



Licenciatura em Engenharia Informática e Computadores
Projeto e Seminário - Semestre de Verão 2018/2019

Relatório de Progresso

**Monitorização de qualidade de serviço em
comunicações ferroviárias**

João Vaz, Nº 41920
41920@alunos.isel.pt
961833140

Luis Vasconcelos, Nº41556
41556@alunos.isel.pt
910365046

Orientado por:

Nuno Cota, ncota@deetc.isel.pt
Ana Beire, anaritabeire@solvit.pt

Índice

1. Introdução	2
1.1 Enquadramento	2
1.2 Objectivos/Funcionalidades	2
1.3 Organização do Documento	2
2. Requisitos do sistema	3
3. Solução Proposta	4
3.1 Arquitetura do sistema	4
4. Implementação	6
4.1 Base de Dados	6
4.2 API	7
4.3 Cliente	8
5. Conclusão	11
6. Referências	12

1. Introdução

1.1 Enquadramento

A monitorização permanente da qualidade de serviço em redes de comunicações móveis ferroviárias é fundamental para a deteção de alterações que coloquem em causa o serviço de comunicações de apoio à exploração. Nesse sentido, foram desenvolvidas unidades embarcadas em comboios (sondas) que realizam de forma autónoma e permanente a monitorização, ativa e passiva, aos sistemas de comunicações móveis ferroviários.

O conjunto de sondas embarcadas, designadas por OBU (*On Board Unit*), efetuam os testes de forma autónoma, sendo a sua administração e monitorização realizada de forma centralizada por um sistema de informação. Este sistema, além de efetuar a administração das sondas, terá como função igualmente servir de repositório de informação e permitir disponibilizar as interfaces necessárias à análise dos dados recolhidos.

1.2 Objectivos

O projeto consiste no desenvolvimento de um sistema de informação que permita realizar a administração de sondas e análise da informação proveniente das sondas embarcadas nos comboios. Este sistema será composto por repositório de dados e aplicação Web que permita a análise e o tratamento de informação proveniente das sondas de forma a permitir ao utilizador uma fácil e objetiva observação dos dados e administração do sistema.

1.3 Organização do Documento

Este documento encontra-se dividido em seis capítulos. O primeiro capítulo enquadra o problema, descreve as motivações e apresenta os objectivos do projecto. No segundo e terceiro capítulo, apresenta-se a abordagem ao presente projecto assim como a descrição da arquitectura proposta. No quarto capítulo, descreve-se a implementação de cada componente do projecto. No quinto capítulo apresenta-se o planeamento de tarefas já realizadas e a realizar.

2. Requisitos do sistema

Em termos de requisitos funcionais, foi definido um conjunto de aspetos funcionais a que o sistema deverá responder, os quais serão descritos de seguida.

Como requisitos obrigatórios temos:

- Aplicação Web Single-Page com integração de mapas
- Gestão de OBUs (*On Board Unit*):
 - Gestão de grupos
 - Gestão de hardware
 - Gestão de configurações
 - Gestão de planos de teste
 - Gestão de ligações
- Apresentação dos dados:
 - Mapas
 - Gráficos
 - Logs
- Gestão de utilizadores
 - Registo de utilizadores
 - Autenticação de utilizadores
 - Gestão de perfis de utilizadores
- Pesquisa de dados por nome, data, local, etc.

Como requisitos opcionais temos:

- Monitorização de sondas em tempo real

3. Solução Proposta

A solução proposta passa pelo desenvolvimento de uma Web API para acesso aos dados reportados pelo conjunto de sondas existente e apresentação de resultados de medidas e testes aos utilizadores, bem como possibilitar a administração das mesmas. A informação proveniente das sondas é já atualmente armazenada numa base de dados.

A informação resultante dos testes deverá ser apresentada de forma geo-referenciada, sobre mapas e através de gráficos. Para disponibilização do serviço de mapas será utilizada a API Leaflet.

De forma a que o cliente tenha acesso aos dados e informações terá de estar registado no sistema de informação, pelo que a solução deverá igualmente incluir toda a componente de gestão de utilizadores e perfis.

