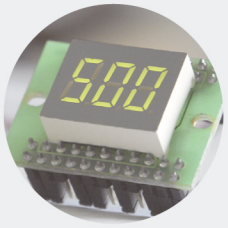
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






GeocarEyes ? Canal de comunicação para a gestão de frotas

HENRIQUE NOBRE DE FIGUEIREDO
novembro de 2017

POLITÉCNICO
DO PORTO

GEOCAREYES – CANAL DE COMUNICAÇÃO PARA A GESTÃO DE FROTAS

Henrique Nobre de Figueiredo

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
2017

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores - Telecomunicações

Candidato: Henrique Nobre de Figueiredo, Nº 1120401, 1120401@isep.ipp.pt
Orientação científica: Carlos Ribeiro Campos, crc@isep.ipp.pt

Empresa: GISGEO

Supervisão: Ricardo Baptista, ricardo.baptista@gisgeo.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
17 de novembro de 2017

Agradecimentos

O momento de entrega e defesa da Tese de Mestrado despoleta uma variedade de sentimentos: a felicidade pela conclusão de um longo percurso académico, o alívio do esforço dedicado para a conclusão desta meta e a excitação do início de um novo percurso, a vida laboral.

A elaboração deste trabalho não teria sido possível sem a colaboração de diversas entidades e pessoas, às quais gostaria de deixar aqui expresso o meu agradecimento pela sua colaboração e apoio na realização deste projeto.

Agradeço ao Engenheiro Carlos Campos, por ter aceitado ser meu orientador científico neste trabalho e pela sua subsequente ajuda e disponibilidade até aos últimos momentos. A sua análise crítica foi indispensável ao aperfeiçoamento desta dissertação.

À empresa GISGEO e a toda a sua equipa, pela oportunidade que me ofereceram e pelo apoio para a conclusão deste Mestrado. Uma empresa com uma boa capacidade de organização, permitindo uma fácil integração e um ótimo ambiente de trabalho em equipa.

Ao Engenheiro Ricardo Baptista, meu orientador de estágio e amigo, por me garantir o acesso a todos os recursos necessários para o desenvolvimento do trabalho, pelas suas críticas construtivas, bem como sugestões que motivaram o meu espirito crítico e me levaram a desenvolver um trabalho mais coerente. Agradeço ainda pela sua disponibilidade, simpatia e amizade demonstrados ao longo deste processo.

Aos meus amigos, em especial aos membros do grupo de Facebook ”Pessoal que quer ser Mestre!”, por todos os momentos partilhados de brincadeira ou trabalho, que ajudaram a manter a motivação e moral para a conclusão deste percurso.

Aos meus tios, por todo o seu apoio e interesse demonstrado durante esta formação. As nossas discussões levaram-me a perceber o valor deste trabalho e consequentemente a orgulhar-me do mesmo.

Aos meus avós, que pelo seu carinho e amor criaram uma base de apoio para um equilíbrio mental, emocional e físico durante o meu percurso académico.

Ao meu irmão, Pedro de Figueiredo, pelo seu grande apoio mesmo com a distância que nos separa, pelos seus inestimáveis conselhos e sugestões e ainda por toda a paciência que teve para me ouvir e ajudar durante este processo.

Ao meu pai, Tomás de Figueiredo, pelo seu constante apoio, carinho e amor, por ser o meu motivador pessoal, por ouvir e me aconselhar em todas pequenas preocupações que foram surgindo e pelas longas conversas entediantes que me deixou continuar mesmo assim.

À minha mãe, Sílvia Nobre, pelo seu amor, carinho e apoio incondicional, pela sua disponibilidade para ler qualquer coisa que eu escrevesse (nem que ela não percebesse nada do assunto) e pelas noites que não a deixei dormir para partilhar as minhas preocupações e entusiasmos.

Finalmente, gostaria de agradecer ao meu grande amigo Teddy, pela sua amizade e carinho incondicional. O seu desejo de querer frequentar um curso superior serviu como principal foco motivador para a conclusão deste percurso.

Resumo

Os sistemas de informação têm estado em constante crescimento ao longo dos anos, permitindo às empresas o estabelecimento de normas e procedimentos, que agilizam as operações empresariais. Destacam-se os sistemas de gestão de frotas, com recurso à geolocalização de veículos, permitindo administrar de forma mais eficiente as atividades das empresas.

Apesar de já existirem produtos com este propósito, a verdade é que são pouco intuitivos e práticos, muitas vezes pela elevada quantidade de informação apresentada, tornando-se disfuncionais. Assim, identificou-se a necessidade de criar uma interface uniforme e intuitiva, com informação precisa e de fácil consulta, para a gestão de frotas de empresa.

Para colmatar esta lacuna, surge o projeto que aqui se apresenta, constituído por uma aplicação *web* de gestão de frotas, que futuramente será também complemento ao produto Geocar da GISGEO. A aplicação desenvolvida corresponde a uma página *web* de gestão de operações das empresas, que terá a forma de uma *WebTV*, ou seja, o seu funcionamento é automatizado e independente da interação do utilizador. O *layout* utilizado é maioritariamente constituído por um mapa sobre o qual é representada a frota de veículos da empresa, os serviços agendados e os respetivos estados, os alertas de irregularidades e perigos, assim como, indicadores de eficiência associados ao funcionamento da empresa e à sua frota. Apesar do cariz automático da aplicação, esta contém ainda uma interface interativa para o administrador, de modo a permitir que este possa configurar os módulos de informação a apresentar.

Com o desenvolvimento deste projeto, pode-se concluir dois aspetos relevantes. Por um lado, a importância crescente da utilização de *software* de georreferenciação na gestão eficiente de frotas, por outro, o valor que os serviços de geolocalização representam nas empresas. Estes aspetos vêm sendo corroborados pela crescente procura nos últimos anos, deste tipo de serviços.

Palavras-Chave

GISGEO, Sistemas de Informação Geográfica, Aplicação *web*, Bases de Dados, *JavaScript*

Abstract

Information systems have been in an ever-growing state in the last few years, allowing companies to set a group of norms and procedures to expedite companies' operations'. The fleet management systems are highlighted, since they allow a more efficient administration of a company operations through the geolocation of vehicles.

Although several products like this already exist, they are unintuitive, mostly because of the high amount of information presented, making them dysfunctional. Therefore, it was identified the need to create a uniform and intuitive interface, practical and easy to consult, for fleet management systems.

To bridge this gap, the project here presented is emerged, a web application for fleet management, that in the future will work as a complement to the product Geocar from the company GISGEO. The application developed corresponds to an operations management webpage, with the format of a WebTV, meaning, its behaviour is automatized and independent of users' interactions. The layout supplied consists, for the most part, of a map. Upon this map is marked the fleet of the customers' vehicles, the scheduled services and their respective state, the irregularity and danger alerts, as well as efficiency indexes regarding the company and its fleet functionality. Despite of the automatic nature of the application, it contains an administration interactive interface that allows the configuration of the information modules to present.

With the development of this project two important aspects can be established. In the one hand, the ever-growing importance of geo referencing software in the effective management of fleets, on the other hand, the value this kind of services adds to a web application, corroborated by the its increased demand in the last few years.

Keywords

GISGEO, Geografic Information Systems, Web application, Database, *JavaScript*

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AJAX – *Asynchronous JavaScript and XML*

API – *Application Programming Interface*

CSS – *Cascading Style Sheets*

CSV – *Comma-separated values*

GIF – *Graphics Interchange Form*

HTML – *HyperText Markup Language*

HTTP – *Hypertext Transfer Protocol*

JPEG – *Joint Photographics Experts Group*

JSON – *JavaScript Object Notation*

JSONP – *JSON with padding*

OHLC – *Open-high-low-close*

PDF – *Portable Document Format*

PHP – *PHP: Hypertext Preprocessor*

PNG – *Portable Network Graphics*

REST – *Representational State Transfer*

SDK – *Software Development Kit*

SIG – *Sistemas de Informação Geográfica*

SOAP – *Simple Object Access Protocol*

SQL – *Structured Query Language*

URI – *Uniform Resource Identifier*

URL – *Uniform Resource Locator*

XML – *eXtensible Markup Language*

1. INTRODUÇÃO

A integração de sistemas de gestão no quotidiano das empresas tem sido um processo crescente ao longo dos anos. Estes sistemas permitem o estabelecimento de normas e práticas de modo a melhor administrar as operações de uma empresa. Assim, os sistemas de gestão estão genericamente associados a todos o tipo de organizações empresariais.

Deste modo, para a grande maioria das empresas, o uso de sistemas de gestão desempenha um papel fulcral no sucesso das mesmas. Associados a este tipo de sistemas, estão os sistemas de informação que fornecem dados aos sistemas de gestão.

1.1. SISTEMAS DE INFORMAÇÃO

Os sistemas de informação disponibilizam diversos tipos de informação, nomeadamente pessoas, recursos ou atividades. Estes sistemas são criados pelos fornecedores de serviços, de modo a criar, recolher, processar, filtrar e distribuir informação.

Estão normalmente associados a outros tipos de sistemas (e.g: sistemas de gestão) e no seu conjunto formam um produto ou um serviço [1][2].

Um caso particular destes sistemas são os Sistemas de Informação Geográfica (SIG) abordados com mais detalhe no capítulo **2.1**.

1.2. CANAIS DE COMUNICAÇÃO

Os sistemas de gestão contemplam ainda o uso de canais de comunicação em conjunção com os sistemas de informação. Os canais de comunicação permitem a disseminação dos dados manipulados pelos sistemas de informação.

Admitindo a grande importância da implementação de um sistema de gestão numa empresa, o uso de canais de comunicação é imperativo para o bom funcionamento do sistema de gestão. Assim, é necessário utilizar canais que garantam que a informação é recebida, processada e retida.

Nos dias de hoje existem diversos tipos de canais de comunicação, sendo que cada um apresenta vantagens e desvantagens, pelo que, é necessário fazer uma escolha de acordo com a natureza da informação a ser partilhada.

Como exemplo de diferentes canais de comunicação, sublinhamos os seguintes:

- **A Fala** – o mais óbvio dos canais de comunicação e talvez o mais usado. É simples e intuitivo permitindo um *feed*back imediato. Apesar disso, obriga à proximidade entre os elementos da comunicação, para além de que, regra geral, a informação transmitida não é guardada.

- **Correio eletrónico** – simples e efetivo, este canal é usado muito frequentemente pela sua versatilidade em termos de do tipo de informação enviada. Por outro lado, este canal de comunicação não oferece garantias de que o recetor irá visualizar o *email*.

- **Intranet** – ideal para situações onde a informação é partilhada por um grupo com acesso à mesma rede.

- **Notificações *Push*** – um método com elevada fiabilidade pois apresenta uma nova janela ou notificação que se sobrepõe a qualquer aplicação dos dispositivos. Poderá também tornar-se abusivo se utilizado em excesso.

- **Eventos** – permitem uma melhor e mais detalhada forma de comunicação, no entanto, obrigam a um maior tempo de organização e implicam maiores custos.

- **Redes Sociais** – adequadas para a partilha de informação com múltiplos utilizadores simultaneamente.

Assim, a comunicação interna de uma empresa pode passar pelo uso de diversos tipos de canais, adequando cada um às suas necessidades comunicação. O projeto desenvolvido e apresentado neste documento corresponde a um sistema de gestão constituído por um sistema de informação (SIG) e um canal de comunicação (*browser/Internet*).

1.3. GISGEO

A empresa GISGEO foi fundada a 31 de Março de 2008 e é especializada na criação de sistema que usam informação geográfica, produzindo soluções *web* ou móveis capazes de obter, analisar, gerir e armazenar dados georreferenciados. A GISGEO está focada no desenvolvimento de ferramentas de apoio à decisão com base geográfica, assentes principalmente em tecnologia *open-source*, tendo aplicações em diversos sectores como transportes, saúde, segurança, desporto, marketing, entre outros [3].

A GISGEO apresenta-se no mercado com as seguintes soluções.

***Geocar* -** é uma solução de gestão de frotas que permite a localização e informação de viaturas ligeiras ou pesadas em tempo real. Este divide-se em duas componentes, uma de *software* que corresponde a uma aplicação de gestão de frotas, e outra de *hardware* correspondente aos diversos dispositivos instalados nas viaturas. Disponibilizando informações como: velocidade, localização, temperatura do motor, combustível, entre outros em tempo real. Os dispositivos comunicam com uma plataforma de suporte à aplicação. A plataforma armazena a informação numa base de dados que é acedida pela aplicação. Esta por sua vez, disponibiliza a informação através de um mapa, permitindo observar os veículos em tempo real, os seus percursos realizados, bem como a informação relacionada com cada veículo. Há também uma vertente de planeamento e análise com o cálculo de rotas, assim como a geração de relatórios com a informação da aplicação sumarizada.

***Geodecide* -** oferece soluções de apoio à decisão com base geográfica, com aplicação na gestão organizacional. A sua ferramenta de planeamento estratégico permite também otimizar recursos e reduzir custos.

***Geomed* –** é uma aplicação *web* que permite a gestão de dados ambientais, demográficos e administrativos como o objetivo de prevenir situações de risco na área da saúde. Permite também o acompanhamento de indivíduos em situação de risco através do uso de dados georreferenciados.

Geospacial - é um gestor e optimizador de recursos e infraestruturas para a manutenção de ativos imóveis. Permite a gestão integrada do território e a reunião de dados de diversas origens, associando os mesmos à área comercial e financeira.

Geosegur - constitui-se como interface para a recolha de informação georreferenciada em tempo real, relativa a agentes de segurança no terreno, posteriormente enviada para um centro de decisão. Permite o acompanhamento de pessoas em risco, controlo de pulseiras eletrónicas e localização de veículos dispondo de opções de segurança (Botões de SOS, bloqueio de ignição, entre outros).

Geosport - é uma plataforma para o controlo e gestão de informação relacionada com desportos de risco, oferecendo a possibilidade da localização em tempo real de indivíduos em zonas remotas bem como a deteção de situações de risco (quedas). Este produto permite também a localização de veículos e a obtenção de dados a eles associados como sejam a temperatura, as rotações do e a velocidade.

Geotravel - é uma aplicação móvel com realidade aumentada melhorando a perceção da realidade sobrepondo elementos virtuais sobre a área circundante para melhorar a gestão de conteúdos georreferenciados.

Geomarketing - oferece uma solução *web* para a visualização de mapas de distribuição de ativos da empresa e de outras entidades. Permite ainda o cálculo de estatísticas económico-financeiras através de valores registados em sistemas de gestão.

Geoindoor - é uma solução direcionada para a localização e controlo de máquinas ou recipientes. Tem como principais características a deteção de roubo e controle da temperatura dos objetos localizados.

1.4. CONTEXTUALIZAÇÃO

Este projeto surgiu no âmbito do Estágio Curricular para a conclusão do Mestrado em Engenharia Eletrotécnica e Computadores e foi proposto pela empresa GISGEO, com o intuito de desenvolver de uma aplicação *web* de gestão de conteúdos. A aplicação irá relacionar-se com um dos produtos da empresa, Geocar, criando um canal de comunicação para os seus clientes empresariais.

A aplicação a ser desenvolvida representa um complemento à solução Geocar, criando um novo serviço para o utilizador empresarial, uma *WebTV*. Esta corresponde à disseminação de conteúdos *web* através de um formato televisivo (automático). O desenvolvimento deste serviço tem o propósito de melhorar a gestão de frotas automóveis através do seu funcionamento automático.

