Instituto Superior de Engenharia de Lisboa (ISEL)
Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores (DEETC)

MEIM
Mestrado em Engenharia Informática e Multimédia

PLANEAMENTO DE VIAGEM EM VELEIRO NAVEGAÇÃO

Rui Filipe Guimarães dos Santos

Júri

Presidente	*Prof. Doutor* Carlos Jorge de Sousa Gonçalves
Vogais	
Arguente	*Prof. Doutor* Rui Manuel Feliciano de Jesus
Orientadores	*Prof. Doutor* Paulo Manuel Trigo Cândido da Silva
	Eng. Hugo Trovão

Março, 2024

Resumo

Na concepção de um trajeto marítimo existem diferentes componentes que se devem ter em conta para o seu bom sucesso.

O planeamento de um trajeto ponto a ponto não implica apenas a utilização de instrumentos de auxílio mais sofisticados, mas também depende da coordenação e organização da equipa de navegação, um dos fatores mais importantes para uma navegação bem-sucedida.

Este trabalho tem como objetivo a criação de um sistema, designado por Travelling Planner, capaz de promover e suportar o trabalho colaborativo para um melhor planeamento de todos os processos inerentes à navegação marítima em diferentes plataformas *web* e móvel. Estes processos estão na sua maioria relacionados com fatores externos à embarcação, como a previsão meteorológica, a definição de pontos geográficos que constituem o trajeto, a definição da distância mínima e máxima da linha costeira, as ajudas à navegação (e.g., docas, faróis), comparação de trajetos já existentes ou rotas preferidas por outros navegadores, a sumarização do trajeto definido e a criação de rotas semi-automáticas.

A realização deste projeto levou a que fossem executados testes com aproximação ao que seria uma utilização considerada normal, de gestão de um trajeto desde o início do seu planeamento ao fim.

Concluindo que com a utilização de diferentes tecnologias e informações públicas, como *Open Street Map*, *Open Sea Map*, *MapLibre* é possível realizar sistemas com funcionalidades de manipulação e extração de informação vetorial para criação de sistemas de informação geográfica.

Palavras chave: Mapa, *tiles*, *raster*, vetorial, renderização

Abstract

In the conception of a maritime there are many different components that must be considered for its final success.

The planning of a course doesn't only imply the utilization of most innovative and sofisticated tools, but also depends in the coordination and organization of the navigation team, which is considered to be the most important factor in the success of a navigation course.

This work has the objective of creating a system, designated by Travelling Planner, capable of promoting and support the collaborative work for a better understanding and planning of all the processes adjacent to maritime navigation. This support will be present in two distinct platforms, web and mobile. Majority of these processes are related with external factors, such as weather forecasts, definition of geographical points that compose the route, definition of coastline distance, navigation aids (docks, lighthouses), comparison with previously create routes and also creation of semi-automated routes.

The realization od this project lead to an execution of test cases with an approximation what should be a normal usage of managing and executing a course from beginning to end.

Utilizing different technologies and public information from open source project like, OpenStreetMap, OpenSeaMap and MapLibre, it was concluded that it is possible to develop systems with functionalities of data manipulation and extraction from vector information.nagelsman

Keywords: Map, tiles, raster, vectorial

Agradecimentos

Ao Professor Doutor Paulo Trigo, pela sua orientação durante todo o desenvolvimento deste projeto e sobretudo, a sua paciência e capacidade de me apoiar e motivar a fazer mais e melhor.

Ao corpo de docente do Instituto Superior de Engenharia Lisboa que ao longos destes anos que tive em contacto, me forneceu uma educação superior com elevada qualidade e pertinência para o meu trajeto profissional. Se em parte hoje me considero bem sucedido no meu trabalho, deve-se aos valores que daqui levo.

Aos meus pais, e em especial à minha mãe, que sempre acreditaram em mim, motivando-me para ser um homem com "H grande", e apesar das infelicidades que a vida nos prega, nunca perderam as suas ondas positivas e contagiantes.

Ao meu irmão, que me apoia em tudo o que me dedico e incentiva-me a ser uma pessoa bem instruída, com valores íntegros e pensamento crítico.

E um especial agradecimento à Ana, minha futura mulher, que em todos os momentos nunca desistiu de me apoiar em todas as minha fases de desenvolvimento pessoal, e se não fosse pela sua persistência e vontade de me ver vencer, não teria terminado este projeto.

"It's not possible, it's necessary"
Cooper, Interstellar 2014

Índice

Lista de Tabelas

Lista de Figuras

Acrónimos

AIS Automatic Identification System

GFS Global Forecast System

GIS Geographic Information System

GPS Global Positioning System

GPX Global Positioning Exchange

JSON JavaScript Object Naming

MVT Mapbox Vector Tile

OSM Open Street Map

PBF Protocol Buffer

Capítulo 1

Introdução

O planeamento de um trajeto é um conjunto de processos que permitem delinear uma sequência de ações para uma abordagem segura e precisa na sua execução [Silveira et al., 2011]. Nestes processos, a definição de pontos geográficos que constituem o trajeto é considerado o mais importante. No caso do planeamento de um trajeto marítimo, estes pontos estão dependentes da origem e destino do trajeto, regras e sinalizações presentes no mar. Estas são maioritariamente representadas por boias de sinalização e por áreas destinadas a diferentes tipos de tráfego marítimo. Uma percentagem significativa da maioria dos acidentes marítimos registados, deveram-se sobretudo a erros e falhas relacionados com humanos, tais como falta de conhecimento, planeamento, análise e comunicação [Conceição, 2018].

A realização deste projeto oferece um sistema que promova o trabalho em equipa, auxilie os navegadores com diferentes graus de conhecimento e objetivos, quer sejam praticantes de navegação por motivos profissionais ou recreativos. Este sistema é constituído por uma componente *web* e móvel. A aplicação *web* fornece ferramentas que permitem a definição e manipulação de pontos geográficos de um trajeto, documentos agregadores da informação de um planeamento, visualização do mapa da terra, sinalizações marítimas, previsões meteorológicas e inserção de embarcações. No caso da aplicação móvel, contém funcionalidades para visualização dos trajetos inseridos na aplicação *web* e dos mapas em ausência de conectividade assim como toda a informação associada ao trajeto.

1.1 Contributos

A realização deste trabalho pretende colmatar as falhas presentes em outros sistemas oferecidos, como suporte ao planeamento de um trajeto marítimo, criação dos trajetos mais relevantes e oferecer uma experiência centrada no utilizador onde se une o conceito de equipa com o planeamento de um trajeto marítimo. Denota-se os seguintes contributos do sistema:

- Associação dos elementos das respetivas equipas a trajetos e funções de cada elemento;
- Ligação entre pontos geográficos para a formação do trajeto.
- Pesquisa de previsões meteorológicas considerando o trajeto pretendido.
- Armazenamento de informações do trajeto em caso de ausência de conexão.
- Sumarização das informações do trajeto em documentos.
- Desenvolvimento de aplicação móvel e *web* que permitem o acompanhamento da equipa durante o trajeto.

1.2 Repositórios GitHub

O sistema realizado está divido em 6 repositórios diferentes, em que cada um corresponde a uma componente do sistema. A decisão da divisão teve como base uma lógica orientada a micro-serviços em que cada repositório representa uma componente do sistema.

- *Gateway* - `https://github.com/rfgsantos/tp-gateway-nginx`
- Gestor de dados - `https://github.com/rfgsantos/tp-cms`
- Gestor de *Vector Tiles* - `https://github.com/rfgsantos/tp-tile`
- Aplicação móvel - `https://github.com/rfgsantos/tp-mobile`
- Aplicação *Web* - `https://github.com/rfgsantos/tp-ui`
- Gestor de *Raster Tiles* - `https://github.com/rfgsantos/tp-raster`

1.3 Estrutura do relatório

O presente relatório está organizado em 7 capítulos, incluindo este primeiro capítulo introdutório do tema:

Capítulo 2: Enquadramento Teórico
Introdução aos conceitos teóricos sobre sistemas de informação geográfica, diferentes projeções, sistemas de coordenadas, divisão do primeiro mapa em secções.

Capítulo 3: Trabalho Relacionado
Descrição dos diferentes sistemas de informação geográfica que constituem o sistema desenvolvido neste trabalho, apresentando as diferenças e valências de cada um.

Capítulo 4: Modelo Proposto
Apresentação do modelo a ser desenvolvido, fundamentação das decisões tomadas, arquitetura adotada e outros conceitos.

Capítulo 5: Implementação do Modelo
Apresentação do sistema desenvolvido do ponto de vista de engenharia e especificação das tecnologias.

Capítulo 6: Validação e Testes
Demonstração dos resultados obtidos de acordo com os cenários de teste aplicados ao sistema.

Capítulo 7: Conclusões e Trabalho Futuro
Conclusão dos resultados obtidos e descrição de trabalho futuro.

1.4 Convenções de escrita

- Algumas expressões de uso comum em inglês (ex: *software*, *hardware*, entre outras) foram mantidas no original.

- Os algoritmos apresentados foram escritos em inglês mantendo coerência

com expressões utilizadas em inglês, favorecendo uma relação mais direta com a terminologia técnica.

- Foi utilizado estilo itálico para expressões em inglês na ausência de tradução.

Capítulo 2

Enquadramento teórico

2.1 Sistemas de representação da terra

Os primeiros mapas representativos do planeta terra usavam uma grelha simples de linhas paralelas e os meridianos com a mesma escala para ambos, denominados por projeção comum ou projeção *Plate Carrée* [Vis, 2018]. Apesar de muito utilizados, resultavam numa distorção de formas e ângulo comparando com a realidade provocando desconfiança por parte dos navegadores nos mapas criados. O desenvolvimento da projeção de *Mercator* teve como objetivo melhorar os mapas anteriores e considerados comuns. Em ambos os casos, projeção comum e *Mercator*, são consideradas projeções cilíndricas. A terra é projetada num cilindro que é cortado tornando-o "aberto", sendo depois desenrolado numa "folha lisa" (Figura 2.1).

A linha do equador automaticamente toca as extremidades do cilindro, sendo o ponto onde é feito este contacto considerado (0,0) e será o ponto de origem do sistema de coordenadas. O valor na abcissa (eixo dos X) do sistema

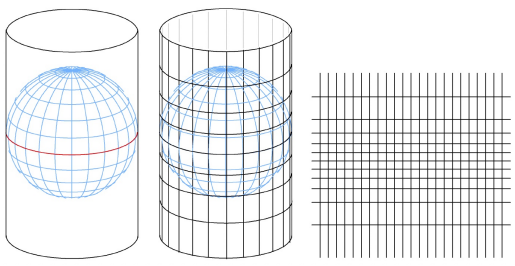

Figura 2.1: Projeção cilíndrica

de coordenadas corresponde à longitude do equador na esfera, variando no intervalo -180º e 180º, onde estes extremos são coincidentes. O valor na ordenada (eixo dos Y) corresponde à latitude, em que este é ângulo que uma linha normal faz com o equador. A aplicação deste método para uma projeção, resulta em representações mais aproximadas à realidade como no exemplo das linhas loxodrómicas na projeção *Mercator* na figura 2.2.

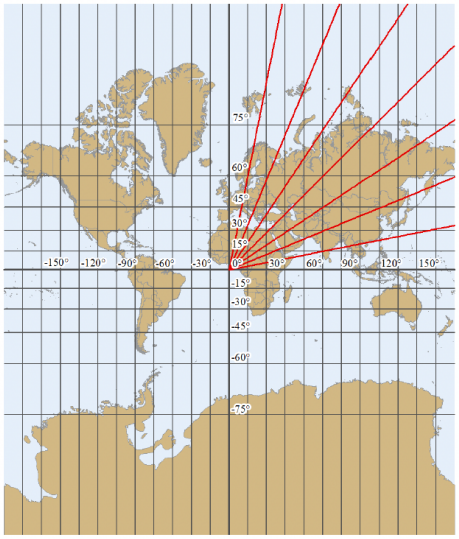

Figura 2.2: Loxodromia na projeção *Mercator*

2.2 *World Map Tiles*

Sistemas com base em *tiles*, utilizam lógicas de mapeamento que representam as posições originais do planeta terra bidimensionalmente, dividindo a superfície formando uma grelha [John T. Sample, 2010]. Esta lógica é o elemento fundamental destes sistemas sendo possível de adaptar para qualquer resolução e projeção. O mapeamento consiste na conversão de uma coordenada geográfica para uma área que esteja coberta por um *tile* do esquema lógico. Este processo começa por representar o planeta terra com uma projeção (Figura 2.3).

Em seguida, os índices dos *tiles* começam no valor 0 sendo sucessivamente incrementados. Na Figura 2.3, está representado o nível de *zoom* 1, em que sub-divide a projeção, neste caso de *Mercator*, em 4 *tiles* iguais. Por cada

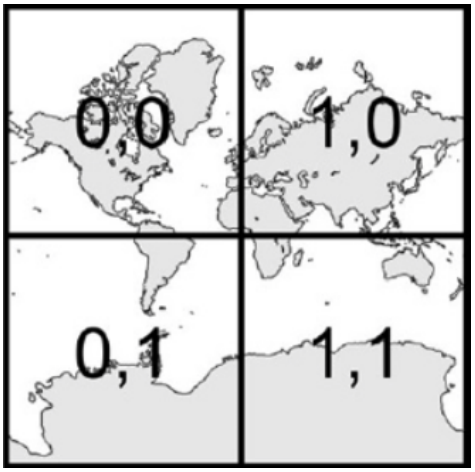

Figura 2.3: Lógica de mapeamento

incremento do nível de *zoom*, cada *tile* é sub-dividido em dois, duplicando o número de *tiles*. O número de *tiles* que constitui cada lado do quadrado em cada nível de *zoom* é calculado equação 2.1.

$$n\_tiles = 2^{zoom} \tag{2.1}$$

e o número de colunas é igual ao número de linhas é definido pela equação 2.2

$$colunas, linhas = (2^{zoom}, 2^{zoom}) \tag{2.2}$$

Na imagem observa-se o primeiro *tile* tem os índices (0,0) e o segundo (1,0) sendo que a alteração é do número representativo da coluna. As equações 2.3 e 2.4 demonstram o cálculo para localização de uma coordenada geográfica específica com um determinado nível de zoom.

$$c = (\lambda + 180.0) * \frac{360.0}{2^i} \tag{2.3}$$

$$r = (\phi + 90.0) * \frac{180.0}{2^{i-1}} \tag{2.4}$$

onde

$c \equiv$ índice horizontal do *tile*

$r \equiv$ índice vertical do *tile*

$\lambda \equiv$ longitude

$\phi \equiv$ latitude

$i \equiv$ nível de *zoom*

A numeração dos *tiles* descrita foi adotada e criada pela empresa *Google*, ignorando as regiões polares o mapa é representado de forma quadrada [John T. Sample, 2010]. Esta não é a única forma de numerar os *tiles* sendo que podem existir múltiplas outras lógicas, como por exemplo a adotada pela empresa *Bing* (figura 2.4). A projeção é a mesma utilizada na figura 2.3 diferenciando-se apenas na numeração.

Figura 2.4: *Bing map tiles scheme*

2.2.1 *Raster Tiles*

Uma imagem digital é uma representação bidimensional de uma imagem que pode ser em formato *raster* ou vetorial. *Raster* (ou *bitmap*) utiliza uma grelha retangular de elementos (pixeis) para demonstrar a imagem [John T. Sample, 2010]. Os pixeis das imagens *raster* representam a cor da imagem no respetivo ponto. Habitualmente, estes elementos não são visíveis a olho nu devido ao seu tamanho reduzido, mas juntos formam o que o olho humano reconhece como uma imagem.

Estas representações digitais são apropriadas para armazenamento de informação geoespacial, incluindo imagens de satélite, aéreas, fotografias, onde a única operação necessária para converter uma imagem digital para imagem geoespacial, é o mapeamento da sua respetiva coordenada geográfica de forma a representar a superfície terrestre [John T. Sample, 2010]. Existem duas formas de fazer este mapeamento:

- Fornecer um retângulo constituído por 2 pontos geográficos (figura 2.5).
- Uma única coordenada de cada pixel em cada dimensão.

Agregando várias imagens geoespaciais, pode-se construir um sistema de *tiles* em que a sua base são representações digitais da superfície terrestre, sendo considerados *Raster Tiles*. A numeração referida na secção anterior é aqui associada a cada uma destas imagens definidas por retângulos.

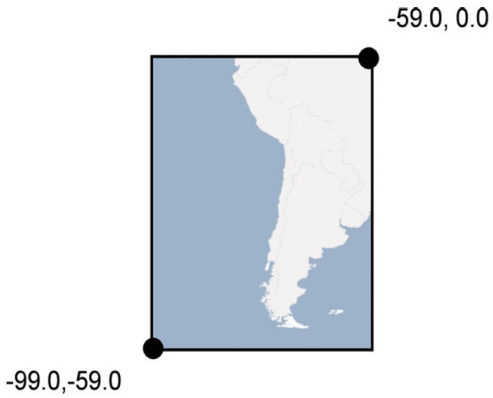

Figura 2.5: *Bounding Rectangle*

2.2.2 *Vector Tiles*

No contexto de sistemas de informação geográfica, dados vetoriais são entendidos como dados definidos por formas geométricas primitivas, tais como pontos, polígonos e linhas. Cada vetor contém características que definem a forma da geometria e a sua posição geográfica. Vetores com geometrias mais complexas também podem ser definidos, contendo formas curvas, cónicas e multi-dimensionais. Adicionalmente, a geometria pode ser uma combinação de várias geometrias [John T. Sample, 2010].

O processo para criação de *tiles* com base em informação vetorial, não é diferente da utilização de imagens. A grande diferença reside na operação de divisão da informação, onde esta não é realizada diretamente na informação original, é no entanto, desenhada no respetivo *tile* a ser criado. No caso dos dados de um objeto pertencerem a múltiplos *tiles*, este é divido e em cada secção é renderizado partes desse objeto (Figura 2.6) [John T. Sample, 2010]. O processo de criação de um *tile* pode ser descrito em pequenas etapas:

1. Escolha do *tile* a ser criado.
2. Determinar o retângulo que define o *tile* (*bounding box*).
3. Extrair a informação vetorial contida dentro da *bounding box*.
4. Renderizar a informação vetorial numa imagem com tamanho correto do *tile*.
5. Armazenar a imagem respetiva do *tile*.

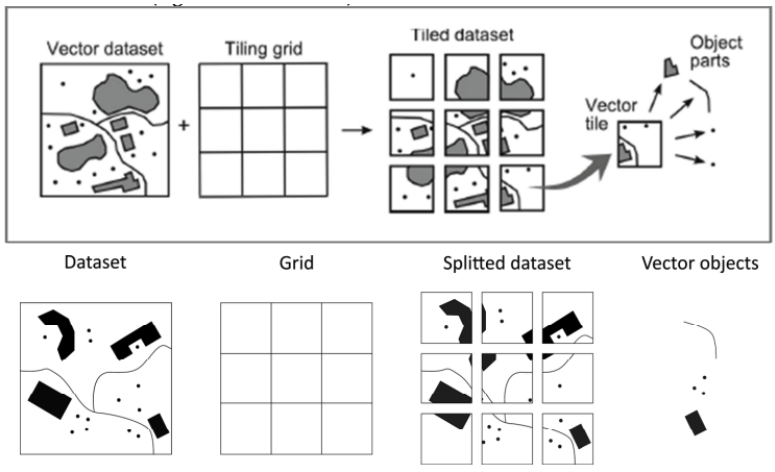

Figura 2.6: Criação de *Vector tile*

Mapas de *Vector tile* representam a opção mais recente. Foram implementados pela primeira vez em 2010 pela Google, primeiro em versão mobile do *Google Maps*, e posteriormente na aplicação *web*.