3.1 Arquitectura do Sistema

Para a realização desta aplicação é necessário desenvolver os componentes da arquitetura, apresentados na Figura 1.

- - **Componente Cliente**

A componente cliente é acedida pelo utilizador através do *browser* e vai ser responsável pela apresentação e gestão dos dados. Vai ser desenvolvida utilizando a linguagem de programação JavaScript (Node.js), a *framework* AngularJS e a ferramenta de desenvolvimento Visual Studio Code.

Esta componente vai disponibilizar a informação já devidamente tratada usando a informação oferecida pela componente servidora.

- - **Componente Servidora**

A componente servidora constitui o meio utilizado pela aplicação cliente para obter, publicar e atualizar dados. Estas operações são realizadas através da interação com a base de dados relacional.

Quanto à Web API vai ser desenvolvida utilizando a linguagem de programação kotlin [5]. A utilização da *framework* Spring MVC [6] permite simplificar e auxiliar o desenvolvimento da API.

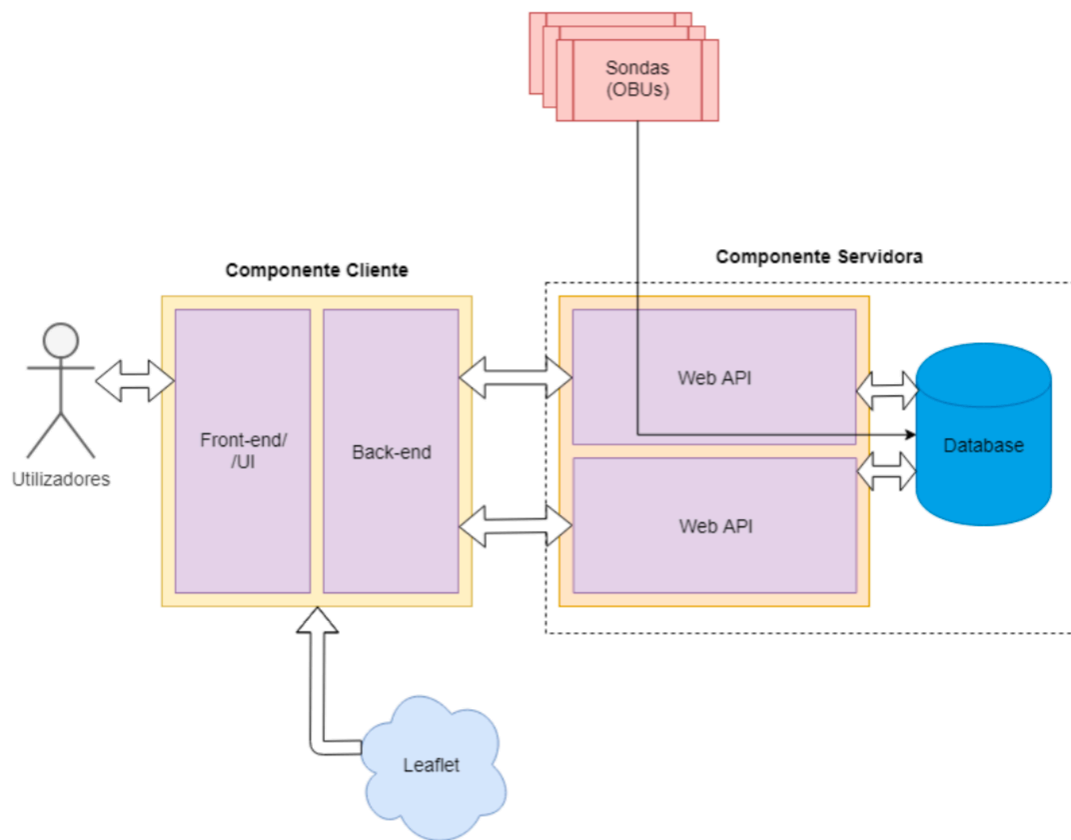


Figura 1 - Arquitectura do Sistema.