1.5. OBJETIVOS

O objetivo principal deste projeto consiste no desenvolvimento de uma aplicação *web* para a visualização de informação relativa à gestão de frotas. Destaca-se a criação de uma interface de gestão de conteúdos, simplificando os processos de criação, gestão, publicação, distribuição e arquivamento de conteúdos, com faixas informativas[1] relevantes para a gestão de frotas automóveis, assim como informação diversa dos serviços agendados que representam as operações a realizar de uma determinada empresa e valores de eficiência diária e semanal, como por exemplo, quilómetros percorridos e combustível. Pretende-se que o desenvolvimento da aplicação *web* contemple as seguintes funcionalidades:

- Disponibilizar através de endereço web a visualização da programação diária estabelecida.
- Visualização de listas diárias dos serviços agendados e respetivos estados, bem como emissão de alertas quando o serviço é concluído, iniciado ou interrompido.
- Visualização do logótipo e do relógio com os fusos horários[2] configuráveis com os serviços agendados.
- Integração de *feeds* de notícias, meteorologia e tráfego rodoviário relacionados com os locais da realização dos serviços (por zonas geográficas: cidades, regiões ou países).
- Integração de uma faixa de notícias com indicadores de eficácia da operação da empresa, disponibilizados pela plataforma de gestão de frotas.

[1] Tira informativa (tendencialmente horizontal) que comummente contém informação de carácter volátil;

[2] Cada uma das 24 áreas em que se divide a Terra (de 15º em 15º admitindo a Terra como uma circunferência) caracterizada por uma posição diferente do Sol no mesmo instante de tempo.

- Inclusão de *layouts* animados em formatos de vídeo, flash ou imagem para divulgação de informação interna, ou mesmo para disponibilização de conteúdos publicitários de parceiros comerciais.

- Configuração através do gestor de conteúdos da programação diária.

- Possibilidade de interatividade do utilizador para a inclusão temporária de conteúdos.

As características aqui enunciadas constituem a base para a elaboração da solução proposta (secção **1.7**), bem como para o subsequente desenvolvimento do trabalho.

1.6. ANÁLISE DE REQUISITOS

Para a elaboração deste projeto há um conjunto de requisitos burocráticos (quer por questões legais, quer pela política interna da empresa), a observar de acordo com as normas definidas pela empresa, a saber:

- Cumprimento da lei de proteção de dados, através do uso de forma transparente dos dados e respeitando sempre a vida privada, bem como os direitos e liberdades da origem dos dados [4].

- Utilização de: as linguagens *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e *JavaScript* para a organização da parte gráfica; a linguagem de programação *PHP: Hypertext Preprocessor* (PHP); e PostgreSQL para o acesso às bases de dados utilizando um módulo PostGIS para o processamento de dados georreferenciados.

- Integração da aplicação a desenvolver no *layout* utilizado pelo produto Geocar, assim como uso *Application Programmable Interface* (API) desenvolvida para a visualização de dados relativos à gestão de frotas.

- Utilização dos mapas HERE assim como as suas APIs, dos quais a empresa possui licenças, para visualização dos mapas e recolha de informação relevante (por exemplo pontos de interesse, tráfego rodoviário, entre outros).

1.7. PROPOSTA DE SOLUÇÃO

Tomando em consideração os objetivos e requisitos do projeto, é necessária a elaboração de uma proposta de solução, que deverá ser também a mais completa e eficiente possível. A solução proposta consiste no desenvolvimento de uma aplicação *web*.

O facto de se partir de uma aplicação *web* prendeu-se com a necessidade de desenvolvimento de três componentes distintas: a parte gráfica (correspondente a um cliente), a parte lógica (correspondente a um servidor) e a fontes de informação. A primeira será desenvolvida através de um *layout* pré-definido pela GISGEO (Geocar v2). A parte lógica será caracterizada por um conjunto de controladores (em PHP) que estabelecem o acesso às diferentes fontes de informação. Por sua vez, as fontes de informação são de vários tipos, nomeadamente bases de dados, serviços *web* e bibliotecas de *Javascript*. A arquitetura da aplicação a desenvolver ilustra-se na figura seguinte.

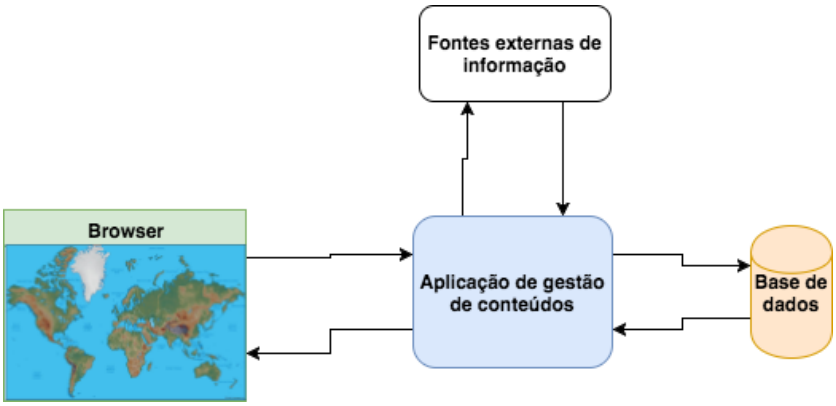


Figura 1 – Arquitetura da aplicação web.

Browser

O *browser* corresponde à componente gráfica da aplicação. A base desta componente será um mapa com as posições de cada veículo representadas, bem como informação de pontos de interesse (bombas de gasolina, áreas de descanso, entre outros), estado do tráfego rodoviário, meteorologia e fuso horário. O conteúdo desta página será ainda animado, mostrando sequencialmente a informação de cada viatura, serviços, alertas, entre outros. O *browser* comunicará com o gestor de conteúdos, através de pedidos *Asynchronous JavaScript And XML* (AJAX), de modo a manter a página atualizada. Os pedidos ao gestor de conteúdos não implicam navegar para outro endereço *web*.

Gestor de conteúdos

O gestor de conteúdos corresponde a uma plataforma de suporte à aplicação a desenvolver. É através deste que será estabelecida a comunicação com o *browser*, base de dados e fontes externas de informação. O gestor de conteúdos executará pedidos à base de dados e a fontes externas de informação que por sua vez irão devolver a informação a ser apresentada no *browser*. Dado que o gestor de conteúdos irá lidar com informação confidencial (por exemplo, credenciais de acesso à API), será necessária a proteção dessa informação, bloqueando o acesso direto à mesma. Deste modo o desenvolvimento do gestor de conteúdos será feito em PHP.

Fontes Externas de Informação

A informação a apresentar na aplicação *web* é gerada através de fontes externas. Desse modo e através do gestor de conteúdos é estabelecida uma comunicação com diversas fontes de informação. Estas fontes podem ser em forma de API (e.g. HERE *Map Tile* API) ou base de dados externas (Base de dados do Geocar).

Base de dados

De forma a armazenar dados relevantes para o funcionamento correto da aplicação, é sugerida a criação de uma nova base de dados ou a adição das seguintes tabelas na base de dados existente do Geocar.

- ***Login*** – Esta tabela deverá conter toda a informação relativa a um utilizador da aplicação. Terá campos como *nome de utilizador*, *password*, *identificador de empresa*, entre outros.

- ***Config*** – Esta tabela será utilizada para armazenar a informação relativa ao modo de funcionamento da aplicação para um utilizador. Deverá ter campos que permitem a ativação e desativação de funcionalidades, bem como um identificador de utilizador e informação relativa à cronologia das animações.

1.8. ORGANIZAÇÃO DO RELATÓRIO

Este relatório encontra-se dividido em seis capítulos. No primeiro, é feita uma introdução ao tema do projeto e uma abordagem à empresa onde o mesmo foi elaborado.

O segundo capítulo faz-se a apresentação de vários conceitos iniciais importantes para a compreensão do trabalho.

O terceiro capítulo diz respeito às diferentes *frameworks*[1] e bibliotecas possíveis para a implementação das funcionalidades do trabalho. Este capítulo começa por abordar as *frameworks* definidas como requisito, fazendo posteriormente uma abordagem a diferentes possibilidades para as necessidades o cumprimento dos objetivos.

O quarto capítulo corresponde ao desenvolvimento do projeto e divide-se em cinco partes: diagramas funcionais, desenvolvimento gráfico, bases de dados, API e desenvolvimento lógico.

O quinto capítulo aborda os testes efetuados à aplicação desenvolvida, enunciando também possíveis implementações e melhorias futuras.

No sexto e último capítulo são frisadas as principais conclusões retiradas ao longo do relatório e é feita uma apreciação geral ao trabalho desenvolvido.

[1] No contexto de desenvolvimento de software, *framework* corresponde a uma abstração que une diferentes projetos de software provendo uma funcionalidade genérica.

2. CONCEITOS

Neste capítulo apresentam-se um conjunto de conceitos, que servirão base de para a melhor compreensão do desenvolvimento do trabalho.

Será feita a definição de conceitos como aplicação *web* e SIG que desempenham um papel fundamental na elaboração do projeto. Posteriormente será também apresentada a definição de API, fazendo a caracterização das suas metodologias de acesso, bem como o formato dos dados utilizados. Finalmente será feita uma leve abordagem a tipo comum de ataque a aplicações *web*.

2.1. SISTEMAS DE INFORMAÇÃO GEOGRÁFICA (SIG)

Dentro dos sistemas de informação surgiram os sistemas de informação geográfica (SIG), caracterizados pelo teor geográfico/topográfico da informação que contêm. São sistemas constituídos por *software* e *hardware* com informação relativa a dados georreferenciados permitindo a sua visualização, análise e gestão.

Estes sistemas constroem-se em vários níveis de informação com diferentes atributos e objetivos (representação em camadas) [5]. Um exemplo de uso de produtos gerados em SIG pode ser visto diariamente nos mapas de previsão meteorológica apresentados na televisão ou nos jornais *online* [6].

É de referir que este tipo de sistemas não só exigem uma grande capacidade do computador utilizado, mas também poderão necessitar de uma formação prévia (do cliente) para o seu uso. A isto acresce o facto de a grande maioria destes sistemas ter um custo elevado [7].

2.1.1. *WEB*-SIG

Inicialmente os SIG estavam apenas estavam disponíveis *offline* na forma de programas como ArcGIS, QGIS, entre outros. No entanto, uma utilização eficiente deste tipo de ferramentas implica uma grande disponibilidade de tempo e de capacidade de processamento, algo que não é possível a todos os utilizadores/dispositivos.

Com a evolução tecnológica passou a ser possível enviar volumes elevados de informação pela *Internet*. Isto permitiu o desenvolvimento de sistemas *Web-SIG* que primam pela sua facilidade de acesso dado o nível de disseminação da *Internet*, bem como pelo reduzido custo quando comparados com os sistemas SIG tradicionais.

Outras vantagens importantes dos *Web*-SIG não presentes nos sistemas tradicionais, são a facilidade das atualizações dos sistemas e a sua independência de um sistema operativo especifico.

Infelizmente, as vantagens apresentadas representam custos elevados bem como a redução da velocidade destes sistemas. Como estes sistemas fazem um grande aproveitamento de capacidades gráficas, a transmissão desta informação pela *Internet* pode revelar-se muito lenta para um utilizador.

Adicionalmente é importante mencionar que estes sistemas não apresentam a funcionalidade/capacidade de processamento dos sistemas tradicionais, mas de igual modo não exigem o consumo de uma elevada quantidade de recursos [8].

A figura 2 representa a arquitetura de um *Web*-SIG.

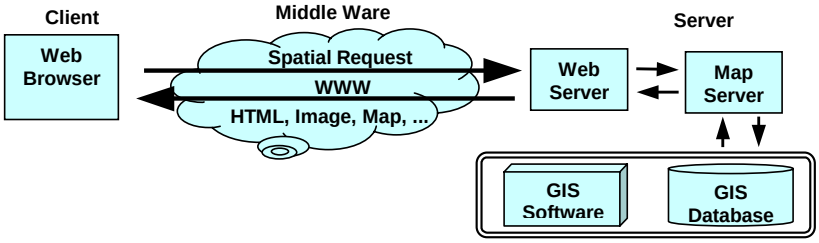


Figura 2 – Arquitetura de um *Web*-SIG [8].

As semelhanças entre uma arquitetura *Web*-SIG e uma aplicação *Web* são bastante evidentes, e serão abordadas na secção seguinte.

2.2. APLICAÇÕES *WEB*

No universo *web*, os sistemas de gestão estão muitas vezes associados a aplicações *web*. Estes são constituídos por sistemas de informação e canais de comunicação, como visto anteriormente, sendo que esta constituição é facilmente observada numa aplicação *web* através da sua definição.

Uma aplicação *web* é definida como *software* que funciona estabelecendo uma conexão cliente/servidor onde o cliente é ligado através de um *browser.*

Assim, é possível extrapolar que nas aplicações *web*, o canal de comunicação corresponderá ao *browser* e o servidor corresponderá ao sistema de informação.

Uma aplicação *web* poderá ainda ser caracterizada pelo modelo de 3 camadas, como ilustrado na figura 3.

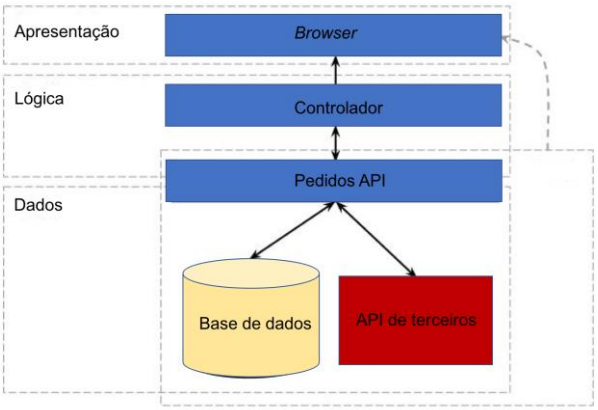


Figura 3 – Representação do modelo de 3 camadas.

Apresentação (*Front-end*) – Camada responsável pela apresentação, corresponde ao que é visualizado no *web browser*. Representa um canal de comunicação.

Lógica – Camada responsável pelo processamento da informação a ser apresentada ou guardada. Representa o controlo entre o canal de comunicação e o sistema de informação.

Dados (*Back-end*) – Camada responsável pela gestão dos dados através do uso de bases de dados. Representa um sistema de informação.

Após esta análise de uma aplicação *web*, é notória a semelhança entre esta arquitetura e a arquitetura de um *Web*-SIG. Usando a figura 2 como referência, é possível fazer a correspondência entre a divisão de um *Web*-SIG e a divisão de uma aplicação *web*, da seguinte forma:

- **Apresentação – *Client***
- **Lógica – *Middleware***
- **Dados – *Server***

Assim, um *Web*-SIG poderá ser considerado um caso particular de uma aplicação *web*.

2.3. *APPLICATION PROGRAMABLE INTERFACE* (API)

Dentro das aplicações *web* mencionadas anteriormente, surge o conceito de API. Uma API poderá ser descrita como um grupo de ordens, funções e/ou protocolos que permitem o desenvolvimento de *software*. API é uma interface de *software* para *software* que define as regras para a comunicação entre duas aplicações sem a necessidade de interação de um utilizador. Estas regras estabelecem a forma como o serviço será entregue e consumido.

As regras impostas pela API definem o protocolo utilizado, os formatos dos pedidos e o tipo e formato das resposta. É considerada uma boa API aquela que fornece todos os blocos de construção necessários, de modo a um programador apenas ter de "montar" os blocos para sua aplicação [9].

Comummente a informação gerada pelos SIG poderá acedida através de uma API (Exemplos disto serão abordados nos capítulos seguintes), sendo que esta serve como ponto de controlo para a troca de informação entre os SIG e uma nova aplicação a desenvolver.

2.4. *SIMPLE OBJECT ACCESS PROTOCOL* (SOAP)

Simple Object Access Protocol (SOAP) é um mecanismo direcionado para a transferência de informação estruturada entre serviços *web*. SOAP não define um modelo de programação ou uma lógica de implementação, a sua função é apenas a criação de mecanismos de encapsulamento e encriptação de dados.

A figura abaixo representa uma simples mensagem SOAP.

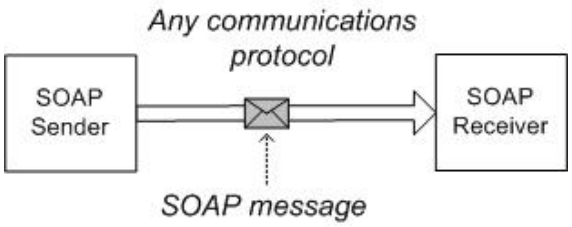


Figura 4 – Transporte de uma mensagem SOAP [10].