2.2.3 Formatos de dados

De acordo com Netek, os formatos mais utilizados para conter os dados vetoriais são *GeoJSON*, *TopoJSON* e *Mapbox Vector Tile* (mvt). O formato *GeoJSON* é baseado em *JSON* (Javascript Obeject Notation), um formato de dados universal, que armazena a informação vetorial. Tem uma estrutura simples e independente da plataforma que a utiliza. *TopoJSON* é uma extensão topológica do formato *GeoJSON* que armazena todas as geometrias e objetos em conjunto e não separadamente. Este formato reduz a quantidade de dados, uma vez que cada parte da geometria é guardada apenas uma vez e cada objeto faz referência à geometria. A utilização deste formato pode reduzir até 80% a informação guardada.

Mapbox Vector Tile é um formato baseado no *Google Protocol Buffer*, que é uma linguagem e mecanismo agnóstico à plataforma para serialização de estruturas de dados. Os *tiles* são organizados de acordo com o esquema da Google utilizando o sistema de coordenadas Web *Mercator* (EPSG: 3857). Cada objeto é guardado relativo à origem de cada *tile*. Os objetos podem ser uma das seguintes geometrias: *Point*, *MultiPoint*, *LineString*, *MultiLineString*, *Polygon* ou *MultiPolygon*.

A modificação dos símbolos e dados é considerada a maior vantagem dos *vector tiles* em comparação com os *raster tile*. O cliente que realiza os pedidos, pode modificar os estilos da informação vetorial renderizada múltiplas vezes. Um estilo contem sempre referência à informação a ser renderizada e suas regras condicionais. Existem imensas normas e especificações que definem como os símbolos devem ser criados e renderizados:

- *Mapbox GL Styles* - Um *standard* para todos os serviços. Criada uma simbologia com recurso ao formato mvt da *Mapbox*. O estilo é guardado num objeto JSON com especificação dos elementos das suas propriedades. A simbologia é então editada com editores visuais (*Mapbox Sudio Style Editor*; *Mapnik*) ou simplesmente editando o ficheiro JSON.

- CartoCSS - Desenvolvido pela *Mapbox* em 2010 como um substituto do *Mapbox GL Styles*. É muito semelhante à tecnologia *Cascading Style Sheets* (CSS) para páginas *Web*.

- Geo Style Sheets - Uma das primeiras especificações. Semelhante ao CSS mas nunca aplicado nos projetos mais pertinentes.

- MapCSS - Estilo maioritariamente conhecido por ser utilizado no projeto OpenStreetMap, também ele baseado em CSS. Utiliza estritamente *tags* do OSM.

Capítulo 3

Trabalho Relacionado

3.1 *OpenSeaMap*

As cartas náuticas são difíceis e caras de obter e, em muitos países, não estão sempre atualizados [Bärlocher, 2013]. O *OpenSeaMap* [Bärlocher, 2013] é uma alternativa gratuita destinada a qualquer utilizador que pretenda ter acesso a informação náutica para navegação marítima. O *OpenSeaMap* pretende ter o mesmo tipo de interação que a Wikipédia, sendo que a informação é partilhada com todos os seus utilizadores e recolhida pelos mesmos voluntariamente com visualização em apenas alguns minutos [Bärlocher, 2013]. De acordo com [Bärlocher, 2013], o sistema do *OpenSeaMap* contém informação de todos os oceanos e rios de todo o mundo. Muitos navegadores, mergulhadores, praticantes de desportos aquáticos, entre outros, são uma parte importante no funcionamento deste modelo (*crowdsourcing*)[1], uma vez que a sua informação é compilada e introduzida no sistema para uma melhor composição da carta náutica.

A utilização de *crowdsourcing* no *OpenSeaMap* permite:

- Atualização - toda a alteração é visível imediatamente;
- Especialização - todos os utilizadores introduzem informação que consideram importante;
- Detalhe e precisão - melhoramento continuo e otimização por parte de todos os utilizadores;

[1] O ato de colher informação de um determinado tema, serviço ou ideias por um grande grupo de pessoas

- Abrangência - informação de diferentes grupos de pessoas, com diferentes atividades.

O *OpenSeaMap* é considerado o apoio tecnológico náutico mais rápido do mundo. A carta náutica não termina apenas na linha costeira, mas fornece detalhes das docas, da infraestrutura associada, rotas, tráfego marítimo, entre outras funcionalidades.

Este sistema é na verdade um sub-sistema de outro (*OpenStreetMap*). O *OpenStreetMap* é uma coleção de informação geoespacial fornecida pela mesmo tipo de modelo, onde existem três fontes de meta-dados[2]:

1. Cerca de um milhão de voluntários fornecedores de informação espacial utilizando GPS[3], em casa ou em viagem. Esta ação por parte dos mesmos, permite a atualização do mapa com uma precisão mais elevada.

2. Organizações e autoridades que acreditam na natureza do projeto. *OpenStreetMap* e fornecem dados voluntariamente.

3. Fotografias aéreas com boa resolução, obtidas através de serviços externos, como a Microsoft e *Bing*, são depois processadas manualmente, sendo convertidas para informação vetorial.

3.2 Aplicações e Sistemas

Foi realizada uma pesquisa sobre outros projetos que providenciassem as mesmas funcionalidades ou outras alternativas, que tornassem este projeto uma mais valia ao ser criado. Muito do trabalho relacionado implica a existência de uma aplicação móvel ou um sistema, que suporte a navegação de uma viagem marítima.

3.3 *C-MAP*

Trata-se de uma empresa [C-MAP, 2023] dedicada à criação de material de suporte especifico para navegação marítima. Os apoios concebidos por esta empresa são maioritariamente cartas náuticas digitais, com múltiplos modelos de apresentação que podem ser úteis.

[2]Dados dados relacionados com outros dados
[3]GPS - *Global Positioning System*

Funcionalidades Gratuitas	**Funcionalidades não gratuitas**
Posição GPS e Navegação total	Cartas náuticas
Mapas *offline* ilimitados	Rotas automáticas e manuais
Previsão meteorológica de 5 dias	Rotas pessoais
Visualização da profundidade	Pontos de interesse
Suporte de AIS	Ficheiros GPX

Tabela 3.1: Funcionalidades gratuitas e não gratuitas da aplicação C-MAP

Esta aplicação é das mais utilizadas por todos os tipos de navegadores, sendo das mais cotadas na *App Store (iOS)* e *Google PlayStore*. Este projeto teve por base muitas características apresentadas nesta aplicação, que são claramente importantes para o mercado onde se pretende inserir.

3.4 *iNavx*

Navionics é também uma empresa que se compromete na criação de mapas cartográficos digitais com o intuito de conter atualizações constantes. Esta empresa posiciona-se no mercado em formato de subscrição com uma duração limite de 1 ano gratuito. É considerada das empresas que fornece informação mais fidedigna, sobretudo pela sua longa presença no mercado da navegação marítima conferindo-lhe maior credibilidade. Algumas das funcionalidade mais importantes incluem:

- Rotas de porto a porto;
- Mapas *offline*;
- Rotas automáticas e manuais;
- Informação meteorológica;
- Diferentes mapas cartográficos;

As duas aplicações constituem um conjunto de funções das quais se considerou de maior relevância, mas que apenas permitem a um conjunto seletivo

de utilizadores dispostos a realizar pagamentos mensais para uso desta tecnologia.

Sistema	Rotas Manuais e Automáticas	Ficheiros auxiliares	Meteorologia	Posição atual	Mapas *offline*
iNavx	✓	✗	✓	✗	✓
C-MAP	✗	✗	✓	✓	✓
Travelling Planner	✓	✓	✓	✓	✓

Tabela 3.2: Sumário das funcionalidades de cada sistema

A Tabela A.1 apresenta um sumário das funcionalidades mais pertinentes na construção de um sistema de informação geográfica, realçando as diferenças entre o *Travelling Planner* e os restantes apresentados nesta secção.

Capítulo 4

Tecnologias de Suporte

PostGIS

A biblioteca *PostGIS* [PostGIS, 2023] é uma extensão que fornece suporte em formatos de dados geográficos e operações sobre os mesmos em bases de dados de *PostgreSQL*.

Estes formatos geográficos, permitem a representação de estruturas que nos são conhecidas na projeção espacial em aplicações (*Google Maps*), como estradas, edifícios, descampados, monumentos, faróis, entre outros. A sua forma geométrica é constituída por pontos, linhas, polígonos, retângulos, quadrados e círculos. Em bases de dados estas são, um conjunto de entidades que são traduzidas em tabelas em que a utilização desta biblioteca, permite adicionar colunas com este formato de dados.

Figura 4.1: Exemplo de forma geométrica

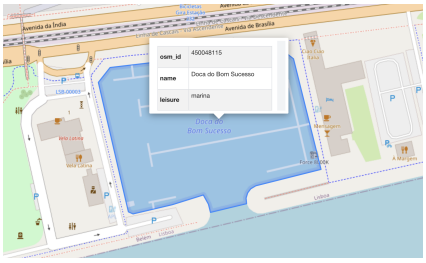

Figura 4.2: Propriedades da forma geométrica

Na Figura 4.1, observa-se um exemplo de geometria com cor azul, que representa uma marina. Este registo está presente numa tabela que contém outras informações alusivas à estrutura em si, como o nome, o identificador

e o tipo de estrutura (figura 4.2). Estes dados são provenientes de uma base de dados pública do sistema *OpenStreetMaps*.

Além de facilitar a utilização de dados com formatos geométricos, a biblioteca dispõe de um conjunto de funções que realizam operações matriciais e aritméticas, obtendo outras geometrias. Operações deste tipo podem ser entendidas como a distância entre duas geometrias, o cálculo da área de uma geometria, a extração de segmentos de reta, o conjunto de pontos constituintes da geometria, a intersecção entre duas geometrias, entre outras.

Bases de dados de informação geográfica

A página *web* oficial [OpenStreetMap, 2023] referencia algumas das fontes possíveis destes dados.

Geofabrik

Geofabrik [Geofabrik, 2023a] é um projeto encarregue de agregar informações correspondentes a infraestruturas e representações geográficas do globo terrestre. Estes projeto está diretamente relacionado com o sistema *OpenStreetMaps* [OpenStreetMap, 2023].

Estes dados foram utilizados no projeto com intuito de criar um ponto de partida para que fosse possível a definição de funções para responder às funcionalidades. As informações geográficas mais relevantes contidas nesta base de dados, são as marinas ou portos náuticos e faróis de sinalização. No contexto do projeto, apenas foram retirados os dados alusivos a Portugal [Geofabrik, 2023b].

Existem dois formatos, *osm* ou *shp*, em que estes dados podem ser carregados, o original e principal do *OpenStreetMap*, que coloca os dados mais aglomerados e de acordo com a formas geométricas, o formato .osm. Os dados do OSM estão em XML. Estes dados são importados com a ferramenta *osm2pgsql* [Osm2pgsql, 2023], executando os comandos:

```
$ osm2pgsql -d gis --create --slim  -G --hstore
--tag-transform-script
openstreetmap-carto.lua -C 2500 --number-processes 1
-S openstreetmap-carto.style
portugal-latest.osm.pbf
```

Eurostat

O Eurostat [Eurostat, 2023] é o ponto central de dados estatísticos e informações gerais da União Europeia. Esta equipa é responsável pela atualização de todos os dados relevantes que estão relacionados com os países da União Europeia.

A informação presente na base de dados do *OpenStreetMap* é suficiente para algumas operações, no entanto, devido à precisão necessária em funções mais cruciais, foi necessário recorrer aos dados do *Eurostat*. É importante que o grau de qualidade dos dados fornecidos nas diferentes bases de dados seja elevado, representando as suas formas naturais e características da superfície terrestre. A região costeira de um país é relevante para uma navegação marítima de caráter obrigatório, tanto para entender a distância a que se encontra da costa, como para realizar cálculos e definir trajetórias com possíveis pontos de auxílio. No trabalho, esta será utilizada para realização de operações de aferição da distância da embarcação a um ponto terrestre e para criação de trajeto semi-automáticos que terão como base a linha costeira portuguesa.

Maplibre JS

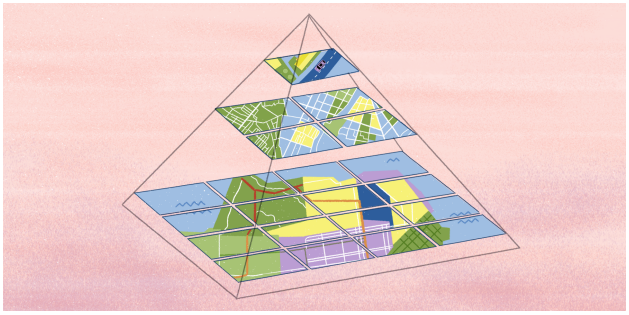

Figura 4.3: Divisão em secções de acordo com *zoom*

A biblioteca *Maplibre* dispõe de um conjunto de funções que permitem facilmente a disponibilização de um mapa com comportamentos de interação por parte de um utilizador. Quando aplicado à linguagem *Web*, em conjunto com as tecnologias de *HTML*, *CSS* e *Javascript*, é necessário definir 3 passos fundamentais para colocar um mapa visível:

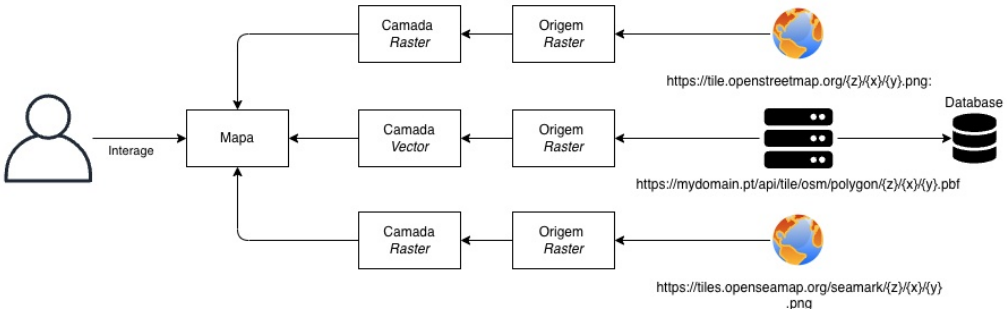

Figura 4.4: Camadas definidas no mapa

1. O contentor do mapa, onde irá assentar todo o tipo de grafismos.
2. As fontes destes grafismos, sejam estas imagens ou informação vetorial;
3. As camadas de apresentação de cada uma destas fontes, bem como ajuste de parâmetros.

Estes conceitos são entendidos no *Maplibre* como a definição de *Sources* (fontes) e *Layers* (camadas). A versatilidade da apresentação destas imagens ou informações, bem como a sua organização, foram pontos a favor para a escolha da biblioteca. Na Figura 4.3, está representada a forma de como são divididas as imagens em secções, de acordo com o nível de *zoom*.

O mapa foi definido de acordo com a figura 4.4. As 3 camadas no diagrama, representam o seguinte:

- A fonte é um servidor de imagens *raster* proveniente do *Open Street Map*, que contém grande parte das infraestruturas, como prédios, comboios, ruas ou paisagens naturais;
- A informação vetorial proveniente da base de dados que renderiza as geometrias, traduzindo-se em conjuntos de pontos, linhas ou polígonos do mapa;
- As imagens que apenas contém informação alusiva às ajudas de navegação, que irão conter a posição das boias de sinalização, os ícones de sinalização de faróis e docas e estradas de navegação. A fonte é um servidor, também do OSM, mas com a derivação de estar apenas relacionado com mar, *Open Sea Map*.

Figura 4.5: Resultado final da união das camadas

Mapbox

O Mapbox é a biblioteca mais conhecida [Mapbox, 2023], no entanto, no desenvolvimento deste projeto, não se optou pela adoção desta biblioteca, uma vez que a grande maioria das funcionalidades pertinentes é dependente de uma chave de acesso, conhecido também como *access token*. Por outro lado, a comunidade que se interessa pela área de sistemas de informação geográfica, teve a iniciativa de criar uma biblioteca que agrega as funcionalidades mais pertinentes do *Mapbox*. Esta biblioteca foi denominada *Maplibre* e será a escolha para este projeto que permitirá a renderização dos *tiles*.

Angular

A interface de utilizador, a escolha da biblioteca *Angular* baseia-se na experiência profissional e na ótica de otimização de aprendizagem das tecnologias. Pretende-se assim colmatar os atrasos do desenvolvimento e melhorar a estruturação do código. O *Angular* é considerada uma *framework*[1] desenvolvida pela *Google*, que visa a programação de interfaces gráficas para *Web* com paradigmas orientados a componentes *web*.

[1]Em conceitos de engenharia de *software*, entende-se por um conjunto de funcionalidades que podem ser reutilizadas para diferentes propósitos.

Nginx

Uma *gateway* é um conceito em engenharia informática que se entende por um único ponto de entrada de um sistema, tornando-o mais fechado ao exterior e com possíveis configurações de segurança centralizadas para todos os pedidos que recebe. É também responsável pelo encaminhamentos dos pedidos para aplicações ou sub-sistemas. O *Nginx* é um *reverse-proxy* servidor *web software* com capacidade de criar regras de roteamento através do endereço resolvido no pedido HTTP.

Django

O gestor de dados utilizará a *framework* da linguagem de programação *Python*, com foco no desenvolvimento *web*, *Django*. Esta *framework* tem funcionalidades que se consideram importantes no desenvolvimento de uma plataforma *web*, como o versionamento da base de dados, a inclusão de um sub-sistema capaz de gerir as informações por ele definidas (CMS) [2]. O versionamento de base de dados é importante para integridade dos dados presentes num sistema.

FastAPI

O servidor de *tiles* vetoriais é o componente responsável pela disponibilização da informação geográfica relacionada com os percursos criados por utilizadores do sistema. Neste bloco, a *framework* de escolhida é a *FastAPI* [FastAPI, 2023], uma micro-biblioteca de *Python* especializada na criação de *API*, sendo a concorrência uma característica importante e presente na definição de pontos de acesso. A alocação de pedidos referentes à renderização de mapas geográficos, tende a colocar o servidor numa gestão massiva dos pedidos recebidos por um cliente apenas. Prevendo a utilização de múltiplos utilizadores em simultâneo, que irá despoletar um aumento exponencial de carga no servidor, sendo uma biblioteca que funciona maioritariamente com pedidos *REST*. [3]

[2] *Content Management System* - designação de um sistema capaz de realizar operações de atualização, remoção, criação e leitura sobre dados alojados num servidor.

[3] *Representational State Transfer* - estilo de arquitetura *web* que define restrições sobre os pedidos que podem ser realizados ao servidor, bem como as suas respostas.

Flutter

O *Flutter* [Flutter, 2023] foi criado pela *Google* com o objetivo de se inserir nos mercados de tecnologia multi-plataforma (*Ionic* e *Xamarim*). As suas características mais importantes são a capacidade de escrita de uma só fonte de código para compilação e execução em múltiplas plataformas (*iOS*, *Android* e *web*). O *Flutter* é conhecido pela sua facilidade de criação de interfaces apelativas ao utilizador, bem como o desempenho em dispositivos móveis.