4. Implementação

4.1 Base de Dados

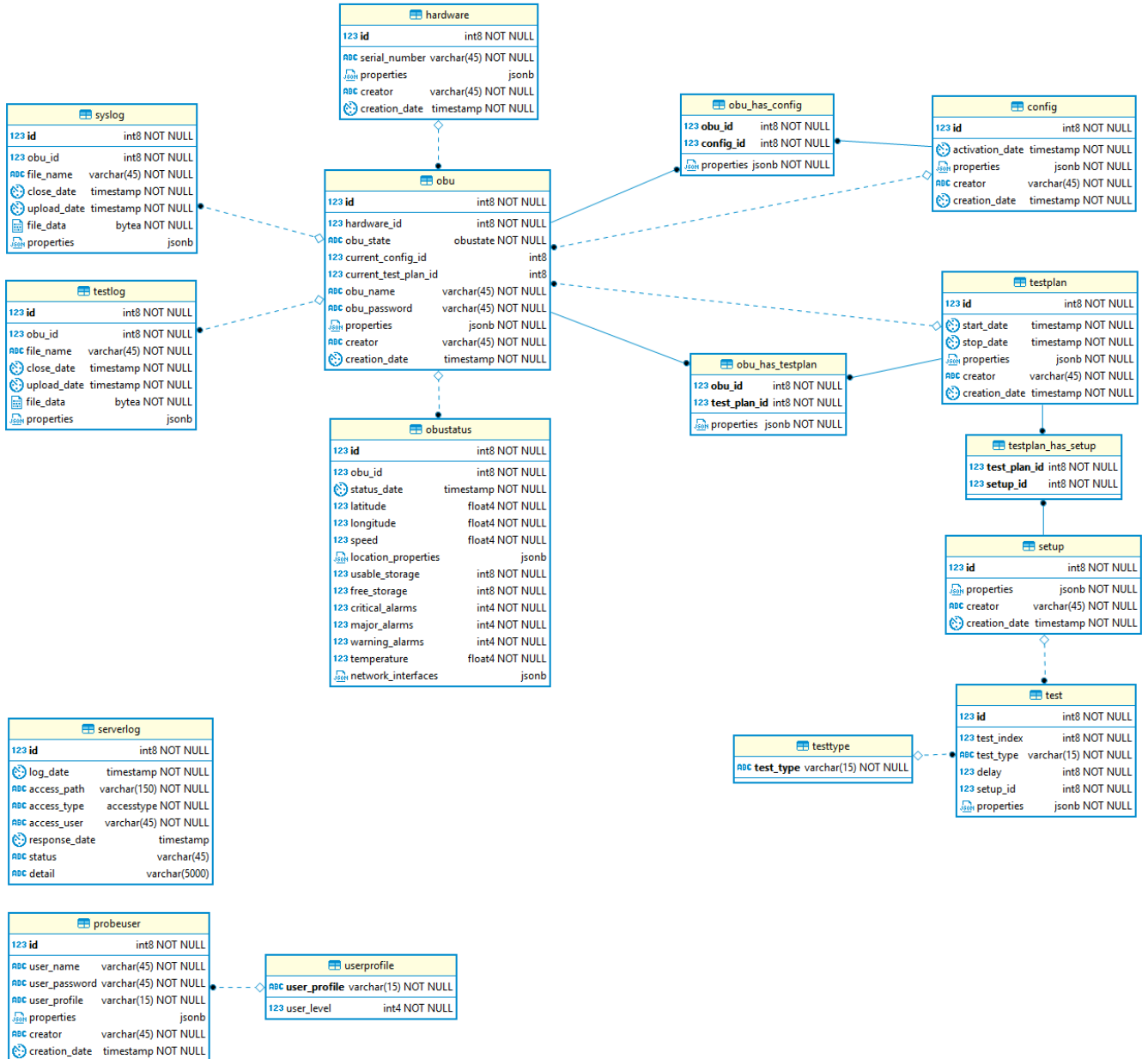


Figura 2 - Modelo Entidade-Associação.

A figura 2 apresenta o Modelo Entidade-Associação da base de dados desenvolvida.

A implementação da Base de Dados foi feita em Sql PostGre com base no Modelo Entidade-Associação já existente que nos foi fornecido posteriormente. Foi necessária uma análise dos dados nos quais iríamos trabalhar com o objectivo de eventualmente acrescentar novos campos ou até mesmo tabelas.