Como é possível observar, SOAP é compatível com qualquer protocolo de comunicação e de igual modo com qualquer linguagem de programação. No entanto, a sua limitação está na falta de compatibilidade com diferentes formatos de dados, apenas sendo possível utilizar *eXtensible Markup Language* (XML) [10].

SOAP apresenta uma *framework* para troca de mensagens SOAP. Esta *framework* define uma série de elemento XML utilizados no encapsulamento de informação XML. Estes elementos facilitam a comunicação em meios heterogéneos onde a interoperabilidade é uma dificuldade.

Os elementos XML principais de uma *framework* SOAP são:

- **Envelope** – Este elemento corresponde à raiz da mensagem SOAP. Através deste elemento, as aplicações facilmente identificam que se trata de uma mensagem SOAP.

- **Header** – É um elemento genérico para informação de controlo (análogo a cabeçalhos de protocolos de comunicação como HTTP). Este campo é opcional, mas quando é utilizado, deverá encontrar-se contido no elemento *Envelope*;

- **Body** – Trata-se de um elemento genérico para a informação a transmitir e poderá conter qualquer número de elementos. Este campo é obrigatório, devendo estar contido no *Envelope*, depois do elemento *Header* se este existir.

- **Fault** – Este elemento é utilizado para representar eventuais erros que possam estar presentes no **Body**.

A figura abaixo apresenta um exemplo de uma mensagem SOAP ligada a uma aplicação bancária.

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Header>
  <!-- security credentials -->
  <s:credentials xmlns:s="urn:examples-org:security">
   <username>dave</username>
   <password>evad</password>
  </s:credentials>
 </soap:Header>
 <soap:Body>
  <x:TransferFunds xmlns:x="urn:examples-org:banking">
   <from>22-342439</from>
   <to>98-283843</to>
   <amount>100.00</amount>
  </x:TransferFunds>
 </soap:Body>
</soap:Envelope>
```

Figura 5 – Exemplo de uma mensagem SOAP para uma transação bancaria [11].

2.5. *REPRESENTATIONAL STATE TRANSFER* (REST)

Representational State Transfer (REST) corresponde a uma arquitetura *Web* para a transferência de informação.

Este conceito aparece muitas vezes associado ao protocolo SOAP, já apresentado, como uma alternativa ao mesmo. No entanto, estes dois conceitos apresentam uma diferença conceptual chave: o REST é uma arquitetura e o SOAP é um protocolo.

Esta diferença torna-se visível quando se definem arquitetura e protocolo, a saber:

- **Arquitetura –** Define um *layout* lógico para o desenvolvimento. É completamente independente do protocolo utilizado.
- **Protocolo –** Define um conjunto de regras e diretivas para o modo de comunicação entre duas máquinas.

REST define ainda uma série de propriedades arquitetónicas:

- **Cliente-Servidor** – A arquitetura do sistema deverá ser cliente-servidor. A separação da interface de utilizador (cliente) e dos dados processados (servidor) permite uma maior portabilidade do cliente para plataformas diferentes, permitindo também uma maior escalabilidade pela simplificação dos componentes do servidor;
- ***Stateless*** – As respostas deverão ser "Sem estado". Significando que a API à qual foi executado o pedido REST não deverá guardar nenhuma informação relativa à

pessoa/dispositivo que efetuou o pedido. Um exemplo prático disto é que os servidores de um serviço REST não irão guardar a sessão do utilizador. Isto tem especial importância na fiabilidade dos sistemas, pois representa um esforço menor para encontrar falhas parciais;

- ***Cache*** (guardar respostas do lado do cliente) – As respostas enviadas pelo serviço REST deverão conter um indicador se são *cacheable* ou não. O uso deste indicador potencia a possibilidade de reduzir ou eliminar parte das comunicações, melhorando a eficiência e a performance do servidor, reduzindo a latência dos clientes.

- **Interface Uniforme** – Os pedidos ao serviço deverão ser consistentes na sua forma de identificação dos recursos, manipulação dos recursos e nas mensagens de estado (e.g: se a página de erro de servidor corresponder a um código 500 para um recurso, deverá manter-se para os restantes recursos). Um exemplo disto poderá ser um serviço que utiliza *HyperText Transfer Protocol* (HTTP) para o acesso aos seus recursos. O *Uniforme Resource Locator* (URL) do recurso deverá ser o mesmo para todos os clientes. Neste caso, como é utilizado o protocolo HTTP, se o utilizador pretender receber informação deverá enviar um pedido GET; caso necessite de enviar informação deverá utilizar um pedido POST [12].

Salienta-se que REST aceita múltiplos tipos de resposta como HTML, *eXtensible Markup Language* (XML), *JavaScript Object Notation* (JSON), *Portable Document Format* PDF, *Joint Photographics Experts Group* (JPEG), texto simples, entre outros.

Um erro comum é admitir que REST utiliza o protocolo HTTP. Apesar de a grande maioria dos serviços REST utilizarem HTTP, isto não é um requisito, sendo que REST não define nenhum protocolo de comunicação obrigatório [13].

2.6. *eXTENSIBLE MARKUP LANGUAGE* (XML)

XML é um formato de texto desenvolvido a pensar no armazenamento e transporte de dados. É um formato que funciona com base em *markups/tags* tal como HTML, no entanto estas *tags* não estão previamente definidas [14].

Outra grande diferença entre HTML e XML, é que este é direcionado para o transporte de dados, não contendo indicações sobre como a informação irá ser visualizada. Pelo contrário, HTML é unicamente utilizado para a visualização de informação.

Dadas as diferentes características, HTML e XML são muitas vezes usados complementarmente, utilizando XML para o transporte de dados e HTML para a sua visualização separando efetivamente a componente gráfica da componente lógica da aplicação.

XML é construído em forma de árvore, nasce num nó "raiz" e expande-se para os nós "filhos". A figura seguinte ilustra esta estrutura.

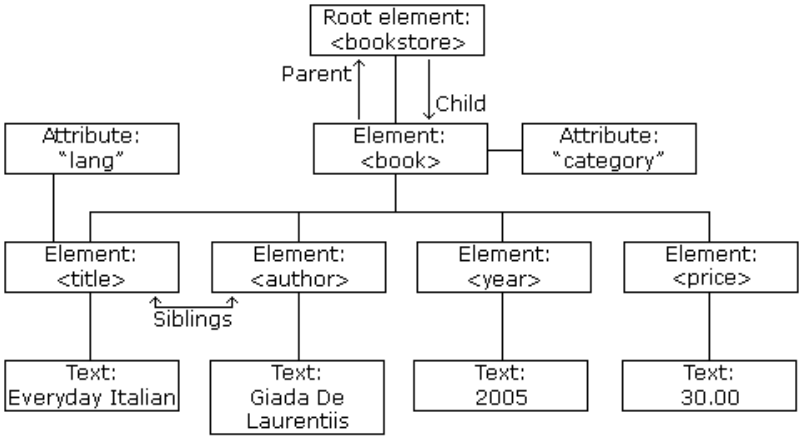


Figura 6 – Exemplo de estrutura XML.

Os nós poderão ser caracterizados através da sua relação com outros nós, isto é, um nó que tenha um nó "filho" é o "pai" desse "filho". De forma semelhante, todos os nós que partilhem o mesmo pai são denominados de "irmãos". No exemplo da figura, o elemento *book* é "pai" do elemento *title* e do elemento *author*, o que significa que *title* e *author* são irmãos.

Todos os elementos poderão ter nós "filhos" e estes por sua vez poderão ter os seus próprios "filhos", repetindo este padrão de acordo desejos do programador. No entanto, XML define que um "filho" não poderá ter mais que um "pai".

A figura apresentada em cima, ilustra também uma boa prática do uso de XML pelo seu carácter auto descritivo. Não é necessário ser o autor do XML da figura, para facilmente perceber que este XML é representativo de uma livraria, contendo os seus livros e elementos chave (título, autor, ano e preço).

XML permite também o uso de atributos associados aos seus elementos. Estes atributos funcionam de maneira análoga a atributos de HTML como *name*, *id* ou *class* e caracterizam um determinado elemento. Novamente não são pré-definidos nomes para os atributos de XML.

Finalmente existem os denominados XML *Namespaces* que são formas de minimizar conflitos causados por nomes iguais para *tags* com informação diferente. Para tal, fazem uso de um prefixo no nome das *tags*, evidenciando o carácter singular de cada elemento.

Um XML *Namespace* indica a localização de um recurso através de um *Uniforme Resource Identifier* (URI) que corresponde a um conjunto de caracteres utilizado para localizar um recurso *Web*. Um exemplo comum de URI é o URL, utilizado para aceder a domínios na *Internet*.

Um elemento XML *Namespace* é definido por um atributo "xmlns" na *tag* inicial do seu elemento, associado ao prefixo definido bem como ao seu URI. Os XML *Namespaces* apresentam o seguinte formato: **xmlns: prefixo="URI"**.

No *snippet* de código XML a seguir apresentado podem-se observar duas tabelas com elementos "filhos" diferentes e consequentemente identificadas com XML *Namespaces* diferentes [15].

```
<h:table xmlns:h="./fruta/" >
  <h:tr>
    <h:td>Maçãs</h:td>
    <h:td>Pêras</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="./carne/" >
  <f:nome>Tabela Anti-Fruta</f:nome>
  <f:largura>80</f:largura>
  <f:altura>120</f:altura>
</f:table>
```

2.7. *JAVASCRIPT OBJECT NOTATION* (JSON)

Como visto anteriormente, as respostas das APIs tendencialmente serão do tipo JSON ou XML, sendo o primeiro tipo mais comum.

JSON é um formato de dados indicado para transmissão. É completamente independente da linguagem de programação onde é utilizado, mas rege-se pelas convenções semelhantes às estabelecidas em linguagens da família C (C, C#, PHP, Java, *JavaScript*, Python, Perl, etc).

Isto significa que a manipulação de elementos JSON é uma tarefa relativamente fácil para a maioria das linguagens.

Um elemento JSON pode ser construído com base em duas estruturas:

- Objeto – Um objeto JSON corresponde a uma lista de elementos não ordenada, associando um nome (*string* na figura abaixo) a um valor (*value*) para cada elemento. A definição do objeto é feita através de chavetas, dentro das quais estará a lista.

- *Array* – Um *array* JSON corresponde, por sua vez, a uma lista ordenada. Esta apenas contém os valores da lista, sendo necessário o utilizador saiba a posição no *array* de cada elemento. A sua definição é feita através de parênteses retos, dentro dos quais estará a lista.

É de notar que estas duas estruturas estão presentes na grande maioria, senão em todas as linguagens de programação, isto alude ao facto de JSON ser uma ótima escolha para a transmissão de dados entre linguagens de programação diferentes.

As figuras 7 e 8 representam um objeto e um *array* na forma JSON, respetivamente.

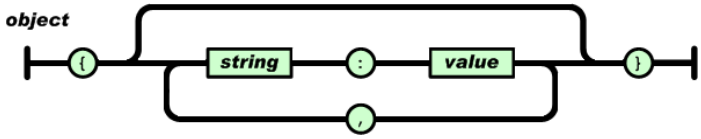


Figura 7 – Objeto JSON.

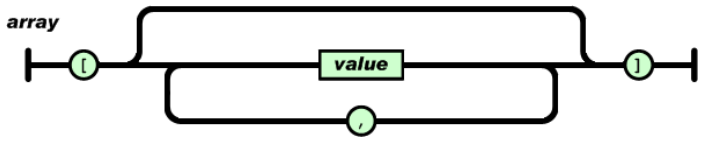


Figura 8 – *Array* JSON.

Destaca-se que o elemento *value* poderá ser de diversos tipos, incluindo *array* e objeto. Isto permite construir elementos JSON de elevada complexidade contendo uma grande quantidade de informação em forma de objetos ou *arrays* dentro de objetos ou *arrays* [16].

2.8. SQL *INJECTION*

No sentido de manter a segurança e integridade da aplicação, foram estudados mecanismos de proteção a ataques através de SQL *Injection.*

Os ataques SQL *Injection* são um dos maiores riscos de segurança no desenvolvimento de aplicações *web*. Estes consistem da inserção de *querys* maliciosas, enviadas através de um parâmetro, forçando assim a sua execução.

O excerto de código abaixo mostra um exemplo de uma *query* vulnerável a este tipo de ataques.

```
query = "SELECT * FROM login WHERE name = '" + clientName + "';";
```

Caso a variável "clientName" (a preencher pelo utilizador) seja um parâmetro recebido diretamente pelo servidor, facilmente um *hacker* insere o seguinte valor nesse campo "a';DROP TABLE login; SELECT * FROM login;".

O uso desse valor resultaria numa *query* deste género:

