.3 Executando o aplicativo

Para iniciar o aplicativo basta executar o seguinte comando no terminal:

```
ionic serve --lab.
```

Isso irá iniciar um servidor local e executar a aplicação. O parâmetro `—lab` é utilizado para abrir o Ionic Lab, que é utilizado para testar o aplicativo em múltiplas plataformas, diretamente pelo navegador.

### 5.1.4 Estrutura do aplicativo

A estrutura de um projeto Ionic consiste nos seguintes componentes:

- **hooks:** Consiste de *scripts* que devem ser executados quando uma tarefa do Cordova é executada;
- **node\_modules:** Onde ficam localizados os pacotes npm descritos no *package.json*;
- **platform:** Código gerado para a plataforma específica, como Android e iOS;
- **plugins:** Contém todos os plugins que foram adicionados ao projeto;
- **resources:** Arquivos extras do projeto, como imagens e ícones;
- **scss:** Consiste de arquivos de estilo no formato *scss*;



- **src:** Código fonte do aplicativo, contém os arquivos HTML e TS;
- **www:** Código que foi gerado após o processo de *build*.

### 5.1.5 Arquivo inicial do aplicativo

`src/index.html` é o arquivo inicial do aplicativo, como um arquivo `index.html` em um projeto angular. Não será necessário alterar nada neste arquivo, pois a principal função dele é incluir os scripts e incluir a tag `<ion-app>`, que é o componente raiz em uma aplicação ionic. Todos os outros componentes estarão dentro dele.

### 5.1.6 Componentes

Os componentes básicos usados neste projeto, que são fornecidos pelo próprio Ionic são:

- *Component:* São elementos gráficos que podem ser reutilizados
- *Page:* São as páginas do aplicativo, contém diversos componentes
- *Pipe:* Componente que altera a informação
- *Provider:* Tipo de componente que fornece acesso aos dados

Tendo em vista que estes componentes são necessários para o desenvolvimento do aplicativo optou-se por abordá-los de forma detalhada conforme as seções que seguem.

#### 5.1.6.1 Component

São os componentes básicos de tela, normalmente criados quando os elementos já inclusos no pacote do Ionic não são suficientes. Neste trabalho foram criados quatro *Components*:

1. Cápsula;
2. Grade de horário semanal;
3. Lista de disciplinas;
4. Lista de disciplinas com ações.

Ao criar um componente são gerados três arquivos, um arquivo *html*, que contém os componentes de visão; um arquivo *ts*, onde está localizada a lógica do componente; e por fim um arquivo *scss*, onde podem ser adicionados outros estilos ao componente.

### 5.1.6.2 Page

São as páginas do aplicativo. O Ionic já possui uma página inicial, que é a *app*. Ela é referenciada dentro do *index.html* pela tag `<ion-app>`. Toda página tem uma tag associada. Neste trabalho, para atender as funcionalidades que queremos, foram criadas sete páginas, que são:

1. Definir critérios;
2. Estatísticas;
3. Grade de horário;
4. Login;
5. Principal;
6. Resultado;
7. Sobre.

Quando uma página é gerada, são criados 4 arquivos, como por exemplo a página de login:

1. *login.html*: Aonde ficam os componentes gráficos da página;
2. *login.module.ts*: Configura injeção de dependências entre outros;
3. *login.scss*: Arquivo que contém os estilos da página;
4. *login.ts*: Arquivo *typescript* que contém a lógica da página;

A estrutura é bem parecida com a do componente, com a diferença de que existe a criação de um arquivo a mais, que é o arquivo *module.ts*. Nesse arquivo está configurada a injeção de dependências, além dos módulos que ele importa e exporta, o que permite que esse módulo seja integrado em outros locais.

Nota-se que é a mesma estrutura que o Angular usa, e essa estrutura deixa bem clara as responsabilidades no padrão MVC, sendo

que a camada de Visão é o *html* da página, e fica totalmente separada da parte lógica da aplicação, que fica no Controller, que é o arquivo *ts*.

Esses componentes são gerados através do próprio Ionic pelo comando: `ionic generate [<tipo>] [<nome>]`, sendo o *tipo* um dos mencionados acima, e entre outros que não utilizamos. O *nome* seria o nome que seria dado ao componente. Por exemplo:

```
ionic generate page login
```

O exemplo acima cria uma página no aplicativo com o nome de login.

### 5.1.6.3 Pipe

Um *pipe* é um componente herdado do Angular, e ele tem como objetivo fazer alguma transformação em algum valor quando ele é mostrado para o usuário. Em nosso aplicativo utilizamos um *pipe* para a internacionalização do aplicativo. No projeto temos arquivos JSON para todos os idiomas que utilizamos. Nesses arquivos JSON, temos as mesmas chaves, porém o conteúdo é diferente em cada idioma, logo para utilizamos o *pipe* de tradução, basta passarmos a chave que desejamos e a linguagem, que ele nos retorna o valor desejado.

Outro exemplo de *pipe* seria a formatação de moeda. A informação que está no modelo pode ser um número, porém ao mostrar para o usuário é mais informativo adicionar o símbolo do Real (R\$). Então esse pipe transformaria uma informação que está somente o número e adicionaria o cifrão. Essa transformação não afetaria o modelo.

### 5.1.6.4 Provider

Um componente ou uma página não deveriam buscar dados diretamente, mesmo que sejam dados de teste. O foco dos componentes e páginas deve ser apenas mostrar os dados. A busca dos dados deve ser feito através dos *providers*.

Os *providers* tem como função acessar bancos de dados ou o *back-end* da aplicação. Para isso ele deve utilizar componentes já presentes no pacote do Ionic, como por exemplo o componente *HTTP*, que é utilizado para realizar chamadas *HTTP* para o acesso à APIs Rest.

Neste trabalho foi utilizado um *provider* para realizar a autenti-

cação com a Setic e outros para acessar o back-end. Também foi criado uma camada adicional no acesso aos dados. Foi criado uma classe chamada de *Api*, a qual tem a responsabilidade de realizar as chamadas pra API Rest. Essa camada não lida com regras de negócio, apenas implementa e simplifica o protocolo *HTTP*.

### 5.1.7 Deploy do aplicativo

Durante o desenvolvimento deste trabalho os testes do aplicativo foram feitos pelo browser, com o comando `ionic serve`. Porém se tem a necessidade de realizar os testes em um dispositivo mobile, para que seja verificado como realmente a aplicação se comporta e ver as situações de uso na realidade. Além disso, muitos plugins nativos só funcionam quando executados por um dispositivo de verdade, como por exemplo *Deeplinks*, que permite fazer links entre sites e aplicativos. Ele faz isso definindo uma URL para o aplicativo, a qual pode ser redirecionado por um site ou outro aplicativo.

Para fazer o deploy do aplicativo em dispositivo Android deve-se ter instalados os seguintes:

- Java JDK;
- Android Studio;
- Android SDK tools atualizados, com os SDKs necessários para o dispositivo.

Para rodar o aplicativo, basta habilitar o modo desenvolvedor no dispositivo Android e então executar o seguinte comando:

```
ionic cordova run android --device
```

Ele irá fazer a montagem do aplicativo, criando um arquivo *apk* e automaticamente instalando-o no dispositivo.

## 5.2 INTERNACIONALIZAÇÃO

Para realizar a internacionalização utilizamos uma ferramenta nativa do Ionic, o *ngx-translate*. Conforme citado anteriormente, esse componente é do tipo *pipe*, assim sua utilização pode ser feita pelo próprio HTML, como por exemplo:

```
<button>{{ 'LOGINBUTTON' | translate }}</button>
```

Esse *pipe* irá buscar automaticamente por arquivos *json* na pasta `i18n` (IONIC, 2018), e estes arquivos são nomeados com as linguagens que se deseja usar. Como neste trabalho foram utilizadas as linguagens em português, inglês e espanhol, foram criados os seguintes arquivos: *pt.json*, *en.json*, *es.json*.

Nestes arquivos estão as traduções para esses valores utilizados pelo *pipe*, assim nos arquivos teriam essas informações:

- **pt:** "LOGINBUTTON": "Entrar"
- **en:** "LOGINBUTTON": "Sign in"
- **es:** "LOGINBUTTON": "Entrar"

Para escolher a linguagem a ser utilizada, deve-se utilizar a classe do Ionic chamada *TranslateService* e chamar o método *use*, passando como parâmetro qual linguagem deve ser utilizada.

## 5.3 REST API

### 5.3.1 Requisitos

Neste trabalho foi desenvolvida uma API Rest em Java utilizando Spring Boot, logo para isso precisamos dos seguintes softwares instalados:

- Java 8;
- Maven;
- Eclipse IDE.

### 5.3.2 Instalação

Para iniciar um projeto Spring Boot, utilizou-se o gerador de projetos online, o Spring Initializr, disponível em (SPRING, 2018).

Esse gerador de projetos cria um projeto Maven básico, com uma classe já configurada como *Spring Boot Initializer*, sendo esta a classe *main* da aplicação responsável por iniciar o projeto.

Após extrair o projeto que foi criado pelo Spring Initializr, foi importado no Eclipse pela opção de importar projeto Maven. Durante a importação ele irá baixar as dependências. Por fim, basta executar a classe inicial, que neste trabalho é a *TantumApplication*, a qual é instanciada através de uma classe do *Spring* chamada *SpringApplication*.

### 5.3.3 Rotas

Para criarmos as rotas e definir quais métodos devem ser executados foi criado um Controller, que faz a ligação das chamadas HTTP para nossos métodos Java. No Spring são utilizadas anotações que fazem esse controle. Como por exemplo: *@RestController*, a qual define uma classe, como um Controller, e que métodos vão estar relacionados com chamadas HTTP.

Utilizamos também em nosso Controller a anotação *@RequestMapping("/v1/")*, que permite definir um mapeamento para todos os métodos. Logo todos os caminhos estão relacionados à versão 1 dessa chamada. Esse versionamento é definido como uma boa prática de APIs RESFful (HALDAR, 2018).

Também foi utilizada neste Controller a anotação *@CrossOrigin*, que permite chamadas de origem diferentes. Caso contrário, por padrão é apenas permitido chamadas de origem local.

Para relacionar os métodos com os caminhos, utilizou-se a anotação *@RequestMapping*, a qual possui como principais parâmetros os *path* e o *method*. Esses parâmetros definem o caminho a ser utilizado e o método HTTP. Como por exemplo o login, que foi definido o caminho `"/v1/login"` utilizando o método HTTP GET.

```
@RequestMapping(
path = "/login", method = RequestMethod.GET)
```

Também há necessidade de se passar parâmetros nas chamadas. Para isso são utilizadas as anotações *@RequestParam*, *@PathVariable* e *@RequestBody*, sendo a última utilizada em métodos HTTP POST. Por exemplo, no login, ele recebe o token de autenticação para verificação do mesmo.

```
login(@RequestParam(value = "token") String token)
```

### 5.3.4 Rotas criadas

Neste trabalho foram criados as seguintes rotas:

- GET `/v1/login`: Recebe o token de autenticação e o validar;
- POST `/v1/calculate-semester`: Faz a sugestão das matérias;
- GET `/v1/schedule/{semester}`: Retorna as matérias do semestre especificado;
- GET `/v1/subjects`: Retorna todas as matérias;
- GET `/v1/statistics`: Retorna as estatísticas do aluno.

O método de calcular o semestre utiliza outras anotações e informações. A anotação `@RequestBody` é utilizada aqui. Ela recebe como parâmetro o atributo `required`, o qual define que o corpo da chamada HTTP é sempre necessário. E na anotação `@RequestMapping` é utilizado outro parâmetro, o `consumes`, que define o tipo do corpo da requisição. Nesse caso é `application/json;charset=UTF-8`.

A busca de matérias pelo semestre utiliza a anotação `@PathVariable`, a qual define uma variável contida na própria URL. Um exemplo de consulta seria: `/v1/schedule/"2018-1"`. A qual buscaria os horários do semestre 2018-1.

Todas as respostas enviam por padrão o código HTTP 200, informando que a chamada ocorreu com sucesso. Caso não seja enviado o parâmetro do token, foi criado um método para que essa falta seja tratada automaticamente. O método criado é `handleMissingToken` e ele retorna o status HTTP 400, de solicitação inválida.

### 5.3.5 Algoritmo para o cálculo do semestre

Conforme mencionado anteriormente, o algoritmo que realiza o cálculo do próximo semestre esta localizado no caminho `/v1/calculate-semester`, sendo utilizado o método `HTTP POST` e enviado no `body` da chamada as restrições que foram definidas no aplicativo.

Para aplicar as restrições, foi utilizando a biblioteca `Choco solver`, as quais são aplicadas através da classe `Model`. Neste trabalho utilizamos o seguinte método desta classe: `addClauseTrue`, que adiciona uma restrição de modo que ela é cumprida caso o resultado da sua condição seja `true`. Esse método recebe como parâmetro uma variável

*booleana*, que é criada utilizando o método `boolVar` também da classe *Model*. Essa variável *booleana* criada possui dois valores como parâmetros, um nome, o qual será identificada essa restrição, e um método que será aplicada a restrição em si.

Assim sendo, neste trabalho foram criadas as seguintes restrições:

Figura 2 – Restrições

```

model.addClauseTrue(model.boolVar(credit_max,
    this.validateClassHourLoad(constraints, currentSubjects, subject)));
model.addClauseTrue(model.boolVar(times,
    this.validateTime(currentSubjects, subject)));
model.addClauseTrue(model.boolVar(period,
    this.validatePeriod(constraints, subject)));
model.addClauseTrue(model.boolVar(subjects_wanted,
    this.validateSubjectsWanted(constraints, subject)));
model.addClauseTrue(model.boolVar(subjects_not_wanted,
    this.validateSubjectsNotWanted(constraints, subject)));
model.addClauseTrue(model.boolVar(requisites,
    this.validateRequisites(subject, subjectsHistory)));

```

As restrições, conforme explicado anteriormente, são criadas através do método `addClauseTrue`, que recebe como parâmetro uma variável criada no método `boolVar`, este que recebe como parâmetros uma *String* para identificação e uma variável *booleana*, que neste trabalho são métodos que validam a restrição.

### 5.3.6 Deploy da API Rest

Neste trabalho o deploy da aplicação foi realizado por um executável *jar*. Para a criação desse arquivo foi adicionado no arquivo *pom.xml* o formato do pacote: `<packaging>jar</packaging>`.

O comando utilizado para a realização da compilação e geração do arquivo executável foi: `mvn clean install`. Esse comando junta todas as dependências que foram adicionadas no arquivo *pom.xml* e cria um arquivo *jar* único e executável, o que torna mais conveniente a execução e distribuição do mesmo.

Esse executável contém um servidor Tomcat, que ao ser executado ele inicia automaticamente e executa a classe *TantumApplication*, pois está marcada com a anotação `@SpringBootApplication`

O servidor por padrão inicia a aplicação na porta 8080. Logo



todas as chamadas para a API Rest devem ter a porta na URL.



## 6 TESTES

### 6.1 PROCESSO DE TESTE

Durante o desenvolvimento da aplicação, foram realizados testes exploratórios, de modo a encontrar *bugs* simples que ocorrem na utilização do usuário.

Para guiar esses testes foi criada uma série de casos de teste, os quais foram elaborados utilizando como base a experiência que os autores adquiriram no Laboratório Bridge. Cada caso é composto por:

- **Grupo:** O módulo (ou a tala) o que o caso de teste tem a finalidade de testar;
- **Item:** Título do caso de teste;
- **Regra:** Regra (ou conjunto de regras) da documentação que o caso de teste abrange;
- **Caso de teste:** Representa os passos de teste;
- **Comportamento esperado:** O que é esperado que aconteça após a execução dos passos de teste;
- **Situação:** Indicação se o comportamento esperado foi alcançado.

### 6.2 TESTES DE INTEGRAÇÃO

Desde o início do projeto, o intuito dos autores foi que ocorresse uma integração com o sistema de autenticação centralizado da SeTIC, para que a aplicação pudesse ser totalmente funcional e que fosse realmente útil na vida dos estudantes da UFSC. Porém, após várias tentativas infrutíferas, inclusive indo ao local e diversas solicitações de serviços que foram criadas por locais diferentes, foi tomada a decisão de não fazer a integração com os dados da UFSC para que o trabalho pudesse ser defendido no semestre corrente.

Uma vez que essa decisão foi tomada, foi elaborada uma série de casos de testes de integração para atestar o correto funcionamento do algoritmo que foi desenvolvido para a geração de horários.

## 6.2.1 Testes do Algoritmo

Neste trabalho realizamos testes unitários para mostrar o funcionamento do algoritmo. O teste define as possíveis restrições definidas pelo usuário e qual o resultado esperado quando se utiliza tais restrições. Foi possível realizar esse teste pois estamos utilizando um conjunto pequeno de disciplinas com a intenção de teste.

Os testes foram criados utilizando a API de testes unitários *JUnit*, pois é a mais conhecida para a plataforma Java. Os testes foram definidos utilizando a estrutura *Give, When, Then* que permite modelar uma história de usuário mais facilmente, sendo que **dado** um pré requisito, **quando** ocorre uma ação **então** eu tenho os seguintes resultados.

Os testes foram feitos utilizando os métodos *assert* do *JUnit*, o qual permite de modo claro saber o que é esperado por um determinado teste.

Foi criado um teste unitário para cada restrição, e com ela definida testamos as disciplinas sugeridas, como por exemplo a restrição de período de estudo.

Com esses testes podemos mostrar que o algoritmo cumpre o que foi pretendido e que as restrições definidas são sempre cumpridas.

Figura 3 – Teste Disciplinas Tarde

```

@Test
public void testMorningAfternoon() {
    // given
    constraints.setPeriods(Arrays.asList(AFTERNOON));

    // when
    NextSemestersDTO result = alg.calculateSemesters(constraints, semesterHistory.getSubjects());

    // then
    assertFalse(result.getNextSemesters().isEmpty());
    assertFalse(result.getSubjectsNotSelected().isEmpty());
    assertTrue(result.getSubjectsNotWantedError().isEmpty());
    assertTrue(result.getSubjectsWantedError().isEmpty());
    for (Semester s : result.getNextSemesters().values()) {
        for (Subject subject : s.getSubjects()) {
            boolean b = subject.getHorarios().stream().map(Period::getPeriodByTime).allMatch(AFTERNOON::equals);
            assertTrue(b);
        }
    }
}

```

## 6.2.2 Teste da API Rest

Para realizar o teste da API Rest foi utilizada uma classe do *Spring* chamada *MockMvc*, a qual consegue-se realizar as chamadas *HTTP* de modo que o teste se comporta como se fosse realizada uma chamada real.

Para utilizar o *MockMvc* deve-se anotar a classe com as anotações mostradas na figura 4.

Figura 4 – Classe Teste

```
@RunWith(SpringRunner.class)
@SpringBootTest(classes = TantumApplication.class)
@AutoConfigureMockMvc
public class TantumControllerTest {

    @Autowired
    private MockMvc mockMvc;
```

Note que esse teste não é executado com o *JUnit*, mas sim com o *SpringRunner*, o que permite a injeção de dependências. Além disso, é utilizada a anotação *@SpringBootTest*, que permite definir o ambiente do Spring que irá rodar, de modo que nossas dependências sejam encontradas. Para isso é passada como parâmetro a classe principal do servidor. Também a anotação *@AutoConfigureMockMvc*, a qual é a responsável pela injeção do *MockMvc*.

Além de serem testadas todas as chamadas, também foram testados casos de erro, como por exemplo um parâmetro em falta.

Na figura 5 é testado não somente o conteúdo, mas todo o *JSON-Path* e o status *HTTP*.

## Figura 5 – Teste

```
@Test
public void scheduleTest() throws Exception {
    this.mockMvc.perform(get("/v1/schedule/2018-1").param("token", token)).andDo(print())
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.exists").exists())
        .andExpect(jsonPath("$.success").isBoolean())
        .andExpect(jsonPath("$.success").value(true))
        .andExpect(jsonPath("$.result").exists())
        .andExpect(jsonPath("$.result.subjects").exists())
        .andExpect(jsonPath("$.result.subjects").isNotEmpty());
}

@Test
public void scheduleTestFail() throws Exception {
    this.mockMvc.perform(get("/v1/schedule/2018-1")).andDo(print())
        .andExpect(status().isBadRequest())
        .andExpect(content().string(bad_request))
        .andExpect(jsonPath("$.exists").exists())
        .andExpect(jsonPath("$.success").isBoolean())
        .andExpect(jsonPath("$.success").value(false))
        .andExpect(jsonPath("$.result").exists())
        .andExpect(jsonPath("$.result").value("token parameter is missing"));
}
```

## 7 O SOFTWARE

### 7.1 VERSÕES PRELIMINARES

Para chegar ao resultado final, diversas versões anteriores foram criadas, servindo como base para a construção e evolução do entendimento e amadurecimento das interfaces do *app*.

Os principais pontos que podem ser destacados das versões preliminares são:

- Menu principal:
  - v1: O menu na versão 1 era composto por pequenos botões, os quais eram responsáveis por guiar o usuário pelas funcionalidades.
  - v2: Na versão 2 houve a mudança de layout, o qual foi dividido com os principais *feedbacks* recebidos na época.
  - v3: Na versão 3 houve mais uma mudança, fazendo com que o menu fosse composto por grandes cards, os quais deveriam ser arrastados para os lados para fazer a transição entre as funções.
- Grade de horários: A grade de horários só possuía a visualização por dia, uma vez que a ideia de trazer a grade semanal ainda não tinha sido concebida.
- Geração de grade de horários: A geração de grade de horários possuía duas vertentes: Usar os critérios definidos anteriormente ou gerar com novos critérios. Foi decidido pela segunda opção para facilitar o entendimento do usuário.
- Definir critérios: A definição de critérios para a geração de grade de horários, por sua vez era realizada em um único passo. Posteriormente se percebeu a facilidade de se realizar essa ação em mais passos.

### 7.2 RESULTADO FINAL

Sobre o *app* em si, serão apresentados a seguir os protótipos de telas concebidos, bem como seu suas funcionalidades e fluxo de navegação.

### 7.2.1 Login

Para ter acesso ao aplicativo, o graduando deve realizar seu login utilizando seu idUFSC e sua correspondente senha de acesso. Há a possibilidade de se manter logado, assim pulando a fase de login sempre que o aluno entrar no sistema.

Há também a opção de ler algumas informações referentes ao sistema acionando o botão Sobre. Uma vez logado, o usuário será redirecionado para o Menu principal.

### 7.2.2 Menu principal

Uma vez no Menu principal, o usuário pode escolher entre as três funcionalidades ofertadas:

- Grade de horários;
- Geração de grade de horários;
- Estatísticas.

### 7.2.3 Grade de horários

Uma vez selecionada a opção Grade de horários, o usuário será redirecionado para a tela de visualização da grade de horários do semestre.

O objetivo desta funcionalidade é a observação dos horários e alocações das aulas do dia. Para visualizar os horários de outros dias, basta arrastar a tela na direção do dia esperado:

- Da direita para esquerda: Avança os dias;
- Da esquerda para direita: Retrocede os dias.

A aba "Semanal" dá acesso aos horários de aula exibindo-os em formato semanal.

### 7.2.4 Geração de horários

Uma vez selecionada a opção Geração de grade de horários, o usuário será redirecionado para a funcionalidade de geração de grades



de horários.

Nesta tela o usuário deve decidir como vai gerar sua grade de horários, de forma a minimizar o tempo de formatura ou a partir de critérios definidos.

Caso a opção selecionada for Gerar grade de horários para minimizar tempo de formatura, redirecionar o usuário para a tela de resultado e, caso haja uma sobrecarga no servidor, para a tela de notificação de conclusão.

Caso a opção selecionada for Definir critérios, redirecionar o usuário para a tela de definição de critérios.

Uma vez acionada a ação Voltar, o aluno deve ser redirecionado para o menu principal do sistema.

O objetivo dessa fase é definir os critérios que nortearão o algoritmo de geração de grades de horário. Os critérios são:

- Turno de estudo;
- Número máximo;
- Matérias que quero cursar nesse semestre.

Ao acionar a opção Gerar grade de horários, redirecionar o usuário para a tela de resultado.

### **7.2.5 Resultado**

A partir dos critérios utilizados, o resultado é exibido da forma como representado no protótipo. O usuário pode navegar entre as opções de geração, excluindo as disciplinas que não lhe agradaram e visualizando o horário diário bem como a localização das salas de aula.

Caso o usuário desejar, ele pode redefinir os critérios de geração a partir da ação de mesmo nome.

Uma vez selecionada a opção Estatísticas, o usuário será redirecionado para a funcionalidade de Estatísticas de conclusão do curso.

### **7.2.6 Estatísticas**

Na tela Estatísticas devem ser apresentadas as estatísticas de conclusão do curso do graduando.

- Gráfico 1: Exibe a porcentagem de semestres cursados pelo aluno

em um gráfico de pizza, seguida pela quantidade de semestres cursados e restantes;

- Gráfico 2: Exibe a porcentagem de disciplinas cursadas pelo aluno em um gráfico de pizza, seguida pelo total de matérias do curso e porcentagem de completude.

Ao acionar a ação Formatura e próximos semestres devem ser exibidos as matérias a serem cursadas nos próximos semestres de acordo com sucessivas gerações do algoritmo.

## 7.3 TESTES DE USABILIDADE

### 7.3.1 Realização

Os testes foram realizados no dia onze de março de 2018, contando com a participação de 8 participantes.

Os participantes foram selecionados de acordo com sua faixa etária, curso realizado e seu nível de escolaridade, o qual variou entre superior incompleto e superior completo. Todos os participantes realizam / realizaram seus cursos na UFSC.

O teste era composto por três documentos os quais estão contidos nos anexos deste trabalho:

1. Termo de consentimento e esclarecimento de participação;
2. Roteiro de teste;
3. Questionário SUS (*System Usability Scale*).

Para a realização dos testes, os participantes leram e preencheram o Termo de consentimento e esclarecimento de participação, de forma a atestar sua efetiva participação no teste. Logo após, receberam os passos de teste, os quais deveriam ser realizados sem a interferência direta dos avaliadores, que observaram os movimentos dos participantes que, por sua vez, falavam em voz alta seus pensamentos e ações realizadas no aplicativo. Durante a execução, as propostas de melhorias e dificuldades também eram citadas pelos participantes.

### 7.3.2 Resultados

O tempo médio de interação dos usuários com o sistema foi de oito minutos, considerando que estes comentavam pontos e já indicavam pontos de melhoria. Porém, ao levar em consideração o tempo que os usuários levaram para acessar as funções e, segundo os próprios participantes, a função mais utilizada seria os próximos horários de aula, a qual pode ser diretamente acessada sem a necessidade de um toque sequer na tela.

Também é importante salientar que os participantes não levaram muito tempo para realizar os passos da geração de horários, considerada por todos e pelos autores como a função mais extensa do aplicativo.

A partir desses raciocínios, a opinião dos autores é que o tempo de interação com o sistema está de acordo com sua proposta de uso.

O critério utilizado para avaliar os resultados dos questionários aplicados é o proposto por (BANGOR; KORTUM; MILLER, 2009), o qual separa a satisfação do usuário em sete níveis, os quais são:

- Pior imaginável: Menos de 12.5 pontos;
- Horrível: Entre 12.6 e 20.3 pontos;
- Pobre: Entre 20.4 e 35.7 pontos;
- Ok: Entre 35.8 e 50.7 pontos;
- Bom: Entre 50.8 e 71.4 pontos;
- Excelente: Entre 71.5 e 85.5 pontos;
- Melhor imaginável: Mais de 85.6 pontos.

A média dos resultados do questionário SUS foi de 92,5 pontos, o que de acordo com os critérios citados acima coloca o aplicativo com a melhor usabilidade possível para o contexto que se propõe.

A nota mais baixa foi de 40 pontos, esta sendo atribuída por um participante. Entendemos que essa nota foi fruto de uma má interpretação do teste, visto que o mesmo comentou várias vezes que as matérias não existiam mesmo sendo informado diversas vezes que os dados para o teste eram fictícios, os quais eram utilizados somente para exemplificar a usabilidade do aplicativo.

A maior nota foi de 95 pontos, esta sendo atribuída por cinco participantes. Entendemos que essa nota foi fruto de todo o trabalho de estudo de usabilidade e várias iterações até conceber o resultado

final. Logo após a completude dos testes, estes usuários elogiaram a facilidade de uso do sistema e seu o provável uso cotidiano.

### 7.3.3 Principais alterações

Os principais pontos de melhoria apontados pelos participantes foram:

- **Login:**
  - Os botões **Login** e **Sobre** possuem o mesmo formato e tamanho, indicando o mesmo nível hierárquico.
- **Menu:**
  - Os próximos horários de aulas poderiam apresentar o seu dia, não somente o horário;
  - Um usuário não identificou o horário das próximas aulas e seguiu diretamente para a grade de horários, não notando a funcionalidade de próximos horários diretamente instalada no menu.
- **Grade de horários:**
  - Em **Grade de horário semanal**, mostrar detalhe da disciplina ao tocar sobre ela, seguindo o comportamento da grade de horário diária.
- **Gerar grade de horários:**
  - Um usuário achou complicada a utilização do componente de seleção de créditos mínimos e máximos;
  - Nos passos 2 e 3, adicionar a funcionalidade de pesquisar as disciplinas enquanto se digita seu nome / código;
  - Ao selecionar uma disciplina, exibir também seu número de créditos;
- **Resultados:**
  - Exibir para o usuário uma indicação que o processo de geração foi concluído;
  - Deixar mais clara a indicação de quais disciplinas matérias não foram selecionadas no resultado e o porque disso;

- Criar uma notificação específica para quando não houver possibilidade de geração de horários devido aos critérios escolhidos.