Nesta fase do projecto trabalhamos maioritariamente com dados que envolvem a tabela ProbeUser e a tabela OBU.

A tabela ProbeUser representa um utilizador e toda a sua informação correspondente, desde o seu *username* ao seu criador passando pelo seu nível de utilizador que lhe permite o acesso ou não a informações mais específicas.

A tabela OBU representa uma sonda, e tem como objectivo principal guardar a informação referente ao seu estado, configuração atual e plano de testes atual. Todas as outras informações pertinentes relativamente a uma sonda encontram-se nas restantes tabelas associadas a esta.

4.2 API

Para a implementação da API foi feita uma análise da informação que necessitamos de disponibilizar para o exterior de acordo com as funcionalidades que vamos implementar e tendo em atenção que poderá não ser apenas o nosso *Cliente* a consumir essa informação.

A nossa API vai dispor de início de um sistema de versões de maneira a impedir que futuras alterações na API quebrem a compatibilidade com aplicações que estejam a usar as versões mais antigas. Para suportar esta implementação adicionamos ao URI da API o número da versão.

O URI Base é dado por: `http://{servidor:porto}/api/v1/`

De seguida apresenta-se uma tabela com os endereços específicos da API (*endpoints*) a disponibilizar:

Tabela 1 - Lista de *endpoints* disponibilizados pela API.

Método	Pedido HTTP	Descrição
GET	users	Retorna todos os utilizadores existentes.
GET	user/{param}	Retorna o utilizador referente ao param, podendo este ser id ou nome de utilizador.
PUT	user/{id}/suspend	Suspende o utilizador referente ao id.
POST	users	Cria um novo utilizador.
POST	login	Retorna o token para efectuar o login.
GET	obus	Retorna todas as OBU's existentes.
GET	obu/{id}	Retorna a OBU referente ao id.

4.3 Cliente

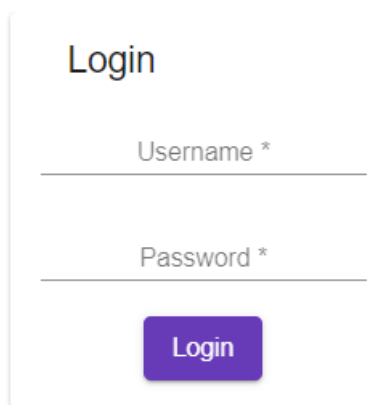
Este componente contém a interface gráfica apresentada ao utilizador.

Os dados apresentados na interface são obtidos através da comunicação com a componente servidora, mais propriamente a API.

A página inicial da aplicação (Figura 3) permite ao utilizador iniciar sessão com o seu nome de utilizador e palavra-passe correspondente. Qualquer tentativa de acesso a outra página sem início de sessão ou com sessão expirada resulta no a esta página. De seguida é redireccionado para a *Home Page* (Figura 4), onde, através do menu lateral (Figura 5), pode escolher uma das várias opções fornecidas com a finalidade de aceder a informações sobre utilizadores, obus, etc. Nas páginas de informações é apresentada uma lista com os detalhes mais importantes sobre, por exemplo, os utilizadores; detalhes como o nome e o seu nível (se é *Admin*, *SuperUser* ou *NormalUser*) (Figura 6).

Certas páginas como as páginas das listas de utilizadores e obus não podem ser acedidas por utilizadores cujo nível seja *NormalUser*, embora outras como a *Home Page* possa ser acedida por qualquer utilizador.

Futuramente os utilizadores suspensos não poderão aceder a qualquer página e serão redirecionados para uma página que dirá que estão suspensos, ou simplesmente no ato de início de sessão um *popup* indicará ao utilizador que está suspenso e que não pode iniciar sessão.



A login form with a white background and a thin grey border. At the top, the word "Login" is centered in a black sans-serif font. Below it are two input fields: "Username *" and "Password *", each with a horizontal line for text entry. At the bottom, there is a purple rectangular button with the word "Login" in white text.

Figura 3 – Menu de login.