Open-Meteo

O *Open-Meteo* [OpenMeteo, 2023] é uma API gratuita que oferece acesso a informações meteorológicas como temperatura atual, direção do vento, velocidade das rajadas de vento, ondulação, entre outras. Este serviço utiliza diversas fontes de informação dos locais provenientes de bancos de dados de cada país. Estes dados são essencialmente ficheiros binários, difíceis de lidar e requerem experiência prévia necessária para processa-los corretamente. Os dados fornecem um nível de precisão adequado a previsões de 14 dias.

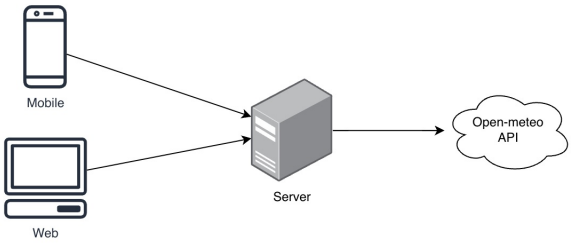

Figura 4.6: Obtenção da informação meteorológica

Na obtenção de dados meteorológicos, os modelos mais conhecidos são o GFS (*Global Forecast System*) e o ECMWF (*European Center Medium-Range Forecast*) [ECMWF, 2023]. No modelo GFS a fonte mais conhecida é o NOAA (*National Oceanic Athmosferic Administration*) [GFS, 2023]. A criação de um padrão global para apresentação destes dados, permite que a sua compatibilidade com sistemas seja um fator preventivo de erros em futuras adoções de outros serviços.

A interação com este serviço é realizada através pedidos HTTP com parametrizações no respetivo *url*. Nestes pedidos devem estar presentes as

informações que se pretendem obter, resultando numa resposta mais concisa dos dados. Na Figura 6.17, estão presentes alguns dos dados que se podem obter com o modelo de dados GFS.

SwitchOSM

O *Switch2OSM* [Switch2OSM, 2023] é um projeto com base nos dados do OSM que tem como objetivo ajudar a incorporação destes serviços por conta própria nos sistemas que desejam introduzir funcionalidades de sistemas de informação geográfica. O *Switch2OSM* tem diversas formas de conseguir servir estes dados em formato gráfico. A página oficial enumera 2 formas de se controlar esta informação para servir aos clientes que vão consumir:

1. Instalação manual num servidor *Ubuntu* ou *Debian*;
2. Instalação através de *Docker* container.

O conjunto de tecnologias para a renderização desta informação, oscila entre *Apache web server*, módulos responsáveis por operações como a leitura dos dados de uma base de dados *PostGIS*, renderização de imagens, mapeamento da imagem com base na informação vetorial, entre outros.

A renderização está relacionada com os componentes do *Mapnik*, biblioteca responsável por mapear os dados vetoriais para uma imagem. Este processo é penoso para os recursos do servidor. Para colmatar este problema, existe um mecanismo que verifica a existência de imagens renderizadas anteriormente.

Capítulo 5

Modelo Proposto

Neste capítulo serão definidos os objetivos do desenvolvimento do sistema, especificar os requisitos funcionais e interligação dos vários componentes.

5.1 Sistema

A figura 5.1 representa uma visão geral da arquitetura do sistema. No lado esquerdo estão representados os dispositivos de cliente que comunicarão com o servidor. A componente *Web* onde serão realizadas grande parte das operações de planeamento e o ponto de acesso à aplicação *web* a ser desenvolvida em *Angular*. Atendendo à naturalidade do projeto, é importante que este seja capaz de responder à necessidade de uma componente móvel, capaz de auxiliar os navegadores nos seus trajetos, centralizando as informações. Esta componente mesma encontra-se ilustrada por um dispositivo móvel ao lado do PC. No servidor *Ubuntu* está ilustrado o ambiente de execução *Docker* com os respetivos *containers* que representam os diferentes módulos que vão processar a informação.

O servidor *Raster Tiles* está separado dos restantes componentes devido a sua elevada necessidade de recursos computacionais para servir estes dados. A separação destes componentes permite que a utilização deste serviço não impacte o funcionamento normal dos restantes.

Enumeram-se as diferentes componentes do sistema:

1. *User Interface* - Interface gráfica do sistema num *Web Browser*;

2. *Gateway* - Ponto único de entrada no sistema;

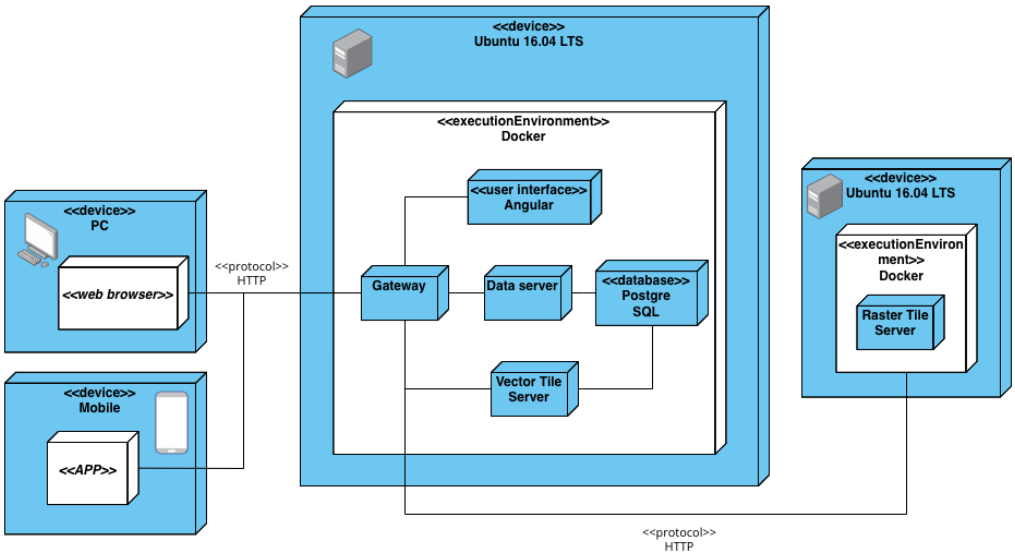

Figura 5.1: Arquitetura do sistema

3. *Database* - Armazenamento dos dados e metadados dos utilizadores;
4. *Tile Server* - Servidor responsável pela informação geoespacial;
5. *Data Server* - Gestão da informação dos utilizadores;
6. *Raster Tile Server* - Servidor responsável pelas imagens *Raster*.

Os componentes do sistema vão utilizar diferentes tecnologias referidas no capítulo 4. Associando os componentes às tecnologias, o *gateway* irá utilizar o *reverse-proxy Nginx*, o *data-server* irá utilizar a tecnologia *Django*, o *Vector Tile Server* a biblioteca *FastAPI* e por último, a aplicação móvel a tecnologia *Flutter*.

5.2 Requisitos

Na presente secção, serão enumerados e fundamentados os requisitos que se pretendem implementar no sistema. Para uma melhor organização, estes requisitos encontram-se organizados por categorias:

1. Equipa

2. Embarcação
3. Planeamento
4. Modo partilhado
5. Autenticação, Registo e Perfil
6. Roteamento
7. Informações gerais

A apresentação dos requisitos será de forma tabular, com um identificador do requisito, referenciando o conceito a que está associado, o número do requisito e a plataforma onde será implementado.

Equipa

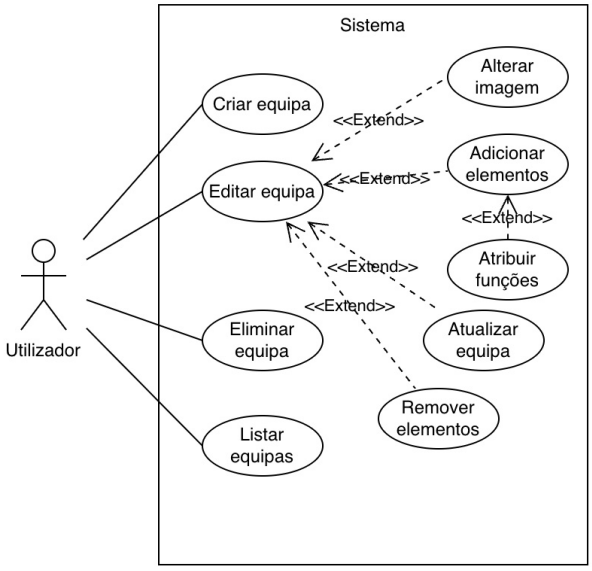

Figura 5.2: Casos de uso da equipa

A informação dos elementos de uma equipa é pertinente para conjugar a informação aquando dos preparativos finais e acompanhamento. Assim, a adição de requisitos que permitam a gestão de equipa cuja entidade no sistema represente os elementos ativos no planeamento é obrigatório.

Na tabela A.7, enumera-se o conjunto de requisitos imprescindíveis. As operações *CRUD* [1], estão presentes na enumeração 1.1 a 1.4. A cada utilizador associado à equipa será atribuída uma função a bordo da embarcação. Estas funções poderão ser, por exemplo, socorrista, *skipper*, mecânico, imediato, "capitão", entre outras. Além da criação de uma equipa, um utilizador deve conseguir visualizar a que equipas pertence, de modo a não impossibilitar integrar múltiplas equipas. A associação de uma imagem à equipa, pretende dar identidade a uma equipa. Uma equipa tem como objetivo a possibilidade de criar planos em conjunto que ambos estejam inseridos, definindo os pontos do trajeto com sentido crítico por parte dos membros.

Embarcação

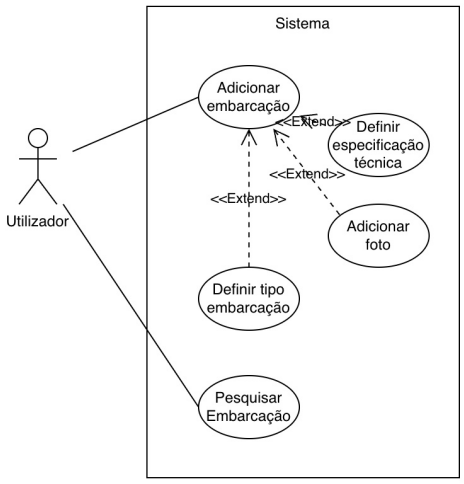

Figura 5.3: Casos de uso das embarcações

Existem informações relevantes para a viagem alusivas às embarcações, que determinam cálculos por parte da equipa, como a quantidade de combustível, o comprimento da embarcação, as medidas das velas. A criação de um banco de dados de embarcações possíveis, seria uma mais valia aquando do planeamento de um trajeto. Os requisitos podem ser consultados no anexo A.6.

[1] *CRUD* entende-se operação de criar, ler, atualizar e remover informações.

Planeamento

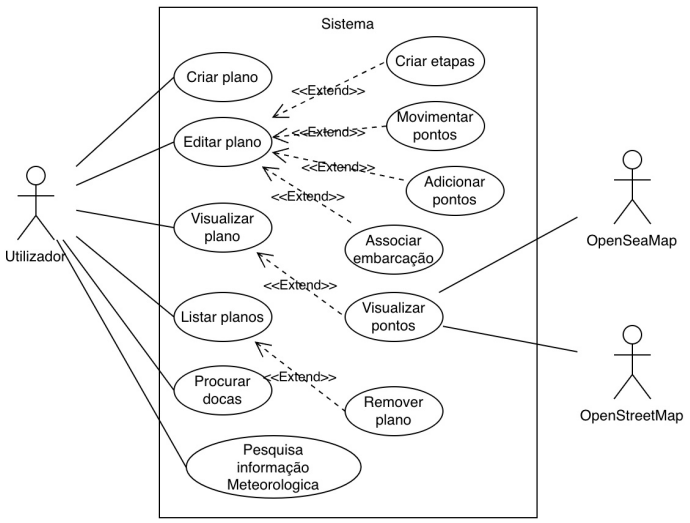

Figura 5.4: Casos de uso do planeamento

As operações básicas (CRUD) sobre a informação fundamental de um planeamento estão presentes na tabela A.5 e enumeradas de 3.1 a 3.4. As meta-informações de cada sub-trajeto, que incluem a distância em milhas náuticas e o tempo em horas e minutos, devem estar presentes na interface gráfica para gestão do caminho a ser percorrido. Destaca-se a importância da manipulação individual destas etapas, sem impacto nas previamente planeadas. A associação de embarcações a um planeamento promove a atualização das meta-informações de fora automática, sempre que esta relação de altera. As informações meteorológicas têm grande impacto nas decisões do planeamento, desde a colocação dos pontos geográficos, às próprias datas em que se pretende realizar o trajeto.

A centralização de informações para os participantes, promove uma melhor organização e um ponto único de divulgação da decisão final.

Modo partilhado

Na presente secção pretende-se realçar os requisitos que possibilitam a decisão simultânea, através de modificações nas características do trajeto, expressar

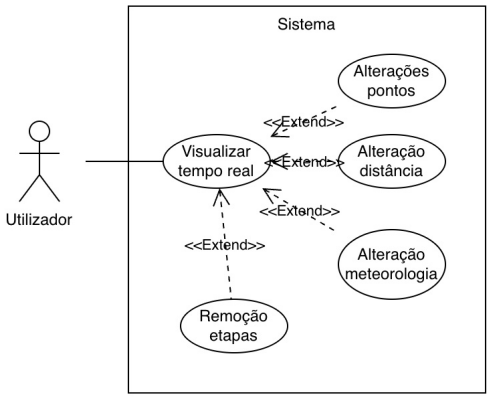

Figura 5.5: Casos de uso da modo partilhado

visualmente as alterações ao planeamento, acessível a todos os elementos da equipa. Os requisitos apresentados na tabela A.4 pretendem promover a importância do trabalho e coordenação da equipa.

Autenticação, Registo e Perfil

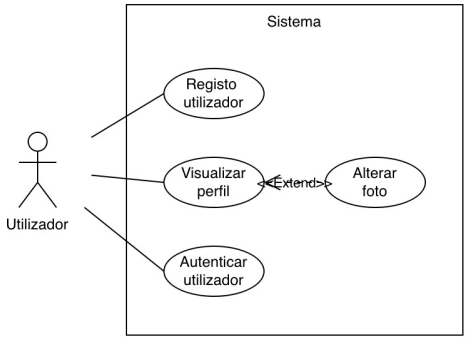

Figura 5.6: Casos de uso do registo, autenticação e perfil

Os requisitos para gestão de utilizadores estão enumerados na tabela A.3. Estes permitem criação de perfis e contas que dão acesso às funcionalidades do sistema. A associação de fotografias aos perfis dos utilizadores facilita a procura destes registos durante a criação de uma equipa (tabela A.7).

Roteamento

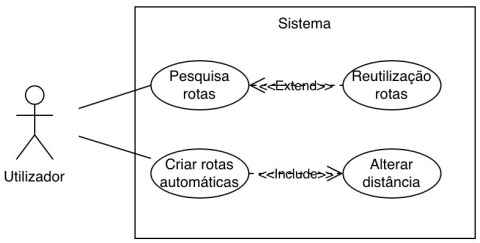

Figura 5.7: Casos de uso do roteamento

A quantidade de trajetos diferentes e planeados por equipas diferentes, e que todos os utilizadores consigam usufruir, agregam informação relevante para abertura de horizontes e perspetivas em futuros planeamentos. Os requisitos na tabela A.2 pretendem introduzir uma automatização para um nível semelhante ás existentes para trajetos terrestres (*Google Maps*).

Informações gerais

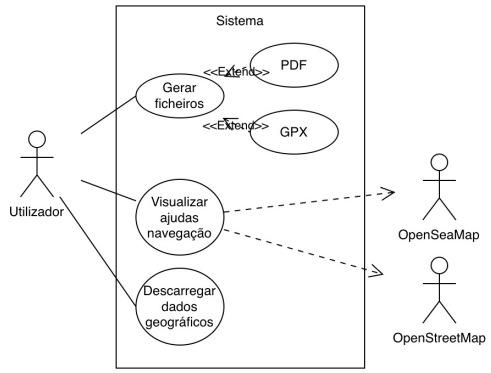

Figura 5.8: Casos de uso das informações gerais

Por informações gerais, entenda-se conjuntos de dados ou funcionalidades, que são complementares às funcionalidades mais centrais do sistema (tabela A.5). Como qualquer tecnologia que visa otimizar áreas de trabalho, esta pode incorrer em falhas, prejudicando a utilização do sistema. Colocando

a hipótese de uma falha, devem existir precauções para colmatar possíveis problemas. Mencionado em secções anteriores, a utilização da aplicação mediante da ausência de conectividade, é um fator obrigatório a solucionar, e por isso, os requisitos listados na tabela A.1, focam-se nestes cenários.

Capítulo 6

Implementação do Modelo

Neste capítulo serão abordadas as especificações técnicas, decisões, pressupostos, vantagens e desvantagens das soluções para cada funcionalidade dos sistema.

6.1 Entidades

As entidades de um sistema constituem a sua representação lógica para armazenamento de informação. Um *User* tem atributos que o definem como único no sistema. No ato de se registar, nem todos os dados são obrigatórios, sendo o mais importante de todos, o *username*. É realizado pelo sistema uma validação da existência na base de dados por outros iguais, obrigando o utilizador a reescrever um novo *username* até ser considerado válido. As restantes propriedades do *User*, têm como objetivo descrever mais detalhadamente o utilizador e assim, ser pesquisado pelos restantes. Este tipo de pesquisas como são referenciadas na secção 5.2, mais especificamente, nas tabelas A.3 e A.7. A propriedade *profile_image* é um conjunto de caracteres alfa-numéricos que são uma compressão em base64 [1] dos meta dados da imagem carregada.

O *User*, este relaciona-se com as entidades *Member*, *Role* e *Team* (figura 6.1). O contexto lógico desta associação, está diretamente relacionado com os requisitos presentes na secção 5.2. A criação de uma equipa é composta por atributos básicas que possibilitam uma descrição detalhada. O atributo *description* visa a escrita do propósito da equipa, o *name* representa o nome

[1] Representação em formata de texto proveniente de dados binários

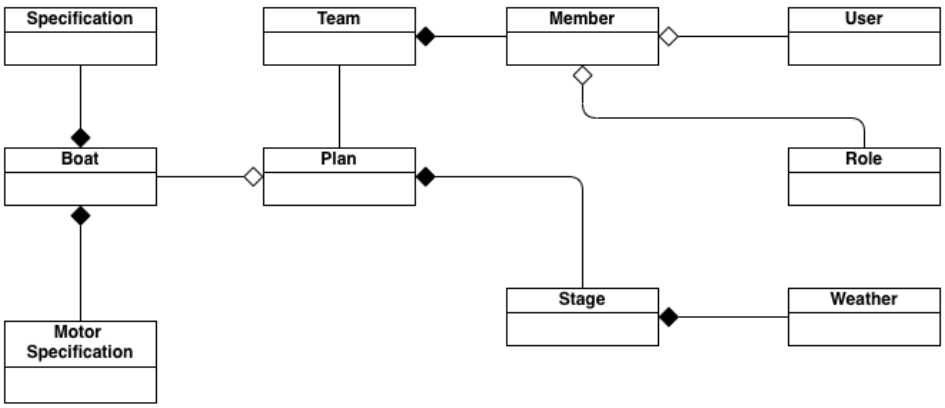

Figura 6.1: Diagrama de classes

da equipa em seguida que é composta por um ou mais *Member*. A entidade *Member* agrega um *User* a um *Role*.