- **Estatísticas:**

- Dois usuários não compreenderam que os gráficos são interativos, e questionaram qual seriam os números dos gráficos.

Figura 6 – Menu principal v1

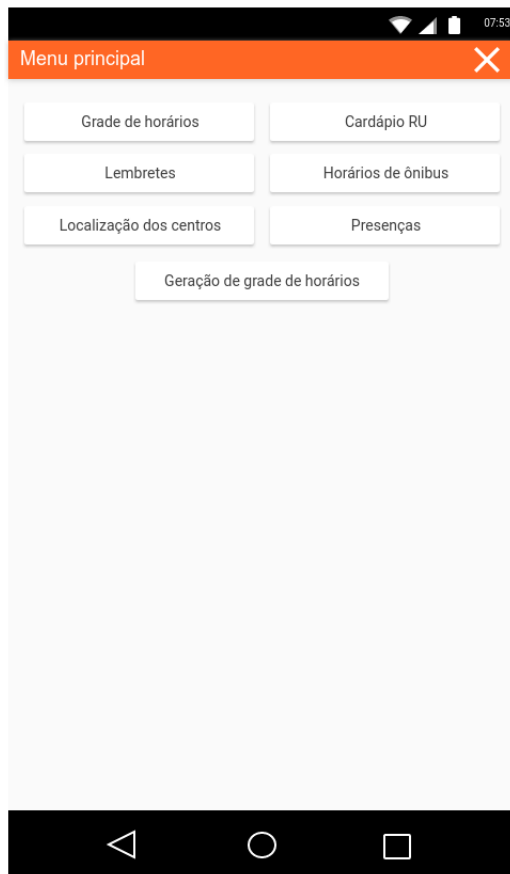
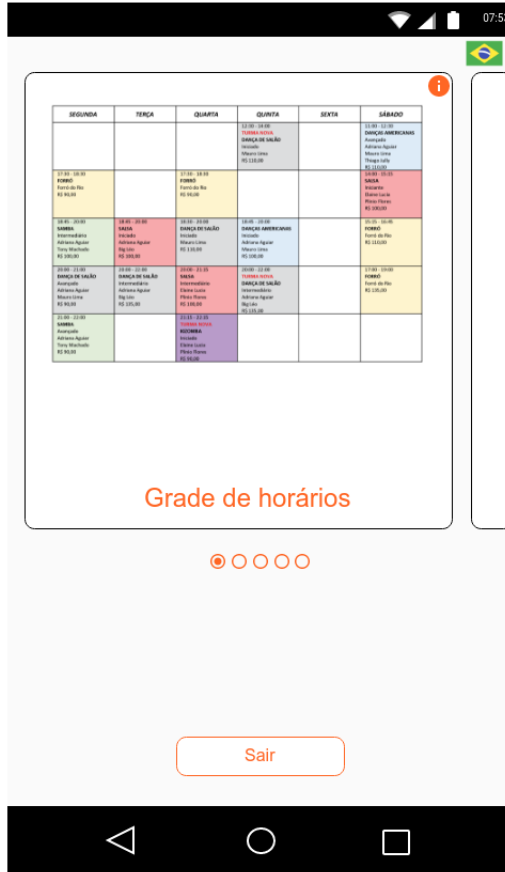


Figura 7 – Menu principal v2



Figura 8 – Menu principal v3



(1).png



Figura 9 – Grade de horários v1



Figura 10 – Grade de horários v2



Figura 11 – Definir critérios

←

Definir critérios para geração

Turno de estudo  Manhã  Tarde  Noite

Número máximo de créditos  +

Considerar matérias equivalentes?  Sim  Não

Matérias que quero cursar nesse semestre  +

Matérias que não quero cursar nesse semestre  +

Gerar grade de horários

Figura 12 – Login

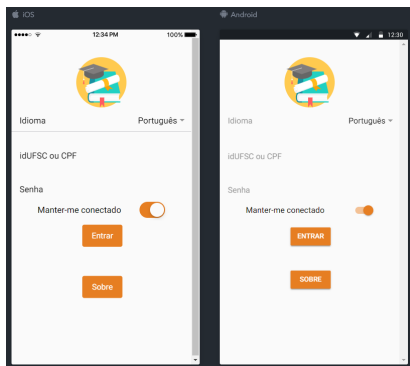


Figura 13 – Menu principal

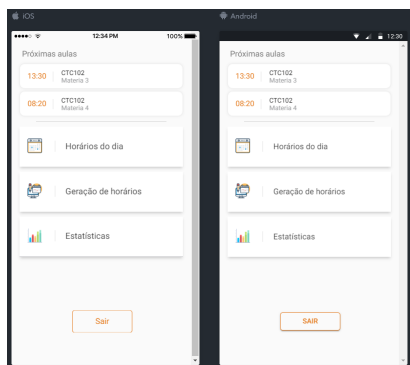


Figura 14 – Horário diário

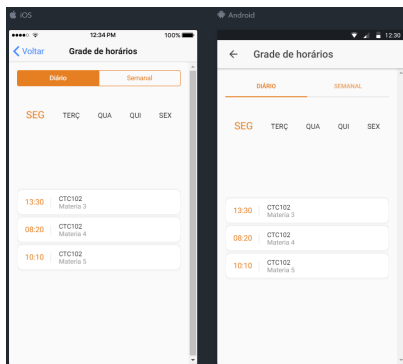


Figura 15 – Horário semanal

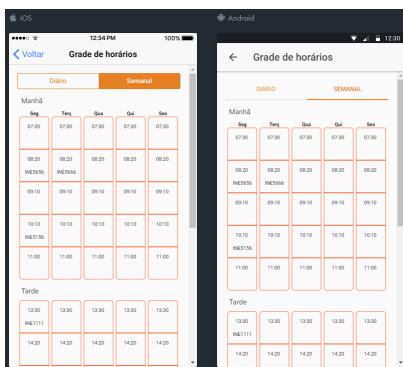


Figura 16 – Geração de grade de horários - Passo 1

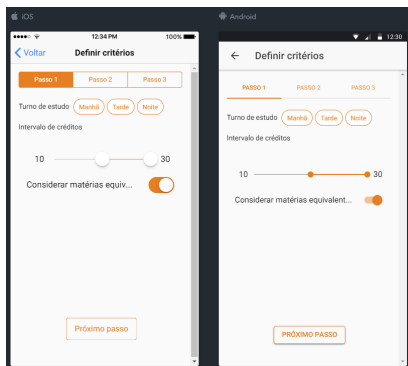


Figura 17 – Geração de grade de horários - Passo 2

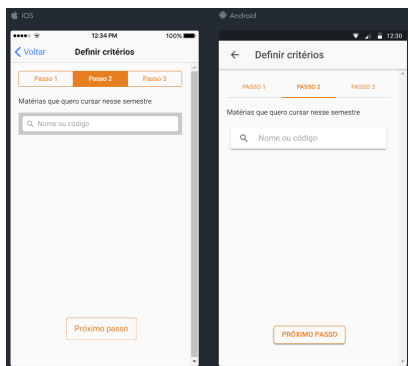


Figura 18 – Geração de grade de horários - Passo 3

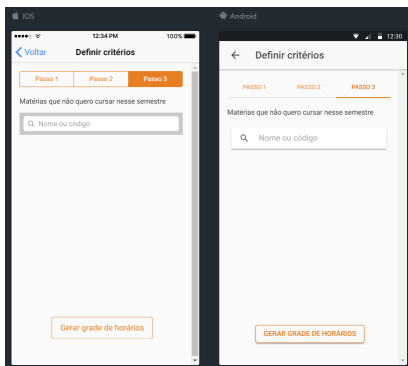


Figura 19 – Geração de grade de horários - Resultado

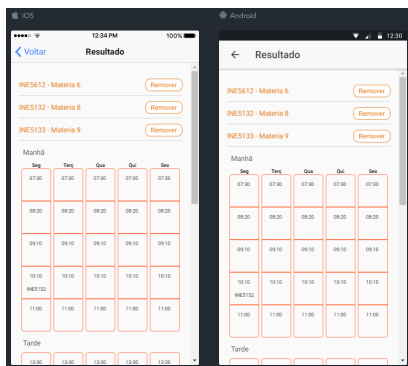


Figura 20 – Estatísticas

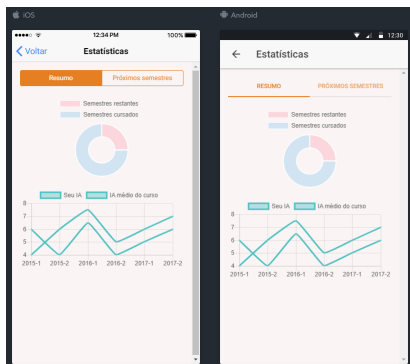
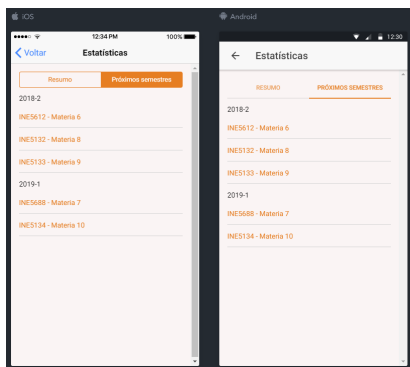


Figura 21 – Formatura e próximos semestres





## 8 CONCLUSÃO

Ao finalizar este trabalho podemos aprender mais sobre a importância do processo completo de desenvolvimento de software, bem como da relevância de cada uma das partes responsáveis por cada artefato dentro do desenvolvimento de um aplicativo. Um ponto importante a ser destacado foi a concepção conjunta do aplicativo, onde ambos os autores participaram ativamente, desde a elaboração dos protótipos, discussões sobre o design e alterações dos mesmos até a completude da documentação e desenvolvimento completo da aplicação.

Também é notória a preocupação com o design da aplicação, item que ambos sempre consideramos muito importante, porém esta foi a primeira vez que este foi testado e validado por usuários através de testes de usabilidade. Teste este que foi importante para o melhor entendimento do usuário final sobre a aplicação e sobre suas opiniões sobre a mesma.

### 8.1 TRABALHOS FUTUROS

Como trabalhos futuros, os autores destacam:

- **Integração com a SeTIC:** Conforme comentado no trabalho, os autores tentaram por várias vezes contato com a superintendência, porém sem sucesso. Seria muito interessante a implementação dessa integração, visto que o algoritmo para a montagem das grades de horários já está desenvolvido;
- **Correções de design:** Conforme comentado na seção de usabilidade, há alterações que podem ser realizadas no *app* para que este melhore ainda mais em sua usabilidade;
- **Disponibilização do *app*:** Uma vez implementadas as alterações colocadas nessa seção, seria interessante disponibilizar a aplicação para os alunos em geral.



## **9 DIREITOS AUTORAIS**

Os autores são os únicos responsáveis pelo conteúdo do material impresso incluído no seu trabalho.



## REFERÊNCIAS

- BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, Usability Professionals' Association, v. 4, n. 3, p. 114–123, 2009.
- CISCON, L. A.; ALVARENGA, G. O problema de geração de horários: um foco na eliminação de janelas e aulas isoladas. In: *XXXVII Brazilian Symposium of Operational Research*. [S.l.: s.n.], 2005.
- CYBIS, W. de A.; BETIOL, A. H.; FAUST, R. *Ergonomia e Usabilidade 3ª edição: Conhecimentos, Métodos e Aplicações*. [S.l.]: Novatec Editora, 2015.
- DABBISH, L. et al. Social coding in github: transparency and collaboration in an open software repository. In: *ACM. Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. [S.l.], 2012. p. 1277–1286.
- EA. EA. 2018. <<http://sparxsystems.com/products/ea/>>. Acessado em 01/02/2018.
- ESSELINK, B. *A Practical Guide to Localization*. John Benjamins Publishing Company, 2000. (Language international world directory). ISBN 9789027219565. <[https://books.google.com.br/books?id=QxFg5AC\\_JZIC](https://books.google.com.br/books?id=QxFg5AC_JZIC)>.
- FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, Irvine, 2000. [Http://www.ics.uci.edu/fielding/pubs/dissertation/top.htm](http://www.ics.uci.edu/fielding/pubs/dissertation/top.htm).
- FREITAS, C. C. et al. Uma ferramenta baseada em algoritmos genéticos para a geração de tabela de horário escolar. *SÉTIMA ESCOLA REGIONAL DE COMPUTAÇÃO Bahia-Sergipe. Vitória da Conquista:[sn]*, 2007.
- GITHUB. *Github*. 2018. <<https://github.com/>>. Acessado em 01/02/2018.

- HALDAR, M. *RESTful API Designing guidelines*. 2018. <<https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9>>. Acessado em 28/04/2018.
- HUYNH, M.; GHIMIRE, P.; TRUONG, D. Hybrid app approach: could it mark the end of native app domination? *Issues in Informing Science and Information Technology*, v. 14, p. 049–065, 2017.
- IONIC. *Ionic Documentation*. 2018. <<https://ionicframework.com/docs/>>. Acessado em 24/05/2018.
- JAFFAR, J.; LASSEZ, J.-L. Constraint logic programming. In: ACM. *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. [S.l.], 1987. p. 111–119.
- JEKYLL. *Jekyll*. 2018. <<https://jekyllrb.com/>>. Acessado em 01/02/2018.
- JÚNIOR, B.; OLIVEIRA, O. de et al. Otimização de horários em instituições de ensino superior através de algoritmos genéticos. Florianópolis, SC, 2000.
- KOOHANG, A. Expanding the concept of usability. *Informing Science: International Journal of an Emerging Transdiscipline*, Informing Science Institute, v. 7, p. 129–141, 2004.
- NIELSEN, J. Usability inspection methods. In: ACM. *Conference companion on Human factors in computing systems*. [S.l.], 1994. p. 413–414.
- NODEJS. *Node.js*. 2018. <<https://nodejs.org>>. Acessado em 01/02/2018.
- PASCHOARELLI, L. C.; SILVA, J. C. P. da. Design ergonômico: uma revisão dos seus aspectos metodológicos. *Conexão-Comunicação e Cultura*, v. 5, n. 10, 2006.
- RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594.
- SHNEIDERMAN, B. *Designing the user interface: strategies for effective human-computer interaction*. [S.l.]: Pearson Education India, 2010.

SILVA, D. J. da; SILVA, G. C. da. Heurísticas baseadas no algoritmo de coloração de grafos para o problema de alocação de salas em uma instituição de ensino superior. *Anais do XLII SBPO*, p. 2839–2849, 2010.

SOMMERVILLE, I. *Engenharia de Software*. Boston, United States: Editora Addison-Wesley, 2011. 592 p.

SOUZA, M. J. F.; COSTA, F.; GUIMARÃES, I. Um algoritmo evolutivo híbrido para o problema de programação de horários em escolas. *Computer*, 2002.

SPRING. *Spring Initializr*. 2018. <<https://start.spring.io/>>. Acessado em 28/04/2018.

SPRINGBOOT. *SpringBoot*. 2018. <<https://projects.spring.io/spring-boot/>>. Acessado em 12/05/2018.

SVN. *SVN*. 2018. <<https://subversion.apache.org/>>. Acessado em 01/02/2018.

TYPESCRIPT. *TypeScript - JavaScript that scales*. 2018. <<https://www.typescriptlang.org/>>. Acessado em 12/05/2018.

VIEIRA, F.; MACEDO, H. Sistema de alocação de horários de cursos universitários: um estudo de caso no departamento de computação da universidade federal de sergipe. *Scientia Plena*, v. 7, n. 3, 2011.

VOEGLER, J.; BORNSCHEIN, J.; WEBER, G. Markdown—a simple syntax for transcription of accessible study materials. In: SPRINGER. *International Conference on Computers for Handicapped Persons*. [S.l.], 2014. p. 545–548.





## 10ANEXOS

### 10.1 CÓDIGO FONTE APLICATIVO

```

import { StorageKeys } from '../../../utils/storage-
keys';
import { Api } from '../api/api';
import { Injectable } from '@angular/core';
import { Storage } from '@ionic/storage';

@Injectable()
export class EstatisticaProvider {

  constructor(public api: Api, private storage:
Storage) {

  }

  getEstatisticas(): any {
    let seq = this.api.get('stacticics',
['token'],
[this.storage.get(StorageKeys.TOKEN)]).share();

    return seq;
  }

}
import 'rxjs/add/operator/map';

import { Injectable } from '@angular/core';
import { Http, RequestOptions, URLSearchParams,
Headers } from '@angular/http';

/**
 * Api is a generic REST Api handler. Set your API
url first.
 */
@Injectable()
export class Api {
  url: string = 'http://localhost:8080/v1';
  // url: string = 'http://192.168.0.3:8080/v1';
  // url: string = 'http://
tccrest.pedro.pacheco.vms.ufsc.br:8080/v1';
  // A24cPmgdCw
  // private url: string = 'https://radiant-
wave-75942.herokuapp.com/v1';

  constructor(public http: Http) {
  }

  get(endpoint: string, params?: any, vals?: any,

```

```

options?: RequestOptions) {
    if (!options) {
        // let headers = new Headers();
        // headers.append('Access-Control-Allow-
Origin', '*');
        options = new RequestOptions();
    }

    // Support easy query params for GET requests
    if (params) {
        let p = new URLSearchParams();
        for (let k in params) {
            p.set(params[k], vals[k]);
        }
        // Set the search field if we have params
and don't already have
        // a search field set in options.
        options.search = !options.search && p ||
options.search;
    }

    return this.http.get(this.url + '/' +
endpoint, options);
}

post(endpoint: string, body: any, params?: any,
vals?: any, options?: RequestOptions) {
    let h: Headers = new Headers();
    h.append('Content-Type', 'application/json;
charset=UTF-8');

    if (!options) {
        options = new RequestOptions();
        options.headers = h;
    }

    if (params) {
        let p = new URLSearchParams();
        for (let k in params) {
            p.set(params[k], vals[k]);
        }
        options.search = !options.search && p ||
options.search;
    }
}

```

```

        return this.http.post(this.url + '/' +
endpoint, body, options);
    }

    put(endpoint: string, body: any, options?:
RequestOptions) {
        return this.http.put(this.url + '/' +
endpoint, body, options);
    }

    delete(endpoint: string, options?:
RequestOptions) {
        return this.http.delete(this.url + '/' +
endpoint, options);
    }

    patch(endpoint: string, body: any, options?:
RequestOptions) {
        return this.http.put(this.url + '/' +
endpoint, body, options);
    }
}
import { Api } from './api/api';
import { LoginProvider } from './login-provider/
login-provider';

export {
    Api,
    LoginProvider
};
import { Constraints } from './../models/
constraints';
import { Injectable } from '@angular/core';
import { Api } from '../providers';
import 'rxjs/add/operator/map';
import { StorageKeys } from './../utils/storage-
keys';
import { Storage } from '@ionic/storage';

@Injectable()
export class SubjectsProvider {

    constructor(public api: Api, private storage:
Storage) {
    }

```

```

    allSubjects(): any {
      let seq = this.api.get('subjects', ['token'],
[this.storage.get(StorageKeys.TOKEN)]).share();
      return seq;
    }

    calculateSemester(constraints: Constraints): any
    {
      let seq = this.api.post('calculate-semester',
constraints, ['token'],
[this.storage.get(StorageKeys.TOKEN)]).share();
      return seq;
    }
  }

import 'rxjs/add/operator/map';
import 'rxjs/add/operator/toPromise';

import { Injectable } from '@angular/core';
import { Http } from '@angular/http';

import { Api } from '../api/api';

@Injectable()
export class LoginProvider {
  _user: any;

  constructor(public http: Http, public api: Api) {
  }

  login(token: string) {
    let seq = this.api.get('login', ['token'],
[token]).share();
    return seq;
  }

  authSetic(code: string) {
    // let api_key = 'oauth';
    // let api_key = 'tccphpils';
    // let secret_key = 'segredo';
    // let secret_key =
'Gto1W3ErSqWdmASpb5CsqqPkgjNv8';
    // let redirect_url = 'ufsclogin://
setic_oauth_example.ufsc.br';
    // let redirect_url = 'tccphpils://

```

```

tccphpils.setic_oauth.ufsc.br';
    // let site = 'https://
sistemas.homologacao.ufsc.br/oauth2.0/';
    // let site = 'https://sistemas.ufsc.br/
oauth2.0/';

    // let url = site + 'accessToken?
grant_type=authorization_code&code=' + code +
'&client_id=' + api_key + '&redirect_uri=' +
redirect_url + '&client_secret=' + secret_key +
'&username=tccphpils&password=VLwwAvvHdNzF';

    // .appendQueryParameter(GRANT_TYPE_PARAM,
GRANT_TYPE)
    // .appendQueryParameter(RESPONSE_TYPE_VALUE,
authorizationToken)
    // .appendQueryParameter(CLIENT_ID_PARAM,
API_KEY)
    // .appendQueryParameter(REDIRECT_URI_PARAM,
REDIRECT_URI)
    // .appendQueryParameter(SECRET_KEY_PARAM,
SECRET_KEY)
    // let url = 'https://
sistemas.homologacao.ufsc.br/oauth2.0/accessToken?
grant_type=authorization_code&code='+code+'&client_id=oauth&redi
// let seq = this.http.post(url, "");
// return seq;
}

}
import { Injectable } from '@angular/core';
import { Api } from '../providers';
import { StorageKeys } from '../../utils/storage-
keys';
import { Storage } from '@ionic/storage';

@Injectable()
export class ScheduleProvider {

    constructor(public api: Api, private storage:
Storage) {

    }

    schedule(semester: string): any {
        let seq = this.api.get('schedule/' + semester,
['token'],

```

```

[this.storage.get(StorageKeys.TOKEN)].share();

    return seq;
}

}

import { platformBrowserDynamic } from '@angular/
platform-browser-dynamic';

import { AppModule } from './app.module';

platformBrowserDynamic().bootstrapModule(AppModule);
import { Component, ViewChild } from '@angular/
core';
import { SplashScreen } from '@ionic-native/splash-
screen';
import { StatusBar } from '@ionic-native/status-
bar';
import { TranslateService } from '@ngx-translate/
core';
import { Config, Nav, Platform } from 'ionic-
angular';

@Component({
  template: `export class MyApp {
  rootPage = 'LoginPage';
}

```

```

@ViewChild(Nav) nav: Nav;

pages: any[] = [
  { title: 'Welcome', component: 'WelcomePage' },
  { title: 'Tabs', component: 'TabsPage' },
  { title: 'Cards', component: 'CardsPage' },
  { title: 'Content', component: 'ContentPage' },
  { title: 'Login', component: 'LoginPage' }, //
  { title: 'Signup', component: 'SignupPage' },
  { title: 'Map', component: 'MapPage' },
  { title: 'Master Detail', component:
'ListMasterPage' },
  { title: 'Menu', component: 'MenuPage' },
  { title: 'Settings', component:
'SettingsPage' },
  { title: 'Search', component: 'SearchPage' },
  { title: 'About', component: 'AboutPage' }, //
  { title: 'Main', component: 'MainPage' }, //
  { title: 'Definir Critérios', component:
'DefineConstraintsPage' }, //
  { title: 'Result', component: 'ResultPage' }
]

constructor(private translate: TranslateService,
private platform: Platform, private config:
Config, private statusBar: StatusBar, private
splashScreen: SplashScreen) {
  this.initTranslate();
}

ionViewDidLoad() {
  this.platform.ready().then(() => {
    // Okay, so the platform is ready and our
    plugins are available.
    // Here you can do any higher level native
    things you might need.
    this.statusBar.styleDefault();
    this.splashScreen.hide();
  });
}

initTranslate() {
  // Set the default language for translation
  strings, and the current language.
  this.translate.setDefaultLang('en');
}

```



```

    if (this.translate.getBrowserLang() !==
undefined) {

this.translate.use(this.translate.getBrowserLang());
    } else {
        this.translate.use('en'); // Set your
language here
    }

this.translate.get(['BACK_BUTTON_TEXT']).subscribe(values
=> {
    this.config.set('ios', 'backButtonText',
values.BACK_BUTTON_TEXT);
    });
}

openPage(page) {
    // Reset the content nav to have just this page
    // we wouldn't want the back button to show in
this scenario
    this.nav.setRoot(page.component);
}
}
// http://ionicframework.com/docs/v2/theming/

// App Global Sass
//
-----
// Put style rules here that you want to apply
globally. These
// styles are for the entire app and not just one
component.
// Additionally, this file can be also used as an
entry point
// to import other Sass files to be included in
the output CSS.
//
// Shared Sass variables, which can be used to
adjust Ionic's
// default Sass variables, belong in "theme/
variables.scss".
//
// To declare rules for a specific mode, create a

```

```

child rule
// for the .md, .ios, or .wp mode classes. The
mode class is
// automatically applied to the <body> element in
the app.
import { CalendarUtils } from '../utils/
calendar';
import { ErrorHandler, NgModule } from '@angular/
core';
import { Http, HttpClientModule } from '@angular/http';
import { BrowserModule } from '@angular/platform-
browser';
import { Camera } from '@ionic-native/camera';
import { GoogleMaps } from '@ionic-native/google-
maps';
import { SplashScreen } from '@ionic-native/splash-
screen';
import { StatusBar } from '@ionic-native/status-
bar';
import { IonicStorageModule } from '@ionic/
storage';
import { TranslateLoader, TranslateModule } from
'@ngx-translate/core';
import { TranslateHttpLoader } from '@ngx-
translate/http-loader';
import { IonicApp, IonicErrorHandler,
IonicModule } from 'ionic-angular';
import { InAppBrowser } from '@ionic-native/in-app-
browser';
import { Deeplinks } from '@ionic-native/
deeplinks';

import { LoginProvider } from '../providers/
providers';
import { Api } from '../providers/providers';
import { MyApp } from './app.component';
import { EstadisticaProvider } from '../providers/
estadistica/estadistica';
import { SubjectsProvider } from '../providers/
subjects/subjects';
import { ScheduleProvider } from '../providers/
horarios/schedule-provider';

// The translate loader needs to know where to
load i18n files
// in Ionic's static asset pipeline.