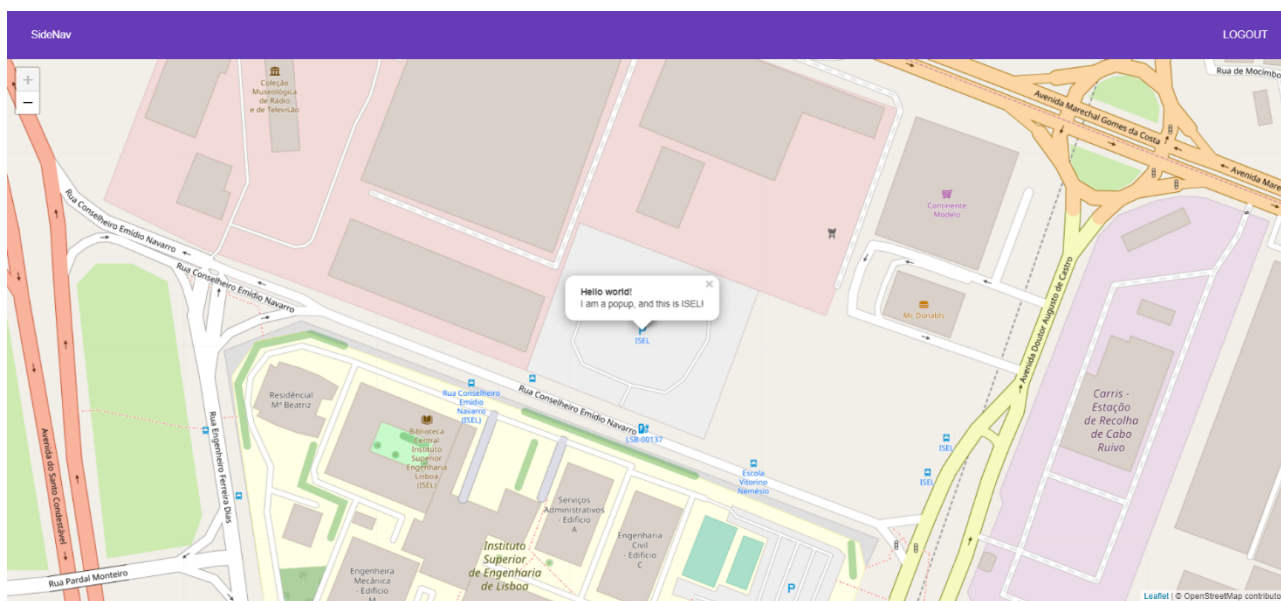


Figura 4 – Mapa da página principal.

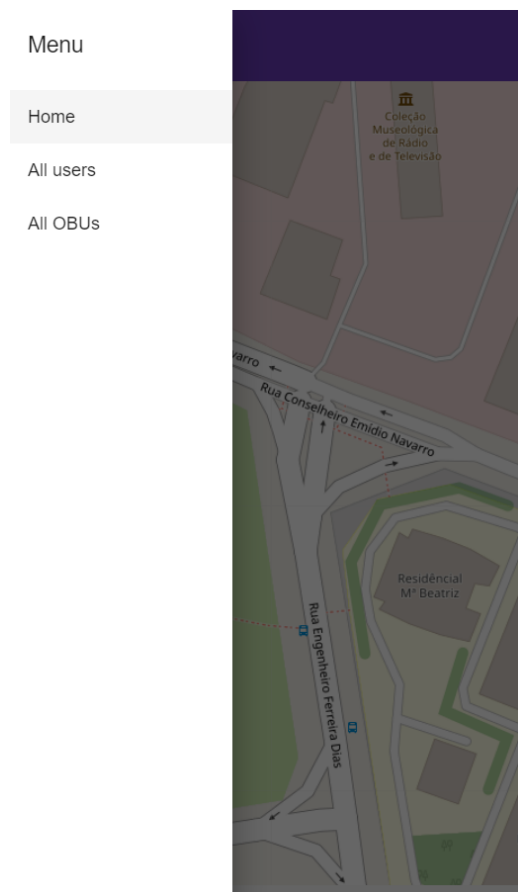


Figura 5 – Menu lateral.