Um dos primeiros passos para a criação de uma equipa, é inerente à relação com a entidade *Plan*. Uma equipa é capaz de criar os planos necessários, independentemente do intervalo temporal em que os pretende realizar. Um plano tem os seguintes atributos considerados essenciais:

- *Name* - Nome do plano;
- *Description* - Descrição livre que pode ser editada pela equipa, om objetivo de conter informações mais especificas sobre o objetivo do plano;
- *Start_date* - Data que se pretende iniciar o plano;
- *Arrival_date* - Data que se pretende terminar o plano ou chegada ao destino;
- *State* - Estado atual do plano (Criado, Em curso ou Terminado).
- *Image* - Imagem ilustrativa do plano;
- *Identifier* - Um identificador único gerado pelo sistema quando este é criado pela primeira vez.

O atributo *Identifier* está diretamente relacionado com o desenvolvimento do Modo Partilhado, que será fundamentado mais à frente neste capítulo.

A entidade *Stage* relaciona-se com o *Plan* numa cardinalidade de 1 para N. Esta prende-se ao facto de a ideia de um plano ser um conjunto de pontos geográficos que formam um caminho a ser seguido pelos navegadores, contendo uma ou mais *Stage*. A utilização do sistema abrange um leque variado de participantes com diferentes objetivos, podendo um planeamento ser, por exemplo, a visita de múltiplas docas desde Lisboa até ao Porto. Considerando o exemplo anterior, deve ser possível dividir o trajeto em subtrajetos, aplicando assim o conceito de etapas (*Stage*). Cada *Stage* contém meta dados diretamente relacionados com a sua geometria (*LineString*). O atributo *Distance* é a distância total da etapa, calculada em milhas náuticas, sendo o tempo medido em horas (*Hours*) e minutos (*Minutes*), dependendo da embarcação escolhida. Ainda relacionado aos atributos da entidade *Stage*, existem referências aos identificadores únicos das geometrias correspondentes aos registos das docas de partida e fim da etapa. A criação de um identificador único para cada etapa (*uui*), está diretamente relacionada com o modo partilhado. Os atributos *start_time* e *end_time* representam o começo e o fim da etapa.

A entidade *Weather* surge por haver necessidade de registar as previsões meteorológicas associadas a cada etapa, uma vez que são fatores diferenciadores do intervalo tempo em que decorrerá o trajeto. Os seus atributos e respetiva origem serão especificados mais à frente neste capítulo.

Por último, a inserção de barcos (*Boats*) e respetivas fichas técnicas (*Specification* e *Motor Specification*) fornecem informações relevantes para serem utilizadas mais tarde.

Pictograma *Open Street Map*

De acordo com [Burk, 1999], um pictograma é uma representação de um ícone em miniatura de um objeto inserido numa entidade. Os pictogramas são utilizados para a extensão de diagramas de entidade relação, inserindo-os nas respetivas entidades a que devem estar associados. Ao inserir estes ícones na representação, entende-se que é uma tabela com um atributo que define uma forma geométrica. Na figura 6.2 estão representadas tabelas provenientes das bases de dados públicas com a omissão de alguns atributos, facilitando

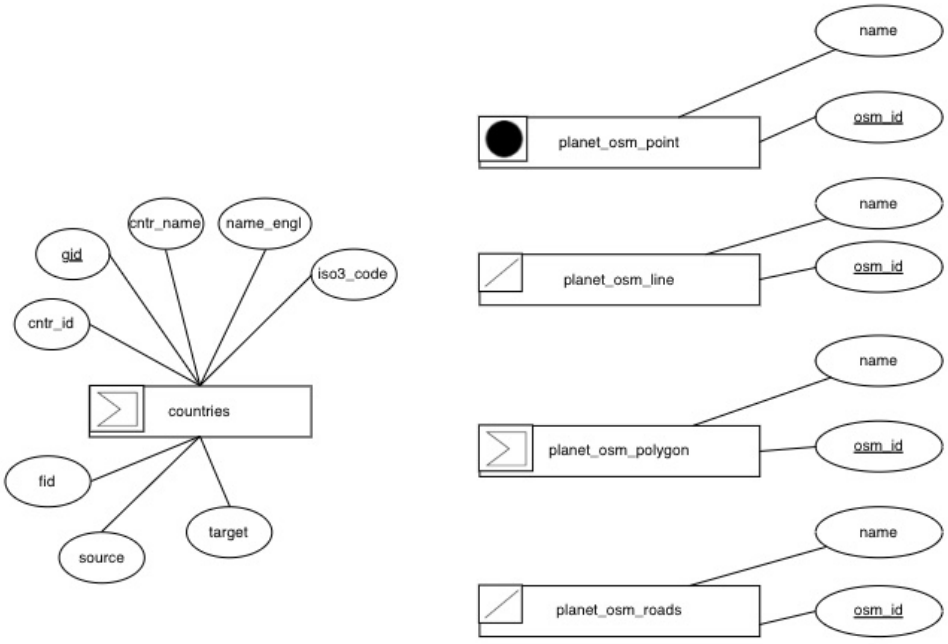

Figura 6.2: Pictograma das entidades

a sua interpretação. A entidade *planet_osm_point* tem um ícone que indica a presença de uma geografia do tipo ponto para todos os registos presentes na tabela. Respetivamente, as entidades *planet_osm_polygon*, *planet_osm_line* e *planet_osm_roads* têm ícones que representam a existência de geometrias do tipo linha, polígono e linha.

6.2 Aplicação *Web*

Abordada a questão de renderização das informações geográficas, prossegue-se para a especificação de toda a aplicação *Web* desenvolvida. Em conformidade com a apresentação dos requisitos em categorias, estas foram igualmente separadas no diagrama de módulos (Figura 6.3).

6.2.1 Planeamento

O planeamento é considerado a categoria mais importante do sistema. Nesta secção serão abordados todos os algoritmos e detalhes das decisões em redor

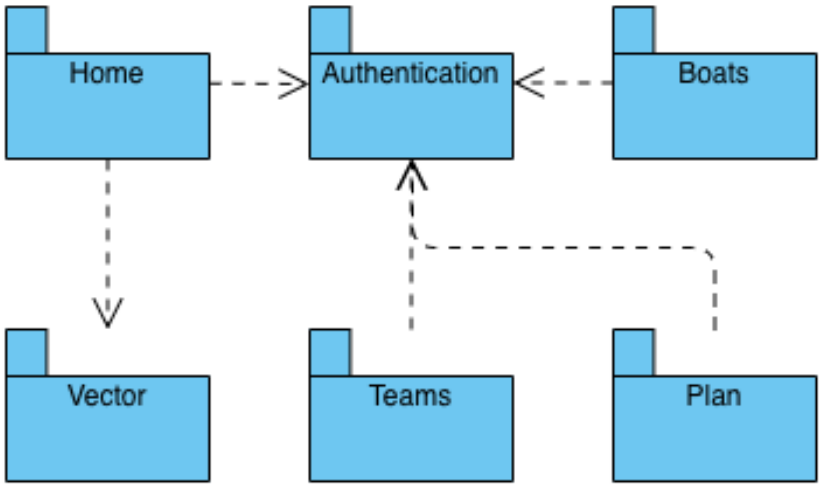

Figura 6.3: Diagrama de módulos

dos casos de uso e funcionalidades que a constituem. Os pontos a serem definidos num trajeto, surgem de uma interação com o mapa por parte do utilizador até ficar completo. A interação com o mapa, permite obter as coordenadas geográficas de cada ponto, latitude e longitude.

O primeiro ponto do trajeto deve ser obrigatoriamente numa marina. Assumindo que a embarcação está contida na geometria da marina no seu arranque, o primeiro ponto de um trajeto é o centroide da geometria correspondente à marina. Este centroide é o centro geométrico de uma geometria, obtido através da função `ST_Centroid` (Figura 6.4).

A camada vetorial que contém as geometrias correspondentes às marinas, é o resultado de uma extração de informação das tabelas com dados geográficos. Esta extração é realizada com uma interrogação *sql* às tabelas em que são agregados os atributos, a definição geométrica e outros atributos calculados (Código 1). Estes dados são transferidos num ficheiro de formato .mvt e renderizado. No código 1, observa-se a criação de um *TileEnvelope* entendido como *BoundingBox*, através dos parâmetros z,x,y que são provenientes do mapa.

A *BoundingBox* é utilizada para intersectar as geometrias da tabela *planet_osm_polygon* filtrando apenas pelas que verificam esta condição, resul-

Algorithm 1 Função de extração Marinas

```
1: envelope ← SELECT ST_TileEnvelope(z,x,y)
2: marinas ← SELECT p, ST_AsMVTGeom(p.way) as geom,
3:       ST_X(ST_Centroid(p.geom)), ST_Y(ST_Centroid(p.geom))
4:       FROM planet_osm_polygon as p WHERE p.leisure='marina'
5: mvt ← SELECT ST_AsMVT(marinas.*, 'marinas', 'geom', 'osm_id')
```

Figura 6.4: Centróide da Doca do Bom Sucesso

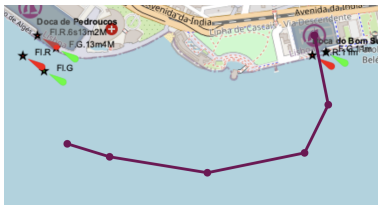

Figura 6.5: Exemplo de trajeto

tando num conjunto de geometrias que serão apresentadas ao utilizador.

O ficheiro .mvt servido, proveniente da função `ST_asMVT` irá conter um objeto representativo das geometrias filtradas e respetivas propriedades. Mediante de um evento de clique na camada vetorial, é possível verificar qual a geometria que se clicou e as suas propriedades. Obtendo estas propriedades, o primeiro ponto de um trajeto será a resultante da coordenada X e coordenada Y, latitude e longitude respetivamente, provenientes das funções `ST_X` e `ST_Y` no código 1. As seguintes ações, irão adicionar pontos no trajeto formando uma linha (figura 6.5).

Informações gerais, como a distância e tempo até ao destino, devem ser calculadas em milhas náuticas por ser a unidade convencional em navegações marítimas. O comprimento de um trajeto pode ser calculado de acordo com a equação 6.1.

$$c/100/1.852 \tag{6.1}$$

Onde c corresponde ao comprimento do trajeto, que é calculado com a função `ST_Length` devolvendo o valor em metros. O valor 1.852 equivale a 1 milha náutica ser 1.852 quilómetros [Moody, 1950].

6.2.2 Rotas semi-automáticas

Atendendo à explicação da composição de um trajeto manual, a criação de uma automatização no processo deve ser vista como um auxílio para com o criador do percurso. Para alcançar esta operação de cálculo destas rotas, foi criado um processo composto por diferentes operações:

1. Criação de função de extração da fronteira de um país europeu à escolha.
2. Criação de função de cálculo da linha costeira de um país europeu à escolha.
3. Criação de função de cálculo dos pontos mais próximos às docas destino e origem contidos na linha costeira.
4. Criação de função de extração de uma sub-linha da linha costeira.

Com base nos dados carregados por parte do *Eursotat*, existe a tabela (*countries*) com as geometrias correspondentes às áreas terrestres dos países europeus. Apenas a geometria de Portugal é relevante no contexto do projeto.

As docas marítimas localizam-se na linha costeira portuguesa. O contorno que define esta linha costeira [2], é parte da fronteira do polígono que representa Portugal. A função `ST_Boundary` calcula esta fronteira, que é a linha poligonal que separa o interior e o exterior de um polígono (figura 6.6).

A linha correspondente à fronteira de Portugal intersecta a fronteira do seu país vizinho (figura 6.7), Espanha. Com este facto, presume-se que para extrair a linha costeira portuguesa podemos calcular a intersecção entre as duas maiores geometrias correspondentes ao território de cada país e em seguida a diferença entre as duas geometrias, Portugal continental e a geometria resultante da intersecção.

Foi criada uma função com a finalidade de extração da linha de contorno de um país (algoritmo 2). Os parâmetros a serem fornecidos são o nome do país, em inglês, e a distância que se pretende obter da linha da original. O tipo de forma geométrica definida para Portugal é um *Multipolygon*, ou seja, composta por múltiplos polígonos, onde são contemplados ilhas, lagos, entre outros. A função `ST_Dump` (Linha 2 algoritmo 2) realiza o processo de

[2]Linha que apenas tem o mar como fronteira

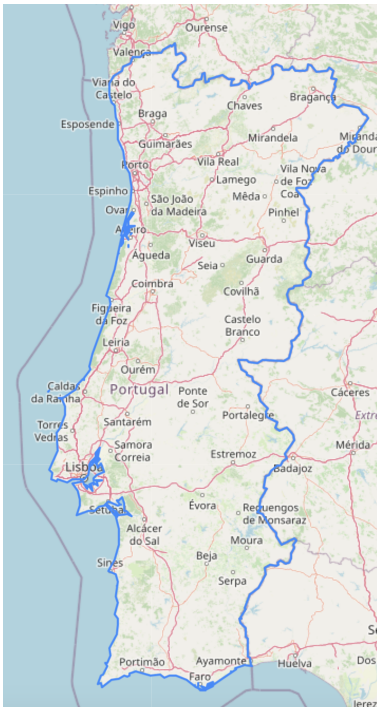

Figura 6.6: Fronteira correspondente à área de Portugal continental

Figura 6.7: Intersecção de fronteiras

extração de todas as geometrias, convertendo-as para geometrias isoladas, criando assim uma lista de geometrias. A lista é ordenada de acordo com a área de cada geometria. Os resultados são limitados apenas a um registo (Linha 5 algoritmo 2), que é o primeiro elemento da lista, onde se sabe ser o registo da geometria com a maior área devido à ordenação realizada.

Considerando a linha de fronteira, foi criada uma nova função com base na anterior, que realiza a operação matricial de diferença entre duas geometrias e devolve a linha costeira (Algoritmo 3).

O primeiro passo é extrair a linha de contorno do país que se pretende obter a linha costeira (Linha 1 algoritmo 3) recorrendo à função criado no Algoritmo 2. Na linha 3 realiza-se a operação de verificação de intersecção entre todos os registos da tabela *countries* e a linha fronteira extraída, resultando uma lista filtrada com apenas os registos dos países que intersectam, semelhante ao ilustrado na figura 6.7. De seguida, na linha 4 são extraídas as linhas de fronteira dos países filtrados. Na linha 5 é calculada a interseção

Algorithm 2 Função de extração das linhas de contorno de um país

```
1: function COUNTRY_VIEW_FUNC (double precision, varchar country)
2:     country_geoms ← ST_Dump(countries.geom) ⊆ countries
3:     country_buffer ← ST_Buffer(country_geoms, precision)
4:     country_boundary ← ST_Boundary(country_buffer)
5:     final_geom ← country_boundaries LIMIT 1
6:     return final_geom
7: end function
```

Algorithm 3 Função de extração da linha costeira

```
1: function COUNTRY_COASTLINE (double precision, varchar country)
2:     c_line ← country_view_func(precision, country)
3:     c_intersect ← c_line ⋂ countries
4:     c_boundary ← ST_Boundary(c_intersect)
5:     c_intersection ← ST_Intersection(c_intersect, c_line)
6:     c_diff ← ST_Difference(c_intersection, ST_Boundary(c_line))
7:     return c_diff
8: end function
```

entre todas as geometrias que intersectam o país que se pretende, resultando na geometria ilustrada na Figura 6.8.

Por último na linha 6, é realizada a operação final de diferença entre a linha de intersecção obtida na linha 6 e a fronteira da geometria do país escolhido, resultando na Figura 6.9. A função `ST_Difference` devolve a parte da geometria que não intersecta com a geometria destino.

Obtendo a linha costeira, é necessário extrair o segmento de reta que representa a distância a ser percorrida entre doca de origem e doca destino. Na geometria representativa da linha costeira podem existir infinitos pontos. A abordagem adotada consiste na descoberta dos pontos mais próximo desta reta às marinas de início e destino de um trajeto.

A função `closest_points_func` (algoritmo 4) tem como objetivo extrair esses pontos. Primeiramente é calculada a a linha costeira do país escolhido na linha 2 com o auxílio da função criada previamente no algoritmo 3. Na linha 3 são calculados todos os centroides das marinas, que estão na tabela `plabet_osm_polygon`. Na linha 4 é utilizada a função `ST_ClosestPoint`, em todos os registos provenientes na linha 3, que tem como objetivo descobrir o ponto mais próximo de uma geometria a outra (figura 6.10).

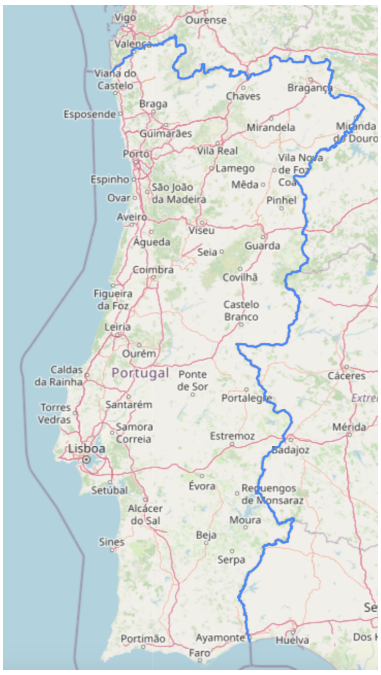

Figura 6.8: Intersecção dos países

Figura 6.9: Linha costeira

Considerando a linha com o ponto representado na linha da figura 6.10, a posição do ponto pode ser entendida como uma fração da linha, assumindo que o inicio da linha é 0 e o final 1, estando o valor desta fração entre este intervalo. O cálculo desta fração é feito pela função `ST_LineLocatePoint` e presente na linha 5. Este valor será importante para o último passo do processo. O retorno desta função constitui o conjunto de registos dos pontos mais próximos da linha costeira de todas as marinas em Portugal e respetivas frações.

O conjunto destas funções permite a concretização final do algoritmo 5,

Algorithm 4 Função de extração dos pontos mais próximos

```
1: function CLOSEST_POINTS_FUNC (double precision, varchar country)
2:     country_line ← country_coastline(precision, country)
3:     centroids ← ST_Centroid(geom) ⊆ planet_osm_polygon
4:     closest_points ← ST_ClosestPoint(centroids, country_line)
5:     points_fraction ← ST_LineLocate(closest_points, country_line)
6:     return closest_points, points_fraction
7: end function
```

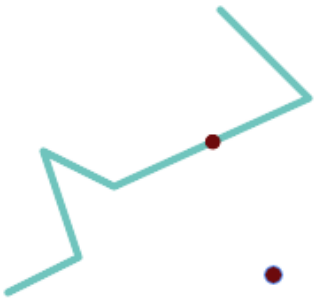

Figura 6.10: Ponto mais próximo da reta a outro ponto

que tem por objetivo criar rotas semi-automáticas facilitando o planeamento do percurso desejado. O código *sql* respetivo, pode ser consultado no anexo A.3.

A funções `ST_LineSubString` (Linha 8 algoritmo 5) permite a extração de um segmento de reta de uma linha. Esta recebe como parâmetros dois valores fracionários que correspondem a pontos da linha. Estes valores fracionários são os calculados no algoritmo 4 e extraídos nas linhas 6 e 7 no algoritmo 5, exemplo ilustrado na figura 6.12. O valor fracionário de uma linha é unidirecional, ou seja, o primeiro ponto da linha é a fração 0 e o último ponto a fração 1. Calculadas as duas frações dos pontos mais próximos às marinas pertencentes à linha costeira, são utilizados para extração do segmento de reta na linha 8. Após a extração do segmento de reta e para finalizar, uma verificação de qual o valor fracionário mais alto é necessária para adicionar os pontos corretamente e formar o trajeto final (figura 6.13). As verificações acontecem nas linhas 10 e 12, em que a `ST_AddPoint` modifica uma geometria fornecida adicionando-lhe um ponto.