```

```

export function HttpLoaderFactory(http: Http) {
  return new TranslateHttpLoader(http, './assets/
i18n/', '.json');
}

@NgModule({
  declarations: [
    MyApp,
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    TranslateModule.forRoot({
      loader: {
        provide: TranslateLoader,
        useFactory: HttpLoaderFactory,
        deps: [Http]
      }
    }),
    IonicModule.forRoot(MyApp),
    IonicStorageModule.forRoot()
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp
  ],
  providers: [
    Api,
    LoginProvider,
    Camera,
    GoogleMaps,
    SplashScreen,
    StatusBar,
    // Keep this to enable Ionic's runtime error
handling during development
    { provide: ErrorHandler, useClass:
IonicErrorHandler },
    EstadisticaProvider,
    ScheduleProvider,
    CalendarUtils,
    SubjectsProvider,
    InAppBrowser,
    Deeplinks
  ]
})
export class AppModule { }

```

```

export class Subject {
    constructor(public nome: string, public
codigo: string, public fase: number, public
professor: string, public aulas: number, public
obrigatoria: boolean, public horarios: string[],
public requisitos: string[]) { }
}export class Dia {
    constructor(public nome: string, public
nomeAbreviado: string, public visible: boolean) { }
}export class Statistics {
    constructor(public semestersRemaining: number,
public semesters: string[], public semestersIA:
number[], public courseIA: number[]) {
    }
    I
}export class Account {
    constructor(public username: string, public
password: string) { }
}export class Constraints {
    constructor(public periods: number[], public
creditMin: number, public creditMax: number, public
equivalent: boolean, public subjectsWanted:
string[], public subjectsNotWanted: string[]) {
    }
}import { Subject } from "./subject";
export class Semestre {
    constructor(public semestre: string, public
subjects: Subject[]) { }
}
// Ionic Variables and Theming. For more info,
// please see:
// http://ionicframework.com/docs/v2/theming/
font-path: "../assets/fonts";

```

```

@import "ionic.globals";

// Shared Variables
//
-----
// To customize the look and feel of this app, you
// can override
// the Sass variables found in Ionic's source scss
// files.
// To view all the possible Ionic variables, see:
// http://ionicframework.com/docs/v2/theming/overriding-ionic-variables/

$text-color:          #000;
$background-color:   #fafafa;

// Named Color Variables
//
-----
// Named colors makes it easy to reuse colors on
// various components.
// It's highly recommended to change the default
// colors
// to match your app's branding. Ionic uses a Sass
// map of
// colors so you can add, rename and remove colors
// as needed.
// The "primary" color is the only required color
// in the map.

$colors: (
  primary:    #488aff,
  secondary:  #32db64,
  danger:     #f53d3d,
  light:      #faf4f4,
  dark:       #222,
  main-color: #e67e22,
);

// App iOS Variables
//
-----
// iOS only Sass variables can go here

```

```
// App Material Design Variables
//
-----
// Material Design only Sass variables can go here

// App Windows Variables
//
-----
// Windows only Sass variables can go here

// App Theme
//
-----
// Ionic apps can have different themes applied,
// which can
// then be future customized. This import comes
// last
// so that the above variables are used and Ionic's
// default are overridden.
@import "ionic.theme.default";

// Ionicons
//
-----
// The premium icon font for Ionic. For more info,
// please see:
// http://ionicframework.com/docs/v2/ionicons/
@import "ionic.ionicons";

// Fonts
//
-----
```

```

@import "roboto";
@import "noto-sans";
page-statistics {

    .scroll-content {
        background-color: map-get($colors, light);
    }

    ion-card-header {
        font-weight: bold;
    }

    .step-hidden {
        display: none;
    }
}

.chart-margin {
    margin-top: 20px !important;
}<ion-header>
  <ion-navbar>
    <ion-title>{{ 'STATISTICS' | translate }}</ion-
title>
  </ion-navbar>
</ion-header>

<ion-content padding>

  <ion-segment
[(ngModel)]="step" (ionChange)="onStepChanged()"
color="main-color">
    <ion-segment-button value="1">
        {{ 'SUMMARY' | translate }}
    </ion-segment-button>
    <ion-segment-button value="2">
        {{ 'NEXT_SEMESTERS' | translate }}
    </ion-segment-button>
  </ion-segment>

  <ion-list [ngClass]="getClass('1')">
    <canvas class="chart-margin"
#doughnutCanvas></canvas>

    <canvas class="chart-margin" #lineCanvas></
canvas>

```

```

</ion-list>

<ion-list [ngClass]="getClass('2')">
  <div *ngFor='let semester of semestersYears'>
    <p>{{ semester }}</p>
    <subject-action-list
[subjects]="getSubjects(semester)" [action]=false></
subject-action-list>
  </div>
</ion-list>

</ion-content>import { StorageKeys } from './../../../../
utils/storage-keys';
import { Statistics } from './../../../../models/
statistics';
import { Component, ViewChild } from '@angular/
core';
import { IonicPage, NavController, NavParams }
from 'ionic-angular';
import { Storage } from '@ionic/storage';

import { Chart } from 'chart.js';
import { EstadisticaProvider } from './../../../../
providers/estadistica/estadistica';
import { TranslateService } from '@ngx-translate/
core';

@IonicPage()
@Component({
  selector: 'page-statistics',
  templateUrl: 'statistics.html',
})
export class StatisticsPage {

  @ViewChild('doughnutCanvas') doughnutCanvas;
  @ViewChild('lineCanvas') lineCanvas;

  doughnutChart: any;
  lineChart: any;

  public statistic: Statistics;

  public semestersYears: any[] = [];

  public semesters: any;

```



```

step: string = '1';

constructor(public navCtrl: NavController,
public navParams: NavParams,
public estadisticaProvider:
EstadisticaProvider,
private storage: Storage,
public translateService: TranslateService)
{
}

ionViewDidLoad() {
this.storage.get(StorageKeys.RESULT).then(res => {
    if (res) {
        for (var property in res) {
            if
(res.hasOwnProperty(property)) {
this.semestersYears.push(property);
            }
        }
        this.semesters = res;
    }
});

this.storage.get(StorageKeys.STATISTIC).then(e => {
    if (e) {
        this.statistic = e;
        this.showDoughnutChart();
        this.showLineChart();
    }
});

this.estadisticaProvider.getEstadisticas()
    .map(res => res.json())
    .subscribe(e => {
        this.statistic = e;
    });

this.storage.set(StorageKeys.STATISTIC, e);
    this.showDoughnutChart();
    this.showLineChart();
});
}

```

```

onStepChanged() {
}

getSubjects(semester: string) {
    return this.semesters[semester].subjects;
}

showLineChart() {
    this.lineChart = new
Chart(this.lineCanvas.nativeElement, {
    type: 'line',
    data: {
        labels: this.statistic.semesters,
        datasets: [
            {
                label:
this.translateService.instant('YOUR_IA'),
                fill: false,
                lineTension: 0.1,
                backgroundColor:
"rgba(75,192,192,0.4)",
                borderColor:
"rgba(75,192,192,1)",
                borderCapStyle: 'butt',
                borderDash: [],
                borderDashOffset: 0.0,
                borderJoinStyle: 'miter',
                pointBorderColor:
"rgba(75,192,192,1)",
                pointBackgroundColor:
"#fff",
                pointBorderWidth: 1,
                pointHoverRadius: 5,
                pointHoverBackgroundColor:
"rgba(75,192,192,1)",
                pointHoverBorderColor:
"rgba(220,220,220,1)",
                pointHoverBorderWidth: 2,
                pointRadius: 1,
                pointHitRadius: 10,
                data:
this.statistic.semestersIA,
                spanGaps: false,
            },
            {

```

```

        label:
this.translateService.instant('AVERAGE_COURSE_IA'),
        fill: false,
        lineTension: 0.1,
        backgroundColor:
"rgba(75,192,192,0.4)",
        borderColor:
"rgba(75,192,192,1)",
        borderCapStyle: 'butt',
        borderDash: [],
        borderDashOffset: 0.0,
        borderJoinStyle: 'miter',
        pointBorderColor:
"rgba(75,192,192,1)",
        pointBackgroundColor:
"#fff",
        pointBorderWidth: 1,
        pointHoverRadius: 5,
        pointHoverBackgroundColor:
"rgba(75,192,192,1)",
        pointHoverBorderColor:
"rgba(220,220,220,1)",
        pointHoverBorderWidth: 2,
        pointRadius: 1,
        pointHitRadius: 10,
        data:
this.statistic.courseIA,
        spanGaps: false,
    }
    ]
}
});
}

showDoughnutChart() {
    this.doughnutChart = new
Chart(this.doughnutCanvas.nativeElement, {
        type: 'doughnut',
        data: {
            labels:
[this.translateService.instant('SEMESTERS_REMAINING'),
this.translateService.instant('SEMESTERS_STUDIED')],
            datasets: [{

```

```

        label: '# of Votes',
        data:
[this.semestersYears.length,
this.statistic.semesters.length],
        backgroundColor: [
            'rgba(255, 99, 132, 0.2)',
            'rgba(54, 162, 235, 0.2)'
        ],
        hoverBackgroundColor: [
            "#FF6384",
            "#36A2EB"
        ]
    ]
    }
}
});
}

getClass(p: string): string {
    return this.step == p ? "" : "step-
hidden";
}

}
import { ComponentsModule } from './../.././
components/components.module';
import { TranslateModule } from '@ngx-translate/
core';
import { NgModule } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { StatisticsPage } from './statistics';

@NgModule({
  declarations: [
    StatisticsPage,
  ],
  imports: [
    IonicPageModule.forChild(StatisticsPage),
    TranslateModule.forChild(),
    ComponentsModule
  ],
})
export class StatisticsPageModule {
import { Constraints } from './../.././models/
constraints';
import { StorageKeys } from './../.././utils/storage-

```

```

keys';
import { Component, ViewChild } from '@angular/
core';
import { IonicPage, NavController, NavParams,
AlertController } from 'ionic-angular';
import { TranslateService } from '@ngx-translate/
core';
import { Storage } from '@ionic/storage';

import { CapsuleComponent } from '../../components/
capsule/capsule';
import { Subject } from '../../models/subject';
import { FormatterUtils } from '../../utils/
formatter';
import { SubjectsProvider } from '../../../
providers/subjects/subjects';

@IonicPage()
@Component({
  selector: 'page-define-constraints',
  templateUrl: 'define-constraints.html',
})
export class DefineConstraintsPage {

  placeholder =
this.translate.instant('NAME_OR_CODE');

  step: string = '1';

  private button: string = this.step == "3" ?
this.translate.instant('GENERATE_TIME_GRID') :
this.translate.instant('NEXT_STEP');

  private periodsSelected: string[] = [];

  private subjectsWanted: Subject[] = [];

  private subjectsExcluded: Subject[] = [];

  @ViewChild(CapsuleComponent) capsuleComponent;

  busca: string;

  private subjects;

  credits = {

```

```

        lower: 20,
        upper: 30
    }

    equivalent = true;

    constructor(public navCtrl: NavController,
                public navParams: NavParams,
                public alertCtrl: AlertController,
                public translate: TranslateService,
                private subjectsProvider: SubjectsProvider,
                private storage: Storage) {

    ionViewDidLoad() {
=> {
        this.storage.get(StorageKeys.RESULT).then((val)
=> {
            if (val) {
                this.navCtrl.push('ResultPage');
            }
        });

        this.storage.get(StorageKeys.ALL_SUBJECTS).then((val)
=> {
            if (val) {
                this.subjects = val;
            }
        });

        this.subjectsProvider.allSubjects()
            .map(res => res.json())
            .subscribe(res => {
                if (res.success) {
                    this.subjects = res.result.subjects;
                }
            });

        this.storage.set(StorageKeys.ALL_SUBJECTS,
        this.subjects);
    }, err => {
        console.error('ERROR', err);
    });
}

ionViewWillEnter(){
    this.step == '1';
}

```

```

}

onPeriodSelected(event: string[]) {
    this.periodsSelected = event;
}

searchSubject(): void {
    this.doCheckbox(this.busca);
}

getClass(step: string): string {
    if (this.step == step) {
        return "step";
    } else {
        return "step step-hidden";
    }
}

onStepChanged(event: any): void {
    this.button = this.step == "3" ?
this.translate.instant('GENERATE_TIME_GRID') :
this.translate.instant('NEXT_STEP');
}

btnNextStepClicked(): void {
    if (this.step == '3') {
        let constraints: Constraints =
this.createConstraints();
        this.storage.set(StorageKeys.CONSTRAINT,
constraints);
        this.navCtrl.push('ResultPage');
    } else {
        this.step = (Number(this.step) +
1).toString();
    }
}

createConstraints(): Constraints {
    let periods: number[] = [];

    if
(this.periodsSelected.indexOf(this.translate.instant('MORNING'))
> -1) {
        periods.push(0);
    }
    if

```

```

    (this.periodsSelected.indexOf(this.translate.instant('AFTERNOON'
> -1) {
        periods.push(1);
    }
    if
    (this.periodsSelected.indexOf(this.translate.instant('NIGHT'))
> -1) {
        periods.push(2);
    }

    let subjectsWantedCode: string[] = [];
    this.subjectsWanted.forEach(s =>
subjectsWantedCode.push(s.codigo));

    let subjectsNotWantedCode: string[] = [];
    this.subjectsExcluded.forEach(s =>
subjectsNotWantedCode.push(s.codigo));

    return new Constraints(periods,
this.credits.lower, this.credits.upper,
this.equivalent, subjectsWantedCode,
subjectsNotWantedCode);
    }

    getPeriods(): string[] {
        let periods: string[] = [];

periods.push(this.translate.instant('MORNING'));

periods.push(this.translate.instant('AFTERNOON'));
        periods.push(this.translate.instant('NIGHT'));

    return periods;
    }

    getSubjectsWanted() {
        return this.subjectsWanted;
    }

    getSubjectsExcluded() {
        return this.subjectsExcluded;
    }

    doCheckbox(search: string) {
        let alert = this.alertCtrl.create();

```



```

alert.setTitle(this.translate.instant('SELECT_SUBJECT'));

let subjects: Subject[] =
this.getPossibleSubjects();
subjects.forEach(s => {
  alert.addInput({
    type: 'checkbox',
    label: s.codigo + " - " + s.nome,
    value: s.codigo
  })
})

alert.addButton(this.translate.instant('BACK_BUTTON_TEXT'));
alert.addButton({
  text: 'Ok',
  handler: (data: any) => {
    this.busca = "";
    if (data) {
      data.forEach(element => {
        console.log(element);
        if (this.step == '2') {

this.subjectsWanted.push(this.getSubjectByCode(element));
        } else if (this.step == '3') {

this.subjectsExcluded.push(this.getSubjectByCode(element));
        }
      });
    }
  })
});

alert.present();
}

getSubjectByCode(code: string): Subject {
  for (let i in this.subjects) {
    let subject: Subject = this.subjects[i];
    if (subject.codigo == code) {
      return subject;
    }
  }
}

```

```

    return undefined;
}

getPossibleSubjects(): Subject[] {
    let subjects: Subject[] = [];

    for (let i in this.subjects) {
        let subject: Subject = this.subjects[i];
        let nome: string = subject.nome;
        let codigo: string = subject.codigo;

        if (this.contains(nome, this.busca) ||
this.contains(codigo, this.busca)) {
            subjects.push(subject);
        }
    }

    return subjects;
}

contains(a: string, b: string): boolean {
    return
FormatterUtils.replaceSpecialChars(a).includes(FormatterUtils.repl
}
}

page-define-constraints {

    .next-step-button {
        border: 1px solid color($colors, main-
color);
        background-color: $background-color;
        color: color($colors, main-color);
        position: absolute;
        left: 50%;
        transform: translate(-50%, -50%);
        border-radius: 5px;
    }

}

.step {
    margin-top: 20px;
    height: 80%
}

```

```

.step-hidden {
  display: none;
}

.button-md:hover:not(.disable-hover) {
  background-color: $background-color !important;
}

.activated {
  background-color: white !important;
}

import { NgModule } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { DefineConstraintsPage } from './define-
constraints';
import { ComponentsModule } from '../components/
components.module';
import { TranslateModule } from '@ngx-translate/
core';

@NgModule({
  declarations: [
    DefineConstraintsPage,
  ],
  imports: [

IonicPageModule.forChild(DefineConstraintsPage),
    TranslateModule.forChild(),
    ComponentsModule
  ],
})
export class DefineConstraintsPageModule {}

<ion-header>

  <ion-navbar>
    <ion-title>{{ 'DEFINE_CRITERIA' | translate }}
  </ion-title>
  </ion-navbar>

</ion-header>

<ion-content padding>
  <ion-segment
[(ngModel)]="step" (ionChange)="onStepChanged($event)"

```

```

color="main-color">
  <ion-segment-button value="1">
    {{ 'STEP_1' | translate }}
  </ion-segment-button>
  <ion-segment-button value="2">
    {{ 'STEP_2' | translate }}
  </ion-segment-button>
  <ion-segment-button value="3">
    {{ 'STEP_3' | translate }}
  </ion-segment-button>
</ion-segment>

<div [ngClass]="getClass('1')">
  <span>{{ 'STUDY_SHIFT' | translate }}</span>
  <capsule
[values]="getPeriods()" (valueSelected)="onPeriodSelected($event
capsule>

    <p>{{ 'CREDIT_INTERVAL' | translate }}</p>
    <ion-item>
      <ion-range dualKnobs="true"
pin="true" [(ngModel)]="credits" color="main-
color" min=10 max=30>
        <ion-label range-left>10</ion-label>
        <ion-label range-right>30</ion-label>
      </ion-range>
    </ion-item>

    <ion-item>
      <ion-label>{{ 'CONSIDER_EQUIVALENT_COURSES'
| translate }}</ion-label>
      <ion-toggle color="main-color"
checked="false" [(ngModel)]="equivalent"></ion-
toggle>
    </ion-item>
  </div>

  <div [ngClass]="getClass('2')">
    <p>{{ 'SUBJECTS_I_WANT_TO_STUDY' | translate }}
</p>
    <ion-searchbar (search)="searchSubject()"
placeholder='{{placeholder}}' [(ngModel)]="busca"></
ion-searchbar>
    <subject-action-list
[subjects]="getSubjectsWanted()"></subject-action-
list>

```

```

</div>

<div [ngClass]="getClass('3')">
  <p>{{ 'SUBJECTS_I_DONT_WANT_TO_STUDY' |
translate }}</p>
  <ion-searchbar (search)="searchSubject()"
placeholder='{{placeholder}}' [(ngModel)]="busca"></
ion-searchbar>
  <subject-action-list
[subjects]="getSubjectsExcluded()"></subject-
action-list>
</div>

<button ion-button class="next-step-
button" (click)="btnNextStepClicked()">{{ button }}
</button>

</ion-content>page-schedule {

  .div-daily {
    margin-top: 50px;
    height: 75%;
  }

  .days {
    display: flex;
    width: 100%;
    justify-content: space-around;
  }

  .day-hidden {
    display: none;
  }

  .day-selected {
    color: #e67e22;
    font-size: 20px;
    position: relative;
    bottom: 5px;
  }

  .step-hidden {
    display: none;
  }

  .segment-sticky {

```

```

        position: sticky;
        top: 0px;
        background-color: $background-color;
    }
}
import { PipesModule } from './../../pipes/
pipes.module';
import { NgModule } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { SchedulePage } from './schedule';
import { TranslateModule } from '@ngx-translate/
core';
import { ComponentsModule } from './../../components/
components.module';

@NgModule({
  declarations: [
    SchedulePage
  ],
  imports: [
    IonicPageModule.forChild(SchedulePage),
    TranslateModule.forChild(),
    ComponentsModule,
    PipesModule
  ],
})
export class SchedulePageModule {
import { StorageKeys } from './../../utils/storage-
keys';
import { SubjectHelper } from './../../utils/
subject-helper';
import { Component, ViewChild } from '@angular/
core';
import { Storage } from '@ionic/storage';
import { IonicPage, NavController, NavParams,
Slides } from 'ionic-angular';

import { CalendarUtils } from './../../utils/
calendar';
import { Dia } from './../../models/dia';
import { Subject } from './../../models/subject';

@IonicPage()
@Component({
  selector: 'page-schedule',

```

```

    templateUrl: 'schedule.html',
  })
  export class SchedulePage {
    private step: string;
    private day: number;
    private days: Dia[];
    private subjects: Subject[];
    @ViewChild('slides') slides: Slides;
    constructor(public navCtrl: NavController,
                public navParams: NavParams,
                private storage: Storage,
                private cal: CalendarUtils) {
      this.day = new Date().getDay() - 1; // starts
monday
      if (this.day < 0) {
        this.day = 0;
      }
      this.days = this.cal.getAllDias();

      this.storage.get(StorageKeys.SCHEDULE).then(d
=> {
        if (d) {
          this.subjects = d;
        }
      });
    }
    ionViewDidLoad(){
      this.step = '1';
    }
    onStepChanged(): void {
    }
    onSlideChanged(event: any): void {
      if (event._activeIndex < 5 &&
event._activeIndex >= 0) {
        this.day = event._activeIndex;
      }
    }
  }

```

```

}

onSwiped(event: boolean): void {
    if (event) {
        this.swipeRight();
    } else {
        this.swipeLeft();
    }
}

swipeRight(): void {
    if (this.day < 5) {
        this.day = this.day + 1;
    }
    this.slides.slideTo(this.day);
}

swipeLeft(): void {
    if (this.day > 0) {
        this.day = this.day - 1;
    }
    this.slides.slideTo(this.day);
}

onDayClicked(day: Dia): void {
    this.day = this.days.indexOf(day);
    this.slides.slideTo(this.day);
}

getSubjects(day?: string): Subject[] {
    return SubjectHelper.list(this.subjects, day);
}

getAllSubjects(): Subject[] {
    return SubjectHelper.list(this.subjects);
}

getClassDay(day: Dia): string {
    let clazz: string = "";
    if (!day.visible) {
        clazz += "day-hidden ";
    }
    if (this.days[this.day].nomeAbreviado ==
day.nomeAbreviado) {
        clazz += "day-selected";
    }
}

```



```

    return clazz;
  }

  getClass(step: string): string {
    if (this.step == step) {
      return "";
    } else {
      return "step-hidden";
    }
  }
}

<ion-header>
  <ion-navbar>
    <ion-title>{{ 'SCHEDULE_GRID' | translate }}</
ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>

  <ion-segment
  [(ngModel)]="step" (ionChange)="onStepChanged()"
  color="main-color" class="segment-sticky">
    <ion-segment-button value="1">
      {{ 'DAILY' | translate }}
    </ion-segment-button>
    <ion-segment-button value="2">
      {{ 'WEEKLY' | translate }}
    </ion-segment-button>
  </ion-segment>

  <div [ngClass]="getClass('1')" class="div-daily">
    <div class="days">
      <span *ngFor="let day of
days" [ngClass]="getClassDay(day)" (click)="onDayClicked(day)">{
</span>
    </div>
    <ion-slides
  [initialSlide]="day" [spaceBetween]=5 #slides
  (ionSlideWillChange)="onSlideChanged($event)">
      <ion-slide>
        <subject-list
  [subjects]="getSubjects('2')" (onSwiped)="onSwiped($event)"></
subject-list>
      </ion-slide>

```

```

    <ion-slide>
      <subject-list
[subjects]="getSubjects('3')" (onSwiped)="onSwiped($event)"></
subject-list>
    </ion-slide>
    <ion-slide>
      <subject-list
[subjects]="getSubjects('4')" (onSwiped)="onSwiped($event)"></
subject-list>
    </ion-slide>
    <ion-slide>
      <subject-list
[subjects]="getSubjects('5')" (onSwiped)="onSwiped($event)"></
subject-list>
    </ion-slide>
    <ion-slide>
      <subject-list
[subjects]="getSubjects('6')" (onSwiped)="onSwiped($event)"></
subject-list>
    </ion-slide>
  </ion-slides>
</div>

```

```

    <div [ngClass]="getClass('2')">
      <week-schedule [subjects]="getAllSubjects()"></
week-schedule>
    </div>

```

```

</ion-content>import { NgModule } from '@angular/
core';
import { IonicPageModule } from 'ionic-angular';
import { ResultPage } from './result';
import { ComponentsModule } from '../components/
components.module';
import { TranslateModule } from '@ngx-translate/
core';

```

```

@NgModule({
  declarations: [
    ResultPage
  ],
  imports: [
    IonicPageModule.forChild(ResultPage),
    TranslateModule.forChild(),
    ComponentsModule

```

```

    ],
  })
  export class ResultPageModule {}
  <ion-header>
    <ion-navbar>
      <ion-title>{{ 'RESULT' | translate }}</ion-
title>
    </ion-navbar>
  </ion-header>

  <ion-content padding>

    <subject-action-list
[subjects]="getSubjects()"></subject-action-list>

    <week-schedule [subjects]="getSubjects()"></week-
schedule>

    <button ion-button
(click)='onRedefineConstraintsClicked()'
class="button-
redefine">{{ 'RESET_GENERATION_CRITERIA' |
translate }}</button>

  </ion-content>
  import { ModalHelper } from '../utils/modal-
helper';
  import { Constraints } from '../models/
constraints';
  import { StorageKeys } from '../utils/storage-
keys';
  import { Component, ViewChild } from '@angular/
core';
  import { IonicPage, NavController, NavParams,
AlertController } from 'ionic-angular';
  import { Storage } from '@ionic/storage';
  import { SubjectsProvider } from '../providers/
subjects/subjects';
  import { Subject } from '../models/subject';
  import { TranslateService } from '@ngx-translate/
core';
  import { Navbar } from 'ionic-angular';

  @IonicPage()
  @Component({
    selector: 'page-result',

```

```

        templateUrl: 'result.html',
    })
    export class ResultPage {

        private constraints: Constraints;

        private subjects: Subject[];

        @ViewChild(Navbar) navBar: Navbar;

        constructor(public navCtrl: NavController,
            public navParams: NavParams,
            private storage: Storage,
            private subjectsProvider: SubjectsProvider,
            public translateService: TranslateService,
            public alertCtrl: AlertController) {
        }

        ionViewDidLoad() {
            this.navBar.backButtonClick = (e: UIEvent) => {
                this.navCtrl.popToRoot();
            }

            this.storage.get(StorageKeys.RESULT).then(res
=> {
                if (res) {
                    this.subjects = res[Object.keys(res)
[0]].subjects;
                }
            });

            this.storage.get(StorageKeys.CONSTRAINT).then((val)
=> {
                if (val) {
                    this.constraints = val;
                }
            });

            this.subjectsProvider.calculateSemester(this.constraints)
                .map(res => res.json())
                .subscribe(res => {
                    if (res.success) {
                        let subjectsError: Subject[] =
res.result.subjectsWantedError;
                        if (subjectsError.length > 0) {
                            let modal = new
ModalHelper(this.translateService, this.alertCtrl);

```

```

modal.createModalError(subjectsError);
    }
    this.storage.set(StorageKeys.RESULT,
res.result.nextSemesters);

this.storage.remove(StorageKeys.CONSTRAINT);
    this.subjects =
res.result.nextSemesters[Object.keys(res.result.nextSemesters)
[0]].subjects;
    }
    }, err => {
        console.error('ERROR', err);
    });
    }
});

// se demorar criar um loading
}

getSubjects() {
    return this.subjects;
}

onRedefineConstraintsClicked() {
    this.storage.remove(StorageKeys.RESULT);
    let last = this.navCtrl.getPrevious();
    if (last && last.id ==
"DefineConstraintsPage") {
        this.navCtrl.pop();
    } else {
        this.navCtrl.popToRoot();
        this.navCtrl.push('DefineConstraintsPage');
    }
}

}
page-result {

    .button-redefine{
        border: 1px solid color($colors, main-
color);
        background-color: $background-color;
        color: color($colors, main-color);
        width: 270px;
        border-radius: 5px;

```

```

        margin-top: 50px;
        margin-bottom: 20px;
        position: absolute;
        left: 50%;
        transform: translate(-50%, -50%);
    }
}
import { Component } from '@angular/core';
import { IonicPage } from 'ionic-angular';

@IonicPage()
@Component({
    selector: 'page-about',
    templateUrl: 'about.html',
})
export class AboutPage {

    constructor() {
    }

}
import { NgModule } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { AboutPage } from './about';
import { TranslateModule } from '@ngx-translate/core';

@NgModule({
    declarations: [
        AboutPage,
    ],
    imports: [
        IonicPageModule.forChild(AboutPage),
        TranslateModule.forChild()
    ],
})
export class AboutPageModule {}
page-about {

    p {
        text-indent: 2.0em;
    }

}
<ion-header>

```

```

<ion-navbar>
  <ion-title>{{ 'ABOUT' | translate }}</ion-
title>
</ion-navbar>
</ion-header>

<ion-content padding>
  <p>{{ 'ABOUT_TEXT' | translate }}</p>
  <p>{{ 'ABOUT_TEXT_2' | translate }}</p>

  <div>Icons made by <a href="https://
www.flaticon.com/authors/smashicons"
title="Smashicons">Smashicons</a> from <a
href="https://www.flaticon.com/"
title="Flaticon">www.flaticon.com</a> is licensed
by <a href="http://creativecommons.org/licenses/by/
3.0/" title="Creative Commons BY 3.0"
target="_blank">CC 3.0 BY</a></div>
  <div>Icons made by <a href="https://
www.flaticon.com/authors/pixel-perfect"
title="Pixel perfect">Pixel perfect</a> from <a
href="https://www.flaticon.com/"
title="Flaticon">www.flaticon.com</a> is licensed
by <a href="http://creativecommons.org/licenses/by/
3.0/" title="Creative Commons BY 3.0"
target="_blank">CC 3.0 BY</a></div>
  <div>Icons made by <a href="https://
www.flaticon.com/authors/popcorns-arts"
title="Icon Pond">Icon Pond</a> from <a
href="https://www.flaticon.com/"
title="Flaticon">www.flaticon.com</a> is licensed
by <a href="http://creativecommons.org/licenses/by/
3.0/" title="Creative Commons BY 3.0"
target="_blank">CC 3.0 BY</a></div>
  <div>Icons made by <a href="http://
www.freepik.com" title="Freepik">Freepik</a> from
<a href="https://www.flaticon.com/"
title="Flaticon">www.flaticon.com</a> is licensed
by <a href="http://creativecommons.org/licenses/by/
3.0/" title="Creative Commons BY 3.0"
target="_blank">CC 3.0 BY</a></div>
  <div>Icons made by <a href="https://
www.flaticon.com/authors/prosymbols"
title="Prosymbols">Prosymbols</a> from <a
href="https://www.flaticon.com/"
title="Flaticon">www.flaticon.com</a> is licensed

```

```

by <a href="http://creativecommons.org/licenses/by/
3.0/" title="Creative Commons BY 3.0"
target="_blank">CC 3.0 BY</a></div>
</ion-content>
import { LoginProvider } from '../providers/
login-provider/login-provider';
import { Component } from '@angular/core';
import { TranslateService } from '@ngx-translate/
core';
import { IonicPage, NavController,
ToastController, LoadingController } from 'ionic-
angular';
import { Storage } from '@ionic/storage';
import { StorageKeys } from '../utils/storage-
keys';

@IonicPage()
@Component({
  selector: 'page-login',
  templateUrl: 'login.html'
})
export class LoginPage {

  showView: boolean = false;

  username: string;
  password: string;

  language = "pt";

  keepLoggedIn: boolean = true;

  constructor(public navCtrl: NavController,
public loginProvider: LoginProvider,
public toastCtrl: ToastController,
public translateService: TranslateService,
private storage: Storage,
public loadingCtrl: LoadingController) {
  // private iab: InAppBrowser,
  // private deeplinks: Deeplinks,
  // private plataform: Platform) {

  //
  this.deeplinks.routeWithNavController(this.navCtrl,
  {

```



```

    //   '*': LoginPage
    // }).subscribe((match) => {
    //   this.browser.close();
    //   // match.$route - the route we matched,
which is the matched entry from the arguments to
route()
    //   // match.$args - the args passed in the
link
    //   // match.$link - the full link data
    //   let code = (match.$link['queryString']);
    //   code = code.replace('code=', "");
    //   code =
code.replace('&state=E3ZYKC1T6H2yP4z', '');
    //   this.loginProvider.authSetic(code)
    //     .map(res => res.json())
    //     .subscribe(res => {
    //       this.doLogin(res['access_token']);
    //     }, err => alert(err));
    // }, (nomatch) => {
    //   console.error('Got a deeplink that
didn\'t match', nomatch);
    // });

    this.translateService.setDefaultLang('pt');

this.storage.get(StorageKeys.LANGUAGE).then(language
=> {
    if (language) {
        this.language = language;
        this.translateService.use(this.language);
    }
});
}

ionViewDidLoad(){
    this.navCtrl.setRoot(this.navCtrl.getActive());
}

ionViewWillEnter() {
    this.username = null;
    this.password = null;
    this.showView = false;

    this.language =
this.translateService.currentLang;