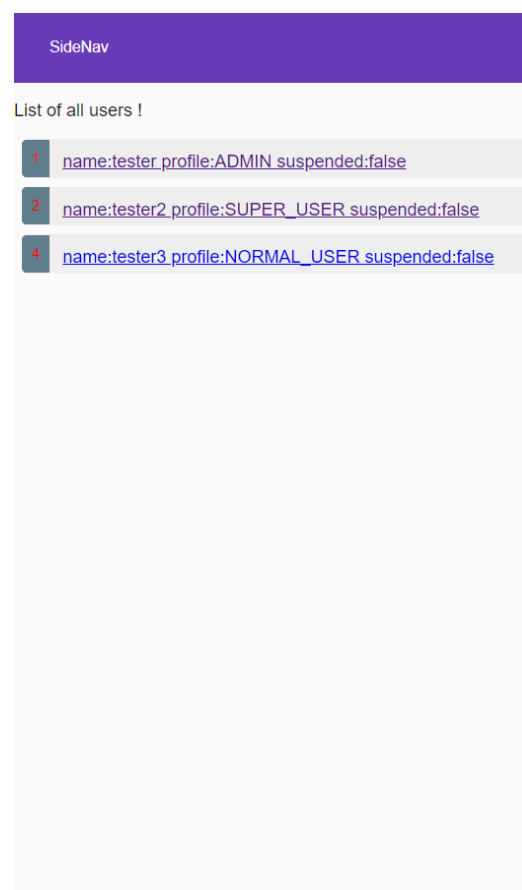


Figura 6 – Página de lista de utilizadores

5. Conclusão

Apresenta-se na figura 7 o plano que está a ser seguido na realização deste projecto. O plano contém as tarefas a realizar bem como os respectivos períodos de tempo. Até ao momento está a ser cumprido o plano que foi proposto.

Encontram-se concluídos o desenvolvimento da arquitectura da solução, o desenho da interface com utilizador da Web API e a implementação do modelo relacional.

Temos já testado com sucesso vários sub-sistemas da nossa aplicação, nomeadamente:

- Login de utilizadores, testado nos navegadores de internet Chrome e Safari, assim como nos sistemas operativos Windows, Linux e MacOS;
- Pedido à API e devolução de resultado em formato JSON;

Os próximos passos passam pela continuação da implementação do Website - gestão de utilizadores e apresentação de dados assim como a continuação da implementação da Web API.

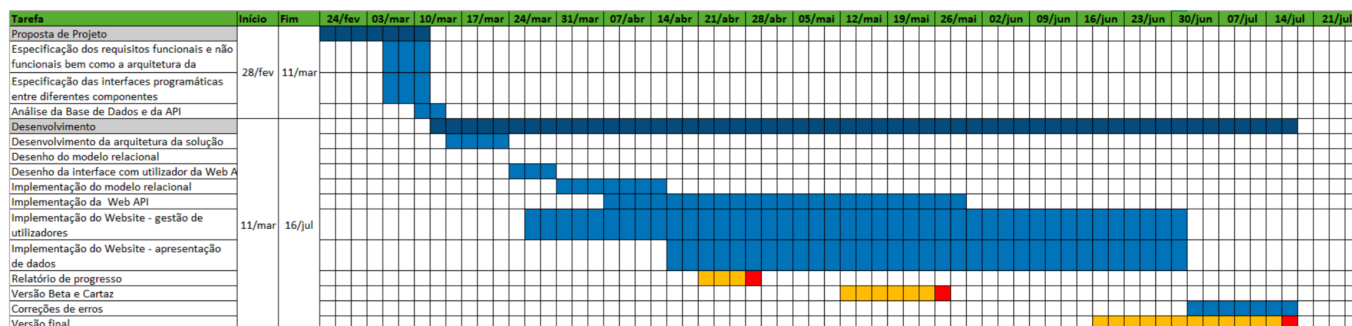


Figura 7 – Planificação temporal das atividade e tarefas associadas ao projecto.

6. Referências

1. <https://leafletjs.com/>, Leaflet
2. <https://nodejs.org/en/>, JavaScript Node.js
3. <https://angularjs.org>, AngularJS
4. <https://code.visualstudio.com>, Visual Studio Code
5. <https://kotlinlang.org/>, Kotlin
6. <https://spring.io/guides/gs/serving-web-content/>, Spring MVC