```
query = "SELECT * FROM login WHERE name = 'a';DROP TABLE login;
SELECT * FROM login;";
```

Como é possível observar, isto representa uma falha gravíssima de segurança, que resultaria na eliminação da tabela *login*. Isto acontece porque o *hacker* inseriu um carácter de fuga de *query*, isto é, um caracter que permite a terminação da *query* original, neste caso " ' ".

Assim, foram utilizados mecanismos de proteção contra estes ataques, nomeadamente o uso de *binds.*

As *binds* fazem um encapsulamento das variáveis enviadas para os campos. Mas a sua funcionalidade de segurança está na análise e interpretação que um *driver* (dependente da linguagem de programação utilizada) faz do valor, eliminando a possibilidade de *queries* maliciosas, uma vez que os caracteres de fuga de *query* são eliminados.

Para além da melhoria da segurança da aplicação através das *binds*, o seu uso aumenta também a velocidade de processamento das *queries*, resultando no crescimento da eficiência das mesmas [17].

Adicionalmente é recomendado o uso de verificações de segurança antes de qualquer *query* ser executada, nomeadamente a verificação do tipo (e.g. um campo dedicado a valores numéricos apenas poderá conter números).

3. *FRAMEWORKS* E TECNOLOGIAS

Neste capítulo serão abordados os diferentes tipos de tecnologias e *frameworks* disponíveis para o desenvolvimento da aplicação.

Numa primeira instância são caracterizadas as *frameworks* utilizadas no desenvolvimento do trabalho. Após isto, serão estudadas as diferentes bibliotecas usadas no carregamento de mapas; Em seguida, serão abordados os diversos tipos de prestadores de serviços para a geração de imagens de mapas. Finalmente será feito um estudo das diferentes funcionalidades necessárias ao projeto, bem como das diferentes opções de implementação. É importante destacar que as tecnologias evidenciadas como melhores, poderão não corresponder às implementadas devido a escolhas internas da empresa.

3.1. BOOTSTRAP

Como já referido, um dos requisitos do trabalho é o seu desenvolvimento num *layout* baseado numa *framework* de *Bootstrap*.

Bootstrap é uma *framework open-source* para o desenvolvimento de projetos *web* baseados em HTML, CSS e *JavaScript*. Esta *framework* apresenta uma variedade de componentes

para o desenvolvimento *front-end*. *Bootstrap* é dependente do uso de *jQuery*, que é uma pequena biblioteca de utilitários de *JavaScript*, que simplifica a manipulação e movimentação da página, o uso de animações, pedidos HTTP e tratamento de eventos. É direcionado para o desenvolvimento de aplicações responsivas, isto é, aplicações cuja componente gráfica é adaptável às características do dispositivo onde é utilizado. As ferramentas da *framework Bootstrap* poderão ser divididas em 4 categorias: *layouts*, conteúdos, componentes e utilitários.

- ***Layouts*** – Estes elementos correspondem a áreas e estilos de *tags* pré-definidas através de uma classe e direcionadas para a organização da estrutura da página. Os *layouts* apresentados correspondem a recipientes para diferentes tipos de conteúdo, como tabelas ou caixas. O uso destes elementos é uma mais-valia evidente na repetição de componentes da página.

- **Conteúdo** – Esta parte representa a adaptação das *tags* HTML (previamente definidas) ao carácter responsivo da aplicação. Contém algumas alterações aos estilos nativos de *tags* HTML (como a *tag "fieldset"* cujas bordas são removidas pelo *Bootstrap*) e acrescenta classes com diferentes estilos pré construídos (como por exemplo uma tabela estilizada a preto com as letras brancas através da classe "*table-inverse*"), mantendo o carácter responsivo da página.

- **Componentes** – Os componentes correspondem a um conjunto de classes de elementos gerais de uma página *web*, definindo também todos os subelementos que possivelmente integrarão o elemento geral. A grande mais-valia do *Bootstrap* poderá ser encontrada nesta categoria, pela vasta quantidade de componentes disponíveis. Exemplos destes componentes poderão ser navegadores, botões, alertas, formulários, caixas modais, *tooltips* (mensagens visualizadas quando um utilizador move o cursor para cima de um elemento), entre muitos outros.

- **Utilitários** – O *Bootstrap* disponibiliza uma série de utilitários, na forma de classes HTML que correspondem a diversos atributos de CSS, sendo que o seu uso já comtempla propriedades responsivas. A utilidade destas classes é evidente para o exemplo das bordas de uma *tag*, que com apenas o uso da classe "border-top-0" poderão ser desenhadas num elemento com a exceção da borda superior [18].

Adicionalmente poderão ser adicionados *plugins* de *jQuery* com diversos propósitos ao *Bootstrap*, como por exemplo um *plugin* para ativar uma lista suspensa quando o rato pairar sobre o elemento, aumentando assim o leque de possibilidades [19].

3.2. POSTGRESQL

Uma das necessidades deste trabalho é o uso de bases de dados para armazenar informação. De acordo com os requisitos, esta deverá ser baseada em PostgreSQL e, portanto, esta subsecção é dedicada a este tema.

O PostgreSQL é um sistema *open-source* de bases de dados do tipo objeto-relacionais. Para uma melhor compreensão do seu funcionamento, é apresentada uma caracterização de bases de dados relacionais e de bases de dados orientadas a objetos. Finalmente é definido o hibrido de ambas, bases de dados objeto-relacionais, caracterizando assim o PostgreSQL [20].

3.2.1. BASES DE DADOS RELACIONAIS

No modelo de bases de dados relacionais, a informação é guardada em tabelas, que podem posteriormente ser acedidas através operadores relacionais como *select* e *project* [21].

As tabelas estabelecem relações entre si, nomeadamente através das denominadas "*Foreign keys*", denotando a interdependência das mesmas. Estas controlam informação de determinadas tabelas cuja informação é dependente de outra tabela. Assim a escrita em tabelas com estas dependências, limita-se a informação contida na tabela que estabelece a dependência (e.g: só poderá ser adicionado um carro a uma tabela "Carros" se a marca do mesmo estiver listada numa tabela "Marcas").

Os operadores relacionais aplicados às bases de dados derivam da álgebra relacional e poderão ser descritos da seguinte maneira:

SELECT – O *select* é um operador cuja resposta corresponde a um conjunto de linhas de uma determinada tabela. É caracterizado pelo uso de uma condição que irá filtrar os resultados, apresentado apenas aqueles que cumprem a condição. Deste modo, é possível considerar o *select* como um "filtro horizontal", significando que este operador irá filtrar as linhas correspondentes à condição.

PROJECT – O *project* é um operador semelhante ao *select*. Caracteriza-se também pelo uso de uma condição que filtra os resultados, correspondendo a resposta aos elementos que cumprem a condição. Difere do *select* pelo facto de filtrar atributos e não elementos, significando isto que a filtragem será feita em termos de colunas em vez de linhas. Assim, e analogamente ao *select*, este operador poderá ser denominado de "filtro vertical".

JOIN – *Join* é um operador utilizado na combinação de diferentes tabelas, estabelecendo a relação entre elas. Este operador faz a junção de dois ou mais grupos de resultados, filtrando-os através de uma condição (caracterizada por elementos de ambos os resultados). De forma semelhante ao *select*, este operador faz uma filtragem horizontal, mantendo os todos os atributos de cada resultado.

DIVISION – O operador *division* é definido como um operador entre duas tabelas, a operação de divisão consiste no estabelecimento de correspondências entre os elementos de uma primeira tabela cujos atributos correspondam à totalidade da segunda tabela. Um exemplo desta operação será a filtragem de alunos que estão inscritos em todas as disciplinas lecionadas por um professor. Neste caso, a primeira tabela compreende o nome dos alunos associado às disciplinas correspondentes e a segunda tabela inclui a lista de disciplinas lecionadas pelo professor. O resultado será o nome dos alunos que frequentam todas as disciplinas do professor [22].

3.2.2. BASES DE DADOS ORIENTADAS A OBJETOS

Bases de dados orientadas a objetos baseiam-se, como o nome indica, no armazenamento da informação em forma de objeto.

Um objeto é caracterizado por dois componentes:

- **Atributos –** Os atributos caracterizam o objeto, definindo a sua função.
- **Métodos –** Os métodos definem o comportamento do objeto e são constituídos por funções ou procedimentos de acesso ao objeto.

A definição de um objeto é dependente de um conceito denominado classe. A classe representa o construtor do objeto, estando nela contidos os atributos e métodos associados. A classe não contém informação, é apenas um modelo para a criação de objetos, estando a informação contida nos objetos criados.

As definições acima descritas aludem a uma grande diferença entre bases de dados relacionais e orientadas a objetos. As bases de dados orientadas a objetos comportam métodos e atributos, definindo, portanto, a informação armazenada e o código a executar para o seu acesso. Por outro lado, as a bases de dados relacionais apenas definem a forma de armazenamento dos dados [23].

O uso de objetos apresenta vantagens no que diz respeito às bases de dados relacionais, pois permite a construção de estruturas mais complexas que de outro modo teriam de ser divididas em diversos atributos. Noto ainda que nas bases de dados orientadas a objetos a informação poderá ser toda manipulada pela aplicação, eliminando a separação lógica entre a base de dados e a aplicação.

3.2.3. BASES DE DADOS OBJETO-RELACIONAIS

Existe ainda uma versão híbrida das duas opções mencionadas. Trata-se de bases de dados objeto-relacionais. Estas são construídas com base no modelo relacional, adicionando o tipo objeto como um possível campo. Assim, é possível manter a simplicidade das bases de dados relacionais, combinando objetos para a representação de estruturas mais complexas.

3.2.4. POSTGIS

A escolha do PostgreSQL para o desenvolvimento deste trabalho advém do módulo PostGIS disponibilizado.

Este módulo permite o tratamento de informação de cariz geográfico. Para tal, apresenta uma série de funções, tipos e funcionalidades de indexação, conferindo rapidez e robustez ao PostgreSQL na manipulação de informação espacial.

Apresenta uma série de construtores para formas geométricas, facilitando assim o armazenamento de pontos, linhas e polígonos.