```

```

    this.checkKeepLoggedIn();
  }

  checkKeepLoggedIn() {
    this.storage.get(StorageKeys.TOKEN).then((acc)
=> {
      if (acc) {
        this.doLogin(acc);
      } else {
        this.showView = true;
      }
    });
  }

  languageChanged(): void {
    this.translateService.use(this.language);
    this.storage.set(StorageKeys.LANGUAGE,
this.language);
  }

  prepareLogin() {
    // let api_key = 'oauth';
    // let api_key = 'tccphpils';
    // let secret_key = 'segredo';
    // let secret_key =
'Gto1W3ErSqWdmASpb5CsqgPkgjNv8';
    // let redirect_url = 'ufsclogin://
setic_oauth_example.ufsc.br';
    // let redirect_url = 'tccphpils://
tccphpils.setic_oauth.ufsc.br';
    // let site = 'https://
sistemas.homologacao.ufsc.br/oauth2.0/';
    // let site = 'https://sistemas.ufsc.br/
oauth2.0/';

    // let url = site + 'authorize?client_id='+
api_key + '&client_secret='+ secret_key
+ '&redirect_uri=' + redirect_url +
'&state=E3ZYKC1T6H2yP4z&response_type=code&bypass_approval_prompt=
&bypass_approval_prompt=true
    // this.browser = this.iab.create(url,
'_system', { location: 'yes'});

    // this.browser.close();
    this.doLogin("AT-10-
XSLAp9Ec0eEHo02aMoUhBzpdU66bTGmoNY0");
  }

```

```

}

doLogin(token: string) {
    this.storage.set(StorageKeys.KEEP_LOGGED_IN,
this.keepLoggedIn);
    let loading = this.loadingCtrl.create({
        content:
this.translateService.instant("PLEASE_WAIT")
    });

    this.storage.set(StorageKeys.TOKEN, token);

    let timeOutid = setTimeout(() => {
        loading.present();
    }, 300);

    this.loginProvider.login(token).subscribe(res
=> {
        if (res && res.status == 200 &&
res.json().success) {
            loading.dismiss();
            clearTimeout(timeOutid);
            this.navCtrl.push('MainPage');
        } else {
            loading.dismiss();
            clearTimeout(timeOutid);
            this.showError();
        }
    }, err => {
        loading.dismiss();
        clearTimeout(timeOutid);
        console.error('ERROR', err);
        this.showError();
    });
}

showError() {
    this.showView = true;
    let toast = this.toastCtrl.create({
        message:
this.translateService.instant('ERROR_CONNECTING_TO_SERVER'),
        duration: 3000,
        position: 'top'
    });
    toast.present();
}

```

```

    }

    onAboutClicked(): void {
      this.navCtrl.push('AboutPage');
    }
  }
}
import { NgModule } from '@angular/core';
import { TranslateModule, TranslateLoader } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { LoginPage } from './login';
import { Http } from '@angular/http';
import { TranslateHttpLoader } from '@ngx-translate/http-loader';

export function TranslateLoaderFactory(http: Http)
{
  return new TranslateHttpLoader(http, './assets/i18n/', '.json');
}

@NgModule({
  declarations: [
    LoginPage
  ],
  imports: [
    IonicPageModule.forChild(LoginPage),
    TranslateModule.forChild(),
    TranslateModule.forChild({
      loader: {
        provide: TranslateLoader,
        useFactory: (TranslateLoaderFactory),
        deps: [ Http ]
      }
    })
  ],
  exports: [
    LoginPage
  ]
})
export class LoginPageModule { }
<ion-content *ngIf='showView' class="login-page">
  <ion-list>
    

```

```

    <ion-item>
      <ion-label>{{ 'LANGUAGE' | translate }}</ion-
label>
      <ion-select
[(ngModel)]="language" (ionChange)="languageChanged()">
        <ion-option value="pt">Português</ion-
option>
        <ion-option value="en">English</ion-option>
        <ion-option value="es">Español</ion-option>
      </ion-select>
    </ion-item>

    <form (submit)="prepareLogin()">
      <ion-list>
        <ion-item>
          <ion-label floating>{{ 'USERNAME' |
translate }}</ion-label>
          <ion-input
type="text" [(ngModel)]="username"
name="username"></ion-input>
        </ion-item>

        <ion-item>
          <ion-label floating>{{ 'PASSWORD' |
translate }}</ion-label>
          <ion-input
type="password" [(ngModel)]="password"
name="password"></ion-input>
        </ion-item>

        <ion-item class="manter-conectado">
          <ion-label>{{ 'KEEP_LOGGED_IN' |
translate }}</ion-label>
          <ion-toggle [(ngModel)]="keepLoggedIn"
color="main-color" [ngModelOptions]="{standalone:
true}"></ion-toggle>
        </ion-item>

        <button ion-button color="main-color"
class="login-button">{{ 'LOGIN_BUTTON' |
translate }}</button>

      </ion-list>
    </form>

```

```

<button ion-button color="main-
color" (click)='onAboutClicked()' class="sobre-
button">{{ 'ABOUT' | translate }}</button>
</ion-list>
</ion-content>page-login {

  .logo-tantum {
    width: 28%;
    margin-left: 36%;
    margin-top: 10%
  }

  .sobre-button {
    margin-top: 100px;
    position: absolute;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 80px;
  }

  .login-button{
    margin-top: 30px;
    position: absolute;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 80px;
  }

  .manter-conectado {
    width: 80%;
    margin-left: 10%;
  }
}

.item-inner {
  border: none !important;
}

.brazil {
  background-image: url(assets/img/brazil.png);
  background-size: contain;
  background-position: 50%;
  background-repeat: no-repeat;
  position: relative;
  display: inline-block;
}

```

```

        width: 1.33333333em;
        line-height: 1em;
    }
    <ion-content padding class="main-content">

        <p class="next-classes">{{ 'NEXT_CLASSES' |
translate }}</p>
        <subject-list [subjects]="getSubjects()"></
subject-list>

        <div class="separator"></div>

        <ion-list class="main-list">
            <ion-item class="main-
item" (click)='onScheduleClicked()'>
                
                <span class="main-div"></span>
                <span class="main-
label">{{ 'DAILY_SCHEDULES' | translate }}</span>
            </ion-item>
            <ion-item class="main-
item" (click)='onScheduleGenerationClicked()'>
                
                <span class="main-div"></span>
                <span class="main-
label">{{ 'SCHEDULE_GENERATION' | translate }}</
span>
            </ion-item>
            <ion-item class="main-
item" (click)='onStatisticsClicked()'>
                
                <span class="main-div"></span>
                <span class="main-label">{{ 'STATISTICS' |
translate }}</span>
            </ion-item>
        </ion-list>

        <button ion-button (click)='onLogOutClicked()'
class="button-log-out">{{ 'SIGN_OUT' | translate }}
</button>

    </ion-content>page-main {

```

```
.main-list {
    height: 60%;
}

.list-ios > .item-block:first-child {
    border-top: none !important;
}

.list {
    margin: 0px !important;
}

.slide-text {
    color: color($colors, main-color);
}

.swiper-pagination-bullet-active {
    background-color: color($colors, main-
color);
}

.main-content {
    display: flex;
    display: -webkit-flex;
    align-items: center;
    justify-content: center;
}

.button-log-out {
    border: 1px solid color($colors, main-
color);
    background-color: $background-color;
    color: color($colors, main-color);
    width: 120px;
    border-radius: 5px;
    margin-top: 20px;
    position: absolute;
    left: 50%;
    transform: translate(-50%, -50%);
}

.main-icon {
    width: 10%;
}

.main-item {
```



```

        height: 80px;
        margin-top: 10px;
        box-shadow: 0 4px 8px -2px rgba(0, 0, 0,
0.2);
        background-color: white;
    }

    .main-div {
        width: 1px;
        border: 1px solid #f1f1f1;
        margin-left: 20px;
        margin-right: 20px;
        position: relative;
        bottom: 8px;
    }

    .main-label {
        position: relative;
        bottom: 8px;
        color: #3d3d3d;
    }

    .next-classes {
        margin-top: 0px;
        margin-bottom: 10px;
        margin-left: 5px;
        color: grey;
        font-size: 16px;
    }

    .separator {
        margin-top: 10px;
        height: 1px;
        background-color: lightgray;
        width: 80%;
        position: relative;
        left: 10%;
    }

    .activated {
        background-color: white !important;
    }

}import { Account } from '../models/account';
import { StorageKeys } from '../utils/storage-
keys';

```

```

import { Subject } from './../../models/subject';
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams }
from 'ionic-angular';
import { Storage } from '@ionic/storage';
import { TranslateService } from '@ngx-translate/
core';
import { ScheduleProvider } from './../../providers/
horarios/schedule-provider';
import { SubjectHelper } from './../../utils/subject-
helper';

@IonicPage()
@Component({
  selector: 'page-main',
  templateUrl: 'main.html',
})
export class MainPage {

  private subjects: Subject[];

  private account: Account;

  constructor(public navCtrl: NavController,
              public navParams: NavParams,
              private storage: Storage,
              public translateService: TranslateService,
              private scheduleProvider: ScheduleProvider) {
  }

  ionViewDidLoad() {
    this.navCtrl.setRoot(this.navCtrl.getActive());

    this.storage.get(StorageKeys.SCHEDULE).then(d
=> {
      if (d) {
        this.subjects = d;
      }
    });

    this.storage.get(StorageKeys.ACCOUNT).then(acc
=> this.account = acc);

    // mudar esse ano automaticamente
    this.scheduleProvider.schedule("2018-1")
      .map(res => res.json())
  }
}

```

```

        .subscribe(res => {
            if (res.success) {
                this.subjects = res.result.subjects;
                this.storage.set(StorageKeys.SCHEDULE,
this.subjects);
            }
        }, err => {
            console.error('ERROR', err);
        });
    }

    onScheduleClicked(): void {
        this.navCtrl.push('SchedulePage');
    }

    onScheduleGenerationClicked(): void {
        this.storage.get(StorageKeys.RESULT).then(res
=> {
            if (res) {
                this.navCtrl.push('ResultPage');
            } else {
                this.navCtrl.push('DefineConstraintsPage');
            }
        });
    }

    onLogOutClicked(): void {
        if (this.navCtrl.length() > 1) {
            this.navCtrl.remove(0);
        }
        this.storage.remove(StorageKeys.TOKEN);
        this.navCtrl.push('LoginPage');
    }

    onStatisticsClicked(): void {
        this.navCtrl.push('StatisticsPage');
    }

    getSubjects() {
        return SubjectHelper.nextTwo(this.subjects);
    }
}
import { PipesModule } from '../pipes/
pipes.module';
import { NgModule } from '@angular/core';

```

```

import { IonicPageModule } from 'ionic-angular';
import { MainPage } from './main';
import { TranslateModule } from '@ngx-translate/
core';
import { ComponentsModule } from '../components/
components.module';

@NgModule({
  declarations: [
    MainPage
  ],
  imports: [
    IonicPageModule.forChild(MainPage),
    TranslateModule.forChild(),
    ComponentsModule,
    PipesModule
  ],
})
export class MainPageModule {}
// The page the user lands on after opening the
app and without a session
export const FirstRunPage = 'TutorialPage';

// The main page the user will see as they use the
app over a long period of time.
// Change this if not using tabs
export const MainPage = 'TabsPage';

// The initial root pages for our tabs (remove if
not using tabs)
export const Tab1Root = 'ListMasterPage';
export const Tab2Root = 'SearchPage';
export const Tab3Root = 'SettingsPage';
<p class="period">{{ morning }}</p>
<table class="week-schedule-
table" (onSwiped)="onSwiped($event)">
  <tr>
    <th>{{ monday }}</th>
    <th>{{ tuesday }}</th>
    <th>{{ wednesday }}</th>
    <th>{{ thursday }}</th>
    <th>{{ friday }}</th>
  </tr>

  <tr *ngFor='let time of morningTimes'>
    <td *ngFor='let day of

```

```

weekDays' [ngClass]='getClass(time)' (click)='onDisciplinaClicker
time)'>
    {{time}}
    <br>
    <br>
    <span [innerHTML]="getInnerHTML(day,
time)"></span>
</td>
</tr>
</table>

<p class="period">{{ afternoon }}</p>
<table class="week-schedule-table">
  <tr *ngFor='let time of afternoonTimes'>
    <td *ngFor='let day of
weekDays' [ngClass]='getClass(time)' (click)='onDisciplinaClicker
time)'>
        {{time}}
        <br>
        <br>
        <span [innerHTML]="getInnerHTML(day,
time)"></span>
    </td>
  </tr>
</table>

<p class="period">{{ night }}</p>
<table class="week-schedule-table">
  <tr *ngFor='let time of nightTimes'>
    <td *ngFor='let day of
weekDays' [ngClass]='getClass(time)' (click)='onDisciplinaClicker
time)'>
        {{time}}
        <br>
        <br>
        <span [innerHTML]="getInnerHTML(day,
time)"></span>
    </td>
  </tr>
</table>
import { TranslateService } from '@ngx-
translate/core';
import { Component, Input } from '@angular/core';
import { AlertController } from 'ionic-angular';

import { Subject } from '../models/subject';
import { ModalHelper } from '../utils/modal-

```

```

helper';

@Component({
  selector: 'week-schedule',
  templateUrl: 'week-schedule.html'
})
export class WeekScheduleComponent {

  @Input() subjects: Subject[]

  morningTimes: string[] = ["07:30", "08:20",
"09:10", "10:10", "11:00"];
  afternoonTimes: string[] = ["13:30", "14:20",
"15:10", "16:20", "17:10"];
  nightTimes: string[] = ["18:30", "19:20",
"20:20", "21:10"];

  weekDays: string[] = ['2', '3', '4', '5', '6'];

  private afternoon: string;
  private morning: string;
  private night: string;

  private monday: string;
  private tuesday: string;
  private wednesday: string;
  private thursday: string;
  private friday: string;

  constructor(private alertCtrl: AlertController,
private translate: TranslateService) {
    this.afternoon =
this.translate.instant('AFTERNOON');
    this.morning =
this.translate.instant('MORNING');
    this.night = this.translate.instant('NIGHT');

    this.monday =
this.translate.instant('MONDAY_SHORT');
    this.tuesday =
this.translate.instant('TUESDAY_SHORT');
    this.wednesday =
this.translate.instant('WEDNESDAY_SHORT');
    this.thursday =
this.translate.instant('THURSDAY_SHORT');
    this.friday =

```



```

    getSubject(day: string, time: string): Subject {
        let horario = day + '.' + time.replace(":",
    '');

        if (this.subjects && this.subjects.length > 0)
        {
            for (let i in this.subjects) {
                for (let j in this.subjects[i].horarios) {
                    if
    (this.subjects[i].horarios[j].startsWith(horario))
                {
                    return this.subjects[i];
                }
            }
        }
        return;
    }
}
week-schedule {

    .primeira-linha {
        border-top-left-radius: 5px;
        border-top-right-radius: 5px;
    }

    .ultima-linha {
        border-bottom-left-radius: 5px;
        border-bottom-right-radius: 5px;
    }

    .week-schedule-table {
        width: 100%;
        font-size: 10px;
    }

    .period {
        margin-left: 8px;
        color: #3d3d3d;
        font-size: 14px;
        margin-top: 15px;
        margin-bottom: -10px;
    }
}
}

```



```

td {
  width: auto;
  text-align: center;
  height: 65px;
  border: 1px solid #ff7043;
}

th, tr {
  width: 30px;
}

table {
  border-spacing: 5px 0px;
  border-collapse: separate;
  margin-top: 20px;
}import { SubjectListComponent } from './subject-
list/subject-list';
import { SubjectActionListComponent } from './
subject-action-list/subject-action-list';
import { NgModule } from '@angular/core';
import { WeekScheduleComponent } from './week-
schedule/week-schedule';
import { CapsuleComponent } from './capsule/
capsule';
import { IonicModule } from 'ionic-angular';
import { PipesModule } from '../pipes/
pipes.module';
import { TranslateModule } from '@ngx-translate/
core';

@NgModule({
  declarations: [WeekScheduleComponent,
    CapsuleComponent,
    SubjectListComponent,
    SubjectActionListComponent],
  imports: [IonicModule, PipesModule,
    TranslateModule.forChild()],
  exports: [WeekScheduleComponent,
    CapsuleComponent,
    SubjectListComponent,
    SubjectActionListComponent]
})
export class ComponentsModule {}
import { Component, Input, Output, EventEmitter }
from '@angular/core';

```

```

import { Platform } from 'ionic-angular';
import { Subject } from '../models/subject';

@Component({
  selector: 'subject-action-list',
  templateUrl: 'subject-action-list.html'
})
export class SubjectActionListComponent {

  @Input() subjects: Subject[];

  @Input() action: boolean = true;

  @Output() onRemoved = new
  EventEmitter<Subject>();

  constructor(public plt: Platform) {
  }

  remove(disciplina): void {
    let index = this.subjects.indexOf(disciplina);
    this.subjects.splice(index, 1);
    this.onRemoved.emit(disciplina);
  }

  getValue(disciplina: Subject): string {
    return disciplina.codigo + " - " +
disciplina.nome;
  }

  getClass(): string {
    let name = "name-subject";
    if (!this.action) {
      name += " name-subject-without-action"
    }
    return name;
  }
}

subject-action-list {

  .name-subject {
    text-overflow: ellipsis;
    white-space: nowrap;
    overflow: hidden;
  }
}

```

```

        display: block;
        width: 75%;
        color: color($colors, main-color);
    }

    .name-subject-without-action {
        width: 100%;
    }

    .remove-subject {
        float: right;
        position: relative;
        bottom: 21px;
        padding: 5px;
        padding-left: 10px;
        padding-right: 10px;
        border: 1px solid color($colors, main-
color);
        border-radius: 10px;
        background-color: white;
        color: color($colors, main-color);
    }

    .base-subject {
        margin-top: 15px;
        height: 30px;
        padding-bottom: 10px;
        border-bottom: 1px solid lightgray;
    }

    .subject-list {
        margin-top: 20px;
    }
}
<div class="subject-list">
  <div class="base-subject" *ngFor='let subject of
subjects'>
    <span
[ngClass]="getClass()" [innerHTML]="getValue(subject)"></
span>
    <button *ngIf="action" class="remove-
subject" (click)="remove(subject)">{{ 'REMOVE' |
translate }}</button>
  </div>
</div>

```

```

</div><ion-chip *ngFor='let v of
values' (click)="onItemClicked(v)" [ngClass]="getClass(v)">
  <ion-label>{{v}}</ion-label>
</ion-chip>.capsule-ative {
  margin-left: 5px;
  background-color: color($colors, main-color);
  border: 1px solid color($colors, main-color);
  color: white;
}

.capsule-inative {
  margin-left: 5px;
  color: color($colors, main-color);
  border: 1px solid color($colors, main-color);
  background-color: white;
}
import { Component, Input, Output, EventEmitter }
from '@angular/core';

@Component({
  selector: 'capsule',
  templateUrl: 'capsule.html'
})
export class CapsuleComponent {

  @Input() values: string[] = [];

  @Output() valueSelected = new
EventEmitter<string[]>();

  allValuesSelected: string[] = [];

  constructor() {
  }

  getClass(periodo: string): string {
    return this.allValuesSelected.indexOf(periodo)
> -1 ? 'capsule-ative' : 'capsule-inative';
  }

  onItemClicked(periodo: string) {
    if (this.allValuesSelected.indexOf(periodo) >
-1){
this.allValuesSelected.splice(this.allValuesSelected.indexOf(per.
1)

```

```

    } else {
      this.allValuesSelected.push(periodo);
    }

    this.valueSelected.emit(this.allValuesSelected);
  }

}

import { TranslateService } from '@ngx-translate/
core';
import { ModalHelper } from '../utils/modal-
helper';
import { AlertController, Platform } from 'ionic-
angular';
import { Component, Input, Output, EventEmitter }
from '@angular/core';
import { Subject } from '../models/subject';

@Component({
  selector: 'subject-list',
  templateUrl: 'subject-list.html'
})
export class SubjectListComponent {

  @Input() subjects: Subject[];

  @Output() onSwiped = new EventEmitter<boolean>();

  constructor(private alertCtrl: AlertController,
public plt: Platform, public translateService:
TranslateService) {
  }

  itemSwiped(s: any) {
    if (s.direction == 2) { // 4 left
      this.onSwiped.emit(true);
    } else {
      this.onSwiped.emit(false);
    }
  }

  getDisciplinas(): Subject[] {
    return this.subjects;
  }

  onDisciplinaClicked(subject: Subject): void {

```

```

    let mh = new
ModalHelper(this.translateService, this.alertCtrl);
    mh.createModal(subject);
}

showLista(): boolean {
    if (this.subjects) {
        return this.subjects.length > 0;
    } else {
        return false;
    }
}

getLocalClass(): string {
    let clazz: string = "disciplia-local"
    if (this.plt.is('windows')) {
        clazz += " disciplia-local-wp"
    }
    return clazz;
}

getNomeClass(): string {
    let clazz: string = "disciplia-nome"
    if (this.plt.is('windows')) {
        clazz += " disciplia-nome-wp"
    }
    return clazz;
}
}
subject-list {

    .horario {
        color: #e67e22;
        font-size: 15px;
        float: left;
    }

    .disciplina-item {
        border: 1px solid #f1f1f1 !important;
        background-color: white;
        border-radius: 10px;
        margin-bottom: 5px;
        font-size: 13px;
        height: 50px;
        text-align: left;
    }
}

```

```

padding-top: 15px;
padding-left: 15px;
box-shadow: 0 0 8px -2px rgba(0, 0, 0,
0.2);
}

.disciplina-div {
width: 1px;
height: 20px;
border: 1px solid #f1f1f1;
margin-left: 13px;
margin-right: 15px;
float: left;
}

.disciplia-local {
position: relative;
bottom: 5px;
}

.disciplia-local-wp {
bottom: 8px !important;
}

.disciplia-nome {
position: relative;
bottom: 5px;
color: #9d9ea0;
text-overflow: ellipsis;
white-space: nowrap;
overflow: hidden;
display: block;
}

.disciplia-nome-wp {
bottom: 8px !important;
}

.nomes {
text-overflow: ellipsis;
white-space: nowrap;
}
}
<div (swipe)="itemSwiped($event)"
*ngIf="showLista()">

```

```

    <div *ngFor='let disciplina of
getDisciplinas()' class="disciplina-
item" (click)="onDisciplinaClicked(disciplina)">
    <div class="horario">{{ disciplina.horarios
| schedule}}</div>
    <div class="disciplina-div"></div>
    <div class="nomes">
    <div
[ngClass]="getLocalClass()">{{ disciplina.horarios
| location }}</div>
    <div
[ngClass]="getNomeClass()">{{disciplina.nome}}</
div>
    </div>
</div>
</div>

<p *ngIf="!showLista()">{{ 'HAS_NOTHING' |
translate }}</p>import { NgModule } from '@angular/
core';
import { SchedulePipe } from './schedule/schedule';
import { LocationPipe } from './location/location';
@NgModule({
  declarations: [SchedulePipe,
  LocationPipe],
  imports: [],
  exports: [SchedulePipe,
  LocationPipe]
})
export class PipesModule {}
import { FormatterUtils } from './../utils/
formatter';
import { Pipe, PipeTransform } from '@angular/
core';

@Pipe({
  name: 'schedule',
})
export class SchedulePipe implements PipeTransform
{

  /**
   * Takes a value like "3.0820-2 / CTC-CTC102"
and return 08:20
   */
  transform(value: string, ...args) {

```



```

    let aux: string = value.toString().split("/")
[0].trim();

    return FormatterUtils.formatHour(aux);
}
}
import { FormatterUtils } from '../../utils/
formatter';
import { Pipe, PipeTransform } from '@angular/
core';

@Pipe({
  name: 'location',
})
export class LocationPipe implements PipeTransform
{

  /**
   * Takes a value like "3.0820-2 / CTC-CTC102"
and return CTC102
   */
  transform(value: string, ...args) {
    let aux: string = value.toString().split("/")
[1].trim();

    return FormatterUtils.formatLocal(aux);
  }
}
import { Subject } from '../../models/subject';

export class SubjectHelper {

  constructor(){
  }

  public static list(subjects: Subject[], day?:
string): Subject[] {
    let newSubjects: Subject[] = [];

    for (let i in subjects) {
      let subject: Subject = subjects[i]
      for (let h in subject.horarios) {
        let time = subject.horarios[h];
        if (day == null) {

newSubjects.push(this.subject(subject, time));

```

```

        } else if (time.startsWith(day)) {
newSubjects.push(this.subject(subject, time));
        }
    }
    }

    return newSubjects;
}

public static nextTwo(subjects: Subject[]):
Subject[] {
    let newSubjects: Subject[] = [];
    let day = new Date().getDay() + 1;
    day = day > 7 ? 2 : day;
    let highDay = 0;
    let lowDay = 99;

    for (let i in subjects) {
        let subject: Subject = subjects[i]
        for (let h in subject.horarios) {
            let time =
Number(subject.horarios[h].substring(0, 1));
            if (time > highDay) {
                highDay = time;
            }
            if (time < lowDay) {
                lowDay = time;
            }
        }
    }

    for (let i in subjects) {
        let subject: Subject = subjects[i]
        for (let h in subject.horarios) {
            let time = subject.horarios[h];
            let currentDay =
Number(subject.horarios[h].substring(0, 1));
            if (day >= highDay && currentDay
== lowDay) {
newSubjects.push(this.subject(subject,
time));
            }
            if (newSubjects.length == 2) {
                return newSubjects
            }
        }
    }
}

```

```

        }
        //
newSubjects.push(this.subject(subject, time));
    }
    }
    return newSubjects;
}

private static subject(subject: Subject, time:
string): Subject {
    subject.horarios = [time];
    return subject;
}

}export class FormatterUtils {

    constructor() {
    }

    /**
     * Method to format de hour from this pattern:
     d.hhmm-n and return hh:mm
     * @param value example 2.0820-2 and return
     08:20
     */
    public static formatHour(value: string) {
        let hour = value.substring(2, 6);

        return hour.slice(0, 2) + ":" +
hour.slice(2);
    }

    public static formatLocal(value: string) {
        return value.split("-")[1];
    }

    public static replaceSpecialChars(value:
string) {
        let newValue: string;

        newValue = value.toLowerCase()
            .replace(/ç/g, "c")
            .replace(/ã/g, "a")
            .replace(/á/g, "a")
            .replace(/à/g, "a")
            .replace(/â/g, "a")

```

```

        .replace(/é/g, "e")
        .replace(/è/g, "e")
        .replace(/ê/g, "e")
        .replace(/ë/g, "e")
        .replace(/í/g, "i")
        .replace(/ó/g, "o")
        .replace(/ò/g, "o")
        .replace(/õ/g, "o")
        .replace(/ô/g, "o")
        .replace(/ú/g, "u")

        return newValue;
    }
}

import { TranslateService } from "@ngx-translate/core";
import { AlertController } from "ionic-angular";
import { Subject } from "../models/subject";

export class ModalHelper {
    constructor(public translateService: TranslateService, public alertCtrl: AlertController){
    }

    public createModalError(subjects: Subject[]) {
        let alert = this.alertCtrl.create({
            title:
this.translateService.instant('SUBJECTS_NOT_SELECTED'),
            subTitle:
this.createContentError(subjects),
            buttons:
[this.translateService.instant('BACK_BUTTON_TEXT')]
        });
        alert.present();
    }

    public createModal(subject: Subject) {
        let alert = this.alertCtrl.create({
            title:
this.translateService.instant('SUBJECT_DETAILS'),
            subTitle: this.createContent(subject),
            buttons:
[this.translateService.instant('BACK_BUTTON_TEXT')]
        });
    }
}

```

```

        alert.present();
    }

    private createContentError(subjects:
Subject[]): string {
        let list: string[] = []
        subjects.forEach(s => list.push(s.nome));

        return list.join(", ");
    }

    private createContent(subject: Subject):
string {
        return
this.p(this.translateService.instant('SUBJECT'))
        + this.p(subject.nome)
        +
this.p(this.translateService.instant('PROFESSOR'))
        + this.p(subject.professor)
        +
this.p(this.translateService.instant('SCHEDULES'))
        + this.p(subject.horarios[0])
        ;
    }

    private p(value: string): string {
        return '<p>'+value+'</p>'
    }
}export class StorageKeys {

    public static CONSTRAINT: string =
"constraint";
    public static RESULT: string = "result";
    public static SCHEDULE: string = "schedule";
    public static ACCOUNT: string = "account";
    public static KEEP_LOGGED_IN: string =
"keep logged in";
    public static LANGUAGE: string = "language";
    public static STATISTIC: string = "statistic";
    public static ALL_SUBJECTS: string =
"all subjects";
    public static TOKEN: string = "token";

}import { TranslateService } from '@ngx-translate/
core';
import { Injectable } from '@angular/core';

```

```

import { Dia } from '../models/dia';

@Injectable()
export class CalendarUtils {

    constructor(public translateService:
TranslateService) {

        public getAllDias(): Dia[] {
            return [
                new
Dia(this.translateService.instant("MONDAY"),
this.translateService.instant("MONDAY_SHORT").toUpperCase(),
true),
                new
Dia(this.translateService.instant("TUESDAY"),
this.translateService.instant("TUESDAY_SHORT").toUpperCase(),
true),
                new
Dia(this.translateService.instant("WEDNESDAY"),
this.translateService.instant("WEDNESDAY_SHORT").toUpperCase(),
true),
                new
Dia(this.translateService.instant("THURSDAY"),
this.translateService.instant("THURSDAY_SHORT").toUpperCase(),
true),
                new
Dia(this.translateService.instant("FRIDAY"),
this.translateService.instant("FRIDAY_SHORT").toUpperCase(),
true),
                new
Dia(this.translateService.instant("SATURDAY"),
this.translateService.instant("SATURDAY_SHORT").toUpperCase(),
false)
            ]
        }
    }
}
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
    <meta charset="UTF-8">
    <title>Ionic App</title>
    <meta name="viewport" content="width=device-
width, initial-scale=1.0, minimum-scale=1.0,