6.2.3 Regulação de distância

Em todas as funções definidas para a criação de rotas semi-automáticas está presente o parâmetro *distance*. Este parâmetro indica qual a distância que se pretende navegar (em quilómetros) da linha costeira de um país. Como especificado no algoritmo 2, observa-se a associação do parâmetro à função `ST_Buffer`, cuja responsabilidade é de expandir uma geometria com base numa constante fornecida também como parâmetro. O parâmetro *distance* é

Algorithm 5 Algoritmo final com funções criadas

Require:

1: `start_osm_id` $\subseteq$ `planet_osm_polygon`;
2: `end_osm_id` $\subseteq$ `planet_osm_polygon`;
3: $distance \geq 0.0$;
4: $country \neq none$;

5: **Begin**
6: $start \leftarrow$ `closest_points_func(distance, country)` $\subseteq$ `start_osm_id`
7: $finish \leftarrow$ `closest_points_func(distance, country)` $\subseteq$ `end_osm_id`
8: $substring \leftarrow$ `ST_LineSubString(country_coastline(distance, country))`

9: **if** $start.fraction > finish.fraction$ **then**
10: $result \leftarrow$ `ST_AddPoint(substring, finish.centroid, start.centroid)`
11: **else**
12: $result \leftarrow$ `ST_AddPoint(substring, start.centroid, finish.centroid)`
13: **end if**
14: $result \leftarrow$ `ST_AsGeoJSON(result)`
15: **End**

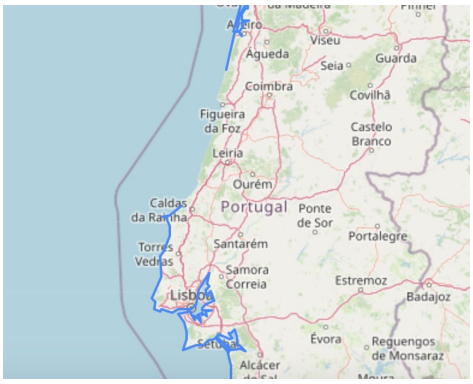

Figura 6.11: Linha costeira sem o segmento de reta extraído

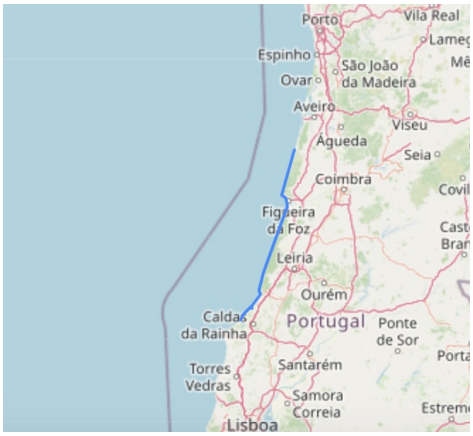

Figura 6.12: Segmento de reta extraído

Figura 6.13: Trajeto gerado entre Doca do Bom Sucesso e Porto de Recreio de Oeiras

uma variável da interface do utilizador visível na figura 6.14 com um *slider*.

A cada variação do valor de distância, a função `generate_linestring` criada na base de dados e com a lógica especificada no algoritmo 5 é executada, resultando em múltiplos cálculos da mesma rota mas com distâncias diferentes à costa.

6.2.4 Pesquisa por rotas existentes

A contínua utilização do sistema contribuirá para um aumento do número de registos e percursos presentes na base de dados. A procura por trajetos existentes resulta numa mais valia para os utilizadores em conseguirem ter uma base para a definição do seu trajeto.

Os operadores agregadores em *SQL* são comuns para casos de somatório

Figura 6.14: Distância de 1km à linha costeira

ou união de múltiplos registos. A função `ST_Union` trabalha sob um conjunto de geometrias unindo-as da qual resulta uma ou várias geometrias. O resultado esperado é de uma linha única. Contudo, a função nem sempre consegue este resultado devido à possível intersecção dos interiores não ser vazia o que origina múltiplas geometrias como exemplo uma *Multi-LineString*, que é um conjunto de linhas. Para contornar este problema, é necessário recorrer à função `ST_LineMerge`, que une os pontos mais próximo de cada linha resultando numa única (figura 6.15).

Figura 6.15: Junção de duas geometrias de cada etapa

A solução adotada assume um sub-conjunto de informações com base nas tabelas `planet_osm_polygon` e `stage`. As informações colocadas neste sub-conjunto são o nome da marina de origem e de destino, o identificador único do plano, as horas e os minutos totais do trajeto, a distância (em milhas náuticas) e o número de etapas.

Pode ser consultado o *script* que cria este sub-conjunto, no anexo A.1. O sub-conjunto tem em consideração as geometrias criadas e armazenadas na tabela *stage*.

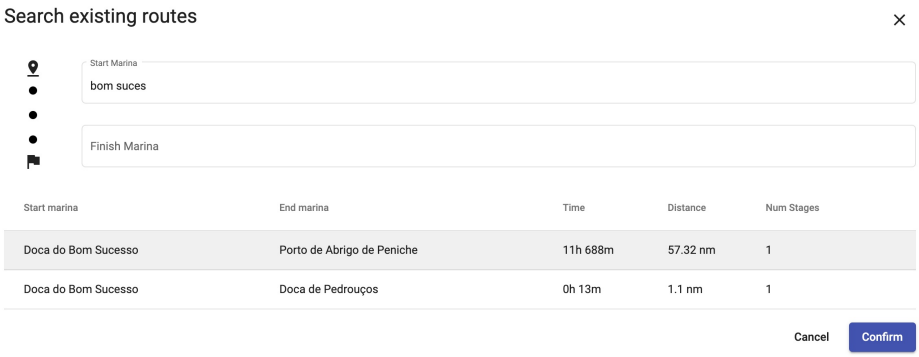

Figura 6.16: Rotas existentes

6.2.5 Extração de informação meteorológica

A extração de informação meteorológica através da API *Open-Meteo* tem múltiplos parâmetros à escolha (Figura 6.17).

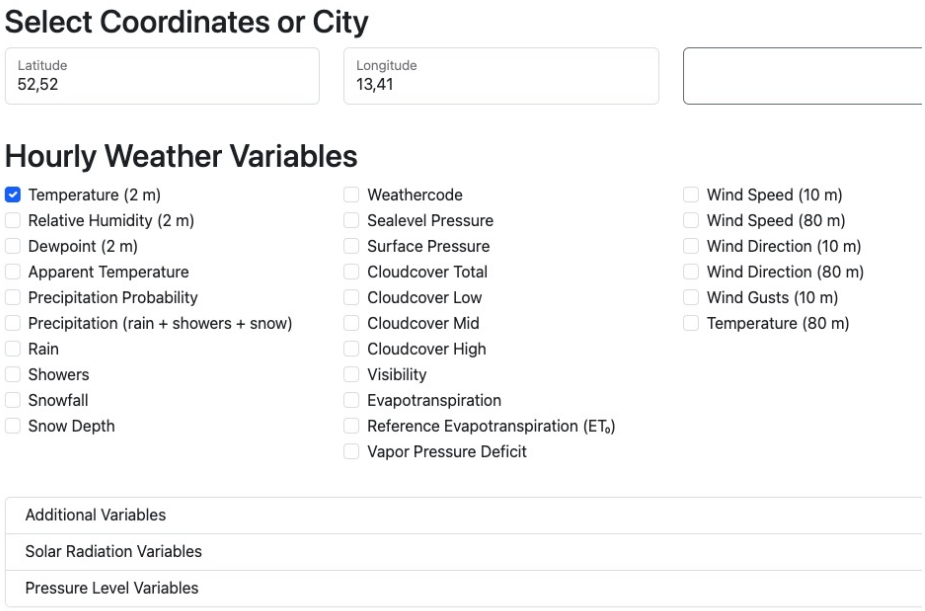

Figura 6.17: Parâmetros de informação meteorológica

Os parâmetros considerados importantes para serem associados a uma etapa, foram os seguintes:

- Temperatura a 2 m (metros acima do nível do água)

- Chuva
- Velocidade do vento a 10 m (metros acima do nível do água)
- Direção do Vento

Um exemplo do url do pedido HTTP, pode ser verificado no algoritmo 6.

Algorithm 6 Exemplo de url de pedido ao serviço

```
lat, lon, start_date, end_date ≠ ∅
url ← https://api.open-meteo.com/v1/forecast?
      latitude{lat}&longitude{lon}&
      hourly=temperature_2m,rain,windspeed_10m
      winddirection_10m&models=gfs_global&
      start_date={start_date}&end_date={end_date}
response ← make_request(url)
```

Para obter estas informações, são necessárias as parametrizações temporais e locais. De forma a agregar a informação sem perder dados, definiram-se 4 intervalos de previsões meteorológicas para serem associados a uma etapa. Para trajetos de curta duração, como o exemplo da Doca do Bom Sucesso até à Doca de Pedrouços, a obtenção dos dados pode ser encurtada em uma das docas devido à sua proximidade. A mesma abordagem não pode ser adotada para trajetos de grande duração, como o exemplo da Doca do Bom Sucesso até Porto de Abrigo de Peniche, uma vez que as condições meteorológicas podem diferir. Para ultrapassar esta dificuldade, deve proceder-se à extração das informações meteorológicas de todos os pontos geográficos pertencentes à etapa, apresentando apenas o intervalo dos valores máximo e mínimo de cada uma das variáveis.

Algorithm 7 Extração dados meteorológicos

Require: $start\_date, end\_date$

$points \leftarrow \{(x_1, y_1)..(x_n, y_n)\} \in Etapa$ ▷ X é Latitude, Y Longitude
$predictions \leftarrow \emptyset$

for $p \leftarrow (x_n, y_n)$ in $points$ **do**
    $prediction \leftarrow request(p.x, p.y, start\_date, end\_date)$
    $predictions.add(prediction)$
end for

O algoritmo 7 demonstra as operações de extração informação meteorológica realizadas e os filtros com base nos intervalos temporais (figura 6.18).

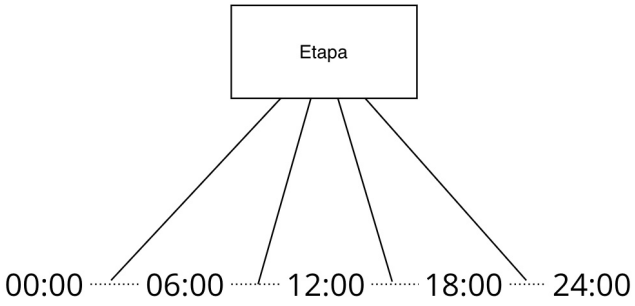

Figura 6.18: intervalos temporais

O resultado final da filtragem desta informação pode ser posteriormente analisado na figura 6.19.

Stage 1

Hour (h)	Temp. (ºC)	Wind speed (km/h)	Wind direction (º)	Rain risk (mm)
0 -6	21.8 -23.2	1.8 -4.8	11 -344	0 -0.3
6 -12	21.8 -23.2	1.8 -4.8	11 -344	0 -0.3
12 -18	21.8 -23.2	1.8 -4.8	11 -344	0 -0.3
18 -24	21.8 -23.2	1.8 -4.8	11 -344	0 -0.3

Figura 6.19: Valores de predições

A filtragem dos dados é feita agregando os dados meteorológicos de cada intervalo de tempo, realizando o cálculo do valor máximo e mínimo de cada intervalo.

6.2.6 Criação de ficheiros

No decorrer de um planeamento, os ficheiros podem ser úteis para colmatar erros do próprio sistema, aumentando a sua robustez. Durante a execução do trajeto, é prática comum a utilização de artefactos físicos, mesmo que exista auxílio tecnológico. A sumarização de todas as informações colhidas e definidas do planeamento são agrupadas num ficheiro .PDF. Numa aproximação às tecnologias mais nativas das embarcações, grande parte destas contém um GPS, com capacidade de processar ficheiros que contenham informação do trajeto.

Ficheiro .GPX

As informações vetoriais associadas à etapa podem ser convertidas para outros tipos de dados, que sejam adequados para operações mais precisas, como a criação do ficheiro GPX. Os ficheiros GPX [3] são um formato de dados estruturados em XML e utilizados para transferir dados GPS para aplicações. A biblioteca utilizada para esta conversão foi a *GeoPandas* [GeoPandas, 2023].

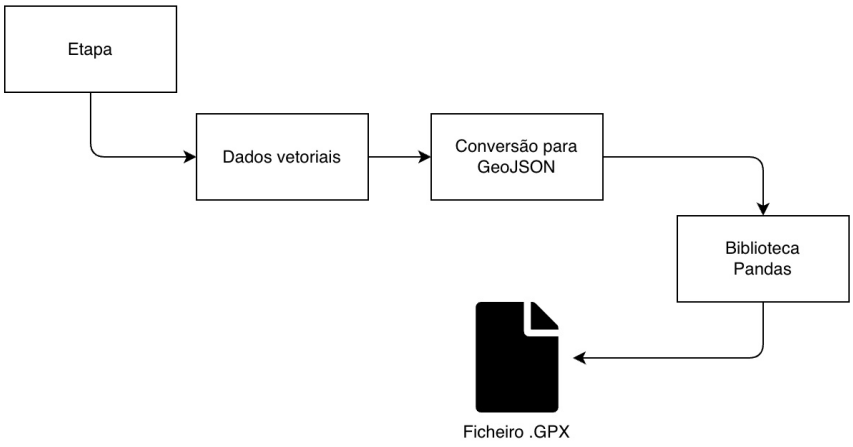

Figura 6.20: Fluxo de informação GPX

No anexo A.2, está representada a interrogação à base de dados pelos dados vetoriais com respetiva conversão para *GeoJSON*. A aplicação utilizada

[3] *GPX - GPS Exchange Format*

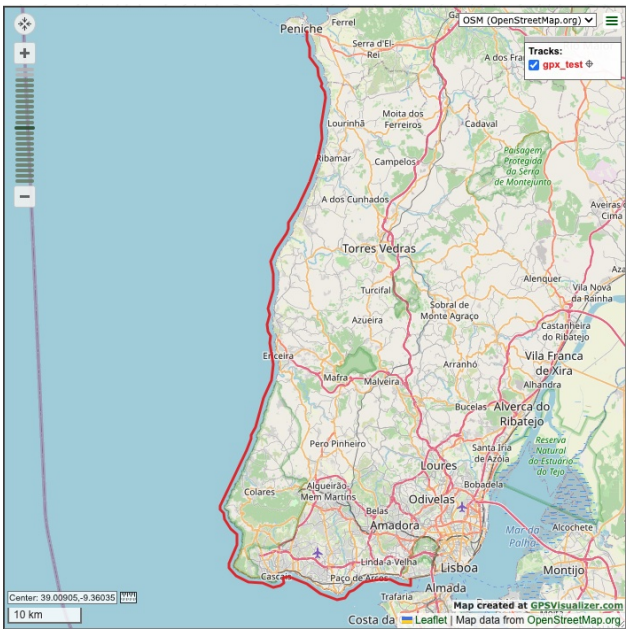

Figura 6.21: Visualização do trajeto no *GPS Visualizer*

para testar os ficheiros gerados, que efetivamente contêm a informação correta e formatada, foi o *GPS Visualizer* [Visualizer, 2022]. Ao fornecer um ficheiro, a aplicação apresenta o seu conteúdo com as mesmas tecnologias abordadas no projeto (figura 6.21).

O trajeto exemplo definido pode ser observado como uma linha vermelha, desde Lisboa até Peniche.

Ficheiro .PDF

A sumarização dos detalhes do trajeto engloba a interrogação à base de dados por dados gerais do plano não matemáticos, dados vetoriais e imagens das ajudas de navegação. As marinas e faróis não são escolhidos pela equipa que definiu o trajeto, por este tipo de interações acrescentar um nível de trabalho que poderá ser automaticamente calculado pelo sistema. Ao utilizar funções que extraem informações mais especificas, com base na geometria do percurso, pode obter-se um polígono correspondente à área que o percurso ocupa no mapa. A extração desta geometria tem como objetivo utilizar operações de intersecção, obtendo sub-conjuntos das geometrias relevantes, como descrito no algoritmo 8.

Algorithm 8 Extração das geometrias com base no trajeto

trajeto ← `st_linemerge(stages)`
bounds ← `st_orientedenvelope(trajeto)`
lighthouses ← (*bounds* ∩ `planet_osm_point`)
marinas ← (*bounds* ∩ `planet_osm_polygon`)

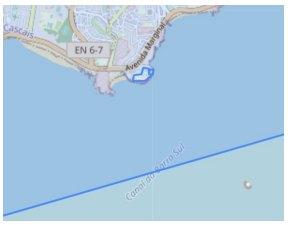

Figura 6.22: Marina dentro da geometria do trajeto

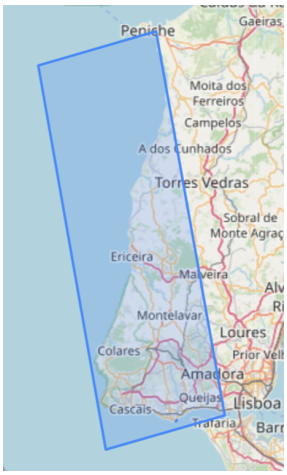

Figura 6.23: Geometria do trajeto

A função `ST_OrienteEnvelope` encarrega-se de calcular a geometria envolvente do percurso, sendo que a intersecção é realizada com a função `ST_Within` (figuras 6.22 e 6.23).

Após a criação dos conjuntos de geometrias correspondentes às marinas e aos faróis, a extração das imagens de todas estas infraestruturas é também importante para a tripulação conseguir estipular a melhor abordagem de aproximação à marina. Nos conjuntos extraídos das marinas e dos faróis, apenas se tem a informação vetorial, ou seja, as coordenadas dos pontos que constituem a geometria. Ao calcular a geometria envolvente de uma marina e as coordenadas dos respetivos pontos geográficos máximos e mínimos, obtém-se algo semelhante ao ilustrado nas figuras 6.24 e 6.25.

De acordo com a documentação do OSM [to Tile, 2022], o cálculo da secção do mundo correspondente a um ponto geográfico, está especificado no algoritmo 9.

Após o cálculo das numerações dos *tiles* com base nos pontos geográficos máximo e mínimo, a substituição manual dos processos realizados pela biblioteca é completada com base nas funções do *PostGIS*. Para se compreender

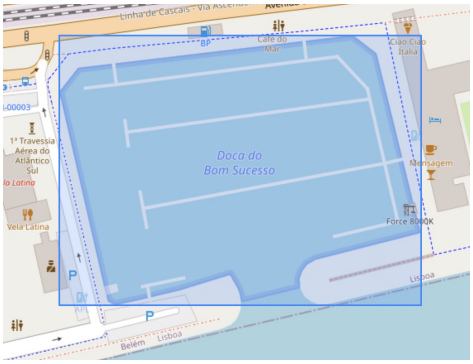

Figura 6.24: Geometria da marina Doca do Bom Sucesso

Figura 6.25: Pontos máximos e mínimos da geometria

Algorithm 9 Coordenadas para numeração *tile*

Require: $lat \neq \emptyset, long \neq \emptyset, zoom \geq 0$

$rad \leftarrow math.radians(lat)$

$n \leftarrow 2.0^{zoom}$

$x\_tile \leftarrow int((long + 180.0)/360.0 * n)$

$y\_tile \leftarrow int((1.0 - math.asinh(math.tan(rad))/\pi)/2.0 * n)$

melhor este processo manual, a imagem 6.26 ilustra as secções pertencentes da marina e respetiva área que ocupa no mapa.