Destacam-se as suas funções de manipulação geométrica, que realizam operações como interpolações, interseções, cálculo de áreas e perímetros, entre outros. Estas funções são dependentes de um uso prévio dos construtores de formas geométricas [24].

3.3. BIBLIOTECAS DE MAPAS

No desenvolvimento de uma aplicação *web*, a sua escalabilidade é um conceito que deverá ser tido em consideração. Num mercado onde os prestadores de serviços de mapas estão cada vez mais competitivos, é necessário construir aplicações que sejam independentes dos prestadores desses serviços.

Assim, surge a necessidade de utilizar uma biblioteca de mapas, que permite ao programador facilmente alterar as fontes de informação, mantendo a mesmo estrutura e lógica de código. Isto evita a necessidade de aprender a trabalhar com as bibliotecas específicas dos prestadores dos serviços, tornando-os irrelevantes para o desenvolvimento (mas não para a qualidade do serviço).

3.3.1. OPENLAYERS VS LEAFLET

O *OpenLayers* consiste de uma biblioteca de elevada funcionalidade para a representações e manipulações de mapas para qualquer aplicação web. O *OpenLayers* destaca-se principalmente pelo grande leque de opções e funcionalidades que acarretam custos em termos de peso/dimensão da biblioteca.

Por sua vez, o *Leaflet* consiste numa biblioteca leve para o uso de mapas em aplicações web. Apresenta uma grande variedade de *plugins* possíveis, tendo menos funcionalidades de base quando comparado com o *OpenLayers*.

Ambas as bibliotecas partilham uma série de componentes chave, nomeadamente:

- **Mapa** – O mapa é a componente principal destas bibliotecas, estando encarregue de alocar o espaço para as imagens do mapa numa determinada *tag* "div". Para um mapa ser visível, este componente deverá ser carregado juntamente com uma camada e uma vista, em simultâneo, associadas ao mapa. No caso particular do *Leaflet*, este elemento poderá ser carregado com um conjunto de opções adicionais. Um exemplo é se o mapa é arrastável ou não (esta funcionalidade torna-se relevante no "modo Animado" da aplicação, que será discutido nos capítulos seguintes).

- **Camada –** As camadas são elementos que funcionam sobre o mapa e carregam um conjunto de subelementos, normalmente do mesmo tipo. São estes elementos que agrupam as imagens de um serviço de mapas e as carregam na divisão do mapa. De

igual modo, um grupo de marcadores poderá ser agrupado numa camada e em seguida adicionado ao mapa. A organização em camadas é bastante vantajosa para a ativação/desativação dos diversos componentes do mapa. Por exemplo, a visualização ou não de um mapa de relevo pode ser alterada com a adição/remoção de uma camada.

- **Marcador** – O marcador é um dos elementos mais básico destas bibliotecas. Corresponde a um ponto, representado por um ícone, sendo utilizado para marcar diferentes tipos de informação no mapa. Este componente apenas está presente no *Leaflet*, dado que o *OpenLayers* o descontinuou na versão 3.0. Desse modo, a representação deste elemento no *OpenLayers* é exequível (através de a associação de um ponto a uma imagem) mas é um processo mais complexo.

- **Vista -** A vista determina a posição central do mapa (correspondendo a um par de coordenadas geográficas), bem como o seu nível de zoom (variante de 1 a 20). Este elemento apenas existe no *OpenLayers* mas poderá ser reproduzido pelas opções de inicialização de um mapa *Leaflet* e posteriormente ser alterado através do método "setView". O *OpenLayers* permite ainda um parâmetro adicional de rotação de modo a apresentar o mapa num determinado ângulo.

Os componentes aqui listados são apenas os fundamentais para o carregamento de um mapa na página (com a exceção do marcador, que foi listado pelo seu elevado uso), existindo uma variedade elevadíssima de outros, nomeadamente formas geométricas ou botões de controlo (e.g: controlo de *zoom*) [25][26].

É importante ainda sublinhar alguns aspetos acerca do *Openlayers*. Tradicionalmente destacava-se pelo seu elevado número funcionalidades, permitindo um grande nível de especificações dos componentes, aumentando, no entanto, a complexidade do código. Tal verificava-se até à versão 2 desta biblioteca, tendo sido retiradas uma série de funcionalidades (como a implementação de marcadores referida anteriormente) na versão 3 (e na atual versão 4). Estas funcionalidades estão a ser transferidas para a forma de *plugin* analogamente ao que acontece com o *Leaflet*, de modo a deixar a biblioteca mais leve. Assim, neste momento, o *OpenLayers* está num processo intermédio onde não apresenta tantas funcionalidades como as versões anteriores e também ainda não é tão leve como o *Leaflet*.

3.4. SERVIÇOS DE MAPAS

Dado que as bibliotecas acima referidas funcionam como interligação entre a aplicação web e os serviços de mapas, torna-se necessária a escolha de um serviço de mapas para utilizar em conjunção com a biblioteca escolhida.

Um dos fatores a ser considerado na gestão de uma frota são os níveis de congestionamento das rotas que os condutores seguem. Uma previsão do fluxo de tráfego pode ser determinante para o cumprimento de horários e a consequente satisfação dos clientes. Assim, salientam-se os prestadores de serviços de mapas que contemplem este recurso, que apenas existe na forma de imagens de mapa e, portanto, apenas empresas que forneçam serviços de mapas o possuem.

3.4.1. OPENSTREETMAPS

OpenStreetMaps é um projeto que funciona através de um mapeamento colaborativo, na elaboração de um mapa mundial livre e editável. Apresenta a vantagem de ser *open-source*, significando que qualquer pessoa ou empresa poderá utilizar estes dados para qualquer propósito, desde que o *OpenStreetMaps* e os seus colaboradores sejam creditados.

Um exemplo de uma imagem gerada por este serviço de mapas poderá ser visualizado na figura seguinte.

Figura 9 – Exemplo de imagem do *OpenStreetMaps*

Esta solução não apresenta a opção de visualização do fluxo de tráfego, impossibilitando a sua utilização no desenvolvimento deste projeto [27].

3.4.2. BING MAPS

O Bing *Maps* é uma alternativa desenvolvida pela Microsoft para a implementação de mapas. O Bing *Maps* disponibiliza uma API através de uma interface REST para a criação de mapas estáticos personalizáveis, a tradução de endereços, a importação de imagens de mapas e o cálculo de rotas. É necessário um registo, quer do utilizador, quer da aplicação onde os mapas serão utilizados. Após o registo, é possível usar a chave gerada para a aplicação, de modo a fazer pedidos ao servidor Bing [28].

Uma das desvantagens deste serviço de mapas é a forma como as imagens são referenciadas. Este identifica cada quadricula do mapa através de uma *quadkey*, ou seja, através de um único valor que contem a informação do *zoom* e localização. As bibliotecas de mapas referidas anteriormente funcionam através de uma divisão em linhas, colunas e *zoom*. Isto significa que na prática o uso deste serviço implica uma adaptação às bibliotecas [29].

A figura seguinte ilustra uma imagem gerada por este serviço.

Figura 10 – Exemplo de imagem Bing *Maps*

Outro ponto desfavorável deste serviço é o facto de não conter informação relativa ao fluxo de tráfego.

3.4.3. GOOGLE MAPS

O Google *Maps* é um dos serviços de mapas mais utilizados no mundo, tanto na forma de mapa interativo, como na forma de API. Esta dispõe de diversas formas de acesso: Android, iOS, *web* (mapas encastrados ou através de *JavaScript*) e serviços *web* [30].

É de salientar que quando a API é acedida através de um serviço *web*, não é possível obter uma resposta de imagens de mapas, assim, a importação de um mapa deverá ser feita previamente através de um mapa encastrado ou de *JavaScript*.

As APIs apresentam uma variedade de serviços: localização de pontos de interesse, cálculo de rotas, valores de elevação, fuso horário e até mesmo o valor dos limites de velocidade. Funciona de maneira análoga ao Bing *Maps*, sendo necessário o registo do utilizador e da aplicação de forma a gerar a chave da API.

A sua limitação está no número de acessos possíveis através da versão grátis do serviço (25000 por dia), sendo que a extensão desse limite está dependente da compra de um plano.

Este serviço fornece também a opção da representação do fluxo de tráfego, apenas possível através da API *JavaScript* [31].

Sublinha-se ainda que é expressamente proibido utilizar as APIs sem o uso de um mapa do Google.

A figura seguinte apresenta um mapa gerado por este serviço.

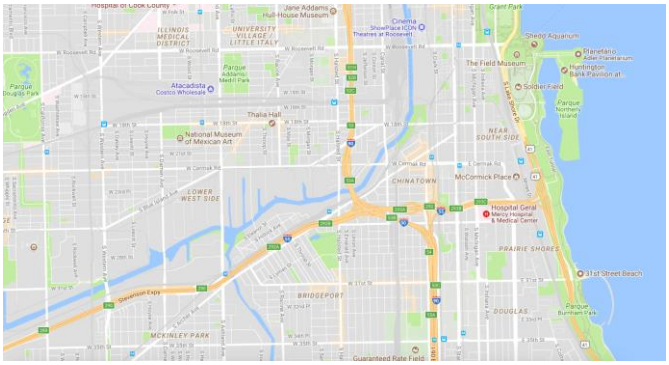


Figura 11 – Exemplo de mapa da Google *Maps*

3.4.4. MAPQUEST

A *MapQuest* é uma empresa especializada na criação de soluções geoespaciais. Dispõe de diversas APIs e *Software Devolopment Kits* (SDK) que estabelecem o acesso a uma variedade de mapas e também a funções fundamentais, tais como a tradução de endereços, o cálculo de rotas, a representação do fluxo de tráfego, entre outras.

As APIs estão disponíveis através de *JavaScript*, mas foram criados *plugins* de integração com a biblioteca *Leaflet*, sendo possível utilizar os recursos da *MapQuest.* Estes *plugins* convertem os objetos *MapQuest* em extensões de objetos do tipo camada *Leaflet.*

O uso destas APIs carece de um registo para a obtenção de uma chave API. O registo poderá ser feito de forma gratuita, permitindo o acesso a todos os recursos disponíveis apenas

limitado por um número máximo de 15000 transações por dia. No caso de o registo ser associado a um plano pago, o número máximo de transações é aumentado (e.g: 30000 transações/dia -> 99$/mês).

A *MapQuest* dispõe também de imagens de mapas com informação relativa ao fluxo de tráfego, no entanto, o foco desta empresa é nos Estados Unidos, sendo que por essa razão, este recurso se encontra limitado ou inexistente para outras localizações [32].

A figura 12 dá-nos uma imagem fornecida por este serviço.

Figura 12 – Exemplo de imagem *MapQuest.*

3.4.5. YANDEX MAPS

Yandex é uma empresa tecnológica com foco no desenvolvimento de produtos e serviços de informação e pesquisa.

Um dos pontos de aplicação desta empresa é o fornecimento de informação relativa a mapas, apresentando um leque variado deste tipo serviços em forma de API. Entre eles destacam-se a geração de mapas, a localização de pontos de interesse, o cálculo de rotas e até a representação do fluxo de tráfego. Estas APIs estão desenvolvidas de modo a serem utilizadas através da sua importação por *JavaScript* ou através de mapas estáticos extensivamente configuráveis.