```

```

maximum-scale=1.0, user-scalable=no">
  <meta name="format-detection"
content="telephone=no">
  <meta name="msapplication-tap-highlight"
content="no">

  <link rel="icon" type="image/x-icon"
href="assets/icon/favicon.ico">
  <link rel="manifest" href="manifest.json">
  <meta name="theme-color" content="#4e8ef7">

  <!-- cordova.js required for cordova apps -->
  <script src="cordova.js"></script>

  <!-- un-comment this code to enable service
worker
  <script>
    if ('serviceWorker' in navigator) {
      navigator.serviceWorker.register('service-
worker.js')
        .then(() => console.log('service worker
installed'))
        .catch(err => console.error('Error', err));
    }
  </script>-->

  <link href="build/main.css" rel="stylesheet">

</head>
<body>

  <!-- Ionic's root component and where the app
will load -->
  <ion-app></ion-app>

  <!-- The polyfills js is generated during the
build process -->
  <script src="build/polyfills.js"></script>
  <script src="build/vendor.js"></script>
  <!-- The bundle js is generated during the build
process -->
  <script src="build/main.js"></script>

</body>
</html>

```

## 10.2 CÓDIGO FONTE DA API REST



```

package com.tantum.app.tantum;

import static org.junit.Assert.assertTrue;

import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class TantumApplicationTests {

    @Test
    public void contextLoads() {
        assertTrue(true);
    }

}

package com.tantum.app.tantum;

import static
org.springframework.test.web.servlet.request.MockMvcRequestBuild
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuild
import static
org.springframework.test.web.servlet.result.MockMvcResultHandler
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers

import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.autoconfigure.web.servlet.AutoConf
import
org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import

```

```

org.springframework.test.context.junit4.SpringRunner;
import
org.springframework.test.web.servlet.MockMvc;

@RunWith(SpringRunner.class)
@SpringBootTest(classes = TantumApplication.class)
@AutoConfigureMockMvc
public class TantumControllerTest {

    @Autowired
    private MockMvc mockMvc;

    private static String token =
"Q$2tQm1Ts*u7qSWK";

    private static String bad_request =
"{\"success\":false,\"result\": \"token parameter
is missing\"}";

    private static String constraints =
"{\"periods\": [0, 1, 2],\"creditMax\":
2,\"creditMin\": 0,\"equivalent\": true,
\"subjectsWanted\": [],\"subjectsNotWanted\": []}";

    @Test
    public void testTest() throws Exception {
        this.mockMvc.perform(get("/v1/
test")).andDo(print())
                .andExpect(status().isOk())
                .andExpect(content().string("This
is a Test, REST API is Working"))
                .andExpect(jsonPath("$").exists())
    }

    @Test
    public void loginTest() throws Exception {
        this.mockMvc.perform(get("/v1/
login").param("token", token)).andDo(print())
                .andExpect(status().isOk())
                .andExpect(content().string("{\"
result\": \"Q$2tQm1Ts*u7qSWK\"}"))
                .andExpect(jsonPath("$").exists())
                .andExpect(jsonPath("$.success").exists())
                .andExpect(jsonPath("$.result").exists())
                .andExpect(jsonPath("$.result").exists())
    }
}

```

```

    }

    @Test
    public void loginTestFail() throws
Exception {
        this.mockMvc.perform(get("/v1/
login")).andDo(print())
                .andExpect(status().isBadRequest)
                .andExpect(content().string(bad_
                .andExpect(jsonPath("$").exists(
                .andExpect(jsonPath("$.success")
                .andExpect(jsonPath("$.success")
                .andExpect(jsonPath("$.result").
                .andExpect(jsonPath("$.result").

        parameter is missing"));
    }

    @Test
    public void scheduleTest() throws
Exception {
        this.mockMvc.perform(get("/v1/
schedule/2018-1").param("token",
token)).andDo(print())
                .andExpect(status().isOk())
                .andExpect(jsonPath("$").exists(
                .andExpect(jsonPath("$.success")
                .andExpect(jsonPath("$.success")
                .andExpect(jsonPath("$.result").
                .andExpect(jsonPath("$.result.su
                .andExpect(jsonPath("$.result.su

    }

    @Test
    public void scheduleTestFail() throws
Exception {
        this.mockMvc.perform(get("/v1/
schedule/2018-1")).andDo(print())
                .andExpect(status().isBadRequest)
                .andExpect(content().string(bad_
                .andExpect(jsonPath("$").exists(
                .andExpect(jsonPath("$.success")
                .andExpect(jsonPath("$.success")
                .andExpect(jsonPath("$.result").
                .andExpect(jsonPath("$.result").

        parameter is missing"));
    }

```

```

    }

    @Test
    public void subjectsTest() throws
Exception {
        this.mockMvc.perform(get("/v1/
subjects").param("token", token)).andDo(print())
            .andExpect(status().isOk())
            .andExpect(jsonPath("$").exists())
            .andExpect(jsonPath("$.success").
exists())
            .andExpect(jsonPath("$.result").
exists())
            .andExpect(jsonPath("$.result.su
ccess").exists())
            .andExpect(jsonPath("$.result.su
ccess").exists())
    }

    @Test
    public void subjectsTestFail() throws
Exception {
        this.mockMvc.perform(get("/v1/
subjects")).andDo(print())
            .andExpect(status().isBadRequest())
            .andExpect(content().string(bad_
request))
            .andExpect(jsonPath("$").exists())
            .andExpect(jsonPath("$.success").
exists())
            .andExpect(jsonPath("$.success").
exists())
            .andExpect(jsonPath("$.result").
exists())
            .andExpect(jsonPath("$.result").
exists());
    }

    @Test
    public void staticticsTest() throws
Exception {
        this.mockMvc.perform(get("/v1/
statictics").param("token", token)).andDo(print())
            .andExpect(status().isOk())
            .andExpect(jsonPath("$").exists())
            .andExpect(jsonPath("$.semesters").
exists())
            .andExpect(jsonPath("$.semesters").
exists())
            .andExpect(jsonPath("$.semesters").
exists())
            .andExpect(jsonPath("$.semesters").
exists())
            .andExpect(jsonPath("$.semesters").
exists())
            .andExpect(jsonPath("$.semesters").
exists())
            .andExpect(jsonPath("$.courseIA").
exists())
            .andExpect(jsonPath("$.courseIA").
exists())
    }

```

```

    }

    @Test
    public void staticticsTestFail() throws
Exception {
        this.mockMvc.perform(get("/v1/
statictics")).andDo(print())
        .andExpect(status().isBadRequest)
        .andExpect(content().string(bad_
        .andExpect(jsonPath("$").exists(
        .andExpect(jsonPath("$.success")
        .andExpect(jsonPath("$.success")
        .andExpect(jsonPath("$.result").
        .andExpect(jsonPath("$.result").

parameter is missing"));
    }

    @Test
    public void calculateSemesterTest() throws
Exception {
        this.mockMvc.perform(post("/v1/
calculate-semester")
        .contentType(MediaType.APPLICATION_
        .content(constraints)
        .param("token",
token))
        .andDo(print())
        .andExpect(status().isOk())
        .andExpect(jsonPath("$").exists(
        .andExpect(jsonPath("$.success")
        .andExpect(jsonPath("$.success")
        .andExpect(jsonPath("$.result").
        .andExpect(jsonPath("$.result.ne
        .andExpect(jsonPath("$.result.ne
        .andExpect(jsonPath("$.result.sul
        .andExpect(jsonPath("$.result.sul
        .andExpect(jsonPath("$.result.sul
        .andExpect(jsonPath("$.result.sul
        .andExpect(jsonPath("$.result.sul
        .andExpect(jsonPath("$.result.sul
        .andExpect(jsonPath("$.result.sul

    }

    @Test
    public void calculateSemesterTestFail()
throws Exception {
        this.mockMvc.perform(post("/v1/

```

```

    calculate-semester")
        .contentType(MediaType.APPLICATION_JSON)
        .andDo(print())
        .andExpect(status().isBadRequest)
        .andExpect(content().string(bad_request))
        .andExpect(jsonPath("$").exists())
        .andExpect(jsonPath("$.success").isFalse())
        .andExpect(jsonPath("$.result").isNotEmpty());
    }

}

package com.tantum.app.tantum;

import static
com.tantum.app.tantum.models.Period.AFTERNOON;
import static
com.tantum.app.tantum.models.Period.MORNING;
import static
com.tantum.app.tantum.models.Period.NIGHT;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;

import java.util.Arrays;

import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.JUnit4;

import com.google.gson.Gson;
import com.tantum.app.tantum.algorithm.Algorithm;
import com.tantum.app.tantum.helper.Helper;
import com.tantum.app.tantum.models.Constraints;
import com.tantum.app.tantum.models.Course;
import com.tantum.app.tantum.models.History;
import
com.tantum.app.tantum.models.NextSemestersDTO;
import com.tantum.app.tantum.models.Period;
import com.tantum.app.tantum.models.Semester;
import

```

```

com.tantum.app.tantum.models.SemesterHistory;
import com.tantum.app.tantum.models.Subject;

@RunWith(JUnit4.class)
public class AlgorithmTest {

    private static Constraints constraints =
new Constraints();

    private static Algorithm alg;

    private static SemesterHistory
semesterHistory;

    private static Course curso;

    @BeforeClass
    public static void before() {

constraints.setPeriods(Arrays.asList(Period.values()));
        constraints.setCreditMax(30);
        constraints.setCreditMin(0);
        constraints.setEquivalent(true);

constraints.setSubjectsWanted(Arrays.asList());

constraints.setSubjectsNotWanted(Arrays.asList());

        String c = Helper.course_test;
        String h =
Helper.class_history_test;

        Gson g = new Gson();
        curso = g.fromJson(c,
Course.class);
        History history = g.fromJson(h,
History.class);

        semesterHistory =
history.getSemesters().stream().reduce((x, y) -> {
x.getSubjects().addAll(y.getSubjects());
        return x;
}).orElse(new SemesterHistory("",
Arrays.asList()));

```

```

        alg = new Algorithm(curso);
        alg.rankDisciplinas();
    }

    @Before
    public void beforeEach() {
constraints.setPeriods(Arrays.asList(Period.values()));
        constraints.setCreditMax(30);
        constraints.setCreditMin(0);
        constraints.setEquivalent(true);

constraints.setSubjectsWanted(Arrays.asList());

constraints.setSubjectsNotWanted(Arrays.asList());

        alg = new Algorithm(curso);
        alg.rankDisciplinas();
    }

    @Test
    public void testConstraints() {
        assertNotNull(constraints);

        assertNotNull(constraints.getPeriods());
        assertNotNull(constraints.getCreditMin());
        assertNotNull(constraints.getCreditMax());
        assertNotNull(constraints.getSubjectsNotWanted());
        assertNotNull(constraints.getSubjectsWanted());
        assertNotNull(constraints.isEquivalent());

        assertFalse(constraints.getPeriods().isEmpty());
        assertTrue(constraints.getCreditMax() > 0);
    }

    @Test
    public void testAllPeriods() {
        // given

```



```

constraints.setPeriods(Arrays.asList(Period.values()));

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

        // then

assertFalse(result.getNextSemesters().isEmpty());
assertTrue(result.getSubjectsNotSelected().isEmpty());
assertTrue(result.getSubjectsNotWantedError().isEmpty());
assertTrue(result.getSubjectsWantedError().isEmpty());
    }

    @Test
    public void testNoPeriods() {
        // given

constraints.setPeriods(Arrays.asList());

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

        // then

assertTrue(result.getNextSemesters().isEmpty());
assertFalse(result.getSubjectsNotSelected().isEmpty());
assertTrue(result.getSubjectsNotWantedError().isEmpty());
assertTrue(result.getSubjectsWantedError().isEmpty());
    }

    @Test
    public void testMorningPeriod() {
        // given

constraints.setPeriods(Arrays.asList(MORNING));

```

```

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

        // then

assertFalse(result.getNextSemesters().isEmpty());

assertFalse(result.getSubjectsNotSelected().isEmpty());

assertTrue(result.getSubjectsNotWantedError().isEmpty());

assertTrue(result.getSubjectsWantedError().isEmpty());
        for (Semester s :
result.getNextSemesters().values()) {
            for (Subject subject :
s.getSubjects()) {
                boolean b =
subject.getHorarios().stream().map(Period::getPeriodByTime).allMatch(
                    assertTrue(b));
            }
        }
    }

    @Test
    public void testMorningAfternoon() {
        // given

constraints.setPeriods(Arrays.asList(AFTERNOON));

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

        // then

assertFalse(result.getNextSemesters().isEmpty());

assertFalse(result.getSubjectsNotSelected().isEmpty());

assertTrue(result.getSubjectsNotWantedError().isEmpty());

assertTrue(result.getSubjectsWantedError().isEmpty());
        for (Semester s :

```

```

result.getNextSemesters().values() {
    for (Subject subject :
s.getSubjects()) {
        boolean b =
subject.getHorarios().stream().map(Period::getPeriodByTime).allMatch(
            assertTrue(b);
        }
    }
}

@Test
public void testMorningNight() {
    // given

constraints.setPeriods(Arrays.asList(NIGHT));

    // when
    NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

    // then

assertTrue(result.getNextSemesters().isEmpty());
assertFalse(result.getSubjectsNotSelected().isEmpty());
assertTrue(result.getSubjectsNotWantedError().isEmpty());
assertTrue(result.getSubjectsWantedError().isEmpty());
}

@Test
public void testCreditMax() {
    // given
    constraints.setCreditMax(2);

    // when
    NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

    // then

assertFalse(result.getNextSemesters().isEmpty());

```

```

assertTrue(result.getSubjectsNotSelected().isEmpty());
assertTrue(result.getSubjectsNotWantedError().isEmpty());
assertTrue(result.getSubjectsWantedError().isEmpty());
        for (Semester s :
            result.getNextSemesters().values()) {
            int totalCredits =
s.getSubjects().stream().mapToInt(Subject::getAulas).sum();
assertTrue(constraints.getCreditMax() >=
totalCredits);
        }
    }

    @Test
    public void testCreditMax0() {
        // given
        constraints.setCreditMax(0);

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

        // then

assertTrue(result.getNextSemesters().isEmpty());
assertFalse(result.getSubjectsNotSelected().isEmpty());
assertTrue(result.getSubjectsNotWantedError().isEmpty());
assertTrue(result.getSubjectsWantedError().isEmpty());
    }

    @Test
    public void testSubjectsWanted() {
        // given

constraints.setSubjectsWanted(Arrays.asList("INE5612"));

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

```

```

        // then

assertFalse(result.getNextSemesters().isEmpty());
assertTrue(result.getSubjectsNotSelected().isEmpty());
assertTrue(result.getSubjectsNotWantedError().isEmpty());
assertTrue(result.getSubjectsWantedError().isEmpty());
        assertTrue(hasSubject(result,
"INE5612"));
    }

    @Test
    public void testSubjectsWantedError() {
        // given

constraints.setSubjectsWanted(Arrays.asList("INE5134"));

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

        // then

assertFalse(result.getNextSemesters().isEmpty());
assertTrue(result.getSubjectsNotSelected().isEmpty());
assertTrue(result.getSubjectsNotWantedError().isEmpty());
assertFalse(result.getSubjectsWantedError().isEmpty());
        assertEquals("INE5134",
result.getSubjectsWantedError().get(0).getCodigo());
        assertFalse(hasSubject(result,
"INE5134"));
    }

    @Test
    public void testSubjectsNotWanted() {
        // given

constraints.setSubjectsNotWanted(Arrays.asList("INE5612"));

```

```

        // when
        NextSemestersDTO result =
alg.calculateSemesters(constraints,
semesterHistory.getSubjects());

        // then

assertFalse(result.getNextSemesters().isEmpty());

assertTrue(result.getSubjectsNotSelected().isEmpty());

assertTrue(result.getSubjectsNotWantedError().isEmpty());

assertTrue(result.getSubjectsWantedError().isEmpty());
        assertFalse(hasSubject(result,
"INE5612"));
    }

    private static boolean
hasSubject(NextSemestersDTO result, String code) {
        return result.getNextSemesters()
            .get(Helper.semesters.get(1))
            .getSubjects()
            .stream()
            .filter(s ->
s.getCodigo().equals(code))
            .findAny()
            .isPresent();
    }
}

package com.tantum.app.tantum.models;

import java.util.List;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class Course {

```

```

        private List<Subject> subjects;
    }
    package com.tantum.app.tantum.models;

    import java.util.List;

    import lombok.AllArgsConstructor;
    import lombok.Getter;
    import lombok.Setter;

    @Getter
    @Setter
    @AllArgsConstructor
    public class History {

        private List<SemesterHistory> semesters;
    }
    package com.tantum.app.tantum.models;

    import java.util.List;

    import
    com.fasterxml.jackson.annotation.JsonIgnoreProperties;

    import lombok.AllArgsConstructor;
    import lombok.Getter;
    import lombok.NoArgsConstructor;
    import lombok.Setter;

    @Getter
    @Setter
    @AllArgsConstructor
    @NoArgsConstructor
    @JsonIgnoreProperties(ignoreUnknown = true)
    public class SeticAuthDTO {

        private List<String> error;

        private String id;
    }
    package com.tantum.app.tantum.models;

    import java.util.LinkedHashMap;

```

```

import java.util.List;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class NextSemestersDTO {

    private LinkedHashMap<String, Semester>
nextSemesters;

    private List<Subject> subjectsWantedError;

    private List<Subject>
subjectsNotWantedError;

    private List<Subject> subjectsNotSelected;
}
package com.tantum.app.tantum.models;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@AllArgsConstructor
public class Login {

    private boolean hasUsername;

    private boolean hasPassword;
}
package com.tantum.app.tantum.models;

import java.util.ArrayList;
import java.util.List;

import lombok.AllArgsConstructor;

```



```

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Semester {

    public List<Subject> subjects = new
    ArrayList<>();

}
package com.tantum.app.tantum.models;

public class SemestersDTO extends
Result<NextSemestersDTO> {

    public SemestersDTO(boolean success,
NextSemestersDTO result) {
        super(success, result);
    }

}
package com.tantum.app.tantum.models;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@AllArgsConstructor
public abstract class Result<R> {

    protected boolean success;

    protected R result;

}
package com.tantum.app.tantum.models;

import java.util.List;

import lombok.AllArgsConstructor;

```

```

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Constraints {

    private List<Period> periods;

    private int creditMax;

    private int creditMin;

    private boolean equivalent;

    private List<String> subjectsWanted;

    private List<String> subjectsNotWanted;

}
package com.tantum.app.tantum.models;

public class LoginDTO extends Result<String> {

    public LoginDTO(boolean success, String
token) {
        super(success, token);
    }

}
package com.tantum.app.tantum.models;

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class SubjectsDTO extends Result<Semester> {

    public SubjectsDTO(boolean success,
Semester result) {
        super(success, result);
    }

}

```

```
        private String semestre;
        private Boolean morning;
        private Boolean afternoon;
        private Boolean night;
    }
    package com.tantum.app.tantum.models;
    import com.fasterxml.jackson.annotation.JsonValue;
    public enum Period {
        MORNING, AFTERNOON, NIGHT;
        // 3.0820-2 : d.hhmm-n
        public static Period
    getPeriodByTime(String time) {
        int hour =
    Integer.valueOf(time.substring(2, 4));
        if (hour <= 12) {
            return MORNING;
        }
        if (hour >= 18) {
            return NIGHT;
        }
        return AFTERNOON;
    }
    @JsonValue
    public int jsonValue() {
        return super.ordinal();
    }
}
package com.tantum.app.tantum.models;
import java.util.List;
import lombok.AllArgsConstructor;
```

```
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Subject {

    private String nome;

    private String codigo;

    private int fase;

    private String professor;

    private int aulas;

    private boolean obrigatoria;

    private List<String> horarios;

    private List<String> requisitos;

}
package com.tantum.app.tantum.models;

import java.util.List;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Statistics {

    private int semestersRemaining;
```

```

        private List<String> semesters;

        private List<Double> semestersIA;

        private List<Double> courseIA;
    }
    package com.tantum.app.tantum.models;

    import java.util.List;

    import lombok.AllArgsConstructor;
    import lombok.Getter;
    import lombok.Setter;

    @Setter
    @Getter
    @AllArgsConstructor
    public class SemesterHistory {

        private String semester;

        private List<String> subjects;
    }
    package com.tantum.app.tantum;

    import java.util.ArrayList;
    import java.util.Arrays;
    import java.util.List;
    import java.util.stream.Collectors;

    import lombok.extern.slf4j.Slf4j;

    import org.springframework.http.HttpStatus;
    import org.springframework.http.MediaType;
    import
    org.springframework.web.bind.MissingServletRequestParameterExceptio
    import
    org.springframework.web.bind.annotation.ControllerAdvice;
    import
    org.springframework.web.bind.annotation.CrossOrigin;
    import
    org.springframework.web.bind.annotation.ExceptionHandler;
    import
    org.springframework.web.bind.annotation.PathVariable;

```

```

import
org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.ResponseStatus;
import
org.springframework.web.bind.annotation.RestController;

import com.google.gson.Gson;
import com.tantum.app.tantum.algorithm.Algorithm;
import com.tantum.app.tantum.helper.Helper;
import com.tantum.app.tantum.models.Constraints;
import com.tantum.app.tantum.models.Course;
import com.tantum.app.tantum.models.History;
import com.tantum.app.tantum.models.LoginDTO;
import
com.tantum.app.tantum.models.NextSemestersDTO;
import com.tantum.app.tantum.models.Result;
import com.tantum.app.tantum.models.Semester;
import
com.tantum.app.tantum.models.SemesterHistory;
import com.tantum.app.tantum.models.SemestersDTO;
import com.tantum.app.tantum.models.Statistics;
import com.tantum.app.tantum.models.Subject;
import com.tantum.app.tantum.models.SubjectsDTO;

@Slf4j
@RequestMapping("/v1/")
@CrossOrigin
@RestController
@ControllerAdvice
public class TantumController {

    @RequestMapping(path = "test", method =
RequestMethod.GET)
    public String test() {
        return "This is a Test, REST API
is Working";
    }

    @RequestMapping(path = "login", method =

```

```

RequestMethod.GET)
    public LoginDTO login(@RequestParam(value
= "token", required = true) String token) {
        // return new
LoginDTO(this.loginService.doAuthenticate(token),
token);
        return new LoginDTO(true, token);
    }

    @RequestMapping(path = "calculate-
semester", method = RequestMethod.POST, consumes =
MediaType.APPLICATION_JSON_UTF8_VALUE)
    public SemestersDTO
calculateSemester(@RequestParam(value = "token",
required = true) String token,
        @RequestBody(required =
true) Constraints constraints) {

        String c = Helper.course_test;
        String h =
Helper.class_history_test;

        Gson g = new Gson();
        Course curso = g.fromJson(c,
Course.class);
        History history = g.fromJson(h,
History.class);
        SemesterHistory xx =
history.getSemesters().stream().reduce((x, y) -> {
x.getSubjects().addAll(y.getSubjects());
        return x;
    }).orElse(new SemesterHistory("",
Arrays.asList()));
        Algorithm a = new Algorithm(curso);
        a.rankDisciplinas();
        NextSemestersDTO result =
a.calculateSemesters(constraints,
xx.getSubjects());

        log.info("calculate-semester");
        return new SemestersDTO(true,
result);
    }

    @RequestMapping(path = "schedule/

```

```

{semester}", method = RequestMethod.GET)
    public SubjectsDTO
schedule(@RequestParam(value = "token", required =
true) String token,
        @PathVariable String
semester) {

    String c = Helper.course_test;
    String h =
Helper.class_history_test;

    Gson g = new Gson();
    Course curso = g.fromJson(c,
Course.class);
    History history = g.fromJson(h,
History.class);

    List<Subject> subjects = new
ArrayList<>();

    for (SemesterHistory sh :
history.getSemesters()) {
        for (String code :
sh.getSubjects()) {
            curso.getSubjects()
                .stream()
                .filter(s
                    .findFirst()
                    .ifPresent(subje

        }

        SubjectsDTO disciplinasDTO = new
SubjectsDTO(true, new Semester(subjects));

        disciplinasDTO.setSemestre(semester);
        log.info("/schedule/" + semester);
        return disciplinasDTO;
    }

    @RequestMapping(path = "subjects", method
= RequestMethod.GET)
    public SubjectsDTO
subjects(@RequestParam(value = "token", required =
true) String token) {

```



```

        String c = Helper.course_test;
        String h =
Helper.class_history_test;

        Gson g = new Gson();
        Course curso = g.fromJson(c,
Course.class);
        History history = g.fromJson(h,
History.class);

        List<Subject> subjects =
curso.getSubjects().stream().filter(s -> {
            for (SemesterHistory sh :
history.getSemesters()) {
                if
                (sh.getSubjects().contains(s.getCodigo())) {
                    return
                    false;
                }
            }
        }).collect(Collectors.toList());

        SubjectsDTO disciplinasDTO = new
SubjectsDTO(true, new Semester(subjects));
        log.info("subjects");
        return disciplinasDTO;
    }

    @RequestMapping(path = "statictics",
method = RequestMethod.GET)
    public Statistics
statictics(@RequestParam(value = "token", required
= true) String token) {
        List<String> semesters =
Arrays.asList("2015-1", "2015-2", "2016-1",
"2016-2", "2017-1", "2017-2");
        List<Double> semestersIA =
Arrays.asList(4.0, 6.0, 7.5, 5.0, 6.0, 7.0);
        List<Double> courseIA =
Arrays.asList(6.0, 4.0, 6.5, 4.0, 5.0, 6.0);

        return new Statistics(2,
semesters, semestersIA, courseIA);
    }

```

```

        @ResponseStatus(HttpStatus.BAD_REQUEST) //
400
    @ExceptionHandler(MissingServletRequestParameterException.class)
    public Result<String>
    handleMissingToken(MissingServletRequestParameterException
    ex) {
        String name =
    ex.getParameterName();
        return new Result<String>(false,
    name + " parameter is missing") {
            };
        }
    }

    package com.tantum.app.tantum.helper;

    import java.util.ArrayList;
    import java.util.Arrays;
    import java.util.List;

    public class ScheduleHelper {

        private static List<String> schedule = new
    ArrayList<>();

        static {

    schedule.addAll(Arrays.asList("0730", "0820",
    "0910", "1010", "1100", "1330", "1420", "1510",
    "1620", "1710", "1830", "1920", "2020", "2110"));
        }

        /**
         * classTime in format: d.hhmm-n
         *
         * @param horario
         * @return
         */
        public static List<String> getNexts(String
    classTime) {
            int index =
    schedule.indexOf(getUnformattedTime(classTime));
            int qt =
    Integer.valueOf(classTime.substring(7));

```

```

        return schedule.subList(index,
index + qt);
    }

    public static String
getUnformattedTime(String classTime) {
        return classTime.substring(2, 6);
    }

    public static String
getFormattedTime(String classTime) {
        String unformattedTime =
getUnformattedTime(classTime);

        StringBuilder sb = new
StringBuilder();

        sb.append(unformattedTime.substring(0, 2));
        sb.append(":");

        sb.append(unformattedTime.substring(2, 4));

        return sb.toString();
    }
}
package com.tantum.app.tantum.helper;

import java.util.Arrays;
import java.util.List;

public class Helper {

    public static String course_test =
"{\"subjects\": [{\"nome\": \"Materia 1\", \"codigo\":
\"INE5666\", \"fase\": 1, \"professor\": \"Professor
aleatório\", \"aulas\": 2, \"obrigatoria\": true,
\"horarios\": [\"3.0820-2 / CTC-CTC102\"],
\"requisitos\": []}, {\"nome\": \"Materia 2\",
\"codigo\": \"INE5661\", \"fase\": 1, \"professor\":
\"Professor aleatório\", \"aulas\":
2, \"obrigatoria\": true, \"horarios\": [\"3.1620-2 /
CTC-CTC102\"], \"requisitos\": []}, {\"nome\":
\"Materia 3\", \"codigo\": \"INE1111\", \"fase\":
1, \"professor\": \"Professor aleatório\", \"aulas\":

```

```

2, \"obrigatoria\":true, \"horarios\":[\"2.1330-4 /
CTC-CTC102\"], \"requisitos\":[], {\"nome\":
\"Materia 4\", \"codigo\": \"INE5656\", \"fase\":
2, \"professor\": \"Professor aleatório\", \"aulas\":
2, \"obrigatoria\":true, \"horarios\":[\"2.0820-2 /
CTC-CTC102\"], \"requisitos\":[\"INE5666\"]},
{\"nome\": \"Materia 5\", \"codigo\": \"INE5156\",
\"fase\":2, \"professor\": \"Professor aleatório\",
\"aulas\":2, \"obrigatoria\":true, \"horarios\":
[\"2.1010-2 / CTC-CTC102\"], \"requisitos\":[]},
{\"nome\": \"Materia 6\", \"codigo\": \"INE5612\",
\"fase\":2, \"professor\": \"Professor aleatório\",
\"aulas\":2, \"obrigatoria\":true, \"horarios\":
[\"5.1330-3 / CTC-CTC102\"], \"requisitos\":[]},
{\"nome\": \"Materia 7\", \"codigo\": \"INE5688\",
\"fase\":3, \"professor\": \"Professor aleatório\",
\"aulas\":2, \"obrigatoria\":true, \"horarios\":
[\"5.1330-3 / CTC-CTC102\"], \"requisitos\":
[\"INE5612\"]}, {\"nome\": \"Materia 8\", \"codigo\":
\"INE5132\", \"fase\":3, \"professor\": \"Professor
aleatório\", \"aulas\":2, \"obrigatoria\":true,
\"horarios\":[\"2.1010-2 / CTC-CTC102\"],
\"requisitos\":[\"INE5656\"]}, {\"nome\": \"Materia
9\", \"codigo\": \"INE5133\", \"fase\":
4, \"professor\": \"Professor aleatório\", \"aulas\":
2, \"obrigatoria\":true, \"horarios\":[\"4.1620-2 /
CTC-CTC102\"], \"requisitos\":[]}, {\"nome\":
\"Materia 10\", \"codigo\": \"INE5134\", \"fase\":
4, \"professor\": \"Professor aleatório\", \"aulas\":
2, \"obrigatoria\":true, \"horarios\":[\"2.1010-2 /
CTC-CTC102\"], \"requisitos\":[\"INE5132\",
\"INE5661\"]}}]}}";