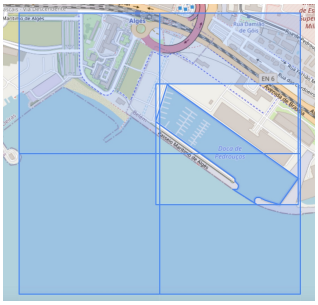

Figura 6.26: *Tiles* correspondente à marina

Observando o polígono que delimita a área correspondente da marina, verifica-se que os pontos intersectam secções diferentes. Extraindo os primeiros *tiles* correspondentes aos pontos que delimitam o máximo e o mínimo, encontram-se os seguintes *tiles* com algumas operações aritméticas (algoritmo 10).

Por fim, a extração destas imagens em separado carece de uma operação

Algorithm 10 Ciclo por secções

$zoom \leftarrow 16$
$long\_max\_tile, lat\_max\_tile \neq \emptyset$
$long\_min\_tile, lat\_min\_tile \neq \emptyset$
$long\_diff \leftarrow |long\_max\_tile - long\_min\_tile|$
$lat\_diff \leftarrow |lat\_max\_tile - lat\_min\_tile|$
for $lat \leftarrow lat\_max\_tile$ to $(lat\_max\_tile + lat\_diff)$ **do**
  for $long \leftarrow long\_max\_tile$ to $(long\_max\_tile + long\_diff)$ **do**
    $raster\_sea \leftarrow request(long, lat, zoom)$
    $raster\_map \leftarrow request(long, lat, zoom)$
  end for
end for

de união das duas formando uma única imagem (figura 6.27), realizada com a biblioteca de *Python*, *Pillow* [Pillow, 2022].

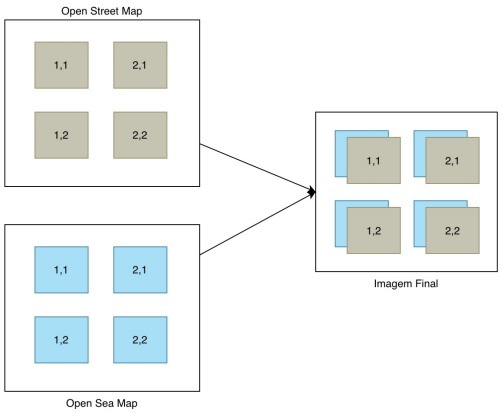

Figura 6.27: Construção da imagem

No final do ficheiro .PDF gerado, as informações apresentadas são as definidas pelos participantes da equipa, de acordo com a seguinte estrutura:

1. Detalhes não vetoriais do plano (nome, descrição, etc)
2. Detalhes da equipa (nome, descrição e elementos)
3. Detalhes dos barcos associados
4. Pontos do trajeto (distância entre cada ponto, coordenadas geográficas)
5. Ajudas à navegação

6.2.7 Modo partilhado

Na maioria das aplicações *web* desenvolvidas, para qualquer propósito, a adoção de uma arquitetura composta por 3 camadas (apresentação, negócio e dados) é suficiente para satisfazer as necessidades das funcionalidades planeadas. Neste registo, a inovação das aplicações *web* é inevitável, assim como as tecnologias que permitem uma solução mais simplificada, resolvendo problemas como a utilização de um sistema com base em dados contínuos. Os dados contínuos são informações modificadas em tempo real e apresentadas à medida que são modificadas.

A aplicações *web* são construídas com base no protocolo HTTP, que assenta na premissa de um pedido obter uma resposta. No entanto, este pedido tem de ser realizado por um interveniente do sistema. As diversas soluções possíveis, aproximam-se pela definição de temporizadores, que realizam pedidos ao fim de um intervalo de tempo ou a criação de uma interação específica, onde o utilizador consegue interagir e obter dados atualizados. Estas soluções podem causar problemas em diversos aspetos do sistema, nomeadamente na performance, na introdução de pedidos, sobrecarregando o sistema com informação irrelevante.

A crescente necessidade de ultrapassar estes problemas e necessidades, levou à criação de uma tecnologia conhecida como *WebSockets*. Segundo [Murley et al., 2021], os *WebSockets* foram desenhados para oferecer aos utilizadores uma solução com níveis de desempenho elevados e nativa, para aplicações que necessitem de informação em tempo real. A API dos *WebSockets* pretende colmatar os problemas descritos, providenciando uso reduzido de recursos para realização de pedidos, comunicação entre o cliente e servidor, reduzindo a largura de banda ocupada.

Para implementar os *WebSockets* é preciso criar uma estrutura de dados que seja enviada para o servidor e, em seguida, transmitida novamente em *broadcast* para todos os clientes ativos. A estrutura de dados pode ser verificada na figura 6.28.

O funcionamento do *WebSocket* necessita da abertura de um canal entre o cliente e o servidor, onde a comunicação pode ser realizada nos dois sentidos. Este canal pode ser entendido como uma subscrição a eventos enviados pelo servidor, semelhante ao demonstrado na figura 6.29.

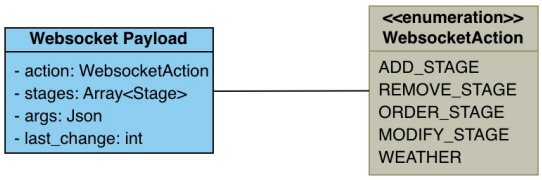

Figura 6.28: Estrutura de dados *WebSocket*

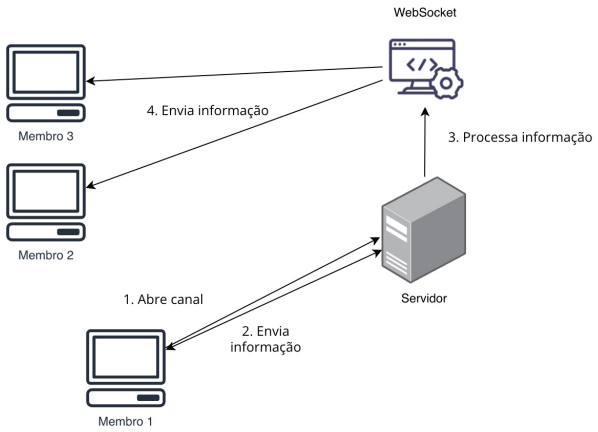

Figura 6.29: Envio de informação por *WebSocket*

6.3 Aplicação Móvel

A aplicação móvel é a última peça do sistema tornando-o mais aproximado ao utilizador. A abrangência de dispositivos, *Android* ou *iOS*, influência a capacidade de adoção do sistema por parte do utilizadores finais. O *Flutter* é uma biblioteca com um conjunto de componentes designados para constante reutilização, possíveis de serem usados em diferentes sistemas operativos e plataformas, o que permite às aplicações acederem diretamente aos seus serviços.

6.3.1 Gestão de estado

O sucesso de um *software* depende da forma como é arquitetado, ainda durante a sua fase de desenvolvimento, podendo surgir alterações. Mesmo na fase de produção, é necessário realizar operações de manutenção e provavelmente acomodação novamente de outras funcionalidades. A forma como os diversos componentes comunicam é também importante. Uma constante em qualquer aplicação com necessidade de autenticação, persistência e pesquisa de dados, é a capacidade de manter o estado durante as interações. Existem bibliotecas no gestor de dependências, que permitem a coordenação deste estado, como o *BLoC (Business Logic Component)*.

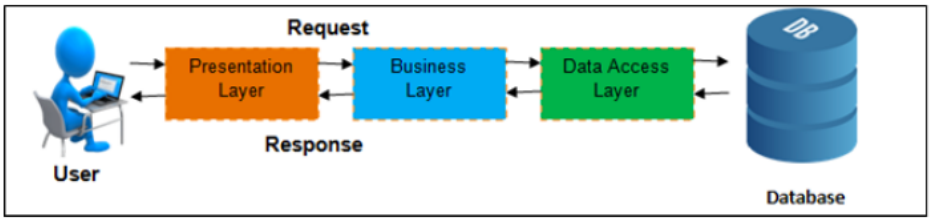

Figura 6.30: Arquitetura BLoC

Na figura 6.30, está ilustrada a arquitetura do padrão BLoC. A camada de apresentação lida apenas com a parte gráfica, isto é, com a apresentação de dados ou informação ao utilizador. É esta a camada que permite que o utilizador interaja com o *software* desenvolvido. A camada de domínio ou lógica de negócio lida com os pedidos do utilizador e produz respostas a esses pedidos. Finalmente, a camada de acesso a dados lida somente com a sua persistência. O BLoC enquadra-se no retângulo azul (Figura 6.31), o

que implica lidar com a comunicação entre a camada de apresentação e a persistência dos dados.

O conjunto de *widgets* que constituem as páginas presentes na camada de apresentação, comunicam com outras páginas através de um controlador de fluxo de informação, semelhante ao ilustrado na figura 6.31.

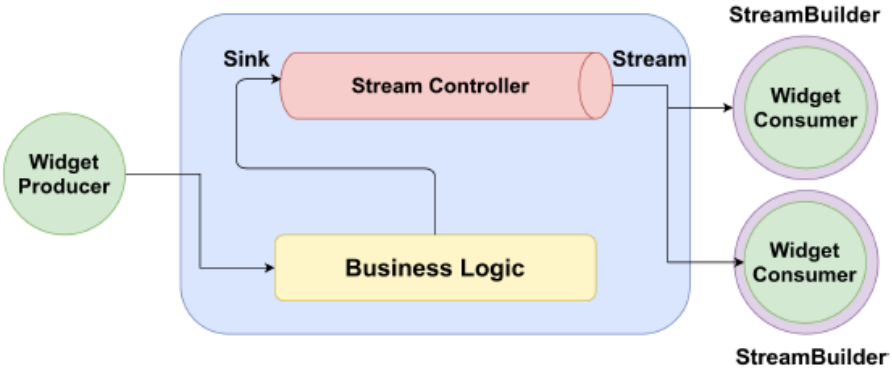

Figura 6.31: Controlador de informação

O controlador de informação (*Stream Controller*) armazena os dados que se pretende manter durante todas as interações, encontrando-se repartido por diferentes secções. Existe um controlador associado a cada ecrã da aplicação e, consequentemente, a cada caso de uso. Estes controladores são código executado no início da aplicação, não tendo qualquer valor inicial. Existem mecanismos de acesso às componentes nativas como a utilização do giroscópio, sensor de movimento, sensor de geolocalização, calendário, notificações, acesso a multimédia, entre outros. A biblioteca *shared_preferences* acede ao disco da plataforma, armazenando dados com formato chave valor, semelhante a um ficheiro *JSON*. Estes dados são a fonte de informação para inicialização dos controladores, suportando os primeiros processos da aplicação. A resolução de estado por parte das bibliotecas, contempla a necessidade de utilização na ausência de conectividade, criando mecanismos que modificam os ecrãs na camada de apresentação indicando ao utilizador as alterações do estado da ligação. Nas figuras 6.32 e 6.33 encontra-se ilustrado o fluxo de operações, que determinam a atualização deste estado com base nas informações dos sensores da plataforma e das interações do utilizador.

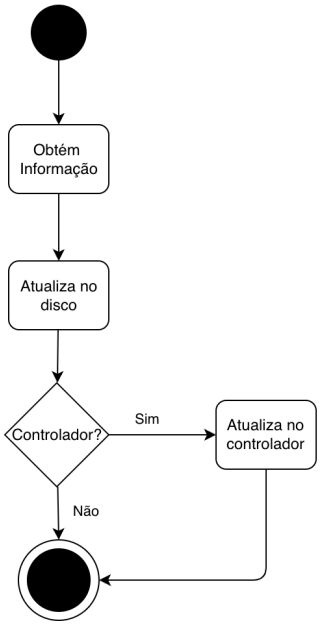

Figura 6.32: Atualização informação no disco e controladores

A apresentação do ecrã é um estado abstrato, sendo que todos os ecrãs planeados para aplicação móvel dependem da ligação à Internet, com exceção de algumas funcionalidades. A abstração representa os múltiplos ecrãs de interface gráfica com necessidade de interação do utilizador.

Observando com mais detalhe, a figura 6.33, por autenticação entende-se que o utilizador tem uma chave cifrada em memória, contendo meta-dados necessários para os pedidos ao servidor serem autenticados. A ausência desta chave provoca um erro uma vez que as informação armazenadas no disco, podem não ser as corretas. Na eventualidade de existir uma nova ligação à Internet, os pedidos serão negados e consequentemente um erro de utilização.

6.3.2 Armazenamento de informação vetorial

Nas componentes relacionadas com a ausência de ligação à Internet, é necessário atualizar o ecrã de acordo com a informação registada na componente *web*, especificamente o trajeto. Este deve ser descarregado antes da data que se prevê a ausência de ligação, armazenando essa informação no disco.

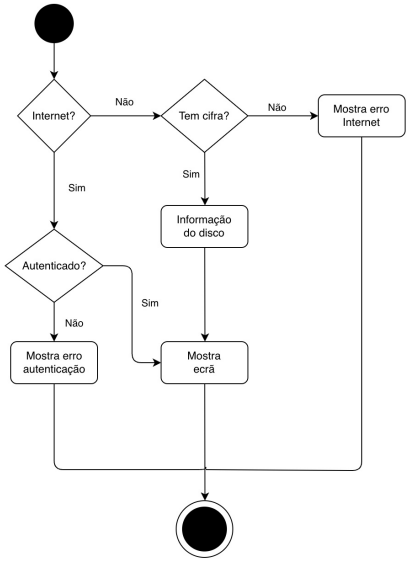

Figura 6.33: Fluxo decisões

A definição de camadas e fontes de informação vetorial e imagens provenientes do serviço OSM é igualmente utilizada na inicialização do mapa (figura 6.34).

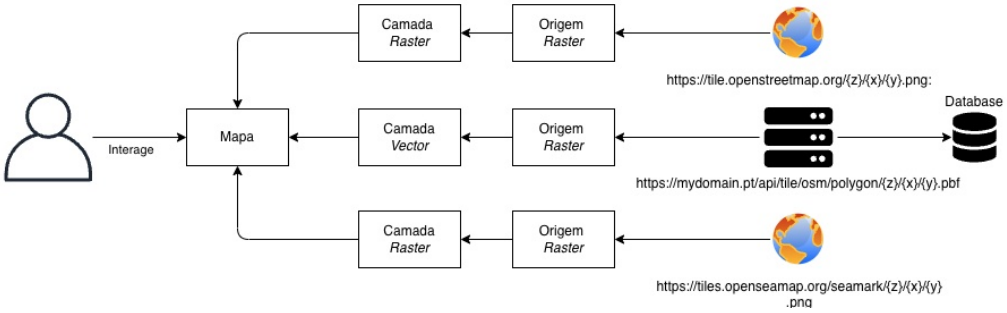

Figura 6.34: Camadas definidas no mapa

Os pedidos durante uma viagem são limitados devido à conexão sendo o aproveitamento da técnica de *cache* uma mais valia para experiência de utilizador. A biblioteca tem uma função que permite obter os dados com base numa área definida pelo utilizador de todas as fontes e camada do mapa definido para o trajeto. Adotando a mesma abordagem definida na extração de marinas na criação do ficheiro .PDF, a área a ser indicada à função a do

percurso definido (Figura 6.35).

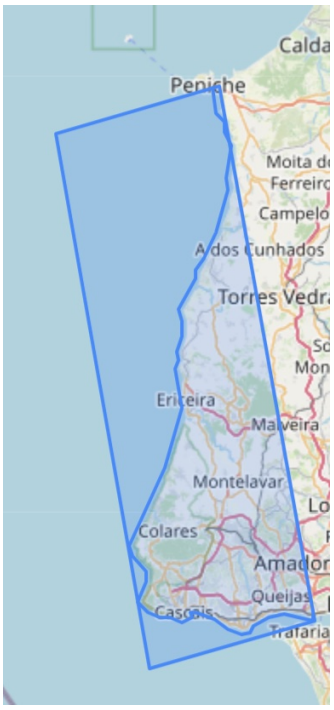

Figura 6.35: Área do percurso

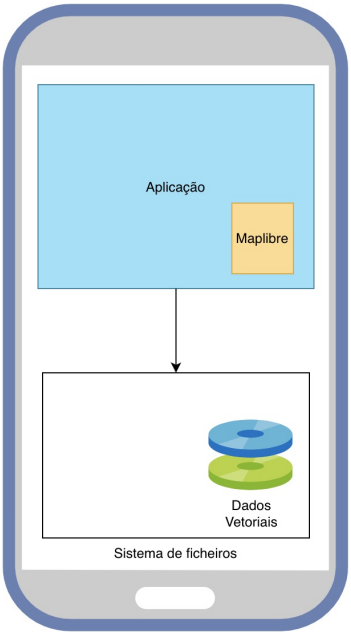

Figura 6.36: Informação no disco telemóvel

Ao fornecendo a área, a definição das camadas, respetivas fontes e níveis de *zoom* máximo e mínimo, a biblioteca armazena todos as informações no disco (Figura 6.36). Desta forma, sempre que o mapa for utilizado por qualquer utilizador e esteja configurado com as mesmas camadas e fontes, estas serão ilustradas no mapa com base na informação do disco.

Nos testes realizados na implementação desta funcionalidade, foi encontrado um problema ao qual está dependente da solução por parte dos *developers* encarregados de manter a biblioteca. Para perceber o problema, temos de entender a natureza da operação. A ação de descarregar os *tiles* é parametrizada com as respetivas fontes, camadas e níveis de zoom. É possível especificar mais do que um nível de zoom, aumentando exponencialmente a quantidade de informação armazenada. O cálculo da quantidade de *tiles* que serão descarregados, está de acordo com a seguinte fórmula:

$$tile_count = (x2 - x1 + 1) * (y1 - y2 + 1)$$

Onde $(x1, y1)$ e $(x2, y2)$ correspondem aos *tiles* do canto superior direito e

canto inferior esquerdo de uma *bounding_box*. Assim, ao realizar o produto entre os dois, obtém-se o número total para um nível de zoom. Na figura 6.37, observa-se o número de *tiles* para cada nível de zoom.

Zoom	Size	Total Size	#tiles	Total #tiles
6	0.1 kB	0.1 kB	1	1
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	223 kB	223 kB	8	9
11	479 kB	702 kB	24	33
12	641 kB	1.3 MB	56	89
13	2.0 MB	3.3 MB	216	305
14	6.0 MB	9.3 MB	848	1153
15	21 MB	30 MB	3255	4408
16	60 MB	90 MB	12 k	17 k
17	177 MB	267 MB	50 k	66 k
18	1023 MB	1.3 GB	197 k	263 k

Figura 6.37: Cálculo de *tiles*

Chegando ao nível 18, a quantidade de espaço ocupado pelos *tiles* aumenta, se for considerado a compressão das imagens não perdendo qualidade, mais memória seria ocupada. Quanto maior for a área da *bounding_box*, o número e *tiles* vai aumentar e consequentemente a memória a ser requisitada. Se um utilizador tiver múltiplos trajetos descarregados para uso em modo desconectado, vai ocupar muita memória do telemóvel e possivelmente impactar a performance da aplicação. Seria oportuno a funcionalidade de remover trajeto anteriormente descarregados que já não fossem necessários. Contudo, existe um problema com a remoção destas informações onde não é possível forçar o seu desaparecimento, como citado pelo criador da biblioteca no seguinte *link*: `https://github.com/mapbox/mapbox-gl-native/issues/7521`. Esta limitação tem impacto na experiência do utilizador, causando um erro de memória e termina a aplicação.