A representação do fluxo de tráfego apenas está disponível em forma de mapa estático. Isto significa que a sua integração para um mapa interativo é na prática impossível mesmo com as bibliotecas de mapas enunciadas anteriormente. Estas, funcionam através de imagens estáticas associadas a uma linha, coluna, *zoom* e tamanho de imagem, em que as imagens criadas pela API *Yandex* estão por sua vez associadas a uma latitude, longitude e *zoom*. Assim a associação dos diferentes parâmetros enunciados é um processo extremamente difícil e moroso.

Estes serviços poderão ser utilizados gratuitamente sem a necessidade de um registo atendendo, no entanto, a algumas condições: uma aplicação desenvolvida com base nestes serviços não poderá ser comercializável; todos os serviços terão de ser utilizados em conjunção com os mapas *Yandex;* uma aplicação apenas poderá fazer 25000 pedidos por dia [33].

No entanto, a compra de um plano comercial desta API permite a alteração das condições enunciadas. Assim, uma aplicação desenvolvida com base nestes serviços poderá ser comercializável e dependendo do plano contratado, o número de pedidos diários possíveis aumentará (e.g: 50000 pedidos – 600,000 *rubles* ≈ 8692.12 € por ano).

Destaca-se que esta empresa se centra nos seguintes países: Rússia, Ucrânia, Bielorrússia, Cazaquistão e Turquia, o que se traduz na escassez ou inexistência de informação para outras áreas do mapa.

Um exemplo de um mapa gerado por este serviço poderá ser visualizado na figura seguinte.

Figura 13 – Exemplo de mapa da *Yandex Maps.*

3.4.6. HERE

HERE é uma empresa especializada na construção de diversas soluções para topografia e mapeamento. Oferece a possibilidade de utilizar as suas APIs e SDK, para além dos seus produtos específicos.

Estas APIs podem ser acedidas através de *JavaScript* ou através de pedidos REST. Ambos os métodos poderão ser utilizados para a grande maioria das APIs, tais como: listagem de pontos de interesse, pesquisa de endereços, representação do fluxo de tráfego, *Geofencing* (verificação se numa determinada área existem pontos previamente estabelecidos), entre outras.

No caso das APIs *JavaScript*, note-se que estas terão de iniciar em primeira instância um mapa interativo, a que posteriormente são adicionadas camadas com os recursos das restantes APIs.

No caso dos pedidos REST a resposta será feita em forma de objeto JSON, não sendo necessário utilizar nenhum mapa para obter esta informação. No entanto, a HERE oferece uma solução REST para o carregamento de mapas que poderá ser utilizada numa biblioteca como *Leaflet* ou *OpenLayers*.

Finalmente existem algumas APIs apenas acedíveis através de pedidos REST, nomeadamente *Toll Cost* API (Custo de portagens) e *Geo Coder Autocomplete* API (Apresenta sugestões de pesquisa). Estas APIs são exclusivas a utilizadores com um plano customizado (de valor elevado, a acertar de acordo com os requerimentos de utilização).

Esta solução necessita do uso de uma chave de API, apenas disponível de forma gratuita por 90 dias após o registo. Nesse período todos os serviços estarão disponíveis para teste [34]. A figura seguinte representa uma imagem fornecida pelo serviço da HERE.

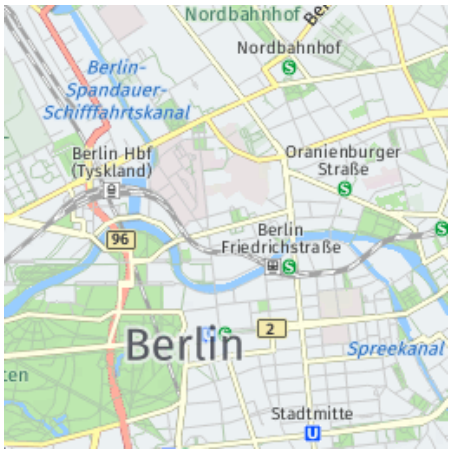


Figura 14 – Exemplo de imagem HERE *Maps.*

3.5. FUSO HORÁRIO

Um dos requisitos deste projeto é a visualização dos fusos horários correspondentes às localidades representadas nos mapas. Para tal é necessário o uso ou de uma API ou biblioteca com a informação necessária.

Nesta subsecção irão ser abordadas as possíveis escolhas para obter esta informação, aferindo-se posteriormente a que mais se adequa ao projeto a desenvolver.

3.5.1. GOOGLE *MAPS TIME ZONE* API

Como referido anteriormente a Google *Maps* disponibiliza uma grande variedade de serviços, entre os quais a informação de fuso horário de uma determinada localização.

A *Time Zone* API funciona através de pedidos REST usando como parâmetros a chave de API, um par de coordenadas e uma data. A data deverá ser apresentada em segundos contabilizados a partir da meia-noite de 1 de Janeiro de 1970, (esta data apenas é utilizada como referência da data atual, para efeitos comparativos). A sua resposta apresenta a diferença em segundos entre a data enviada como parâmetro e a data correspondente no local especificado. A resposta apresenta ainda a diferença em relação ao horário de verão (também em segundos).

Os valores apresentados poderão ser positivos ou negativos, indicado a operação a efetuar para obter a data do local especificado [35].

3.5.2. MOMENT.JS

É uma biblioteca de *JavaScript* que contem a informação dos fusos horários de acordo com a localização. Permite diversos formatos de datas e apresenta funções de comparação e conversão de fusos horários. Para requerer um fuso horário é necessário passar a data a comparar e a localização (País/Cidade).

Esta biblioteca está mais direcionada para a conversão e formatação da data e apesar de ser vantajosa, (dado que não necessita de registo nem de uma chave para a API), apenas permite a busca de fusos horários por País/Cidade limitando bastante a precisão do fuso horário [36].

3.5.3. *GEONAMES*

Geonames é uma base de dados geográfica com cobertura pelo mundo inteiro. Apresenta diversos serviços *web* REST, entre eles, informação do fuso horário. Para aceder a essa informação é necessário fazer um pedido REST à API com os valores de latitude e longitude correspondentes. Requer um registo, mas não necessita de uma chave de API, sendo que apenas o nome de utilizador é incorporado como parâmetro. Para além do fuso horário, responde também com os momentos de nascer e pôr do sol. Esta base de dados tem a vantagem de fazer os pedidos com base em coordenadas geográficas (maior precisão), mas requer um registo [37].

3.5.4. *TIMEZONEDB*

TimeZoneDB disponibiliza uma base de dados para o fuso horário de diferentes localizações no mundo.

A informação está disponível para *download* em tipo *Comma-separeted Values* (CSV) ou *Structed Query Language* (SQL). É também possível utilizar uma API para fazer pedidos REST. Esta API a mesma está limitada a 1 *query* por segundo para uma utilização grátis e requerendo também registo de modo a obter-se uma chave de API.

A pesquisa pode ser feita por coordenadas geográficas, zona, cidade, endereço IP. Os dois últimos métodos mencionados apenas estão disponíveis para utilizadores que pagam o serviço [38].

3.6. PREVISÃO METEOROLÓGICA

Um ponto importante a desenvolver no projeto é à representação da informação relativa à meteorologia nos diversos pontos do mapa. Aqui novamente será necessário o uso de uma biblioteca ou de uma API que faça a gestão e disponibilize esta informação.

3.6.1. *OPENWEATHERMAP*

O *OpenWeatherMap* disponibiliza uma API com a informação do tempo para diferentes localizações. Dispõe de informação meteorológica momentânea, assim como previsões meteorológicas a 5 dias (informação separada de 3 em 3 horas) e a 16 dias (informação separada diariamente).

É necessário efetuar um registo bem como o uso de uma chave de API, sendo que o livre acesso está limitado a 60 *queries* por minuto.

Permite a pesquisa por nome de cidade, coordenadas geográficas, código postal e por área retangular. Responde com valores de temperatura e humidade do ar, pressão atmosférica, altura pluviométrica, velocidade do vento e nebulosidade [39].

3.6.2. *ACCUWEATHER*

AccuWeather é uma interface de informação meteorológica, sendo que disponibiliza uma API para o uso externo dessa informação.

Contém informação de localização, fuso horário, condições meteorológicas, alertas e previsões. Necessita de um registo e do uso de uma chave de API, sendo que o acesso à API apenas é de livre acesso durante 6 meses.

Para a poder fazer um pedido de informação meteorológica é necessário primeiro fazer um pedido à API de localização para este responder com um identificador dessa localização. Posteriormente, este é utilizado para um novo pedido à API de meteorologia. Esta envia uma resposta com a informação do fuso horário e localização, bem como informação de temperatura e previsões meteorológicas [40].

3.6.3. *WEATHER UNDERGROUND*

A *Weather Underground* é outra interface que contém diversos tipos de informação meteorológica. Disponibiliza uma API que funciona através da realização de pedidos REST, requerendo um registo e o uso de uma chave de API para executar os pedidos.

Apresenta serviços de localização, previsões meteorológicas, *webcams* de estações meteorológicas, astronomia e almanaque. Aceita parâmetros de cidade, coordenadas geográficas, códigos de aeroporto e endereços IP [41].

3.6.4. *SIMPLEWEATHER.JS*

SimpleWeather.js trata-se de uma biblioteca de *JavaScript* de baixo peso, que importa informação meteorológica com base em pedidos à API do Yahoo.

Permite obter a informação com base no nome da localização ou nas suas coordenadas geográficas. A sua resposta contém diversos dados meteorológicos, nomeadamente a temperatura e humidade do ar, velocidade do vento e altura de precipitação, fazendo a sua previsão até 10 dias [42].

3.6.5. HERE *WEATHER* API

A HERE *Weather* API é uma das diversas API disponibilizadas pela HERE, focada nas condições meteorológicas. Este tipo de API apenas está disponível através de pedidos REST, não disponibilizando uma versão *JavaScript* ao contrário da HERE *Places* API e da HERE *Map Tile* API. Esta API permite a pesquisa por nome de localidade e coordenadas geográficas, respondendo com dados de temperatura, precipitação, velocidade do vento e pressão atmosférica.

Para o uso desta API é necessário ainda o uso de *JSON with padding* (JSONP). O JSONP corresponde a um método de transmissão de dados JSON utilizando uma tag "*script*" em vez de um objeto "*XMLHttpRequest"*. O uso de JSONP é necessário para permitir partilha de recursos entre domínios diferentes (*Cross-Domain Requests*) [43].

3.6.6. *WORLD WEATHER ONLINE*

World Weather Online apresenta informação meteorológica de diversas localidades no mundo. Esta informação pode ser acedida utilizando a sua API que apresenta dados meteorológicos atuais e de previsão, assim como dados meteorológicos para localizações montanhosas, marítimas ou aéreas.

Necessita de um registo e o uso de uma chave de API e o seu acesso grátis está limitado a 6 dias.

Permite a pesquisa por nome de cidade, coordenadas geográficas, código postal e por endereço IP. Responde com valores de temperatura e humidade do ar, pressão atmosférica, volume de pluviométrico, velocidade do vento e nebulosidade [44].