```

```

    public static String class_history_test =
    "{\"semesters\":[{\"semester\":\"2015-1\",
    \"subjects\":[\"INE5666\", \"INE5661\",
    \"INE1111\"]}, {\"semester\":\"2015-2\",
    \"subjects\":[\"INE5656\", \"INE5156\"]}]}}";

```

```

    public static List<String> semesters =
    Arrays.asList("2018-1", "2018-2", "2019-1",
    "2019-2", "2020-1", "2020-2", "2021-1", "2021-2");
}
package com.tantum.app.tantum;

```

```

import
org.springframework.boot.autoconfigure.web.ErrorController;
import org.springframework.stereotype.Controller;

@Controller
public class TantumErrorController implements
ErrorController {

    // https://stackoverflow.com/questions/31134333/this-application-has-no-explicit-mapping-for-error

    private static final String ERROR_PATH = "/"
error";

    @Override
    public String getErrorPath() {
        return ERROR_PATH;
    }

}

package com.tantum.app.tantum.algorithm;

public class ConstraintsNames {

    private ConstraintsNames() {

    }

    private static final String credit_max =
"credit_max";
    private static final String times = "times";
    private static final String period =
"period";
    private static final String subjects_wanted
= "subjects_wanted";
    private static final String
subjects_not_wanted = "subjects_not_wanted";
    private static final String requisites =
"requisites";

}

package com.tantum.app.tantum.algorithm;

import static
com.tantum.app.tantum.algorithm.ConstraintsNames.credit_max;
import static

```

```

com.tantum.app.tantum.algorithm.ConstraintsNames.period;
import static
com.tantum.app.tantum.algorithm.ConstraintsNames.requisites;
import static
com.tantum.app.tantum.algorithm.ConstraintsNames.subjects_not_wanted;
import static
com.tantum.app.tantum.algorithm.ConstraintsNames.subjects_wanted;
import static
com.tantum.app.tantum.algorithm.ConstraintsNames.times;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.function.Function;
import java.util.stream.Collectors;
import java.util.stream.Stream;

import lombok.Getter;

import org.chocosolver.solver.Model;
import org.chocosolver.solver.Solver;
import
org.chocosolver.solver.variables.impl.FixedBoolVarImpl;

import com.tantum.app.tantum.helper.Helper;
import com.tantum.app.tantum.models.Constraints;
import com.tantum.app.tantum.models.Course;
import
com.tantum.app.tantum.models.NextSemestersDTO;
import com.tantum.app.tantum.models.Period;
import com.tantum.app.tantum.models.Semester;
import com.tantum.app.tantum.models.Subject;

@Getter
public class Algorithm {

    private Map<String, Subject> subjects;

    private Map<String, Integer> rank;

    private Map<Integer, Map<String,
List<String>>> course;

```

```

        private LinkedHashMap<String, Semester>
semesters = new LinkedHashMap<>();

        private List<Subject> subjectsWantedError;

        public Algorithm(Course curso) {
            this.subjects =
curso.getSubjects().stream().collect(Collectors.toMap(Subject::g
Function.identity()));
            // recebe a lista de disciplinas
do curso e monta do map
            this.course = new HashMap<>();

curso.getSubjects().stream().forEach(d -> {
                if (!
this.course.containsKey(d.getFase())) {
this.course.put(d.getFase(), new HashMap<>());
                }

this.course.get(d.getFase()).put(d.getCodigo(),
d.getRequisitos());
            });

        public void rankDisciplinas() {
            this.rank = new HashMap<>();

            int qtFases = this.course.size();
            // para cada materia da fase,
começando pela ultima fase
            for (int i = qtFases; i >= 1; i--)
            {
                for (String materia :
this.course.get(i).keySet()) {
                    // controle da
disciplina
                    if
(this.rank.containsKey(materia)) {
                        // se ja
existe a disciplina, soma 1
                        Integer
aux = this.rank.get(materia);
                        aux++;
                    }
                }
            }
        }
    }
}

```

```

this.rank.replace(materia, aux);
                                } else {
                                // senao
adiciona com valor 1
this.rank.put(materia, 1);
                                }

                                // controle dos
requisitos
                                // para cada
requisito da materia
                                for (String
requisito : this.course.get(i).get(materia)) {
                                if
(this.rank.containsKey(requisito)) {
                                //
se ja existe o requisito, soma 1
Integer aux = this.rank.get(requisito);
aux++;
this.rank.replace(requisito, aux);
                                } else {
                                //
senao passa o valor da materia
this.rank.put(requisito, this.rank.get(materia));
                                }
                                }
                                }
}

private List<String> getSubjectsByRank() {
    return this.rank.entrySet()
                .stream()
                .sorted((e1, e2) -
> Integer.compare(e2.getValue(), e1.getValue()))
                .map(Entry::getKey)
                .collect(Collectors.toList());
}

private Solver

```



```

applyConstraints(Constraints constraints,
List<String> subjectsHistory, Subject subject,
List<Subject> currentSubjects) {
    Model model = new Model();
    Solver solver = model.getSolver();

    model.addClauseTrue(model.boolVar(credit_max,
this.validateClassHourLoad(constraints,
currentSubjects, subject)));

    model.addClauseTrue(model.boolVar(times,
this.validateTime(currentSubjects, subject)));

    model.addClauseTrue(model.boolVar(period,
this.validadePeriod(constraints, subject)));

    model.addClauseTrue(model.boolVar(subjects_wanted,
this.validateSubjectsWanted(constraints,
subject)));

    model.addClauseTrue(model.boolVar(subjects_not_wanted,
this.validateSubjectsNotWanted(constraints,
subject)));

    model.addClauseTrue(model.boolVar(requisites,
this.validateRequisites(subject,
subjectsHistory)));

        return solver;
    }

    public NextSemestersDTO
calculateSemesters(Constraints constraints,
List<String> subjectsHistory) {
    List<String> rankDisciplinas =
this.getSubjectsByRank();

    rankDisciplinas.removeAll(subjectsHistory);
    List<String> newSubjectsHistory =
new ArrayList<>(subjectsHistory);
    List<Subject> subjectsSemester =
null;

    int i = 0;
    while (!rankDisciplinas.isEmpty())

```

```

{
    i++;
    if
(this.semesters.isEmpty()) {
        subjectsSemester =
this.calculateFirstSemester(constraints,
newSubjectsHistory, rankDisciplinas);
    } else {
        subjectsSemester =
this.calculateSemester(constraints,
newSubjectsHistory, new ArrayList<>(),
rankDisciplinas);
    }

    if
(subjectsSemester.isEmpty()) {
        break;
    }

this.semesters.put(Helper.semesters.get(i), new
Semester(subjectsSemester));

subjectsSemester.stream().map(Subject::getCodigo).forEach(codigo
-> {
    rankDisciplinas.remove(codigo);
    newSubjectsHistory.add(codigo);
});
    }
    return new NextSemestersDTO(
        this.semesters,
this.subjectsWantedError, new ArrayList<>(),
rankDisciplinas.stream()
                                .map(this.subjectsWantedError)
                                .collect(Collectors.toList()));
}

private List<Subject>
calculateFirstSemester(Constraints constraints,
List<String> subjectsHistory, List<String>
subjectsRemaining) {

```

```

        this.subjectsWantedError = new
ArrayList<>();
        List<Subject> currentSubjects = new
ArrayList<>();
        List<String> newSubjectsRemaining
= new ArrayList<>(subjectsRemaining);

        for (String s :
constraints.getSubjectsWanted()) {
            Subject subject =
this.subjects.get(s);
            Solver solver =
this.applyConstraints(constraints,
subjectsHistory, subject, currentSubjects);
            boolean solve =
solver.solve();
            if (solve) {
currentSubjects.add(subject);
            } else {
this.subjectsWantedError.add(subject);
            }
        }

constraints.getSubjectsNotWanted().forEach(newSubjectsRemaining:
currentSubjects.stream().map(Subject::getCodigo).forEach(newSubj

// clear subjects wanted and not
wanted

constraints.setSubjectsWanted(Arrays.asList());
constraints.setSubjectsNotWanted(Arrays.asList());

        return
this.calculateSemester(constraints,
subjectsHistory, currentSubjects,
newSubjectsRemaining);
    }

    private List<Subject>
calculateSemester(Constraints constraints,
List<String> subjectsHistory, List<Subject>

```

```

currentSubjects, List<String> subjectsRemaining) {
    List<Subject> newCurrentSubjects =
new ArrayList<>(currentSubjects);

        for (String d : subjectsRemaining)
    {
        Subject disciplina =
this.subjects.get(d);
        Solver solver =
this.applyConstraints(constraints,
subjectsHistory, disciplina, newCurrentSubjects);
        boolean solve =
solver.solve();
        if (solve) {
newCurrentSubjects.add(disciplina);
        } else {
            if (!
this.checkCargaHorariaOk(solver)) {
                break;
            }
        }
    }

        return newCurrentSubjects;
    }

    private boolean validateRequisites(Subject
disciplina, List<String> subjects) {
        return
subjects.containsAll(disciplina.getRequisitos());
    }

    private boolean checkCargaHorariaOk(Solver
solver) {
        return
Stream.of(solver.getModel().getVars())
            .filter(v ->
v.getName().equals(credit_max))
            .map(v ->
((FixedBoolVarImpl) v).getValue() == 1 ? true :
false)
            .findAny().orElse(false);
    }

    private boolean

```

```

validateSubjectsNotWanted(Constraints constraints,
Subject subject) {
    if
    (constraints.getSubjectsNotWanted().isEmpty()) {
        return true;
    } else {
        return !
constraints.getSubjectsNotWanted().contains(subject.getCodigo())
    }
}

private boolean
validateSubjectsWanted(Constraints constraints,
Subject subject) {
    if
    (constraints.getSubjectsWanted().isEmpty()) {
        return true;
    } else {
        return
constraints.getSubjectsWanted().contains(subject.getCodigo());
    }
}

private boolean
validateClassHourLoad(Constraints constraints,
List<Subject> currentSubjects, Subject subject) {
    int cargaHoraria =
currentSubjects.stream().mapToInt(Subject::getAulas).sum()
+ subject.getAulas();

    return cargaHoraria <=
constraints.getCreditMax();
}

private boolean validadePeriod(Constraints
settings, Subject disciplina) {
    return disciplina.getHorarios()
        .stream()
        .map(Period::getPeriodByTime)
        .allMatch(settings.getPeriods());
}

private boolean validateTime(List<Subject>
currentSubjects, Subject currentDisciplina) {
    for (Subject s : currentSubjects) {
        for (String time :

```

```

s.getHorarios()) {
    (currentDisciplina.getHorarios().contains(time)) {
        if
        return
    }
    }
    }
    return true;
}
}
package com.tantum.app.tantum;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.context.annotation.ComponentScan;

@ComponentScan(basePackages = { "com.tantum" })
@SpringBootApplication
public class TantumApplication {

    public static void main(String[] args) {

SpringApplication.run(TantumApplication.class,
args);
    }
}
package com.tantum.app.tantum.services;

import java.util.Objects;

import javax.annotation.PostConstruct;

import lombok.extern.slf4j.Slf4j;

import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import com.tantum.app.tantum.models.SeticAuthDTO;

@Slf4j
@Service

```

```

public class SeticService {

    private RestTemplate restTemplate;

    private static final String URL = "https://
sistemas.ufsc.br/oauth2.0/profile?access_token=";
    private static final String
VINCULO_PESSOA_WS = "https://
ws.homologacao.ufsc.br/rest/
CadastroPessoaUsuarioService/vinculosPessoa";
    private static final String
MATRICULAS_CAGR_WS = "https://
ws.homologacao.ufsc.br/rest/CAGRUusuarioService/
matriculas";
    private static final String
INFORMACAO_ALUNO_CAGR_WS = "https://
ws.homologacao.ufsc.br/rest/CAGRUusuarioService/
informacaoAluno";

    private static final String AUTH_HEADER =
"Authorization";
    private static final String BEARER =
"Bearer ";

    @PostConstruct
    private void init() {
        this.restTemplate = new
RestTemplate();
    }

    public Boolean doAuthenticate(String
token) {
        try {
            SeticAuthDTO dto =
this.restTemplate.getForObject(URL + token,
SeticAuthDTO.class);
            log.info("Authntentication
successful, id: " + dto.getId());
            return
Objects.isNull(dto.getError());
        } catch (Exception e) {
            log.error("Error in
authentication with token: " + token + " " + e);
        }
        return false;
    }
}

```

```

        public String getSubjects(String token) {
            return
this.restTemplate.postForObject(INFORMACAO_ALUNO_CAGR_WS,
this.getHeader(token), String.class);
        }

        public String getMatriculas(String token) {
            return
this.restTemplate.postForObject(MATRICULAS_CAGR_WS,
this.getHeader(token), String.class);
        }

        public String getInfo(String token) {
            return
this.restTemplate.postForObject(VINCULO_PESSOA_WS,
this.getHeader(token), String.class);
        }

        private HttpEntity<String>
getHeader(String token) {
            HttpHeaders headers = new
HttpHeaders();
            headers.add(AUTH_HEADER, BEARER +
token);

            return new
HttpEntity<>("parameters", headers);
        }
    }
    {
        "semesters": [
            {
                "semester": "2015-1",
                "subjects": ["INE5666", "INE5661",
"INE1111"]
            },
            {
                "semester": "2015-2",
                "subjects": ["INE5656", "INE5156"]
            }
        ]
    }
    {
        "disciplinas": [
            {

```



```
"nome": "Materia Aleatoria 1",
"codigo": "INE5666",
"fase": 1,
"professor": "Professor aleatório",
"aulas": 2,
"obrigatoria": true,
"horarios": ["3.0820-2 / CTC-CTC102"],
"requisitos": []
},
{
  "nome": "Materia Aleatoria 2",
  "codigo": "INE5661",
  "fase": 1,
  "professor": "Professor aleatório",
  "aulas": 2,
  "obrigatoria": true,
  "horarios": ["3.1620-2 / CTC-CTC102"],
  "requisitos": []
},
{
  "nome": "Materia Aleatoria 3",
  "codigo": "INE1111",
  "fase": 1,
  "professor": "Professor aleatório",
  "aulas": 2,
  "obrigatoria": true,
  "horarios": ["2.1330-4 / CTC-CTC102"],
  "requisitos": []
},
{
  "nome": "Materia Aleatoria 4",
  "codigo": "INE5656",
  "fase": 2,
  "professor": "Professor aleatório",
  "aulas": 2,
  "obrigatoria": true,
  "horarios": ["2.0820-2 / CTC-CTC102"],
  "requisitos": [
    "INE5666"
  ]
},
{
  "nome": "Materia Aleatoria 5",
  "codigo": "INE5156",
  "fase": 2,
  "professor": "Professor aleatório",
```

```

    "aulas": 2,
    "obrigatoria": true,
    "horarios": ["2.1010-2 / CTC-CTC102"],
    "requisitos": []
  },
  {
    "nome": "Materia Aleatoria 6",
    "codigo": "INE5612",
    "fase": 2,
    "professor": "Professor aleatório",
    "aulas": 2,
    "obrigatoria": true,
    "horarios": ["5.1330-3 / CTC-CTC102"],
    "requisitos": []
  },
  {
    "nome": "Materia Aleatoria 7",
    "codigo": "INE5688",
    "fase": 3,
    "professor": "Professor aleatório",
    "aulas": 2,
    "obrigatoria": true,
    "horarios": ["5.1330-3 / CTC-CTC102"],
    "requisitos": [
      "INE5612"
    ]
  },
  {
    "nome": "Materia Aleatoria 8",
    "codigo": "INE5132",
    "fase": 3,
    "professor": "Professor aleatório",
    "aulas": 2,
    "obrigatoria": true,
    "horarios": ["2.1010-2 / CTC-CTC102"],
    "requisitos": [
      "INE5656"
    ]
  },
  {
    "nome": "Materia Aleatoria 9",
    "codigo": "INE5133",
    "fase": 4,
    "professor": "Professor aleatório",
    "aulas": 2,
    "obrigatoria": true,

```

```
    "horarios": ["4.1620-2 / CTC-CTC102"],
    "requisitos": []
  },
  {
    "nome": "Materia Aleatoria 10",
    "codigo": "INE5134",
    "fase": 4,
    "professor": "Professor aleatório",
    "aulas": 2,
    "obrigatoria": true,
    "horarios": ["2.1010-2 / CTC-CTC102"],
    "requisitos": [
      "INE5132",
      "INE5661"
    ]
  }
]
}
```

### 10.3 DOCUMENTAÇÃO

## # bridge-docs

Repositório base para documentação de projetos, utilizando [Jekyll](<https://jekyllrb.com/>) para geração de sites estáticos de documentação através de arquivos escritos em Markdown.

### ## Como adicionar a um projeto

Adicione este sistema de documentação a um projeto com o comando:

```
```sh
$ git subtree add --squash --prefix docs/
git@github.com:laboratoriobridge/docs.git master
```
```

Uma pasta `docs/` será criada com os fontes deste repositório. Modifique os arquivos de acordo com as necessidades do projeto.

### ## Como atualizar

A partir da raiz de um projeto usando este sistema de documentação, execute:

```
```sh
$ git subtree pull --squash --prefix docs/
git@github.com:laboratoriobridge/docs.git master
```
```

Sendo `docs/` o nome da pasta que contém os fontes do sistema de documentação.

Este comando irá atualizar a versão local da documentação com as últimas atualizações do repositório base. Os arquivos específicos da versão local serão mantidos. Resolva os conflitos, caso existam, e commite o resultado.

```
---
layout: doc
title: Login
id: login
order: 1
---
```

### ### Campos

- Idioma;
- IDUFSC ou CPF;
- Senha ID UFSC;
- Manter conectado.

### ### Regras

**\*\*1.\*\*** Os campos devem vir preenchidos por padrão de acordo com o seguinte mapeamento:

| Campo            | Valor       |
|------------------|-------------|
| Idioma           | Português   |
| IDUFSC ou CPF    | Em branco   |
| Senha ID UFSC    | Em branco   |
| Manter conectado | Selecionado |

**\*\*2.\*\*** Sobre o campo **\*\*Idioma\*\***:

- Deve possuir as seguintes opções de preenchimento:

- Português;
- English;
- Español.

- Uma vez selecionado o idioma, todo o sistema deve mudar se adequar a linguagem escolhida pelo usuário.

**\*\*3.\*\*** Ao acionar a ação **\*\*Manter conectado\*\***, pular a etapa de login a cada entrada do usuário no sistema.

**\*\*4.\*\*** Ao acionar a ação **\*\*Sobre\*\*** redirecionar o usuário para a tela {% link sobre %}.

### ### Internacionalização

|           |          |
|-----------|----------|
| Português | Inglês   |
| Espanhol  |          |
| -----     | -----    |
| Idioma    | Language |
| Idioma    |          |

|                  |                  |          |
|------------------|------------------|----------|
| IDUFSC ou CPF    | IDUFSC           |          |
| IDUFSC           |                  |          |
| Senha ID UFSC    | ID UFSC password | ID UFSC  |
| contraseña       |                  |          |
| Manter conectado | Keep connected   | Mantener |
| conectado        |                  |          |
| Entrar           | Log in           |          |
| Entrar           |                  |          |
| Sobre            | About            |          |
| Sobre            |                  |          |

### ### Protótipos de tela

```

![Login.](img/login.PNG "Protótipo 1 - Login.")
*Protótipo 1 - Login.*---
layout: doc
title: Sobre
id: sobre
parent: login
order: 1
---
```

### ### Conteúdo

A tela de sobre do aplicativo é composta pelas seguintes mensagens (de acordo com a linguagem de exibição).

#### ##### Português

O objetivo dessa aplicação é auxiliar você a montar sua grade de horários para o próximo semestre de forma a minimizar seu tempo de formação e encaixar as disciplinas nos turnos de estudo que são mais convenientes com sua rotina. Também serão disponibilizadas as funções de visualização de grade de horários e estatísticas de acompanhamento do curso.

Este aplicativo foi desenvolvido como TCC (Trabalho de conclusão de curso) pelos alunos Isaac Silva e Pedro Pacheco do curso de Sistemas de Informação.

#### ##### Inglês

The purpose of this application is to help you set up your schedule for the next semester in order to minimize your training time and fit the disciplines in the study shifts that are more convenient with your routine. Also available are the grid display functions and course tracking statistics.

This application was developed as a Course Completion Work by students Isaac Silva and Pedro Pacheco of the course of Information Systems.

#### ##### Espanhol

El objetivo de esta aplicación es ayudarle a montar su cuadrícula de horarios para el próximo semestre para minimizar su tiempo de formación y encajar las disciplinas en los turnos de estudio que son más convenientes con su rutina. También se pondrán a disposición las funciones de visualización de rejilla de horarios y estadísticas de seguimiento del curso.

Esta aplicación ha sido desarrollada a través de estudios de aprendizaje por parte de los alumnos de Isaac Silva y Pedro Pacheco del curso de sistemas de información.---

```
layout: doc
title: Introdução
id: introducao
permalink: /
---
```

#### \*\*Módulos:\*\*

```
- {% link login %};
- {% link menu_principal %}.
- {% link geral %};---
layout: doc
title: Grade de horários
id: grade_horarios
parent: menu_principal
order: 1
---
```

#### ### Modos de exibição



- Diário;
- Semanal.

### ### Campos

- Diário:
  - Dia da semana;
  - Horário de início da disciplina;
  - Sala de aula;
  - Nome da disciplina.
- Semanal:
  - Dia da semana;
  - Horário de início da disciplina;
  - Sala de aula.

### ### Regras

**\*\*1.\*\*** O módulo deve ser aberto por padrão de acordo com o seguinte mapeamento:

- **\*\*Aba:\*\*** Diário;
- **\*\*Dia da semana:\*\*** Dia da semana no qual o aplicativo está sendo utilizado.

**\*\*2.\*\*** Sobre os dias que não possuem disciplinas alocadas:

- Caso não seja fim de semana: Exibir o dia normalmente, porém apresentar a mensagem: "Não há aulas de nenhuma disciplina neste dia";
- Caso seja fim de semana: Não exibir o dia.

**\*\*3.\*\*** Os nomes das disciplinas não devem ser traduzidos para outros idiomas.

### ##### Diário

**\*\*4.\*\*** Para realizar a transição entre os dias da semana, o usuário deverá arrastar a tela para avançar (da direita para a esquerda) ou retroceder (da esquerda para a direita) entre os dias da semana.

**\*\*5.\*\*** Os dias da semana devem ser apresentados em

sua forma abreviada.

**\*\*6.\*\*** O dia da semana que está sendo exibido deve receber um destaque de acordo com o seguinte protótipo.

**\*\*7.\*\*** Ao selecionar uma disciplina da listagem, redirecionar o usuário para a tela de detalhes da disciplina `{% link detalhes_disciplina %}`.

#### ##### Semanal

**\*\*8.\*\*** Ao selecionar uma disciplina da listagem, redirecionar o usuário para a tela de detalhe da disciplina `{% link grade_horarios_semanal %}`.

#### ### Internacionalização

|                   |           |         |
|-------------------|-----------|---------|
| Português         | Inglês    |         |
| Espanhol          |           |         |
| -----             | -----     |         |
| -----             |           |         |
| Grade de horários | Timetable | Horario |
| de horarios       |           |         |
| Diário            | Daily     |         |
| Diario            |           |         |
| Semanal           |           | Weekly  |
| Semanal           |           |         |

#### ##### Diário

|           |        |          |  |
|-----------|--------|----------|--|
| Português | Inglês | Espanhol |  |
| -----     | -----  | -----    |  |
| SEG       | MON    | LUN      |  |
| TERÇ      | TUE    | MAR      |  |
| QUA       | WED    | MIÉ      |  |
| QUI       | THU    | JUE      |  |
| SEX       | FRI    | VIE      |  |
| SAB       | SAT    | SÁB      |  |

#### ##### Semanal

|           |        |  |
|-----------|--------|--|
| Português | Inglês |  |
| Espanhol  |        |  |
| -----     | -----  |  |

```

----- |
| Segunda |           | Monday   |
Lunes    |           |           |
| Terça   |           | Tuesday  |
Martes   |           |           |
| Quarta  |           | Wednesday|
Miércoles|           |           |
| Quinta  |           | Thursday |
Jueves   |           |           |
| Sexta   |           | Friday   |
| Viernes |           |           |
| Sábado  |           | Saturday |
| Sábado  |           |           |

```

### # Protótipos de tela

```

![Horário diário.](img/horarios_dia.PNG "Protótipo
1 - Horário diário.") *Protótipo 1 - Horário
diário.*

```

```

![Horário semanal.](img/horarios_semanal.PNG
"Protótipo 2 - Horário semanal.") *Protótipo 2 -
Horário semanal.*---

```

```

layout: doc
title: Notificação sobre completude de geração
id: notificacao_resultado
parent: gerar_grade_horarios
order: 3
---

```

### # Regras

**\*\*1.\*\*** Esta tela deve ser exibida somente se houverem muitos acessos ao serviço de geração de grades de horários.

**\*\*2.\*\*** Ao acionar a opção **\*\*Notificar-me ao final do processo\*\*** o usuário deve ser notificado ao final da geração.

### # Internacionalização

```

| Português | Inglês | Espanhol |
| ----- | ----- | ----- |
| Resultado | Result | Resultado |
| Devido ao grande número de acessos ao sistema de

```

montagem de horários, seu resultado pode demorar um pouco. Deseja ser notificado quando o processo finalizar? | Due to the large number of accesses to the scheduling system, its result may take a while. Do you want to be notified when the process ends? | Debido al gran número de accesos al sistema de montaje de horarios, su resultado puede tardar un poco. ¿Desea ser notificado cuando el proceso finalice? |  
 | Notificar-me ao final do processo | Notify me at the end of the process | Notificarme al final del proceso |

---

layout: doc  
 title: Definir critérios - Passo 1  
 id: definir\_criterios  
 parent: gerar\_grade\_horarios  
 order: 1

---

### # Campos

- Turno de estudo;
- Intervalo de créditos;
- Considerar matérias equivalentes?

### # Regras

**\*\*1.\*\*** Os campos devem vir preenchidos por padrão de acordo com o seguinte mapeamento:

| Campo

| Valor

|  
 | ----- |

|  
 | Turno de estudo  
 | Turno (s) de estudo do curso

|  
 | Intervalo de créditos |  
 Intervalo padrão de créditos que podem ser cursados do curso |

| Considerar matérias equivalentes | Não

|

**\*\*2.\*\*** 0 campos **\*\*Turno de estudo\*\*** deve ser composto pelas seguintes opções:

- Manhã;
- Tarde;
- Noite.

**\*\*3.\*\*** 0 campo **\*\*Intervalo de créditos\*\*** não deve permitir valores excedentes ao máximo e mínimo de créditos do curso.

**\*\*4.\*\*** Ao acionar a opção **\*\*Próximo passo\*\*** o usuário deve ser redirecionado para a tela **\*\*LINK\*\***.

## # Internacionalização

| Português

| Inglês

Espanhol

| ----- |

----- |

----- |

| Definir critérios para geração | Define  
criteria for generation | Definir criterios para  
la generación |

| Passo 1

| Step 1

| Paso

1

| Passo 2

Step 2

Paso

2

| Passo 3

Step 3

Paso

3

| Turno de estudo

```

        | Study shift
        | Turno de
estudio |
|
Manhã   | Morning
        |
Mañana  |
|
Tarde   | Afternoon
        |
Tarde   |
|
Noite   | Night
        |
Noche   |
| Intervalo de créditos
Credit interval | Credit
interval
| Considerar matérias equivalentes? | Consider
equivalent courses? | Considerar materias
equivalentes? |
| Próximo
passo    |
Next step
        | Proximo
passo    |

```

### # Protótipos de tela

```

![Definir critérios - Passo 1.](../img/
definir_criterios_1.PNG "Protótipo 1 - Definir
critérios - Passo 1.") *Protótipo 1 - Definir
critérios - Passo 1.*---
layout: doc
title: Definir critérios - Passo 3
id: definir_criterios_passo_3
parent: definir_criterios
order: 3
---
Titulo: Definir critérios - Passo 3

```

### # Campos

- Matérias que não quero cursar nesse semestre.

## # Regras

**\*\*1.\*\*** Sobre o campo **\*\*Matérias que não quero cursar nesse semestre\*\***:

- Deve permitir a pesquisa sobre todas as disciplinas de graduação oferecidas pela UFSC;
- A pesquisa pode ser realizada tanto por código quanto por nome da disciplina;
- Um vez selecionando uma disciplina, inserí-la na lista contida a baixo do campo;
- Cada item da listagem deve possuir a ação **\*\*Remover\*\***, a qual deve remover a disciplina da lista.

**\*\*2.\*\*** Ao acionar a opção **\*\*Próximo passo\*\*** o usuário deve ser redirecionado para a tela **\*\*LINK\*\***.