6.3.3 Seguimento de posição

A posição atual da equipa em relação ao trajeto é imprescindível para a realização de um percurso. Esta posição, pode ser obtida através de mecanismos provenientes da biblioteca *Maplibre* sendo que existe uma configuração para associar um comportamento lógico ao valor emitido a cada alteração. Este tipo de alterações, são eventos provenientes das componentes nativas. O evento é originado no sensor de geolocalização. A utilização destes sensores é da responsabilidade do utilizador permitir que a aplicação os possa aceder, onde a negação deste acesso resulta numa incorreta utilização da aplicação.

Ao obter esta posição e definição de comportamentos, a correlação com os pontos definidos é visível no mapa, sendo que ambos são renderizados pela biblioteca com ícones diferentes. Opcionalmente, é demonstrado um painel lateral com a sumarização de todos os pontos definidos do trajeto (figura 6.38).

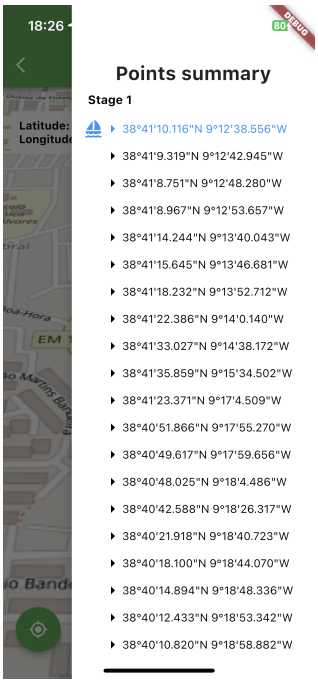

Figura 6.38: Sumarização dos pontos

Observando a figura 6.38, observa-se que um dos pontos listados se encontra sublinhado a azul, sendo que este é considerado o ponto mais próximo da posição atual do dispositivo do utilizador. Este cálculo é realizado com base

na distância entre dois pontos, seguindo a fórmula de *Haversine* [Distance, 2023], que dita o seguinte:

$$d = 2rsin^{-1}\left(\sqrt{sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + cos(\phi_1)cos(\phi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) * 6,357 \tag{6.2}$$

Onde ϕ_1 e ϕ_2 representam as latitudes dos dois pontos e λ_1 e λ_2 as longitudes. Com base neste cálculo, é alterada a cor do ponto que é renderizado na lista e também no mapa. A utilização da equação 6.2 permite ter em conta a curvatura da terra invés de uma linha reta sem qualquer tipo de variação angular.

6.4 Criação do servidor de *raster tiles*

A origem das imagens que são conhecidas como *raster*, são fornecidas por um serviço gratuito do OSM. Estas imagens são utilizadas por todos os componentes de renderização do mapa. A utilização deste serviço cria uma dependência de terceiros o que pode ser prejudicial. Estando o sistema sujeito a alterações de funcionamento de outros, aumenta o seu risco de erros. Colocando esta responsabilidade no lado do sistema, aumenta a sua resiliência para erros.

Dos métodos oferecidos para servir esta informação ao sistema, o escolhido foi o *Docker*. Na arquitetura (no capítulo 5) está presente a utilização deste contentor.

Na figura 6.39, estão ilustradas as tecnologias que estão a ser utilizadas pela imagem *Docker*. São fornecidos parâmetros à imagem para que individualmente execute os comandos para importar os dados sozinho e em seguida fornecer estes mesmos dados de acordo com os pedidos do cliente do sistema. Estas imagens são guardadas numa posição de memória que está associada ao contentor, acedendo as estas imagens sempre que necessário.

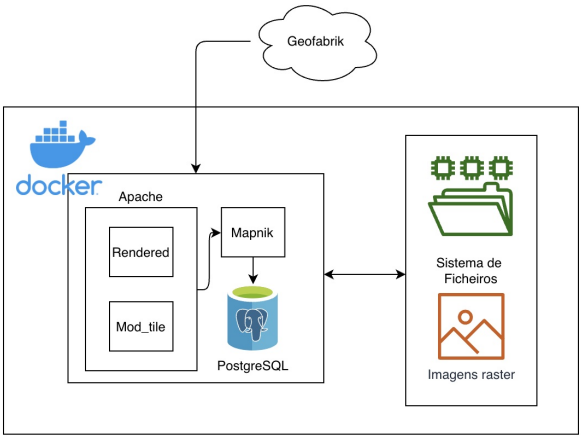

Figura 6.39: Imagem *Docker*

Capítulo 7

Validação e Testes

Com base no sistema desenvolvido, foram delineados diferentes casos de teste com objetivo de aferir o comportamento no decorrer das ações. Os casos definidos pretendem replicar situações que se consideram reais e que se relacionam diretamente com os casos de uso especificados no capítulo 5. A execução destes casos de teste não teve um conjunto de utilizadores para responder a questionários de satisfação sobre critérios de avaliação de *User Experience* e *Usability*. Estes resultados devem ser interpretados como funcionamento esperado considerado normal mas não ótimo, ou seja, que não foi validado por utilizadores finais. No decorrer deste capítulo serão apresentados os resultados dos diferentes casos de teste, em que o foco será a comparação do resultado obtido com o resultado esperado.

Resultados

Os resultados são provenientes de um conjunto de casos de teste que incidem em diferentes funcionalidades. Esta abordagem deve-se à grande quantidade de passos necessários até ao planeamento de um trajeto ocorrer. Posto isto, será organizado em sub-secções com os respetivos resultados obtidos, onde apenas o último caso é realizado na aplicação móvel e os restantes na aplicação web. Numa perspetiva menos detalhada, a experiência pretende simular uma interação de um utilizador com o sistema pela primeira vez e que este consiga planear um trajeto desde o Terminal Fluvial do Seixal até ao Terminal Fluvial do Cais do Sodré.

Registo no sistema e criação de uma equipa

Na primeira ação registo no sistema acedeu-se à página inicial, introduzindo as informações pedidas para o registo (Figura 7.1):

- *Username* - rfgsantos
- *First name* - Rui
- *Last name* - Santos
- *Email* - ruis@gmail.com
- *Password* - Adminadmin

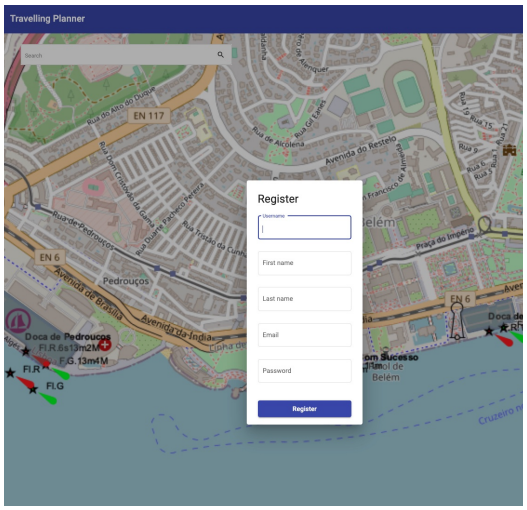

Figura 7.1: Registo no sistema

O registo foi concluído com sucesso tendo o sido adicionado na base de dados corretamente o tuplo. Criado o utilizador, procedeu-se à autenticação no sistema sendo um pré-requisito para aceder às restantes funcionalidades. A autenticação é realizada de acordo com as informações introduzidas no registo, sendo a mesma também realizada com sucesso. Nas Figuras 7.2 e 7.3 observa-se a introdução e extração das informações do utilizador.

O acesso a outras secções do sistema são realizadas através do menu lateral, onde estão presentes hiperligações para as diferentes áreas. Acedendo à hiperligação de "Equipas" é possível visualizar as equipas a que o utilizador

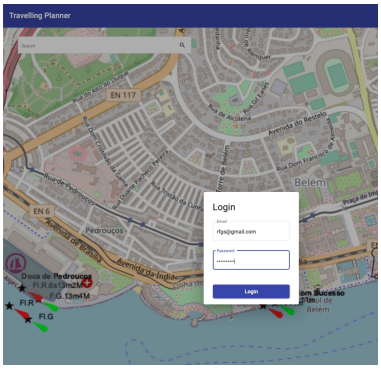

Figura 7.2: Autenticação

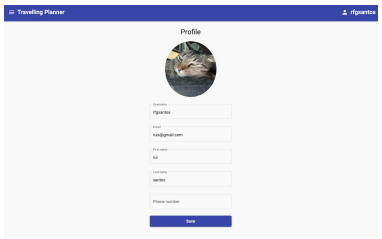

Figura 7.3: Perfil

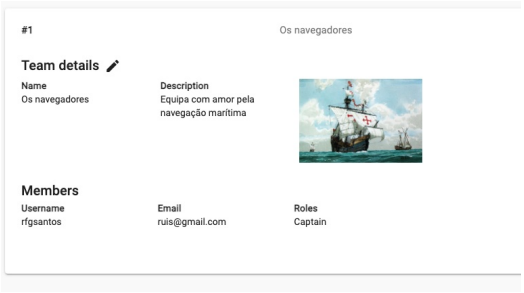

Figura 7.4: Equipa

pertence e criar novas. A lista de equipas deve estar vazia por se tratar de um utilizador novo sem qualquer relação com outras entidades do sistema. Ao criar uma equipa, introduziu-se as seguintes informações:

- **Name** - Os navegadores
- **Description** - Equipa com amor pela navegação maritima
- **Members** - Membro único, o próprio utilizador (adicionado automaticamente pelo sistema)

No fim da introdução das informações, foi criada a equipa e em seguida adicionada uma imagem (Figura 7.4). O resultado obtido está de acordo com o esperado no A.1.1.

Inserir um barco

A embarcação a ser utilizada nos caso de testes é um catamarã que pertence à frota da empresa TTSL. Para inserir uma embarcação no sistema é necessária a respetiva ficha técnica, que pode ser encontrada na página oficial da empresa [TTSL, 2023].

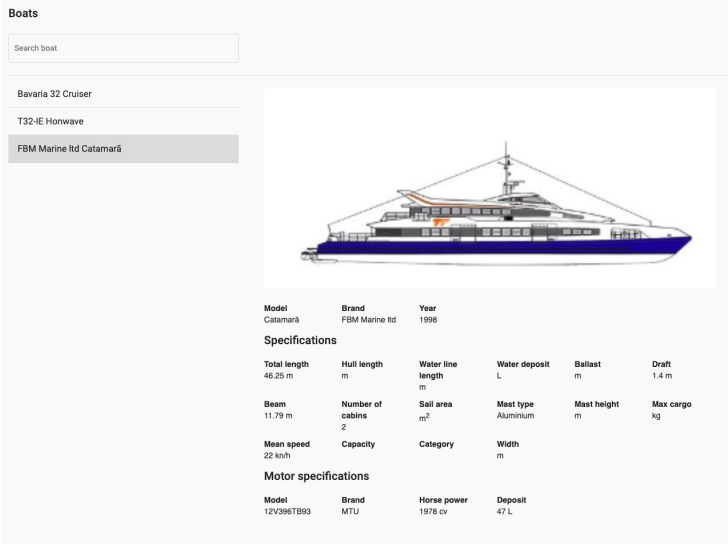

Figura 7.5: Catamarã

A especificações disponibilizadas são reduzidas em relação a outras que se pode encontrar de outros barcos. No entanto, as que se consegue extrair são importantes para o planeamento, tais como a velocidade média, respetiva marca e modelo, tipo de embarcação e o seu motor. Estas características vão ser utilizadas no cálculo do tempo necessário para realizar o percurso. Os dados inseridos foram os seguintes:

- **Model** - Catamarã
- **Brand** - FBM Marine ltd
- **Total length** - 46.25m
- **Beam** - 11.79m
- **Draft** - 1.4m

- **Mean speed** - 22kn/h
- **Motor specifications** - 12V396TB93 MTU 1978cv 47L

Após a inserção dos dados da embarcação (Figura 7.5), verificou-se que foi persistido no sistema com sucesso e possível de ser procurado futuramente. O resultado obtido está de acordo com o resultado esperado no caso de uso A.1.2.

Criação de um plano

A criação de um plano é feita através de 4 passos, especificações gerais (datas, nome e descrição), escolha da equipa, escolha da embarcação e definição dos pontos. Esta secção foca-se apenas nos primeiros 3 passos, sendo estes preliminares ao acesso das funcionalidades que que permitem a definição de pontos. Nas figuras 7.6, 7.7 e 7.8 estão presentes os passos necessários até ser possível definir os pontos.

Das informações introduzidas, são realçadas as seguintes:

- **Name** - Travessia Seixal - Cais do Sodré
- **Description** - Travessia experimental do Seixal até ao Cais do Sodré
- **Start date** - 08/09/2023
- **End date** - 08/09/2023
- **Team** - Os navegadores
- **Boat** - FBM Marine ltd Catamarã

A embarcação e equipa escolhida estão de acordo com os casos de teste anteriores 7 e 7. Revisitando a página principal dos planos, é possível aceder novamente ao plano introduzido, concluindo assim que o resultado obtido está de acordo com o resultado esperado no caso de uso A.1.3.

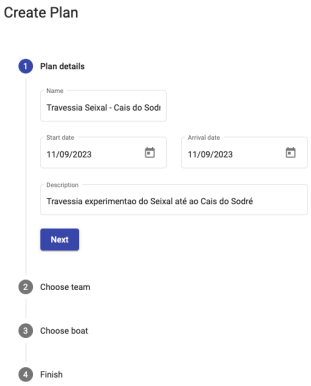

Figura 7.6: Informação geral

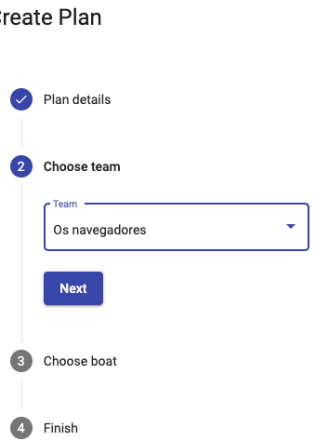

Figura 7.7: Escolha de equipa

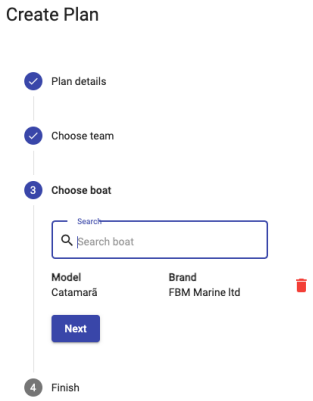

Figura 7.8: Escolha de embarcação

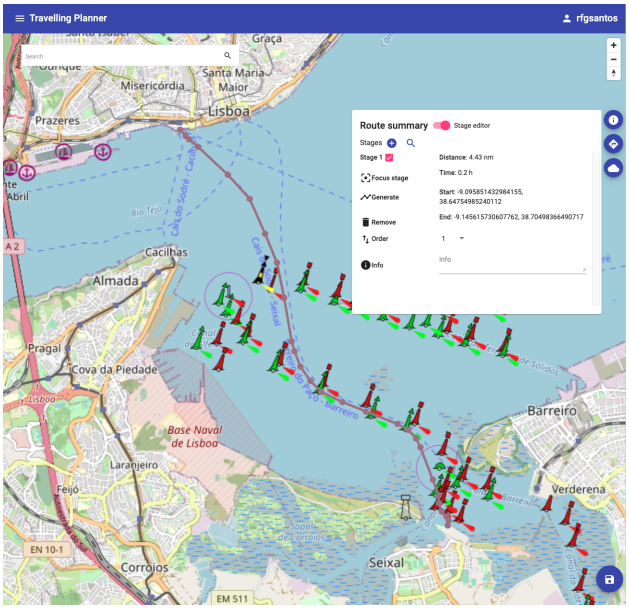

Figura 7.9: Trajeto Seixal - Cais do Sodré

Definição de pontos do trajeto

Acedendo ao ecrã inicial do planeamento, existem menus laterais que permitem a configuração do trajeto. Estas configurações variam entre modos de edição ou apenas de visualização. Ativando o modo de edição é possível começar a definir pontos do trajeto. Os pontos definidos podem ser visualizados na figura 7.9.

No menu lateral onde estão as informações sumarizadas da rota definida, que são especificamente a distância em milhas náuticas - 4.43 *Nautical Miles*, o tempo necessário para completar o trajeto - 0.20 Horas que equivale a 12 minutos e as coordenadas geográficas do ponto de inicial e o final, que correspondem aos centroides do Terminal Fluvial do Seixal (-9.095851, 38.647549) e do Terminal Fluvial do Cais do Sodré (-9.145616, 38.704983). Na base de dados do sistema foi criada a linha ilustrada na figura 7.10. O trajeto é pequeno e por isso apenas foi criada uma etapa.

De acordo com os resultados obtidos e comparando com o resultado esperado (caso de uso A.1.4) é alcançado o objetivo final em que o trajeto está traçado e persistido na base de dados corretamente.

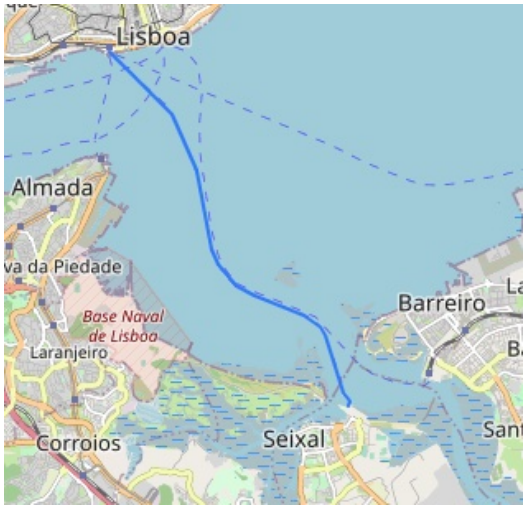

Figura 7.10: Trajeto na base de dados

Execução do trajeto

O trajeto planeado é considerado apenas uma experiência parcial ao que se pretende com uso final, ou seja, a travessia do Seixal ao Cais do Sodré, não foi realizada num barco pessoal mas com recurso a transportes públicos. No entanto, o teste conseguiu fornecer informações valiosas no contexto de perceber se os cálculos realizados de acordo com a embarcação estão corretos ou aproximados.

Comparando os diferentes usos da aplicação, num contexto de transporte público, os pontos definidos no trajeto podem nem sempre ser cumpridos uma vez que não é o próprio utilizador da embarcação a velejar. Contudo, com auxílio dos *raster tiles* do *OpenSeaMap*, existe informação sobre o trajeto habitual da ligação entre os dois terminais fluviais, tendo os pontos sido definidos com base neste trajeto.

O inicio do trajeto deu-se aproximadamente às 17:07 do dia 8 de setembro de 2023 como demonstrado na figura 7.11, onde tinham decorrido apenas 1 minuto e 41 segundos desde o começo do trajeto.

Na figura 7.12 estão presentes todos os pontos definidos do trajeto. O ícone perto em um do pontos está a ilustrar a proximidade da localização atual de acordo com os pontos. Na figura 7.11 observa-se a posição atual com as respetivas coordenadas por extenso.

No decorrer do percurso foi verificada uma boa capacidade do dispositivo

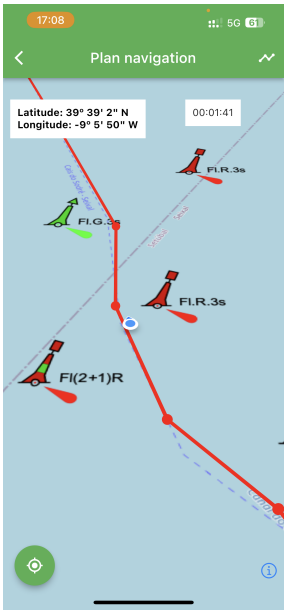

Figura 7.11: Inicio do percurso

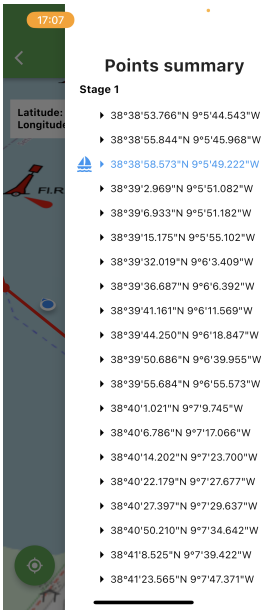

Figura 7.12: Pontos do percurso

e da aplicação móvel em conseguir obter a posição atual, também ilustrada nas figuras 7.13 e 7.14 onde na última está presente uma fotografia da boia de sinalização.

Chegando ao final do percurso, foi indicado ao sistema que este tinha terminado e assim registado a hora e data. Verificando as figuras 7.15, 7.16 e 7.17.

Atendendo ao valores de inicio e fim do tempo decorrido no trajeto, observa-se a duração de aproximadamente 13 minutos e 47 segundos. O desvio da previsão inicial pode ser considerado curto uma vez que estava previsto ter uma duração de 12 minutos e o diferencial entre os valores é de 1 minuto e 47 segundos. Esta diferença temporal pode ser entendida como um desvio do percurso face ao planeado, tendo em conta que a embarcação era conduzida por pessoas não pertencentes ao plano. O facto de não ter sido um participante do plano a conduzir a embarcação, não invalida que o plano não fosse alvo de alterações no tempo decorrido. O mesmo tipo de situações poderiam ocorrer por fatores externos, como por exemplo, alterações marítimas que implicassem manobras de desvio do trajeto planeado. Concluindo que o resultado esperado no caso de uso A.1.5 foi alcançado com sucesso.

Figura 7.13: Boia de sinalização no mapa

Figura 7.14: Fotografia da boia

Figura 7.15: Trajeto terminado no Terminal Fluvial do Cais do Sodré

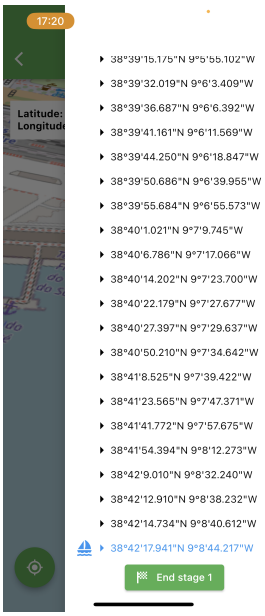

Figura 7.16: Ponto final do trajeto

end_time timestamp with time zone	start_time timestamp with time zone
2023-09-08 17:21:09.334336+01	2023-09-08 17:07:22.406525+01

Figura 7.17: Tempo do trajeto

Capítulo 8

Conclusões e Trabalho Futuro

Neste projeto foi desenvolvida uma ferramenta de auxilio ao planeamento de trajetos de viagem em veleiro. Numa análise preliminar do problema, chegou-se à conclusão que o sistema final poderia incidir em 3 diferentes áreas de interesse:

- Logística;
- Navegação;
- Mineração de dados;

O projeto desenvolvido teve um foco maior nos elementos relacionados com a navegação.

Na investigação de como criar e manipular informação vetorial foram encontradas dificuldades em como a disponibilizar graficamente. As bibliotecas capazes de criar ilustrações e processamento computacional gráfico, não continham informação especifica em relação à estruturação dos ficheiros a serem utilizados. As bibliotecas, que maioritariamente se destinam a *web browsers*, estão apenas focados na computação gráfica das informações que recebem de forma a conseguirem aplicar eventos com base nas ações do utilizador. Estes eventos trabalham sobre a informação com diferentes formatos do gerado inicialmente pelo servidor. A algoritmia que um servidor deve implementar para criar um ficheiro capaz de conter informação relevante e requisitada por qualquer cliente é pouco explorada por parte da comunidade. Grande parte das questões feitas à comunidade são muito específicas, sendo a procura por

soluções complicada e consequentemente os métodos mais aplicados requerem experimentar diferentes valores para funções.

A utilização de serviços externos (*Open Sea Map*, *Open Street Map*, *Open Meteo*)para a criação deste sistema foi crucial uma vez que sem estes não teria sido possível a visualização das ajudas de navegação. A decisão criar uma componente dedicada apenas à renderização de *raster tiles* do mundo teve como objetivo retirar esta dependência de serviços externos. Por ser uma componente que exigia muito tempo de descoberta e teste, foi movida para trabalho futuro.

Os testes realizados conseguiram demonstrar que o sistema tem capacidade de guardar e manipular a informação vetorial do planeamento.

Trabalho futuro

Nesta área de estudo, a exploração de funcionalidades que tenham um impacto positivo na experiência de utilizador é vasto. Tomando o exemplo das aplicações Google Maps e Waze, apesar de se focarem apenas na navegação terrestre, a sua evolução ao longos dos ano foi notória. A capacidade de um sistema se conseguir moldar à necessidades de um utilizador, é um fator de peso sendo capaz de influenciar a decisão de escolha de aplicações. Na sua essência, esta aplicações contêm a mesma informação vetorial que sempre tiveram, não contemplando alterações das estruturas. Os trabalhos futuros especificados nesta secção estão diretamente relacionados com a evolução da aplicação focando-se na experiência do utilizador.

A componente de inteligência artificial e mineração de dados, inicialmente discutida no inicio do projeto, poderá a vir ser explorada no futuro. No entanto, sendo um ponto relativamente extenso e dependente de dados existentes, foi excluído. Os trajetos armazenados no sistema contêm informações relevantes do planeamento, como por exemplo, os dados meteorológicos.

A funcionalidade de gerar trajetos com base na linha costeira do país é limitada. A linha é gerada com base na especificidade da informação vetorial o que pode originar a muito pontos ou poucos pontos. Esta oscilação de pontos é entendida como qualidade de informação, ou seja, se está o mais idêntico possível á linha costeira real ou não. O utilizador pode mover os pontos do trajeto, mas não os consegue eliminar ou adicionar pontos, sendo

este um ponto a favor da experiência de utilizador em falta.

Com o decorrer da utilização contínua do sistema, será criada um banco de dados de múltiplos trajetos. Estes trajetos terão associados aos mesmos, docas e informações gerais (tempo, distância, meteorologia). Futuramente, poderiam ser cruzados múltiplos trajetos de diferentes equipas e com base na biblioteca *pgRouting* [pgRouting, 2023], aplicar algoritmos de inteligência artificial e gerar os melhores trajetos para o destino final.

Independência de serviços externos

A criação de camadas capazes de visualizar as ajudas à navegação, elevariam o sistema para um patamar de independência de serviços externos. Esta independência não reflete a necessidade de atualização dos dados, mas das aplicações a serem executadas como serviços *web* prontos para serem consumidos livremente por qualquer utilizador. Explorando a biblioteca *Mapnik* e o respetivo contentor *Docker* que aloja a lógica de renderização dos *raster tiles*, seria uma possibilidade de modificar o contentor original para renderizar esta nova camada.

Apêndice A

Anexos

A.1 Casos de Uso

A.1.1 Registo no sistema e criação de uma equipa

Pré-condições Nenhuma

Descrição Pretende-se que um utilizador se registe no sistema e crie uma equipa.

Resultado esperado Um utilizador consiga autenticar-se no sistema após o seu registo e a criação da sua equipa fique persistida no sistema.

A.1.2 Inserir um barco

Pré-condições O utilizador estar registado e autenticado no sistema.

Descrição Pretende-se que um utilizador insira uma embarcação no sistema.

Resultado esperado O utilizador crie uma embarcação no sistema e persista.

A.1.3 Criação de um plano

Pré-condições O utilizador estar registado e autenticado no sistema e associado a uma equipa.

Descrição Pretende-se que um utilizador crie um plano.

Resultado esperado O utilizador crie um plano e consiga persistir o seu registo na base de dados do sistema.

A.1.4 Definição de pontos do trajeto

Pré-condições Existência de um plano criado e que o utilizador pertença à equipa desse plano.

Descrição Pretende-se que um utilizador consiga definir os pontos de um trajeto e visualizar respetivas informações.

Resultado esperado O trajeto é persistido sistema e possível de ser acedido na componente móvel para inicio.

A.1.5 Execução do trajeto

Pré-condições Existência de um plano criado trajeto definido.

Descrição Pretende-se que um utilizador consiga executar o plano definido através da componente móvel.

Resultado esperado O trajeto é realizado com sucesso com auxilio da componente móvel, sendo possível de verificar a proximidade com os pontos definidos e tempo decorrido no trajeto.

A.2 Tabelas de requisitos

Requisito	Função	Plataforma
R7.1	Sumarização das informações de um planeamento num documento gerado automaticamente	*Web* e Aplicação
R7.2	Adicionar informação relevante respetiva a uma doca	*Web* e Aplicação
R7.3	Visualização das ajudas à navegação durante o percurso	Aplicação
R7.4	Descarregar os dados geográficos relativos ás etapas de um planeamento para uso *offline*	Aplicação

Tabela A.1: Requisitos funcionais do categoria Informações Gerais

Requisito	Função	Plataforma
R6.1	Criação de rotas semi-automáticas com doca origem a doca destino	*Web*
R6.2	Alteração da distância da linha costeira	*Web*
R6.3	Procura por rotas existentes de outras equipas	*Web*
R6.5	Reutilização de rotas previamente realizadas e completadas	*Web*

Tabela A.2: Requisitos funcionais do categoria Roteamento

Requisito	Função	Plataforma
R5.1	Registo de um utilizador no sistema	*Web* e Aplicação
R5.2	Autenticação de um utilizador no sistema	*Web* e Aplicação
R5.3	Alteração de dados pessoais do utilizador	*Web* e Aplicação
R5.4	Associar uma fotografia de perfil	*Web* e Aplicação
R5.5	Modificar a fotografia de perfil	*Web* e Aplicação
R5.6	O utilizador registado é o mesmo nas duas plataformas	*Web* e Aplicação
R5.7	O utilizador registado deve conseguir alterar a sua password	*Web* e Aplicação

Tabela A.3: Requisitos funcionais do categoria Autenticação, Registo e Perfil

Requisito	Função	Plataforma
R4.1	Visualização em tempo real das alterações de coordenadas dos pontos geográficos	*Web*
R4.2	Visualização em tempo real das remoções de etapas de navegação	*Web*
R4.3	Visualização da alteração da distância de navegação	*Web*
R4.4	Visualização da alteração de datas do trajeto	*Web*
R4.5	Visualização da alteração de informação meteorológica do trajeto	*Web*
R4.6	Visualização do autor das alterações	*Web*

Tabela A.4: Requisitos funcionais do categoria modo partilhado

Requisito	Função	Plataforma
R3.1	Criação de um plano	*Web*
R3.2	Edição de um plano	*Web*
R3.3	Remoção de um plano	*Web*
R3.4	Atualização de um plano	*Web*
R3.5	Adição de pontos geográficos a um plano	*Web*
R3.6	Remoção da totalidade dos pontos geográficos a um plano	*Web*
R3.7	Procura de dados meteorológicos associados ao plano	*Web*
R3.8	Alteração de datas do plano	*Web*
R3.9	Movimentação de pontos geográficos individuais	*Web*
R3.10	Visualização do plano	*Web* e Aplicação
R3.11	Procura por nomes de docas	*Web* e Aplicação
R3.12	Criação de etapas de navegação	*Web*
R3.13	Possibilidade de focar uma etapa em especifico	*Web*
R3.14	Alteração da ordem das etapas de navegação	*Web*
R3.15	Visualização dos pontos geográficos pertencentes ao trajeto	*Web* e Aplicação
R3.16	Listagem dos planos existentes	*Web* e Aplicação
R3.17	Associação de uma embarcação existente a um plano	*Web*

Tabela A.5: Requisitos funcionais do categoria planeamento

Requisito	Função	Plataforma
R2.1	Adicionar uma embarcação no sistema	*Web*
R2.2	Adicionar especificação técnica de uma embarcação	*Web*
R2.3	Definir tipo de embarcação	*Web*
R2.4	Adicionar foto da embarcação	*Web*
R2.5	Procurar por embarcação	*Web*

Tabela A.6: Requisitos funcionais do categoria embarcações

Requisito	Função	Plataforma
R1.1	Criação de uma equipa	*Web* e Aplicação
R1.2	Edição de uma equipa	*Web* e Aplicação
R1.3	Eliminação de uma equipa	*Web* e Aplicação
R1.4	Atualização de uma equipa	*Web* e Aplicação
R1.5	Adicionar utilizadores registados à equipa	*Web* e Aplicação
R1.6	Atribuição de funções a elementos da equipa	*Web* e Aplicação
R1.7	Remover elementos de uma equipa	*Web* e Aplicação
R1.8	Alteração da imagem da equipa	*Web* e Aplicação
R1.9	Listagem das equipas	*Web* e Aplicação

Tabela A.7: Requisitos funcionais do categoria equipa

A.3 *Scripts SQL*

```
CREATE OR REPLACE VIEW existing_routes AS
SELECT ts.name          AS start_marina,
       te.name          AS end_marina,
       route.plan_id    AS plan_id,
       route.minutes    AS minutes,
       route.hours      AS hours,
       route.distance   AS distance,
       route.num_stages AS num_stages
FROM (SELECT p.id                            AS plan_id,
             COUNT(*)                        AS num_stages,
             SUM(s.hours)                    AS hours,
             SUM(s.minutes)                  AS minutes,
             SUM(s.distance)                 AS distance,
             ST_LineMerge(st_union(s.geom)) AS geom
      FROM stage AS s
               INNER JOIN plan AS p ON p.id = s.plan_id
      GROUP BY p.id) AS route
         INNER JOIN planet_osm_polygon AS ts
         ON ST_Within(ST_Startpoint(route.geom), ts.way)
         AND ts.leisure = 'marina'
         INNER JOIN planet_osm_polygon AS te
         ON ST_Within(ST_Endpoint(route.geom), te.way)
         AND te.leisure = 'marina'
```

Figura A.1: *SQL script* da criação das *View*

```
SELECT ST_AsGeoJSON(ST_LineMerge(ST_Union(s.geom)))::jsonb
FROM stage AS s
WHERE plan_id = '{plan_id}';
```

Figura A.2: Conversão para estrutura de dados GeoJSON

```
CREATE OR REPLACE function generate_linestring(
    start_osm_id integer,
    finish_osm_id integer,
    distance float,
    country varchar
)

DECLARE
    start     record;
    finish    record;
    substring geometry;
    result    json;
BEGIN
    SELECT *
    INTO finish
    FROM closest_points_func($3, $4) as cpm
    WHERE cpm.osm_id = $1;

    SELECT *
    INTO start
    FROM closest_points_func($3, $4) as cpm
    WHERE cpm.osm_id = $2;

    SELECT ST_LineSubstring(ST_LineMerge(p.geometry),
                            least(start.fraction, finish.fraction),
                            greatest(finish.fraction, start.fraction))
    INTO substring
    FROM country_coastline($3, $4) as p;

    if start.fraction > finish.fraction then
        SELECT ST_AsGeoJson(
            ST_AddPoint(
                ST_AddPoint(substring, finish.centroid, 0),
                start.centroid))
        INTO result;
    else
        SELECT ST_AsGeoJson(
                ST_AddPoint(
                    ST_AddPoint(substring, start.centroid, 0),
                    finish.centroid))
        INTO result;
    end if;
    return result;
END;
```

Figura A.3: *SQL script* da função final de criação de rotas semi-automáticas

Bibliografia

[Bärlocher, 2013] Bärlocher, M. (2013). Openseamap - the free nautical chart. *Hydro International*, p. 28–33.

[Burk, 1999] Burk, S. S. R. R. V. S. C. T. E. (1999). Spatial pictogram enhanced conceptual data models and their translation to logical data models.

[C-MAP, 2023] C-MAP (2023). C-map application. `https://www.c-map.com/app/`.

[Conceição, 2018] Conceição, V. (2018). *Designing for Safe Maritime Navigation Studying Control Processes for Bridge Teams*. PhD thesis.

[Distance, 2023] Distance (2023). Distance calculator. `https://www.calculator.net/distance-calculator.html`.

[ECMWF, 2023] ECMWF (2023). Database of weather data. `https://www.ecmwf.int/en/about`.

[Eurostat, 2023] Eurostat (2023). Eurostat. `https://ec.europa.eu/eurostat/web/main/about-us/who-we-are`.

[FastAPI, 2023] FastAPI (2023). Micro framework for rest application development in python. `https://fastapi.tiangolo.com/`.

[Flutter, 2023] Flutter (2023). Open source library for cross-platform application development. `https://flutter.dev/`.

[Geofabrik, 2023a] Geofabrik (2023a). Geofabrik home website. `https://www.geofabrik.de/`.

[Geofabrik, 2023b] Geofabrik, E. E. S. (2023b). Europe. https://download.geofabrik.de/europe/portugal-latest-free.shp.zip.

[GeoPandas, 2023] GeoPandas (2023). Python library to handle lgeospatial data made easy. https://geopandas.org/en/stable/.

[GFS, 2023] GFS (2023). Noaa gfs. https://www.ncei.noaa.gov/products/weather-climate-models/global-forecast.

[John T. Sample, 2010] John T. Sample, E. L. (2010). *Tile-Based Geospatial Information Systems*. Springer.

[Mapbox, 2023] Mapbox (2023). Library render map tiles. https://www.mapbox.com/.

[Moody, 1950] Moody, A. B. (1950). The nautical mile. *The International Hydrographic Review*.

[Murley et al., 2021] Murley, P., Ma, Z., Mason, J., Bailey, M., e Kharraz, A. (2021). Websocket adoption and the landscape of the real-time web. p. 1192–1203.

[OpenMeteo, 2023] OpenMeteo (2023). Free to use weather api. https://open-meteo.com/en/about.

[OpenStreetMap, 2023] OpenStreetMap (2023). Openstreetmap system application. https://wiki.openstreetmap.org/wiki/Main_Page.

[Osm2pgsql, 2023] Osm2pgsql (2023). Library to import osm data. https://osm2pgsql.org/.

[pgRouting, 2023] pgRouting (2023). pgrouting project. https://pgrouting.org/.

[Pillow, 2022] Pillow (2022). Pillow. https://pillow.readthedocs.io/en/stable/.

[PostGIS, 2023] PostGIS (2023). Postgis psc osgeo. https://postgis.net/.

[Silveira et al., 2011] Silveira, P., Teixeira, A., e Guedes Soares, C. (2011). Analysis of maritime traffic off the coast of portugal.

[Switch2OSM, 2023] Switch2OSM (2023). Switch2osm. https://switch2osm.org/.

[to Tile, 2022] to Tile, L. (2022). Latlon to tile. https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames#Tile_numbers_to_lon..2Flat._2/.

[TTSL, 2023] TTSL (2023). Ttsl. https://ttsl.pt/terminais-e-frota/frota/aroeira-carnide-se-e-sao-juliao/.

[Vis, 2018] Vis, M. (2018). History of mercator projection.

[Visualizer, 2022] Visualizer, G. (2022). Gps visualizer. https://www.gpsvisualizer.com/.