## # Internacionalização

| Português                                    | Inglês  | Espanhol                                       |
|--|---|--|
| Definir critérios para geração               | Define criteria for generation                  | Definir criterios para la generación           |
| Passo 1                                      | Step 1  | Paso 1   |
| Passo 2                                      | Step 2  | Paso 2   |
| Passo 3                                      | Step 3  | Paso 3   |
| Matérias que não quero cursar nesse semestre | Subjects I don't want to study in this semester | Materias que no quiero cursar en este semestre |
| Remover                                      | Remove  | Eliminar                                       |
| Gerar grade de horários                      | Generate time grid                              | Generar cuadrícula de horas                    |

## # Protótipos de tela

```
![Definir critérios - Passo 3.](../img/definir_criterios_3.PNG "Protótipo 1 - Definir critérios - Passo 3.") *Protótipo 1 - Definir critérios - Passo 3.*---
layout: doc
title: Resultado da geração de grade de horários
```

```
id: resultado_da_geracao_de_grade
parent: gerar_grade_horarios
order: 3
---
```

## # Regras

**\*\*1.\*\*** Caso só haja uma opção de geração, não exibir a opção de alterar opções de geração.

**\*\*2.\*\*** Caso haja mais de uma opção de grade de horários gerada, apresentar no topo da tela opções para visualizar as diferentes opções.

**\*\*3.\*\*** Uma lista deve ser apresentada após as diferentes opções com as matérias que foram selecionadas pelo algoritmo. Ao acionar a ação **\*\*Remover\*\***, a matéria deve ser removida da lista e a grade de horários deve ser atualizada.

**\*\*4.\*\*** Ao selecionar uma disciplina da grade de horários, o detalhe a ser exibido deve seguir o especificado no requisito {% link detalhes\_disciplina %}.

**\*\*5.\*\*** O comportamento da grade de horários está contido no requisito {% link grade\_horarios\_semanal %}.

**\*\*6.\*\*** Ao acionar a opção **\*\*Redefinir critérios de geração\*\*** o usuário deve ser redirecionado para a tela {% link definir\_criterios %}.

## # Internacionalização

```
| Português |
Inglês |
Espanhol |
| ----- |
----- |
----- |
| Resultado |
Result |
Resultado |
| Opção 1 |
| Option
```



```

1                               |
Opção                           |
1                               |
| Opção 2                       | Option
2                               |
Opção                           |
2                               |
| Opção 3                       | Option
3                               |
Opção                           |
3                               |
| Remove                         |
Remove                           |
Eliminar                         |
| Redefinir critérios de geração | Reset
generation criteria             | Redefinir los criterios
de generación                  |

```

### # Protótipos de tela

```

![Resultados da geração de horários.](../img/
resultado.PNG "Protótipo 1 - Resultados da geração
de horários.") *Protótipo 1 - Resultados da
geração de horários.*

```

```

---
layout: doc
title: Definir critérios - Passo 2
id: definir_criterios_passo_2
parent: definir_criterios
order: 2
---

```

### # Campos

- Matérias que quero cursar nesse semestre.

### # Regras

**\*\*1.\*\*** Sobre o campo **\*\*Matérias que quero cursar nesse semestre\*\***:

- Deve permitir a pesquisa sobre todas as

disciplinas de graduação oferecidas pela UFSC;  
 - A pesquisa pode ser realizada tanto por código quanto por nome da disciplina;  
 - Um vez selecionando uma disciplina, inserí-la na lista contida a baixo do campo;  
 - Cada item da listagem deve possuir a ação **\*\*Remover\*\***, a qual deve remover a disciplina da lista.

**\*\*2.\*\*** Ao acionar a opção **\*\*Próximo passo\*\*** o usuário deve ser redirecionado para a tela **\*\*LINK\*\***.

### # Internacionalização

|   |  |        |
|---|--|--------|
| Português                                   |  | Inglês |
|   |  |        |
| Espanhol                                    |  |        |
| -----                                       |  |        |
| -----                                       |  |        |
| Definir critérios para geração              |  |        |
| Define criteria for generation              |  |        |
| Definir criterios para la                   |  |        |
| generación                                  |  |        |
| Passo 1                                     |  | Step 1 |
|   |  | Paso   |
| 1   |  |        |
| Passo 2                                     |  | Step 2 |
|   |  | Paso   |
| 2   |  |        |
| Passo 3                                     |  | Step 3 |
|   |  | Paso   |
| 3   |  |        |
| Matérias que quero cursar nesse semestre    |  |        |
| Subjects I want to study in this semester   |  |        |
| Materias que quiero cursar en este semestre |  |        |
| Remover                                     |  |        |

```

Remove
Eliminar
| Próximo passo
| Next
step
| Proximo
paso
|

```

## # Protótipos de tela

```

![[Definir critérios - Passo 2.](../img/
definir_criterios_2.PNG "Protótipo 1 - Definir
critérios - Passo 2.") *Protótipo 1 - Definir
critérios - Passo 2.*---
layout: doc
title: Gerar grade de horários
id: gerar_grade_horarios
parent: menu_principal
order: 2
---
```

```

{% link definir_criterios %}---
layout: doc
title: Menu principal
id: menu_principal
order: 2
---
```

## ### Opções

- Próximas aulas;
- Grade de horários;
- Geração de grade de horários;
- Estatísticas;
- Sair.

## ### Regras

**\*\*1.\*\*** Sobre a opção **\*\*Próximas aulas\*\***:

- Deve exibir os dois próximos horários de aulas, da forma:
  - Definir como vai ser feita a passagem das aulas.

- Ao selecionar uma disciplina da listagem, redirecionar o usuário para a tela de detalhes da disciplina {% link detalhes\_disciplina %}.

**\*\*3.\*\*** Cada opção deve possuir um ícone informativo. Ao selecionar a informação, cada módulo deve possuir sua respectiva informação de contexto, de acordo com o seguinte mapeamento:

- Grade de horários:

| Idioma    | Texto   |
|-----------|---|
| Português | Esta opção dá acesso a sua grade de horários, podendo ser visualizada de forma diária ou semanal. Selecione um horário específico para obter mais detalhes      |
| Inglês    | This option gives access to your schedule grid, can be viewed daily or weekly. Select a specific time for more details  |
| Espanhol  | Esta opción da acceso a su cuadrícula de horarios, pudiendo ser visualizada de forma diaria o semanal. Seleccione una hora específica para obtener más detalles |

- Geração de grade de horários:

| Idioma    | Texto   |
|-----------|---|
| Português | Esta opção dá acesso a geração a grade de horários, a qual pode ser gerada a partir dos critérios padrões ou customizáveis                        |
| Inglês    | This option gives access to generation the grid of schedules, which can be generated from the standard or customizable criteria                   |
| Espanhol  | Esta opción da acceso a generación a la cuadrícula de horarios, la cual puede ser generada a partir de los criterios estándares o personalizables |

- Estatísticas:

| Idioma    | Texto                                |
|-----------|--------------------------------------|
| Português | Esta opção dá acesso as estatísticas |

sobre a sua formação, as quais dizem respeito ao progresso do seu curso e quais matérias são sugeridas por semestre |  
 | Inglês | Esta opción da acceso a las estadísticas sobre su formación, que se refieren al progreso de su curso y qué materias se sugieren por semestre |  
 | Espanhol | This option gives access to the statistics about your graduation, which relate to the progress of your course and what subjects are suggested per semester |

**\*\*4.\*\*** Ao selecionar um módulo o usuário deve ser redirecionado para sua respectiva tela, conforme o seguinte mapeamento:

```
- **Grade de horários:** {% link grade_horarios %};
- **Geração de grade de horários:** {% link gerar_grade_horarios %};
- **Estatísticas:** .
```

**\*\*5.\*\*** Ao acionar a ação **\*\*Sair\*\***, o usuário deve ter sua seção encerrada, sendo redirecionado para a tela de login {% link login %}.

### ### Internacionalização

```
| Português |
Inglês |
Espanhol |
| ----- |
| ----- |
| Grade de horários |
Timetable | Horario de
horarios |
| Geração de grade de horários | Generation of
time grid | Geração de grade de horários |
| Estatísticas |
Statistics |
Estadística |
| Sair | Log
out | Finalizar la sesión
|
```

### ### Protótipos de tela

```
![[Horário semanal.]](img/main.PNG "Protótipo 1 -  
Menu principal.") *Protótipo 1 - Menu principal.*
```

```
---
```

```
layout: doc  
title: Estatísticas  
id: estatisticas  
parent: menu_principal  
order: 2
```

```
---
```

## # Resumo

É composto pelo resumo da formação do aluno, desde sua entrada na UFSC.

Composto por:

- Gráfico de semestres;
- Gráfico de IA.

O gráfico de semestres é composto por:

- Semestres cursados, o qual é calculado a partir do semestre de entrada do aluno na UFSC;
- Semestres restantes, o qual é calculado a partir da geração de próximos semestres do próprio aplicativo.

O gráfico de IA é composto por:

- Seu IA, populado pelo IA do aluno no decorrer dos semestres;
- IA médio do curso, populado pelo IA médio do curso no decorrer dos semestres.

## # Próximos semestres

Composto pelos próximos semestres que devem ser cursados pelo aluno.

Os semestres são calculados levando em consideração o último semestre gerado através do app e que o aluno seja aprovado em todas as disciplinas.

Composto por:

- Semestre;
- Código;
- Descrição da disciplina.

### # Internacionalização

```

| Português                               |
Inglês                                   |
Espanhol                                 |
| ----- |
----- |
| Estatísticas                            |
Statistics                               |
Estadística                             |
| Resumen                                |
| Resume                                 |
Resumen                                  |
| Próximos semestres                     | Next
semesters                               | Proximos
semestres                               |
| Semestres restantes                    | Semesters
remaining                               | remaining
| Semestres cursados                    | Semesters
semesters                               | Completed
| Semestres completos                    |
|
| Seu IA                                  | Your
IA                                       | Su IA
| IA médio do curso                     | Average IA from
course | IA medio del curso             |

```

### ### Protótipos de tela

```

![Resumo.](img/estatisticas.PNG "Protótipo 1 -
Resumo.") *Protótipo 1 - Resumo.*

```

```

![Próximos semestres.](img/proximos_semestres.PNG
"Protótipo 2 - Próximos semestres.") *Protótipo 2
- Próximos semestres.*---

```

```

layout: doc
title: Detalhes da disciplina
id: detalhes_disciplina
parent: geral

```

order: 1

---

## # Campos

- Disciplina;
- Professor;
- Horários.

## # Regras

**\*\*1.\*\*** No campo **\*\*Disciplina\*\*** deve ser exibido o nome da disciplina selecionada;

**\*\*2.\*\*** Sobre o campo **\*\*Professor\*\***:

- Devem ser exibidos os nomes dos professores que lecionam a disciplina selecionada;
- Caso haja mais de um professor ministrante:
  - Mudar o nome do campo para

**\*\*Professores\*\***;

- Apresentar o nome de cada professor em uma linha.

**\*\*3.\*\*** Sobre o campo **\*\*Horários\*\***:

- Devem ser exibidos os horários de aula da disciplina selecionada;
- Cada registro deve possuir as seguintes informações:
  - Dia da semana;
  - Horário de início;
  - Horário de término;
  - Sala de aula.
- Cada registro deve ser apresentado da forma:
  - [Dia da semana]: [Horário de início] - [Horário de término] ([Sala de aula])
- Exemplos:
  - Segunda: 18:30 - 20:10 (LIICT8)
  - Quinta: 13:30 - 15:10 (CTC203)

**\*\*4.\*\*** Ao acionar a ação **\*\*Voltar\*\*** o usuário deve ser redirecionado para a tela na qual solicitou os detalhes da disciplina.

## # Internacionalização



|                           |                |
|---------------------------|----------------|
| Português                 | Inglês         |
| Espanhol                  |                |
| -----                     | -----          |
|                           |                |
| Detalhes da disciplina    | Course Details |
| Detalles de la asignatura |                |
| Disciplina                | Course         |
| Disciplina                |                |
| Professor                 | Teacher        |
| Profesor                  |                |
| Horários                  | Schedules      |
| Horarios                  |                |
| Segunda                   | Monday         |
| Lunes                     |                |
| Terça                     |                |
| Tuesday                   |                |
|                           |                |
| Quarta                    |                |
| Wednesday                 |                |
|                           |                |
| Quinta                    |                |
| Thursday                  |                |
| Jueves                    |                |
| Sexta                     |                |
| Friday                    |                |
| Viernes                   |                |
| Sábado                    |                |
| Saturday                  |                |
| Sábado                    |                |
| Voltar                    |                |
| Back                      |                |
| Regreso                   |                |

### # Protótipos de tela

```

![Detalhes da disciplina.](img/
disciplina_detail.PNG "Protótipo 1 - Detalhes da
disciplina.") *Protótipo 1 - Detalhes da
disciplina.*---
layout: doc
title: Grade de horários semanal
id: grade_horarios_semanal
parent: geral
order: 1
---
```

## # Campos

- Dia da semana;
- Horário de início da disciplina;
- Sala de aula.

## # Regras

**\*\*1.\*\*** Sobre os dias que não possuem disciplinas alocadas:

- Caso não seja fim de semana: Exibir o dia normalmente, porém apresentar a mensagem: "Não há aulas de nenhuma disciplina neste dia";
- Caso seja fim de semana: Não exibir o dia.

**\*\*2.\*\*** Os nomes das disciplinas não devem ser traduzidos para outros idiomas.

**\*\*3.\*\*** Ao selecionar uma disciplina da listagem, redirecionar o usuário para a tela de detalhe da disciplina {% link detalhes\_disciplina %}.

## # Internacionalização

| Português | Inglês    | Espanhol  |
|-----------|-----------|-----------|
| Segunda   | Monday    | Lunes     |
| Terça     | Tuesday   | Martes    |
| Quarta    | Wednesday | Miércoles |
| Quinta    | Thursday  | Jueves    |
| Sexta     | Friday    | Viernes   |
| Sábado    | Saturday  | Sábado    |

## # Protótipos de tela

```
![[Horário semanal.](img/horarios_semanal.PNG
"Protótipo 1 - Horário semanal.") *Protótipo 1 -
Horário semanal.*---
layout: doc
title: Geral
id: geral
order: 1
---
```

Neste módulo estão contidos os requisitos que serão utilizados em mais de um ponto do sistema.

- {% link detalhes\_disciplina %};
- {% link grade\_\_horarios\_semanal %}.

layout: page

title: Sobre

permalink: /sobre/

icon: icon-about

---

Documentação gerada em **\*\*{{ site.time | date: '%d/%m/%y, às %Hh %Mmin' }}\*\***

---

Este site é uma tentativa de padronizar o processo de escrita, geração e submissão de documentação de projetos.

O código-fonte deste site pode ser usado como base para a criação de documentação específica de projetos. Ao mesmo tempo, este site pode prover guidelines para o processo de documentação.

## 10.4 CASOS DE TESTE

| Grupo | Item                            | Regra  | Caso de teste  |
|-------|---------------------------------|--|--|
| Login | Conteúdo inicial dos campos     | 1. Os campos devem vir preenchidos por padrão de acordo com o seguinte mapeamento:<br><br>Idioma - Português<br>IDUFSC ou CPF - Em branco<br>Senha ID UFSC - Em branco<br>Manter conectado - Selecionado | 1. Abrir o aplicativo  |
|       | Idiomas disponíveis             | 2. Sobre o campo Idioma:<br><br>Deve possuir as seguintes opções de preenchimento:<br><br>- Português;<br>- English;<br>- Español.   | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Idioma";  |
|       | Adequação ao idioma selecionado | 2. Uma vez selecionado o idioma, todo o sistema deve mudar se adequar a linguagem escolhida pelo usuário.  | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Idioma";<br>3. Selecionar um dos idiomas disponíveis;<br>4. Clicar em "Ok".   |
|       | Ação "Manter conectado"         | 3. Ao acionar a ação Manter conectado, pular a etapa de login a cada entrada do usuário no sistema.  | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Manter conectado";<br>3. Usar um login válido;<br>4. Logar no sistema;<br>5. Fechar o aplicativo;<br>6. Abrir o aplicativo. |
|       | Navegação para a tela "Sobre"   | 4. Ao acionar a ação Sobre redirecionar o usuário para a tela  | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Sobre";   |
|       | Internacionalização             | Seção "Internacionalização"  | 1. Abrir o aplicativo;<br>2. Observar as labels;<br>3. Mudar o idioma;<br>4. Observar as labels  |

| Comportamento esperado   | Situação |
|--|----------|
| Os campos da tela de login devem vir preenchidos com os valores mencionados na regra especificada. |          |
| Devem haver 3 opções de idioma:<br><br>- Português;<br>- English;<br>- Español.                    |          |
| Os campos da tela de login alterar seu idioma para o idioma selecionado.                           |          |
| O aplicativo deve ser aberto diretamente no menu principal.  |          |
| O sistema deve navegar para a tela sobre   |          |
| Os campos da tela de login alterar seu idioma para o idioma selecionado.                           |          |

| Grupo          | Item                        | Regra  | Caso de teste   |
|----------------|-----------------------------|--|---|
| Menu Principal | Conteúdo inicial dos campos | 1. Os campos devem vir preenchidos por padrão de acordo com o seguinte mapeamento:<br><br>Próximas aulas: Deve apresentar as duas próximas aulas | 1. Abrir o aplicativo   |
|                | Horários do dia             | 1. Ao acionar a ação Horários do dia redirecionar o usuário para a tela  | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Horários do dia";    |
|                | Gerar grade de horários     | 1. Ao acionar a ação Geração de horários redirecionar o usuário para a tela  | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Geração de horários" |
|                | Estatísticas                | 1. Ao acionar a ação Estatísticas redirecionar o usuário para a tela   | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Estatísticas";       |

| Comportamento esperado   | Situação |
|--|----------|
| Os campos da tela de principal devem vir preenchidos com os valores mencionados na regra especificada. |          |
| O sistema deve navegar para a tela Horários do dia   |          |
| O sistema deve navegar para a tela Geração de horários   |          |
| O sistema deve navegar para a tela Estatísticas  |          |



| Grupo               | Item                                    | Regra   |
|---------------------|---|---|
| Geração de horários | Conteúdo inicial dos campos             | 1. Os campos devem vir preenchidos por padrão de acordo com o seguinte mapeamento:<br>- Segmentos com dos passos 1, 2 e 3             |
|                     | Passo 1                                 | 2. Deve estar no passo 1 e conter os seguintes campos:<br>- Turnos de estudo;<br>- Intervalo de créditos;<br>- Matérias equivalentes; |
|                     | Turnos de estudo                        | 3. Sobre o campo Turnos de estudo:<br>Deve possuir as seguintes opções de preenchimento:<br>- Manhã;<br>- Tarde;<br>- Noite.          |
|                     | Intervalo de créditos                   | 3. Sobre o campo intervalo de créditos:<br>Deve permitir o usuário escolher um intervalo de valores para o crédito máximo e mínimo    |
|                     | Considerar matérias equivalentes        | 3. Sobre o campo matérias equivalentes:<br>Deve permitir o usuário escolher se deseja considerar matérias equivalentes                |
|                     | Passo 2                                 | 4. Deve estar no passo 2 e conter os seguintes campos:<br>- Matérias que quero cursar nesse semestre                                  |
|                     | Busca de matérias que deseja cursar     | 5. Sobre o campo busca de matérias<br>Deve permitir o usuário escolha as matérias que deseja cursar no próximo semestre               |
|                     | Passo 3                                 | 5. Deve estar no passo 2 e conter os seguintes campos:<br>- Matérias que não quero cursar nesse semestre                              |
|                     | Busca de matérias que não deseja cursar | 6. Sobre o campo busca de matérias<br>Deve permitir o usuário escolha as matérias que não deseja cursar no próximo semestre           |

| Caso de teste  | Comportamento esperado   | Situação |
|--|--|----------|
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";  | Os campos da tela de geração de horários devem vir preenchidos com os valores mencionados na regra especificada. |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 1   | Os campos do passo 1 devem vir preenchidos com os valores mencionados na regra especificada.                     |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 1<br>4. Selecionar os turnos                    | Usuário deve poder escolher qualquer uma das opções, sendo estas<br>- Manhã;<br>- Tarde;<br>- Noite.             |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 1<br>4. Selecionar os intervalo                 | O aplicativo deve permitir escolher qualquer valor entre o limite permitido                                      |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 1<br>4. Selecionar matérias equivalentes        | Deve-se permitir a seleção ou não de matérias equivalentes   |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 2   | Os campos do passo 2 devem vir preenchidos com os valores mencionados na regra especificada.                     |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 1<br>4. Busca de matérias que deseja cursar     | Ao buscar as matérias deve-se trazer os valores que foram pesquisados.   |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 3   | Os campos do passo 3 devem vir preenchidos com os valores mencionados na regra especificada.                     |          |
| 1. Abrir o aplicativo;<br>2. Selecionar a opção "Gerar horários";<br>3. Estar no passo 1<br>4. Busca de matérias que não deseja cursar | Ao buscar as matérias deve-se trazer os valores que foram pesquisados.   |          |

| Grupo           | Item                        | Regra   | Caso de teste  |
|-----------------|-----------------------------|---|--|
| Horários do dia | Conteúdo inicial dos campos | 1. Os campos devem vir preenchidos por padrão de acordo com o seguinte mapeamento:<br>- Segmentos com as informações Diário e Semanal<br>- Estar selecionado opção Diário<br>- Estar no dia de hoje | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Horários do dia"  |
|                 | Segmento diário             | 2. Deve possuir as seguintes opções dia:<br>- Segunda, Terça, Quarta, Quinta e Sexta;<br>- Deve estar no dia de hoje;   | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Horários do dia"  |
|                 | Ação "clique na disciplina" | 2. Uma vez selecionado a disciplina o aplicativo deve mostrar um modal com informações mais detalhadas da disciplina  | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Horários do dia"<br>3. Selecionar uma disciplina                            |
|                 | Segmento semanal            | 2. Deve mostrar a grade de horário semanal  | 1. Abrir o aplicativo;<br>2. Selecionar a opção "Horários do dia"<br>3. Selecionar uma disciplina<br>4. Selecionar "Semanal" |

| Comportamento esperado   | Situação |
|--|----------|
| Os campos da tela de horários do dia devem vir preenchidos com os valores mencionados na regra especificada. |          |
| Devem haver os dias da semana como possíveis dias  |          |
| Deve mostrar o modal com as informações da disciplina que foi clicada  |          |
| Deve abrir o segmento de horário semanal e mostrar a grade semanal   |          |

| Grupo        | Item                        | Regra   | Caso de teste   |
|--------------|-----------------------------|---|---|
| Estatísticas | Conteúdo inicial dos campos | <p>1. Os campos devem vir preenchidos por padrão de acordo com o seguinte mapeamento:</p> <ul style="list-style-type: none"> <li>- Segmentos com as informações Resumo e Próximos semestres</li> <li>- Gráfico de tipo rosca com a quantidade de semestres cursados e próximos semestres</li> <li>- Gráfico do tipo linha com o histórico do IA pessoal e do curso</li> </ul> | <ol style="list-style-type: none"> <li>1. Abrir o aplicativo;</li> <li>2. Selecionar a opção "Estatísticas";</li> </ol>   |
|              | Segmento Estatísticas       | <p>Deve possuir os seguintes gráficos</p> <ul style="list-style-type: none"> <li>- Gráfico de tipo rosca com a quantidade de semestres cursados e próximos semestres</li> <li>- Gráfico do tipo linha com o histórico do IA pessoal e do curso</li> </ul>   | <ol style="list-style-type: none"> <li>1. Abrir o aplicativo;</li> <li>2. Selecionar a opção "Estatísticas";</li> <li>3. Clicar no segmento "Resumo"</li> </ol>       |
|              | Segmento Próximos Semestres | <p>Deve possuir uma lista com os possíveis próximos semestres</p>   | <ol style="list-style-type: none"> <li>1. Abrir o aplicativo;</li> <li>2. Selecionar a opção "Estatísticas";</li> <li>3. Clicar no segmento "Próximos Seme</li> </ol> |

| Comportamento esperado  | Situação |
|---|----------|
| Os campos da tela de estatísticas devem vir preenchidos com os valores mencionados na regra especificada.       |          |
| Os campos da tela de resumo devem vir preenchidos com os valores mencionados na regra especificada.             |          |
| Os campos da tela de próximos semestres devem vir preenchidos com os valores mencionados na regra especificada. |          |

## 10.5 ARTIGO

# Desenvolvimento de um aplicativo de auxílio de tomada de decisão na escolha de grade de horários utilizando Ionic

Isaac Luiz da Silva<sup>1</sup>, Pedro Henrique Pacheco<sup>2</sup>

<sup>1</sup>Departamento de informática e estatística – Universidade Federal de Santa Catarina (UFSC)  
Caixa Postal 15.064 – 88.036-020 – Florianópolis – SC – Brasil

isaacluizs@gmail.com, pedro.pacheco@outlook.com

**Abstract.** *Existem diversos programas destinados à solução de problemas de timetabling voltados para professores e instituições. Porém, nossos estudos não evidenciaram a existência de alguma ferramenta que seja mais voltada para as necessidades específicas dos estudantes. Nesse contexto, nosso trabalho terá como objetivo a criação de um aplicativo que possua uma boa usabilidade e que possua como funcionalidade principal auxiliar os alunos na escolha de disciplinas de modo que se possa realizar a completude da sua graduação no menor tempo possível. O aplicativo foi criado utilizando o Ionic Framework, que é um framework destinado à criação de aplicativos híbridos, com o qual é possível, com o mesmo código fonte, criar aplicativos para diversas plataformas. A documentação do aplicativo foi criada utilizando uma ferramenta chamada Jekyll, a qual cria páginas web estáticas utilizando markdown. Para resolver o problema de timetabling é utilizada uma técnica conhecida por constraint programming (programação com restrições), a qual nos permite modelar diversas restrições, como horário, período e fazer a sugestão de possíveis matérias.*

**Resumo.** *There are several programs aimed at solving timetabling problems for teachers and institutions. However, our studies did not show the existence of any tool more strongly focused on the specific needs of the students. In this context, our work will aim at the creation of an application that has a good usability and that has as main functionality to assist students in the choice of disciplines so that graduation can take place as soon as possible. The application was created using the Ionic Framework, which is a framework for creating hybrid applications, which can, with the same source code, create applications for various platforms. The application documentation was created using a tool called Jekyll, which creates static web pages using markdown. To solve the problem of timetabling, a technique known as constraint programming is used, which allows us to model several constraints, such as time, period and suggestion of possible matters.*

## 1. Objetivos específicos

Analisar, desenvolver e testar o aplicativo Tantum. a priori composto por:

- Algoritmo de montagem de grade de horários autônoma;
- Consulta aos horários das aulas do semestre;
- Consulta às estatísticas de conclusão do curso e próximos semestres de acordo com previsão realizada pelo algoritmo.



Escrever a documentação em arquivos .md, e apresentar através da ferramenta Jekyll, composta por:

- Requisitos funcionais;
- Requisitos não funcionais e
- Protótipos de telas.

**Etapa 1** - Fundamentação teórica: O início do projeto é dado pela pesquisa e análise dos conceitos básicos que nortearão o trabalho, de forma a conceder o arcabouço de conhecimento para os autores. A revisão foi focada em *Timetabling*, *Constraint programming*, Ionic, Análise de requisitos, Jekyll e Usabilidade.

**Etapa 2** - Revisão de trabalhos correlatos: O foco desta etapa é realizar uma pesquisa sobre trabalhos correlatos e similares de forma a compreender o estado da arte e o que já foi desenvolvido nesta área. A revisão levou em consideração algumas das palavras chave do trabalho, as quais foram:

- Grade de horários;
- Timetabling;
- Constraint programming;

**Etapa 3** - Análise, implementação e teste: O foco desta etapa é a realização de três tarefas principais e suas respectivas subtarefas:

1. **Análise:** O objetivo da tarefa de análise é a produção de protótipos de tela, os quais auxiliarão na elucidação dos requisitos (funcionais e não funcionais) gerando como artefato de saída a documentação do projeto. Para alcançar tais objetivos, a tarefa foi dividida em duas fases:
  - (a) Prototipação: Nesta fase haverá a concepção dos protótipos de tela do projeto, sendo estes gerados e logo depois validados com possíveis usuários e designers de modo a atestar a usabilidade e facilidade do aplicativo. Após uma rodada de validação, os protótipos serão alterados, aplicando as alterações julgadas coerentes e depois passando novamente pela rodada de avaliação, até que se alcance um ponto ótimo de usabilidade e facilidade de uso do projeto.
  - (b) Documentação de requisitos: A partir dos protótipos concebidos na fase anterior e entrevistas informais com os usuários serão elucidados os requisitos do aplicativo. A documentação será escrita em arquivos de texto (formato .md) que serão armazenados em nuvem e submetidos a controle de versão através do Github [Dabbish et al. 2012]. Para visualização da documentação de forma mais clara será utilizada a ferramenta Jekyll (<https://jekyllrb.com/>), a qual gera páginas Web a partir dos arquivos .md e um *template* de montagem de páginas. Esse conjunto de requisitos será utilizado na etapa de implementação de forma a guiar o desenvolvimento das funcionalidades do projeto e na elaboração do manual de utilização.
2. **Implementação:** O objetivo da tarefa de implementação é a produção do artefato principal do projeto: o aplicativo. O desenvolvimento será baseado a partir dos requisitos e protótipos concebidos na etapa 1, este também interferindo nos requisitos definidos, uma vez que alguns problemas e abordagens são encontradas somente na fase de efetiva codificação das regras já apontadas. Para alcançar tais objetivos, a etapa foi dividida em três fases:

- (a) Desenvolvimento do algoritmo de montagem de grade de horários: Nesta fase será desenvolvida a parte mais complexa do projeto: O algoritmo de montagem de grade de horário. Algoritmo esse que se baseará nas matérias que o aluno já cursou para sugerir as próximas disciplinas que o mesmo deve cursar no próximo semestre, respeitando o número máximo de créditos do curso e as matérias que ainda lhe restam cursar. Essa fase ocorrerá em paralelo com a concepção e avaliação cíclica dos protótipos, já que o desenvolvimento do algoritmo é independente da escolha de layout do aplicativo e demais decisões tomadas na fase de prototipação.
  - (b) Desenvolvimento do servidor *rest*: Para não sobrecarregar o aplicativo e evitar que o mesmo possua um elevado consumo de recursos e bateria do aparelho, a equipe decidiu por concentrar todo o processamento em um servidor remoto, deixando o *app* somente para visualização e solicitação de informações para o servidor. Nesta fase ocorrerá o desenvolvimento do servidor *rest* que conterá o algoritmo e fará o gerenciamento de carga de acessos.
  - (c) Desenvolvimento do aplicativo: O objetivo desta fase é a implementação do *app* a partir dos requisitos e protótipos concebidos na etapa 1. Para alcançar esse objetivo será utilizado o *framework* de desenvolvimento Ionic, o qual gera aplicativos para os dois sistemas operacionais para celulares mais utilizados atualmente (Android e iOS) a partir de código escrito em linguagem Javascript.
3. **Teste:** A partir do aplicativo desenvolvido, serão realizados testes exploratórios e dedicados a contextos controlados para atestar a qualidade e confiabilidade do artefato.

**Etapa 4 - Estudo de usabilidade:** O estudo de usabilidade será realizado em todas as etapas e tarefas anteriores do projeto, de forma que o conceito de boa usabilidade do aplicativo seja construído antes mesmo deste ser implementado. Para alcançar tal objetivo a etapa foi dividida em duas fases:

1. **Validação na concepção dos protótipos:** Durante a concepção dos protótipos serão chamados possíveis usuários do sistema de forma a atestar sua facilidade de uso e entendimento dos termos que foram utilizados no aplicativo através de testes de usabilidade informais e comentários sobre suas opiniões. Essas avaliações serão cíclicas, até que se alcance um ponto ótimo de usabilidade e facilidade de uso do aplicativo.
2. **Validação após o término da implementação:** Após a implementação, um outro grupo de possíveis usuários será chamado para utilizar diretamente o *app* em um dispositivo móvel, uma vez que este já se encontrará implementado. A partir da aplicação de testes de usabilidade moderados pelos próprios autores, será oferecido um roteiro de teste e serão coletadas informações através do questionário SUS (*System Usability Scale*), de forma a atestar formalmente a qualidade da usabilidade e satisfação do usuário.

Estado da arte

Observamos que diversos trabalhos acadêmicos tratam da alocação de horários para disciplinas de um determinado curso a partir dos horários em que os professores ministrantes estão disponíveis, exemplos são [Freitas et al. 2007], [Vieira and Macedo 2011],

[Júnior et al. 2000], [da Silva and da Silva 2010], [Ciscon and Alvarenga 2005]. Outros trabalhos focam na alocação de salas para as aulas, por exemplo [Souza et al. 2002]. A partir de pesquisas em repositórios de trabalhos acadêmicos e artigos científicos - principalmente Google Scholar e IEEE - utilizando as palavras-chave Grade de horários e Timetabling.

Por esse motivo, resolvemos utilizar como estado da arte os sistemas que hoje auxiliam os graduandos da UFSC a montarem sua grade de horários semestral. A seguir detalharemos as três principais ferramentas normalmente utilizadas para a realização da tarefa:

**Ferramenta 1:** Matrufsc2 (<https://matrufsc2.appspot.com/>)

O Matrufsc 2 é o sistema mais utilizado pelo alunos para realizar a simulação de sua grade de horários devido a sua interface amigável, a funcionalidade de salvar seus horários e garantir que o processo possa ser concluído posteriormente. Isso sem citar a grande quantidade de disciplinas que podem ser selecionadas, essas mantendo seus horários e professores ministrantes atualizados a cada troca de semestre.

**Ferramenta 2:** SisAcad (<https://admin.inf.ufsc.br/>)

O SisAcad (Sistema Acadêmico) foi concebido a partir do interesse de disponibilizar aos graduandos uma forma simplificada de apresentar as principais informações para nortear e situar o estudante sobre o grau de completude do curso. Ele está disponível somente para alunos dos cursos de Ciências da Computação e Sistemas de Informação e não foram encontradas informações sobre os criadores bem como a data de lançamento da ferramenta.

**Ferramenta 3:** CAGR (<http://cagr.sistemas.ufsc.br/>)

O CAGR (Sistema Acadêmico de Graduação) é o sistema responsável por gerenciar todas as informações do aluno no que se refere a administração escolar. Alguns exemplos de funcionalidades são:

- Calendários acadêmicos;
- Currículos dos cursos;
- Cadastro de turmas;
- Cadastro do Aluno;
- Espelho de matrícula;
- Geração de atestado de matrícula, Histórico Escolar e controle curricular;
- Solicitação de matrícula a cada início de semestre letivo.

Sobre a funcionalidade de solicitação de matrícula, o CAGR conta com a listagem de disciplinas sempre atualizadas, porém peca na questão de usabilidade e design, com informações confusas e sem nenhum auxílio para a resolução de conflitos de horários de matérias. Outro ponto importante é o fato do CAGR ser o único meio de oficializar a matrícula do aluno junto a UFSC, já que nenhuma ferramenta o faz por si só.

De forma a comparar as características específicas e guiar o desenvolvimento do projeto proposto, foi concebida uma tabela com as funcionalidades principais de cada sistema apresentado e o sistema proposto:

De acordo com o apresentado nas descrições dos sistemas e na tabela abaixo, chegamos à conclusão que nenhuma das aplicações oferecidas atualmente na UFSC atende as

Table 1. Comparação entre sistemas

| Característica                           | Tantum | Matrufsc | Sisacad | CAGR |
|--|--------|----------|---------|------|
| Resolução de conflitos                   | X      | X        |         |      |
| Escolha de disciplinas                   | X      | X        |         | X    |
| Montagem autônoma                        | X      |          | X       |      |
| Escolha de turno de estudo               | X      |          |         |      |
| Apresentação de disciplinas equivalentes | X      |          |         |      |
| Avaliação do histórico do aluno          |        |          | X       |      |
| Avaliação de desempenho do aluno         |        |          | X       |      |
| Provável semestre de formatura           | X      |          | X       | X    |
| Matrícula definitiva                     |        |          |         | X    |
| Múltiplos planos de matrícula            |        | X        |         | X    |

necessidades que enfrentamos a cada nova etapa de matrícula, assim gerando a motivação para a execução do presente projeto.

#### Análise

O padrão de escrita da documentação utilizado neste projeto foi baseado no padrão adotado no Laboratório Bridge, local de trabalho dos autores deste trabalho. No laboratório, foi definido um padrão de documentação que tem como objetivo a padronização dos documentos gerados pelos diferentes analistas do projeto, além de guiar a escrita de documentações dos diferentes módulos.

### 1.1. A documentação do TCC

A documentação do aplicativo foi baseada no padrão de documentação utilizado no laboratório Bridge.

A documentação foi organizada de acordo com o contexto de cada documento, assumindo a seguinte estrutura ao final do processo:

- Geral: Composto por componentes e telas que são utilizados mais de uma vez no aplicativo;
- Login: Composto pelas telas que formam o Login do usuário bem como a tela de “Sobre” do sistema;
- Menu principal: Composto pelas telas que podem ser acessadas através do Menu principal, englobando os módulos do aplicativo.

Cada documento, por sua vez, possui as seguintes seções:

- Campos: Seção responsável pela elucidação de quais campos devem estar contidos na tela, bem como seus nomes. A ordem dos campos nessa seção deve ser a mesma da tela.
- Regras: Seção responsável por enumerar as regras que a tela e os campos deverão seguir. Por padrão, a primeira regra apresenta o mapeamento dos campos e como estes devem vir preenchidos quando a tela for aberta;

- Internacionalização: Seção responsável por apresentar como cada label da tela deverá se comportar em cada língua;
- Dicionário de dados: Seção responsável por apresentar a composição de informações de cada campo, tamanho mínimo e máximo de entradas, seus tipos e configurações;
- Protótipos de telas: Seção responsável por apresentar os protótipos de tela.

## 2. Desenvolvimento

O desenvolvimento deste trabalho consistiu em: criação do aplicativo e uma API REST, a qual teria a geração da grade de horários, que enviaria pro aplicativo.

O aplicativo foi desenvolvido utilizando o *Ionic Framework*. Ele cria *Portable Web Apps* (PWA), que é um híbrido de uma pagina web convencional e um aplicativo nativo, de modo que o resultado é algo que se parece com um aplicativo nativo, mas é uma pagina web. Assim o desenvolvimento e teste de um PWA é muito rápido e fácil.

A API REST foi feita em Java, utilizando o *Spring Boot*. Sua utilização torna o desenvolvimento da aplicação mais simples, pois com servidores embarcados não existe a necessidade do *deploy* de um arquivo WAR.

### 2.1. Ionic

#### 2.1.1. Pré-requisitos

Para desenvolver com Ionic, é necessário o *nodejs*, pois ele necessita do *node package manager* (npm).

Também é necessário o *Apache Cordova*, que é a base no qual o Ionic é desenvolvido, e por fim o próprio Ionic.

Para instalar os mesmos, deve-se executar:

```
npm install -g cordova ionic
```

Isso irá instalar a versão mais atual do Cordova e do Ionic.

#### 2.1.2. Estrutura do aplicativo

A estrutura de um projeto Ionic consiste nos seguintes componentes:

- **hooks:** Consiste de *scripts* que devem ser executados quando uma tarefa do Cordova é executada;
- **node\_modules:** Onde ficam localizados os pacotes npm descritos no *package.json*;
- **plataform:** Código gerado para a plataforma específica, como Android e iOS;
- **plugins:** Contém todos os plugins que foram adicionados ao projeto;
- **resources:** Arquivos extras do projeto, como imagens e ícones;
- **scss:** Consiste de arquivos de estilo no formato *scss*;
- **src:** Código fonte do aplicativo, contém os arquivos HTML e TS;
- **www:** Código que foi gerado após o processo de *build*.

### 2.1.3. Arquivo inicial do aplicativo

`src/index.html` é o arquivo inicial do aplicativo, como um arquivo `index.html` em um projeto angular. Não será necessário alterar nada neste arquivo, pois a principal função dele é incluir os scripts e incluir a tag `<ion-app>`, que é o componente raiz em uma aplicação ionic. Todos os outros componentes estarão dentro dele.

### 2.1.4. Componentes

Os componentes básicos usados neste projeto, que são fornecidos pelo próprio Ionic são:

- *Component*: São elementos gráficos que podem ser reutilizados
- *Page*: São as páginas do aplicativo, contém diversos componentes
- *Pipe*: Componente que altera a informação
- *Provider*: Tipo de componente que fornece acesso aos dados

### 2.1.5. Páginas

O Ionic já possui uma página inicial, que é a *app*. Ela é referenciada dentro do `index.html` pela tag `<ion-app>`. Toda página tem uma tag associada. Neste trabalho, para atender as funcionalidades que queremos, foram criadas sete páginas, que são:

1. Definir critérios;
2. Estatísticas;
3. Grade de horário;
4. Login;
5. Principal;
6. Resultado;
7. Sobre.

Quando uma página é gerada, são criados 4 arquivos, como por exemplo a página de login:

1. *login.html*: Aonde ficam os componentes gráficos da página;
2. *login.module.ts*: Configura injeção de dependências entre outros;
3. *login.scss*: Arquivo que contém os estilos da página;
4. *login.ts*: Arquivo *typescript* que contém a lógica da página;

### 2.1.6. Deploy do aplicativo

Durante o desenvolvimento deste trabalho os testes do aplicativo foram feitos pelo browser, com o comando `ionic serve`. Porém se tem a necessidade de realizar os testes em um dispositivo mobile, para que seja verificado como realmente a aplicação se comporta e ver as situações de uso na realidade. Além disso, muitos plugins nativos só funcionam quando executados por um dispositivo de verdade, como por exemplo *Deeplinks*, que permite fazer links entre sites e aplicativos. Ele faz isso definindo uma URL para o aplicativo, a qual pode ser redirecionado por um site ou outro aplicativo.

## 2.2. Spring Boot

### 2.2.1. Instalação

Para iniciar um projeto Spring Boot, utilizou-se o gerador de projetos online, o Spring Initializr, disponível em [Spring 2018].

Esse gerador de projetos cria um projeto Maven básico, com uma classe já configurada como *Spring Boot Initializer*, sendo esta a classe *main* da aplicação responsável por iniciar o projeto. Por fim, basta executar a classe inicial, que neste trabalho é a *TantumApplication*, a qual é instanciada através de uma classe do *Spring* chamada *SpringApplication*.

### 2.2.2. Rotas

Para criarmos as rotas e definir quais métodos devem ser executados foi criado um Controller, que faz a ligação das chamadas HTTP para nossos métodos Java. No Spring são utilizadas anotações que fazem esse controle. Como por exemplo: *@RestController*, a qual define uma classe, como um Controller, e que métodos vão estar relacionados com chamadas HTTP.

Também foi utilizada neste Controller a anotação *@CrossOrigin*, que permite chamadas de origem diferentes. Caso contrário, por padrão é apenas permitido chamadas de origem local.

Para relacionar os métodos com os caminhos, utilizou-se a anotação *@RequestMapping*, a qual possui como principais parâmetros os *path* e o *method*. Esses parâmetros definem o caminho a ser utilizado e o método HTTP. Como por exemplo o login, que foi definido o caminho `"/v1/login"` utilizando o método HTTP GET.

Também há necessidade de se passar parâmetros nas chamadas. Para isso são utilizadas as anotações *@RequestParam*, *@PathVariable* e *@RequestBody*, sendo a última utilizada em métodos HTTP POST. Por exemplo, no login, ele recebe o token de autenticação para verificação do mesmo.

### 2.2.3. Rotas criadas

Neste trabalho foram criados as seguintes rotas:

- GET `/v1/login`: Recebe o token de autenticação e o validar;
- POST `/v1/calculate-semester`: Faz a sugestão das matérias;
- GET `/v1/schedule/{semester}`: Retorna as matérias do semestre especificado;
- GET `/v1/subjects`: Retorna todas as matérias;
- GET `/v1/statistics`: Retorna as estatísticas do aluno.

### 2.2.4. Algoritmo para o cálculo do semestre

O algoritmo que realiza o cálculo do próximo semestre esta localizado no caminho `/v1/calculate-semester`, sendo utilizado o método *HTTP POST* e enviado no *body* da

chamada as restrições que foram definidas no aplicativo.

Para aplicar as restrições, foi utilizando a biblioteca *Choco solver*, as quais são aplicadas através da classe *Model*. Neste trabalho utilizamos o seguinte método desta classe: `addClauseTrue`, que adiciona uma restrição de modo que ela é cumprida caso o resultado da sua condição seja *true*. Esse método recebe como parâmetro uma variável *booleana*, que é criada utilizando o método `boolVar` também da classe *Model*. Essa variável *booleana* criada possui dois valores como parâmetros, um nome, o qual será identificada essa restrição, e um método que será aplicada a restrição em si.

Figure 1. Restrições

```

model.addClauseTrue(model.boolVar(credit_max,
    this.validateClassHourLoad(constraints, currentSubjects, subject)));
model.addClauseTrue(model.boolVar(times,
    this.validateTime(currentSubjects, subject)));
model.addClauseTrue(model.boolVar(period,
    this.validatePeriod(constraints, subject)));
model.addClauseTrue(model.boolVar(subjects_wanted,
    this.validateSubjectsWanted(constraints, subject)));
model.addClauseTrue(model.boolVar(subjects_not_wanted,
    this.validateSubjectsNotWanted(constraints, subject)));
model.addClauseTrue(model.boolVar(requisites,
    this.validateRequisites(subject, subjectsHistory)));

```

### 2.2.5. Deploy da API Rest

Neste trabalho o deploy da aplicação foi realizado por um executável *jar*. Para a criação desse arquivo foi adicionado no arquivo *pom.xml* o formato do pacote: `<packaging>jar</packaging>`.

O comando utilizado para a realização da compilação e geração do arquivo executável foi: `mvn clean install`. Esse comando junta todas as dependências que foram adicionadas no arquivo *pom.xml* e cria um arquivo *jar* único e executável, o que torna mais conveniente a execução e distribuição do mesmo.

Esse executável contém um servidor Tomcat, que ao ser executado ele inicia automaticamente e executa a classe *TantumApplication*, pois está marcada com a anotação `@SpringBootApplication`

## 3. Processo de teste

Durante o desenvolvimento da aplicação, foram realizados testes exploratórios, de modo a encontrar *bugs* simples que ocorrem na utilização do usuário.

Para guiar esses testes foi criada uma série de casos de teste, os quais foram elaborados utilizando como base a experiência que os autores adquiriram no Laboratório Bridge. Cada caso é composto por:

- **Grupo:** O módulo (ou a tala) o que o caso de teste tem a finalidade de testar;
- **Item:** Título do caso de teste;
- **Regra:** Regra (ou conjunto de regras) da documentação que o caso de teste abrange;
- **Caso de teste:** Representa os passos de teste;



- **Comportamento esperado:** O que é esperado que aconteça após a execução dos passos de teste;
- **Situação:** Indicação se o comportamento esperado foi alcançado.

#### 4. Testes de integração

Desde o início do projeto, o intuito dos autores foi que ocorresse uma integração com o sistema de autenticação centralizado da SeTIC, para que a aplicação pudesse ser totalmente funcional e que fosse realmente útil na vida dos estudantes da UFSC. Porém, após várias tentativas infrutíferas, inclusive indo ao local e diversas solicitações de serviços que foram criadas por locais diferentes, foi tomada a decisão de não fazer a integração com os dados da UFSC para que o trabalho pudesse ser defendido no semestre corrente.

Uma vez que essa decisão foi tomada, foi elaborada uma série de casos de testes de integração para atestar o correto funcionamento do algoritmo que foi desenvolvido para a geração de horários.

##### 4.1. Testes do Algoritmo

O teste define as possíveis restrições definidas pelo usuário e qual o resultado esperado quando se utiliza tais restrições. Foi possível realizar esse teste pois estamos utilizando um conjunto pequeno de disciplinas com a intenção de teste.

Os testes foram criados utilizando a API de testes unitários *JUnit*, pois é a mais conhecida para a plataforma Java. Os testes foram definidos utilizando a estrutura *Give, When, Then* que permite modelar uma história de usuário mais facilmente, sendo que **dado** um pré requisito, **quando** ocorre uma ação **então** eu tenho os seguintes resultados.

Os testes foram feitos utilizando os métodos *assert* do *JUnit*, o qual permite de modo claro saber o que é esperado por um determinado teste.

Foi criado um teste unitário para cada restrição, e com ela definida testamos as disciplinas sugeridas, como por exemplo a restrição de período de estudo.

Com esses testes podemos mostrar que o algoritmo cumpre o que foi pretendido e que as restrições definidas são sempre cumpridas.

##### 4.2. Teste da API Rest

Para realizar o teste da API Rest foi utilizada uma classe do *Spring* chamada *MockMvc*, a qual consegue-se realizar as chamadas *HTTP* de modo que o teste se comporta como se fosse realizada uma chamada real.

Note que esse teste não é executado com o *JUnit*, mas sim com o *SpringRunner*, o que permite a injeção de dependências. Além disso, é utilizada a anotação *@SpringBootTest*, que permite definir o ambiente do Spring que irá rodar, de modo que nossas dependências sejam encontradas. Para isso é passada como parâmetro a classe principal do servidor. Também a anotação *@AutoConfigureMockMvc*, a qual é a responsável pela injeção do *MockMvc*.

Além de serem testadas todas as chamadas, também foram testados casos de erro, como por exemplo um parâmetro em falta.

## 5. Testes de usabilidade

### 5.1. Realização

Os testes foram realizados no dia onze de março de 2018, contando com a participação de 8 participantes.

Os participantes foram selecionados de acordo com sua faixa etária, curso realizado e seu nível de escolaridade, o qual variou entre superior incompleto e superior completo. Todos os participantes realizam / realizaram seus cursos na UFSC.

O teste era composto por três documentos os quais estão contidos nos anexos deste trabalho:

1. Termo de consentimento e esclarecimento de participação;
2. Roteiro de teste;
3. Questionário SUS (*System Usability Scale*).

Para a realização dos testes, os participantes leram e preencheram o Termo de consentimento e esclarecimento de participação, de forma a atestar sua efetiva participação no teste. Logo após, receberam os passos de teste, os quais deveriam ser realizados sem a interferência direta dos avaliadores, que observaram os movimentos dos participantes que, por sua vez, falavam em voz alta seus pensamentos e ações realizadas no aplicativo. Durante a execução, as propostas de melhorias e dificuldades também eram citadas pelos participantes.

### 5.2. Resultados

O tempo médio de interação dos usuários com o sistema foi de oito minutos, considerando que estes comentavam pontos e já indicavam pontos de melhoria. Porém, ao levar em consideração o tempo que os usuários levaram para acessar as funções e, segundo os próprios participantes, a função mais utilizada seria os próximos horários de aula, a qual pode ser diretamente acessada sem a necessidade de um toque sequer na tela.

Também é importante salientar que os participantes não levaram muito tempo para realizar os passos da geração de horários, considerada por todos e pelos autores como a função mais extensa do aplicativo.

A partir desses raciocínios, a opinião dos autores é que o tempo de interação com o sistema está de acordo com sua proposta de uso.

O critério utilizado para avaliar os resultados dos questionários aplicados é o proposto por [Bangor et al. 2009], o qual separa a satisfação do usuário em sete níveis, os quais são:

- Pior imaginável: Menos de 12.5 pontos;
- Horrível: Entre 12.6 e 20.3 pontos;
- Pobre: Entre 20.4 e 35.7 pontos;
- Ok: Entre 35.8 e 50.7 pontos;
- Bom: Entre 50.8 e 71.4 pontos;
- Excelente: Entre 71.5 e 85.5 pontos;
- Melhor imaginável: Mais de 85.6 pontos.

A média dos resultados do questionário SUS foi de 92,5 pontos, o que de acordo com os critérios citados acima coloca o aplicativo com a melhor usabilidade possível para o contexto que se propõe.

A nota mais baixa foi de 40 pontos, esta sendo atribuída por um participante. Entendemos que essa nota foi fruto de uma má interpretação do teste, visto que o mesmo comentou várias vezes que as matérias não existiam mesmo sendo informado diversas vezes que os dados para o teste eram fictícios, os quais eram utilizados somente para exemplificar a usabilidade do aplicativo.

A maior nota foi de 95 pontos, esta sendo atribuída por cinco participantes. Entendemos que essa nota foi fruto de todo o trabalho de estudo de usabilidade e várias iterações até conceber o resultado final. Logo após a completude dos testes, estes usuários elogiaram a facilidade de uso do sistema e seu o provável uso cotidiano.

O software

## 6. Resultado final

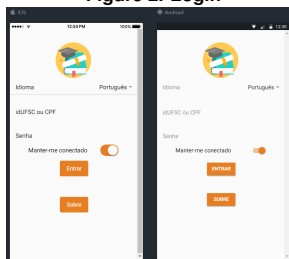
Sobre o *app* em si, serão apresentados a seguir os protótipos de telas concebidos, bem como suas funcionalidades e fluxo de navegação.

### 6.1. Login

Para ter acesso ao aplicativo, o graduando deve realizar seu login utilizando seu idUFSC e sua correspondente senha de acesso. Há a possibilidade de se manter logado, assim pulando a fase de login sempre que o aluno entrar no sistema.

Há também a opção de ler algumas informações referentes ao sistema acionando o botão Sobre. Uma vez logado, o usuário será redirecionado para o Menu principal.

Figure 2. Login



### 6.2. Menu principal

Uma vez no Menu principal, o usuário pode escolher entre as três funcionalidades ofertadas:

### 6.3. Grade de horários

Uma vez selecionada a opção Grade de horários, o usuário será redirecionado para a tela de visualização da grade de horários do semestre.

O objetivo desta funcionalidade é a observação dos horários e alocações das aulas do dia. Para visualizar os horários de outros dias, basta arrastar a tela na direção do dia esperado:

- Da direita para esquerda: Avança os dias;
- Da esquerda para direita: Retrocede os dias.

A aba "Semanal" dá acesso aos horários de aula exibindo-os em formato semanal.

#### 6.4. Geração de horários

Uma vez selecionada a opção Geração de grade de horários, o usuário será redirecionado para a funcionalidade de geração de grades de horários.

Nesta tela o usuário deve decidir como vai gerar sua grade de horários, de forma a minimizar o tempo de formatura ou a partir de critérios definidos.

Caso a opção selecionada for Gerar grade de horários para minimizar tempo de formatura, redirecionar o usuário para a tela de resultado e, caso haja uma sobrecarga no servidor, para a tela de notificação de conclusão.

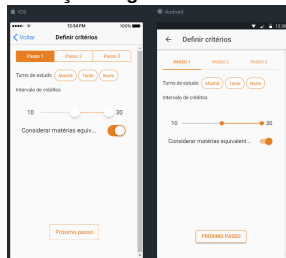
Caso a opção selecionada for Definir critérios, redirecionar o usuário para a tela de definição de critérios.

Uma vez acionada a ação Voltar, o aluno deve ser redirecionado para o menu principal do sistema.

O objetivo dessa fase é definir os critérios que nortearão o algoritmo de geração de grades de horário. Os critérios são:

- Turno de estudo;
- Número máximo;
- Matérias que quero cursar nesse semestre.

Figure 3. Geração de grade de horários - Passo 1



Ao acionar a opção Gerar grade de horários, redirecionar o usuário para a tela de resultado.

## 6.5. Resultado

A partir dos critérios utilizados, o resultado é exibido da forma como representado no protótipo. O usuário pode navegar entre as opções de geração, excluindo as disciplinas que não lhe agradaram e visualizando o horário diário bem como a localização das salas de aula.

Caso o usuário desejar, ele pode redefinir os critérios de geração a partir da ação de mesmo nome.

Uma vez selecionada a opção Estatísticas, o usuário será redirecionado para a funcionalidade de Estatísticas de conclusão do curso.

## 6.6. Estatísticas

Na tela Estatísticas devem ser apresentadas as estatísticas de conclusão do curso do graduando.

- Gráfico 1: Exibe a porcentagem de semestres cursados pelo aluno em um gráfico de pizza, seguida pela quantidade de semestres cursados e restantes;
- Gráfico 2: Exibe a porcentagem de disciplinas cursadas pelo aluno em um gráfico de pizza, seguida pelo total de matérias do curso e porcentagem de completude.

Ao acionar a ação Formatura e próximos semestres devem ser exibidos as matérias a serem cursadas nos próximos semestres de acordo com sucessivas gerações do algoritmo.

### Conclusão

Ao finalizar este trabalho podemos aprender mais sobre a importância do processo completo de desenvolvimento de software, bem como da relevância de cada uma das partes responsáveis por cada artefato dentro do desenvolvimento de um aplicativo. Um ponto importante a ser destacado foi a concepção conjunta do aplicativo, onde ambos os autores participaram ativamente, desde a elaboração dos protótipos, discussões sobre o design e alterações dos mesmos até a completude da documentação e desenvolvimento completo da aplicação.

Também é notória a preocupação com o design da aplicação, item que ambos sempre consideramos muito importante, porém esta foi a primeira vez que este foi testado e validado por usuários através de testes de usabilidade. Teste este que foi importante para o melhor entendimento do usuário final sobre a aplicação e sobre suas opiniões sobre a mesma.

## 7. Trabalhos futuros

Como trabalhos futuros, os autores destacam:

- **Integração com a SeTIC:** Conforme comentado no trabalho, os autores tentaram por várias vezes contato com a superintendência, porém sem sucesso. Seria muito interessante a implementação dessa integração, visto que o algoritmo para a montagem das grades de horários já está desenvolvido;
- **Correções de design:** Conforme comentado na seção de usabilidade, há alterações que podem ser realizadas no *app* para que este melhore ainda mais em sua usabilidade;

- **Disponibilização do app:** Uma vez implementadas as alterações colocadas nessa seção, seria interessante disponibilizar a aplicação para os alunos em geral.

## References

- Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123.
- Ciscon, L. A. and Alvarenga, G. (2005). O problema de geração de horários: um foco na eliminação de janelas e aulas isoladas. In *XXXVII Brazilian Symposium of Operational Research*.
- da Silva, D. J. and da Silva, G. C. (2010). Heurísticas baseadas no algoritmo de coloração de grafos para o problema de alocação de salas em uma instituição de ensino superior. *Anais do XLII SBPO*, pages 2839–2849.
- Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012). Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM.
- Freitas, C. C., Guimarães, P. R., Neto, M. C., and Barboza, F. (2007). Uma ferramenta baseada em algoritmos genéticos para a geração de tabela de horário escolar. *SÉTIMA ESCOLA REGIONAL DE COMPUTAÇÃO Bahia-Sergipe. Vitória da Conquista:[sn]*.
- Júnior, B., de Oliveira, O., et al. (2000). Otimização de horários em instituições de ensino superior através de algoritmos genéticos.
- Souza, M. J. F., Costa, F., and Guimarães, I. (2002). Um algoritmo evolutivo híbrido para o problema de programação de horários em escolas. *Computer*.
- Spring (2018). Spring Initializr.
- Vieira, F. and Macedo, H. (2011). Sistema de alocação de horários de cursos universitários: um estudo de caso no departamento de computação da universidade federal de sergipe. *Scientia Plena*, 7(3).