3.7. PONTOS DE INTERESSE NO MAPA

Um ponto relevante do desenvolvimento da aplicação pretendida é a adição de pontos de interesse (postos de combustível, áreas de descanso, transportes, cafés, etc) no mapa. Para ter acesso a esta informação, é necessário o uso de uma API que a contenha.

Note-se que as APIs aqui referidas já foram abordadas anteriormente, sendo que as principais empresas associadas a este serviço são também prestadores de serviços de mapas. Deste modo, esta subsecção focar-se-á no modo de funcionamento e nas principais restrições das APIs.

3.7.1. GOOGLE *PLACES* API

Google *Places* API contém a informação de localização e outros detalhes sobre pontos de interesse.

O pedido é feito através das coordenadas geográficas e de um raio associado, sendo que pode ser ainda especificado o tipo de ponto de interesse a pesquisar. A resposta contém informação da morada, assim como o nome, horário de funcionamento e classificação [45].

3.7.2. HERE *PLACES* API

A HERE disponibilizou uma API com informação sobre pontos de interesse para implementação conjunta com os mapas HERE.

Esta API está disponível em duas modalidades, REST e *JavaScript*. Caso se deseje integrar esta API com uma biblioteca de carregamento de mapas, é necessário utilizar os pedidos REST. A versão *JavaScript* está disponível apenas através da implementação da versão *JavaScript* da HERE *Maps* API.

A API recebe como parâmetros as coordenadas geográficas em forma de ponto, caixa, círculo ou polígono. Pode ser ainda especificado o tipo de ponto de interesse (e.g. áreas de descanso, cafés, restaurantes, aeroportos, etc), assim como pode ser feito um pedido cuja resposta será na forma de sugestões. A resposta contém o nome, morada e o horário de funcionamento, e no caso das estações de combustível poderá obter-se a informação do preço dos combustíveis (caso esteja disponível).

Para o uso desta API é necessário ainda o uso de JSONP de modo a poder fazer-se uma partilha de recursos de domínios diferentes (*Cross-Domain Requests*) [46].

3.7.3. BING *LOCATIONS* API

A *Locations* API do Bing, é uma API direcionada para a informação de localizações geográficas através do uso de pedidos REST. Esta API poderá receber como parâmetros a morada, um ponto geográfico ou um nome a procurar [47].

3.7.4. YANDEX PLACES API

Yandex disponibiliza um serviço *web* com informação de pontos de interesse, na forma de uma REST API, dependente de uma chave de API obtida após o registo.

É possível fazer uma pesquisa por nome ou morada, ou ainda por negócio através do nome, número de telefone ou tipo de serviço do negócio. Há ainda a opção de fazer esta pesquisa por coordenadas geográficas. Este serviço dá-nos acesso a informação do nome, localização, categoria, número de telefone, horário de funcionamento e se é aceite ou não cartão de crédito na localização pesquisada.

É importante sublinhar que este serviço está direcionado para um público-alvo, sendo que a informação disponível corresponde maioritariamente à zona da Rússia, tendo informação muito escassa em relação a outros países [48].

3.8. GERAÇÃO DE GRÁFICOS

Um dos objetivos deste projeto é a visualização de indicadores de eficiência de cada empresa/cliente. Nesse sentido, foi proposta uma forma intuitiva de fazer a sua visualização, pela utilização de gráficos. Esta subsecção explora diferentes bibliotecas para a elaboração de gráficos (através de código *JavaScript*).

3.8.1. CHART.JS

O Chart.js é uma biblioteca de desenho gráfico, *open-source* e baseada em *JavaScript*, que permite a criação de diversos tipos de gráficos, nomeadamente: linhas, barras, estrela, circulares, polares, espalhamento e bolhas.

Os gráficos criados poderão ser personalizados através de diversas opções tais como a cor dos seus constituintes, o nome e escala dos eixos e até a informação a apresentar nos diversos pontos (visível quando o cursor está em cima dos pontos) [49].

Esta biblioteca é a menos completa das opções listadas, em que as funcionalidades que inclui, estão presentes também nas outras opções a seguir apresentadas. Assim, na abordagem às próximas bibliotecas destacar-se-ão as funcionalidades particulares.

3.8.2. AMCHARTS

A AmCharts é uma empresa especializada na construção de *software* de apoio à visualização de dados.

A sua diversidade de gráficos é aparente, destacando-se os gráficos *Open-high-low-close* (OHLC) e *Mekko* (ou gráfico de mosaico).

Os gráficos OHLC são caracterizados pelo uso de valores variáveis, isto é, cada valor apresentado no mapa é representado por uma linha, resultante dos limites de variação do valor.

Os gráficos *Mekko* são gráficos semelhantes a gráficos de barras, mas com a característica de se poderem empilhar, possibilitando assim a análise de 2 variáveis no mesmo gráfico.

Os gráficos amCharts apresentam-se como uma das soluções mais completas em termos de funcionalidade. No entanto, o uso desta biblioteca implica um pagamento. Assim, tendo em conta que as funcionalidades específicas desta biblioteca não são obrigatórias na elaboração deste trabalho, considera-se desnecessário o pagamento deste serviço [50].

3.8.3. GOOGLE *CHARTS*

O Google *Charts* é a opção do Google para a representação de gráficos. É apresentada na forma de uma biblioteca de *JavaScript*, definindo uma série de classes para o uso destes gráficos. Estes apresentam um vasto leque de tipos: linhas, barras, estrela, cronológicos, circulares, polares, cascata, arvore, espalhamento, bolhas, entre muitos outros.

Para além das inúmeras possibilidades de seleção de gráficos, o *Google Charts* disponibiliza também um conjunto de opções de modo a estilizar os gráficos de acordo com o desejo do utilizador. Uma das opções mais peculiar e apenas associada a este serviço é a possibilidade do uso de uma mira que localiza o utilizador no gráfico.

Ao contrário da maioria dos serviços da Google, este serviço é grátis, podendo ser utilizado por qualquer utilizador que apresente as devidas licenças [51].

4. DESENVOLVIMENTO

Neste capítulo será feita uma descrição da solução desenvolvida, tendo em conta os requisitos enunciados na secção **1.6**. Na primeira parte serão descritos os casos de uso, bem como fluxogramas funcionais que descrevem o decorrer da aplicação desenvolvida. Posteriormente, são elaboradas as suas diferentes funcionalidades gráficas correspondentes ao cliente. Após isto são caracterizados os servidores utilizados através da descrição da base de dados utilizada, seguida das diferentes APIs necessárias para o desenvolvimento. Em seguida, é descrita a componente lógica da aplicação (servidor da aplicação), caracterizando cada controlador utilizado para aceder aos diversos recursos (bases de dados e APIs). Finalmente, serão feitas algumas considerações relativas ao desenvolvimento, destacando-se as dificuldades/obstáculos que possam surgir, bem como os métodos para os superar.

4.1. DIAGRAMAS DE FUNCIONAMENTO

De modo a melhor entender os diversos componentes desenvolvidos na aplicação, foram elaborados fluxogramas de funcionamento das diversas fases da aplicação.

4.1.1. CASOS DE USO

O primeiro passo para a definição das funcionalidades a desenvolver, foi a criação de um diagrama de casos de uso para a aplicação. A figura 15 representa o diagrama de casos de uso para a aplicação desenvolvida.

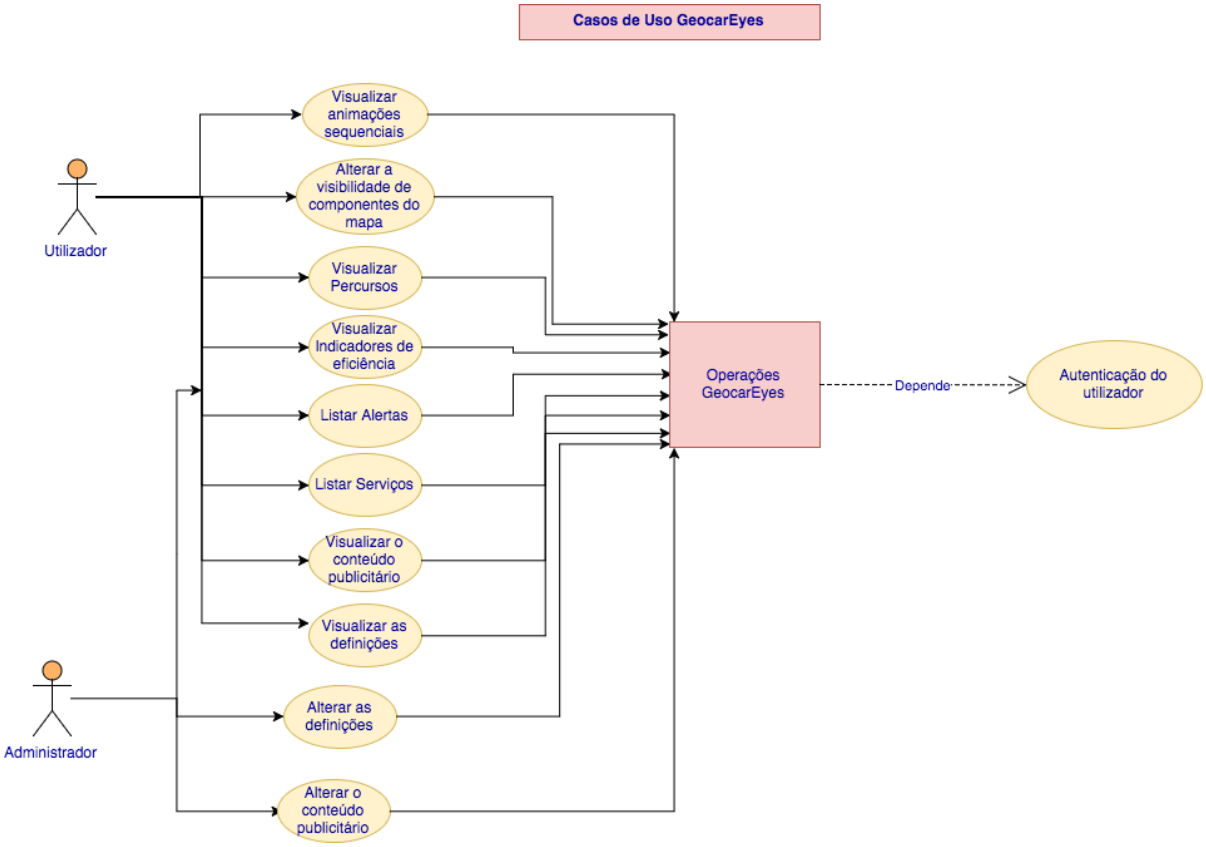


Figura 15 – Diagrama de casos de uso da aplicação GeocarEyes.

Como é possível observar, os utilizadores têm acesso à alteração da visibilidade de componentes do mapa, à visualização de percursos, indicadores de eficiência, alertas e serviços. A visualização sequencial das animações apenas acontece no modo Animado descrito com mais detalhe na secção **4.2.2**.

Os utilizadores podem ainda listar o conteúdo publicitário e as definições e o ator "Administrador" apenas difere de um utilizador normal, pela sua capacidade de alteração das definições e do conteúdo publicitário.

É de notar ainda que este conjunto de operações da aplicação GeocarEyes é sempre dependente da autenticação do utilizador.

4.1.2. FLUXOGRAMAS FUNCIONAIS

De modo a descrever o funcionamento da aplicação, foram criados fluxogramas funcionais. Estes encontram-se divididos em diferentes subprocessos. Cada subprocesso será abordado com mais detalhe nos próximos subtópicos.

Funcionamento geral

O funcionamento geral da aplicação poderá ser descrito por dois processos principais:

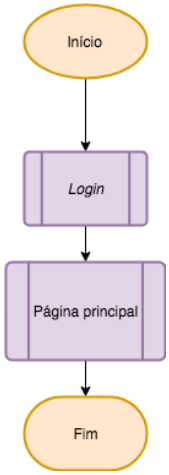


Figura 16 – Fluxograma base da aplicação GeocarEyes.

Um utilizador terá de fazer *login* e posteriormente tem acesso a uma página principal que contém todas as funcionalidades da aplicação.

Login

O funcionamento da página de *login* poderá ser descrito da seguinte maneira:

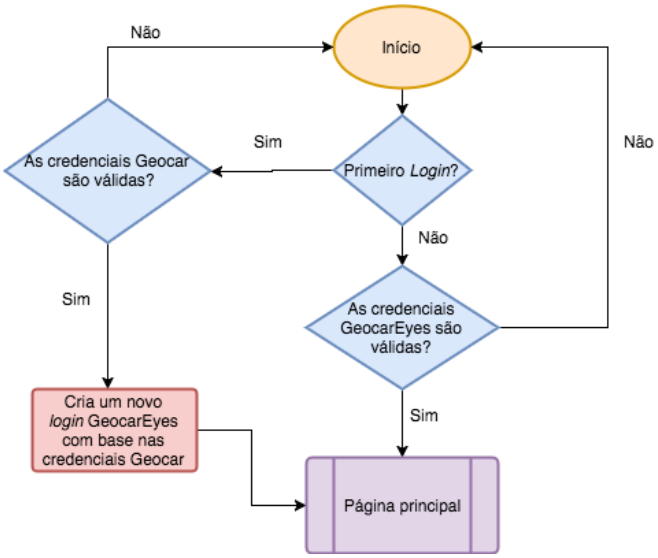


Figura 17 – Fluxograma do processo de *login*.

Como foi possível observar na figura 17, a aplicação é dependente de um registo prévio no Geocar. Isto deve-se ao facto de a aplicação ser dependente da informação contida no Geocar. Assim, o programa importa as credenciais Geocar e cria um novo *login* no GeocarEyes.

Caso não seja o primeiro *login* a aplicação irá buscar as credenciais GeocarEyes, o que acontece por ser possível a alteração de *password* para a aplicação GeocarEyes e, portanto, as credencias das duas aplicações poderem não coincidir.

O programa verifica as credenciais e caso estas sejam válidas redireciona o utilizador para a página principal.

Página principal

A figura seguinte ilustra o funcionamento da página principal.

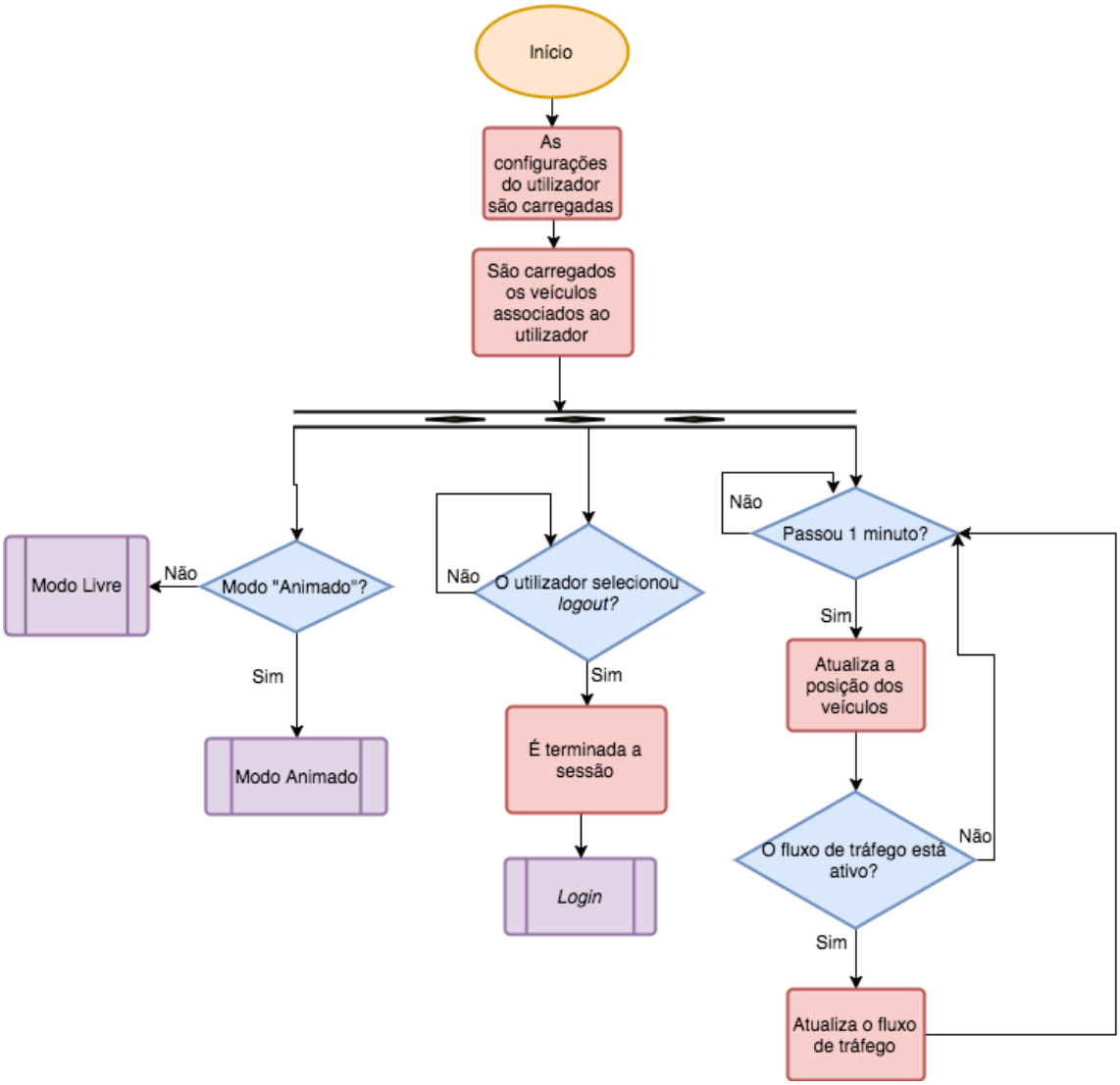


Figura 18 – Fluxograma do processo da página principal.

A página carrega as definições do utilizador e em seguida carrega o mapa com os veículos associados ao utilizador. O programa fica então a decorrer em 3 processos: o primeiro é a atualização dos veículos e do fluxo de tráfego (caso esta opção esteja selecionada) com um período de um minuto; o segundo corresponde ao término de sessão, que caso seja selecionado redireciona o utilizador para a página de *login*; finalmente o programa verifica o modo selecionado e inicia um dos dois processos, modo Livre ou modo Animado, de acordo com as definições.

Modo Livre

O modo Livre poderá ser descrito pela figura 19:

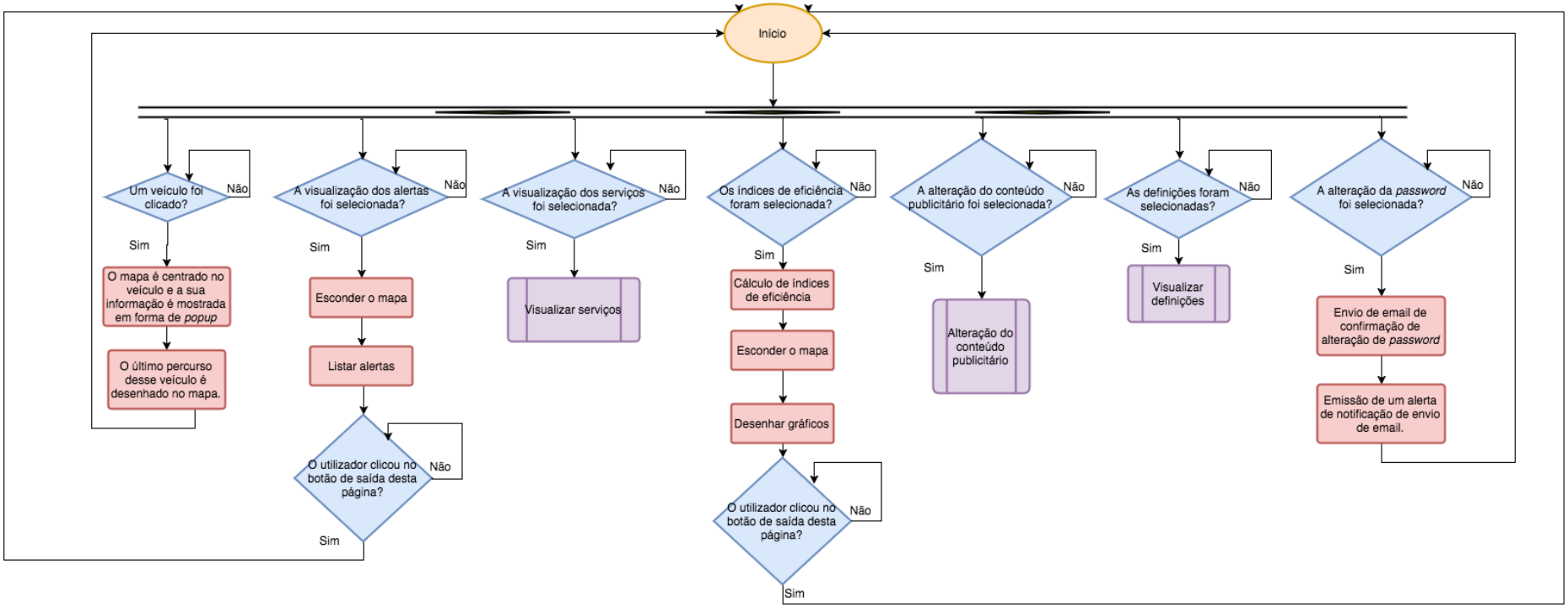


Figura 19 – Fluxograma do modo Livre.

O modo Livre divide-se em 7 processos que correspondem à seleção de diferentes componentes da página:

- **Veículo –** Caso um veículo seja selecionado, o mapa é centrado no mesmo, abrindo um *popup* com a informação associada. O último percurso desse veículo é também desenhado.

- **Alertas –** A seleção dos alertas esconde o mapa do utilizador, carregando uma lista dos últimos alertas (até 30 dias atrás). O programa fica então à espera que o utilizador saia desta página.

- **Alteração da *password* –** A seleção desta opção corresponde a um pedido de alteração de *password.* A aplicação envia um email de confirmação para o utilizador, contendo um link para página de alteração de *password* (apresentada de mais abaixo). O programa emite também um alerta a informar o utilizador do envio do email.

- **Índices de eficiência –** A seleção dos índices de eficiência despoleta um processo de cálculo dos mesmos. Este é feito antes do mapa ser escondido pois poderá ser um processo moroso. Seguidamente, serão desenhados gráficos de acordo com os cálculos efetuados, ficando a página à espera da saída do utilizador.

Os processos despoletados pela seleção dos serviços, da alteração do conteúdo publicitário e das definições são de um grau de complexidade maior e, portanto, serão discutidos de forma independente posteriormente.

Visualização dos serviços

A visualização dos serviços poderá ser caracterizada pelo seguinte fluxograma:

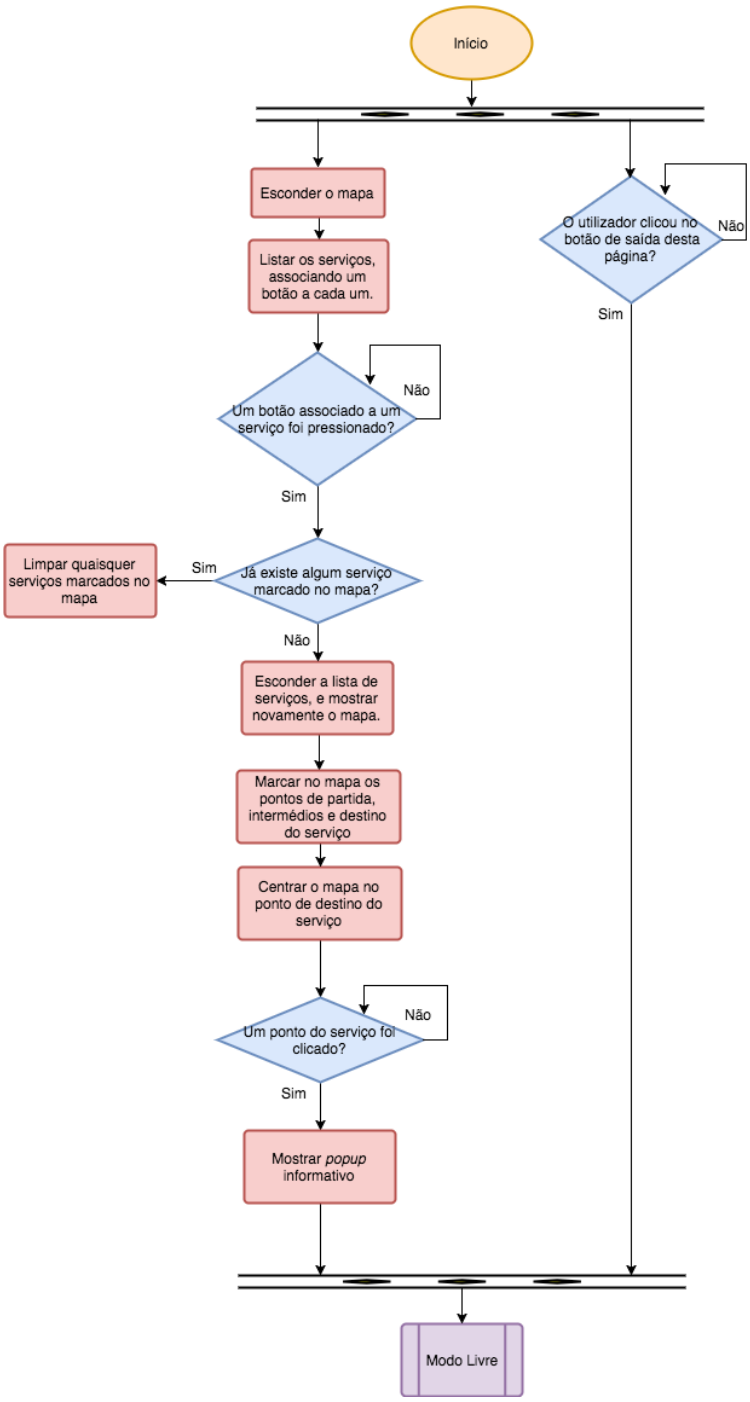


Figura 20 – Fluxograma da visualização de serviços.

Como é possível observar, a aplicação começa por esconder o mapa, listando de seguida todos os serviços com um botão associado. O programa fica então à espera do acionamento de algum dos botões. Quando isto acontece é primeiro verificado se existe algum serviço previamente representado no mapa e caso exista os marcadores associados a este são limpos. Posteriormente a lista de serviços é ocultada mostrando novamente o mapa. São então marcados no mapa todos os pontos relevantes do serviço selecionado (origem, destino e pontos intermédios). Este são ainda clicáveis, mostrando um *popup* com a sua informação.

A qualquer momento do decorrer deste processo, o utilizador poderá interromper o seu funcionamento, saindo da página de serviços.

Alteração do conteúdo publicitário

O processo de alteração do conteúdo publicitário encontra-se caracterizado no fluxograma seguinte.

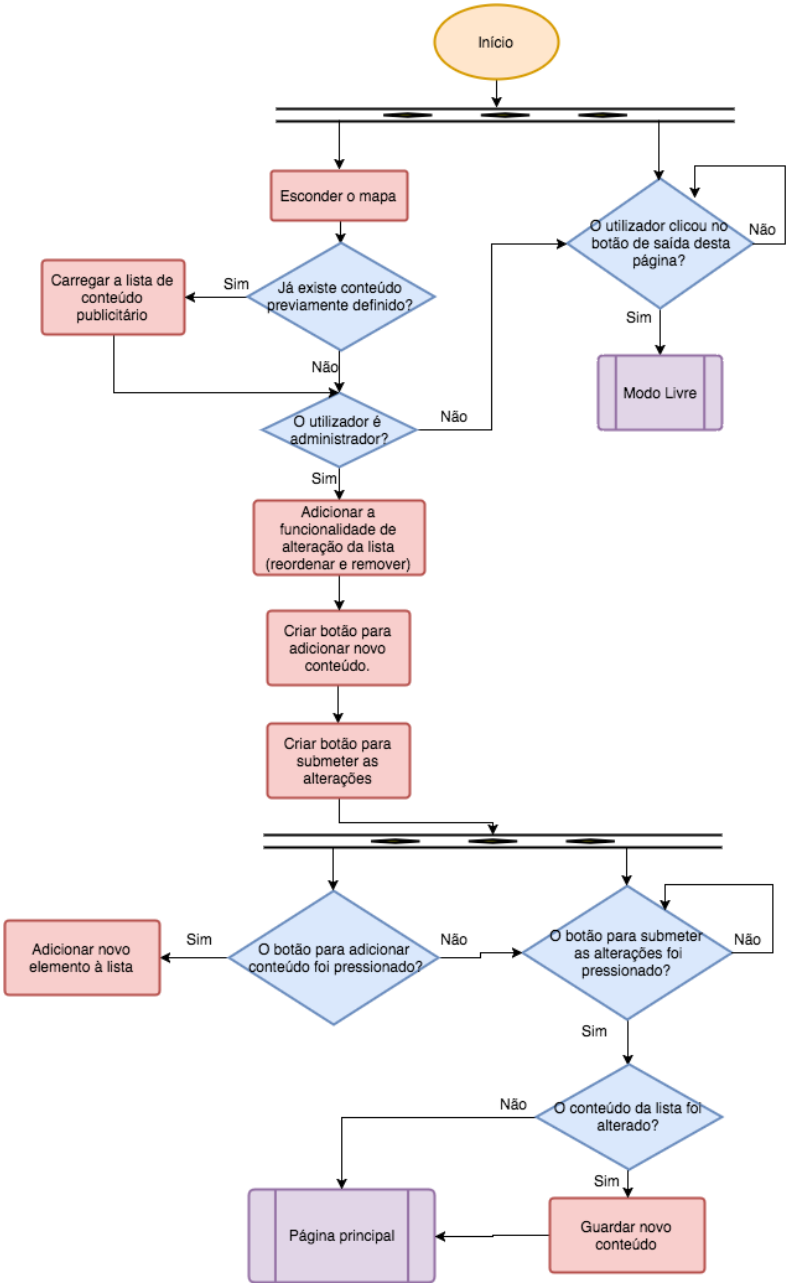


Figura 21 – Fluxograma do processo de alteração de conteúdo publicitário.

Esta página está direcionada para um utilizador administrador, significando que um utilizador comum terá pouco acesso às funcionalidades da página. Apenas poderá visualizar a lista previamente definida ou sair da página.

Para um utilizador administrador são adicionadas outras funcionalidades como a alteração do conteúdo previamente inserido (reordenação da lista e eliminação de elementos) e a possibilidade de adicionar novo conteúdo.

Após as alterações desejadas o utilizador poderá pressionar o botão de submissão para guardar as alterações. Neste momento, o programa analisa se efetivamente aconteceram alterações na lista e caso tenham acontecido, grava-as. No entanto, se o utilizador navegar para fora da página, as alterações serão perdidas. Finalmente o utilizador é redirecionado para a página principal (é necessário que a página principal seja recarregada de modo a que as novas configurações se tornem efetivas).

Definições

Este processo inicia-se com a ocultação do mapa, seguida do carregamento das definições previamente estabelecidas. Para além destas, são também listadas as opções alternativas para cada funcionalidade configurável.

Em seguida, é feita a verificação do nível do utilizador e caso este corresponda a um administrador, é gerada a opção de alteração das definições. O programa fica então à espera da submissão das novas alterações, guardando-as após a ação do botão de submissão.

Finalmente o utilizador é redirecionado para a página principal, recarregando-a novamente de modo a tornar as alterações efetivas.

A página das definições pode ser descrita funcionalmente pela figura apresentada de seguida.

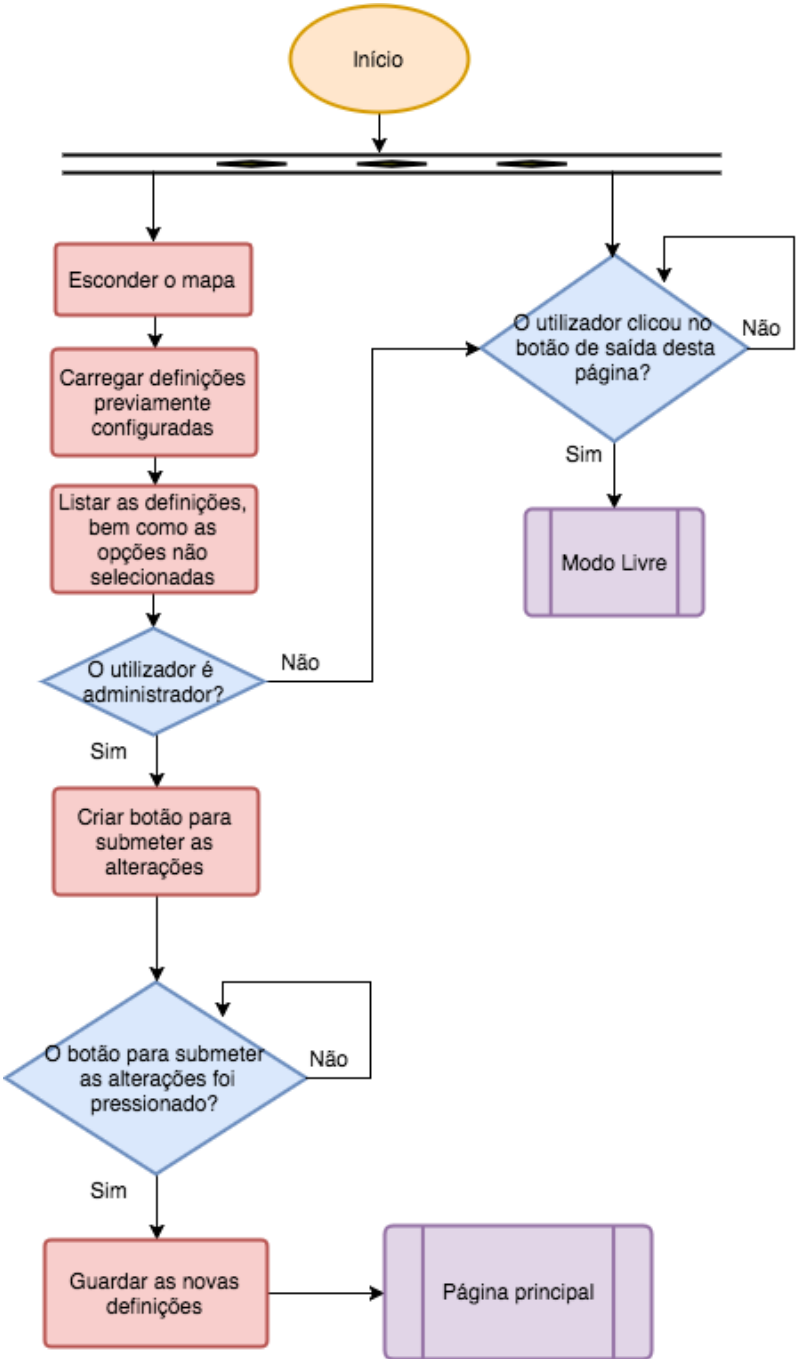


Figura 22 – Fluxograma do funcionamento da página das definições.

Modo Animado

O modo Animado corresponde à visualização de animações sequenciais descritas nos casos de uso. Este poderá ser descrito pelo seguinte fluxograma:

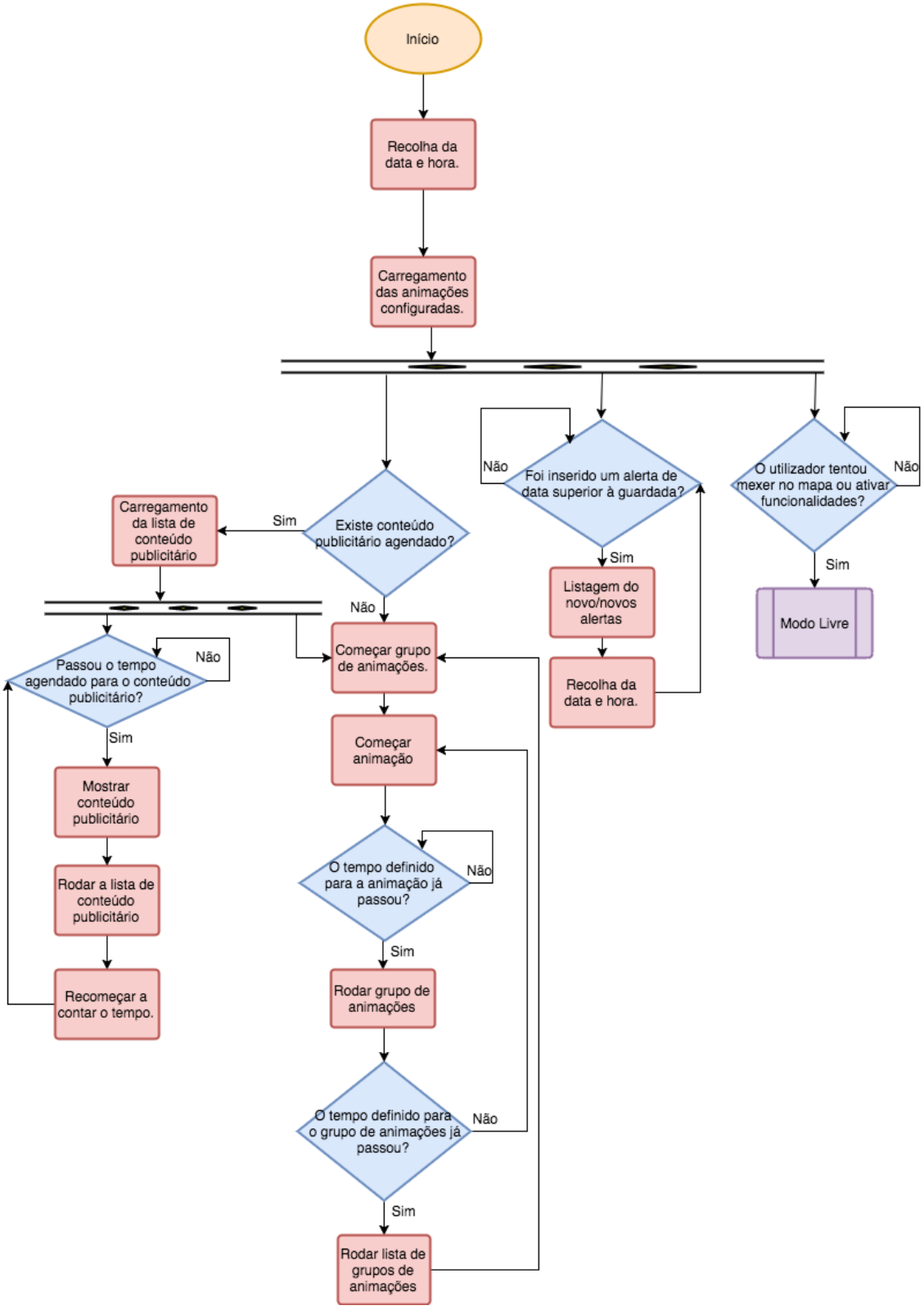


Figura 23 – Fluxograma do modo Animado.

Quando este modo é selecionado, a primeira ação do programa é a recolha da data e da hora. De seguida são carregadas as animações configuradas e caso exista conteúdo publicitário agendado, esta lista também será carregada.

Inicia-se então um grupo de animações que contém várias séries de animações. O programa executa cada animação da primeira série e rodando-a quando a animação termina. É então verificado se a série de animações foi totalmente percorrida, rodando o grupo de animações se isto se verificar. No caso de o grupo de animações não ter sido finalizado, é iniciada uma nova animação da série iniciada.

Durante a execução das animações, o programa poderá interrompê-las, mostrando o conteúdo publicitário agendado numa *modal box* (uma caixa centrada no ecrã, tornado o fundo da página opaco).

Finalmente, o programa executa também uma verificação periódica da inserção de um novo alerta, com base na data e hora recolhida no início da aplicação. Caso sejam detetados novos alertas, é feita a sua listagem e é recolhida a data e a hora do último gerado (de modo a evitar a repetição do alerta).

Destaca-se que neste modo não é suposto existir interação do utilizador. Por esta razão, caso o utilizador tente interagir com a aplicação, todas as animações são interrompidas e o programa passa para o modo Livre.

Página de alteração da password

Como descrito anteriormente, quando é despoletado um pedido de alteração de *password* é enviado um email ao utilizador de modo a confirmar esta ação. Este *email* contém um *link* que permite o acesso à página de alteração de *password.*

O funcionamento desta página encontra-se representado na figura 24, apresentada de seguida.

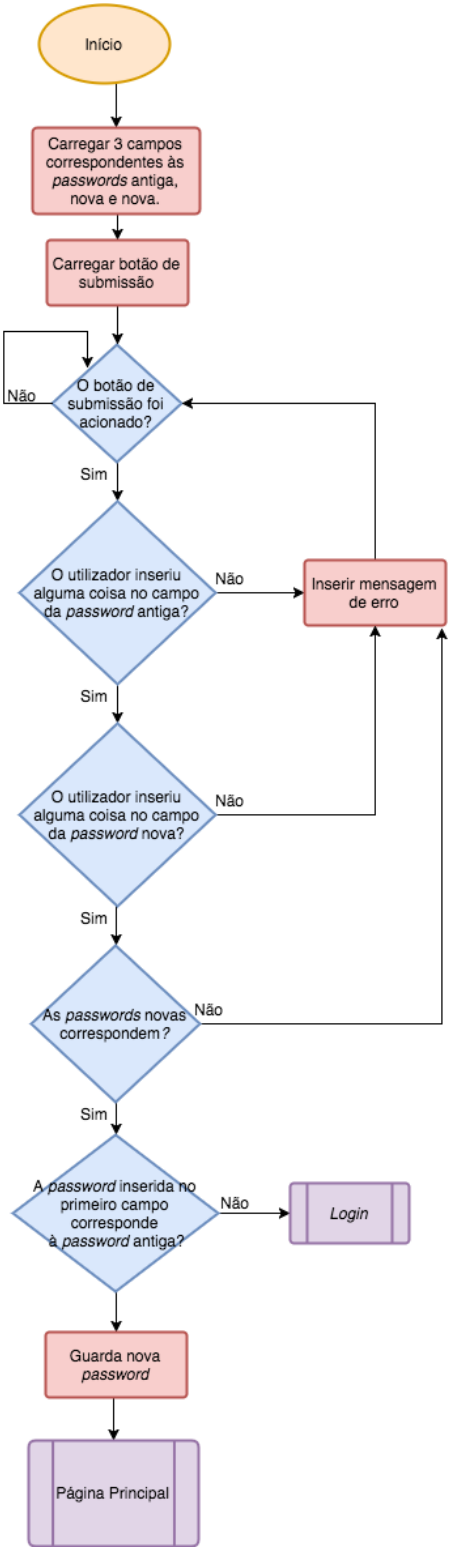


Figura 24 – Fluxograma do processo de alteração de *password*.

Esta página carrega 3 campos para a introdução de *passwords*. Estes correspondem respetivamente, à *password* antiga, à *password* nova e a uma repetição da *password* nova. De seguida é carregado um botão de submissão. No momento de tentativa de submissão, é verificado o preenchimento dos campos, fazendo-se as verificações adicionais:

- Verificação se a *password* nova tem pelo menos 8 caracteres, contendo uma maiúscula, uma minúscula e um dígito.

- Verificação se a *password* nova corresponde à repetição da *password* nova.

- Verificação se o campo para a primeira *password* corresponde à *password* antiga.

Se estas condições se verificarem, a nova *password* é guardada e o utilizador é redirecionado para a página principal.

Caso contrario, a página irá emitir uma série de erros, e o botão de submissão não será ativo. A última verificação feita implica uma conexão às bases de dados e, portanto, remete para a navegação fora da página, sendo o utilizador redirecionado para a página de *login*.

Destaca-se que o acesso a esta página é dependente de um início de sessão previamente feito. Por essa razão, o sucesso desta submissão envia o utilizador para a página principal. De igual modo, o insucesso desta operação resulta no envio do utilizador para o *login*, dado que as credenciais inseridas estão incorretas, terminando a sua sessão.

4.2. DESENVOLVIMENTO GRÁFICO

Uma das componentes mais relevantes desta aplicação é a apresentação gráfica da informação. Esta subsecção tem o intuito descrever todas as funcionalidades gráficas implementadas neste trabalho.

A aplicação poderá ser dividida em dois grupos:

- **Modo Livre** - uma página iterativa na qual o utilizador deve acionar as funcionalidades que desejar.

- **Modo Animado** - uma página animada onde as funcionalidades vão sendo despoletadas ao longo do tempo, de acordo com as configurações previamente definidas.

De acordo com os requisitos definidos, a aplicação foi desenvolvida com base num *layout* fornecido pela empresa. Este *layout* é baseado em *Bootstrap* e a sua representação poderá ser vista na figura 25 para o modo Livre e figura 26 para o modo Animado.

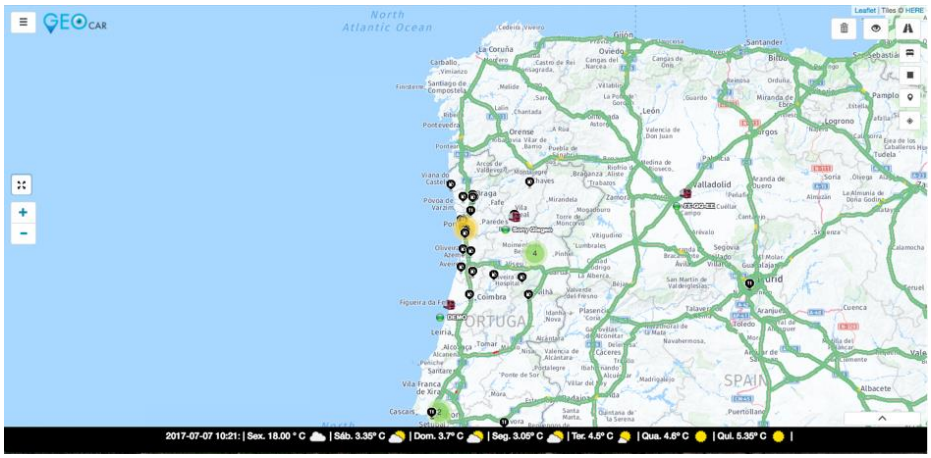


Figura 25 – *Layout* GeocarEyes no modo Livre.

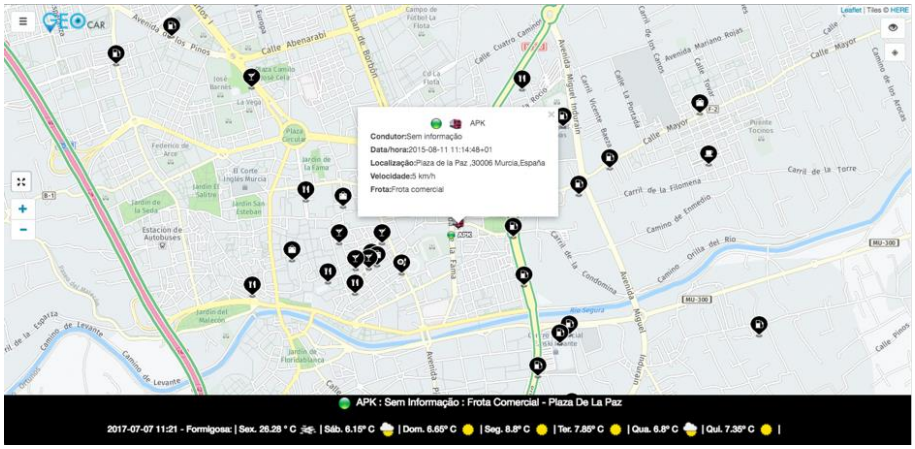


Figura 26 – *Layout* GeocarEyes no modo Animado.

Como foi visto anteriormente na secção **3.1**, o uso de *Bootstrap* é dependente de *jQuery*. Assim, dado que *jQuery* já é necessário para o desenvolvimento do trabalho, foi utilizado no trabalho diversas vezes, substituindo a necessidade de usar *JavaScript* nativo.

4.2.1. MODO LIVRE

Neste modo, o utilizador poderá mexer livremente no mapa, bem como ativar/desativar funcionalidades através de botões.

De seguida serão descritas a diferentes funcionalidades possíveis para este modo.

Botões do mapa

De forma a permitir interatividade, foram criados botões para ativação e desativação de recursos (normalmente marcadores) representados no mapa. Estes recursos estão

previamente estabelecidos nas definições e a alteração do seu estado através dos botões não será gravada. Estes botões situam-se no canto superior direito da página.

Alteração do tipo de mapa – Esta funcionalidade é auto descritiva, estando implementada através do uso de diferentes camadas *Leaflet* que são ligadas ou desligadas de acordo com a opção selecionada. Os diferentes tipos de mapas são gerados pela HERE *Maps* API (descrita na secção **3.2**) dado que o acesso à mesma já se encontra contratado pela empresa GISGEO. A figura 27 apresenta o seletor de tipos de mapas, estando representadas as diferentes opções da seleção na figura 28.

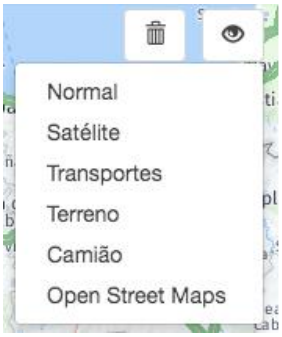


Figura 27 – Botão de alteração do tipo de mapa.

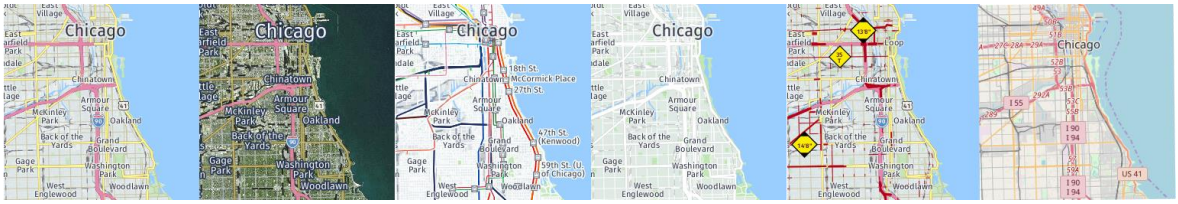


Figura 28 – Exemplos de imagens geradas pelos diferentes tipos de mapas possíveis.

***Toggle* do fluxo de tráfego –** Este botão está encarregue de controlar a visualização do fluxo de tráfego. Na figura seguinte é possível observar o aspeto deste botão.

Figura 29 – Botão de *toggle* do fluxo de tráfego.

O fluxo de tráfego corresponde a uma camada *Leaflet* independente do tipo de mapa, representada pelas linhas da estrada com gradações de cor diferentes. Estas linhas são geradas pela HERE *Maps* API e têm o aspeto seguinte:

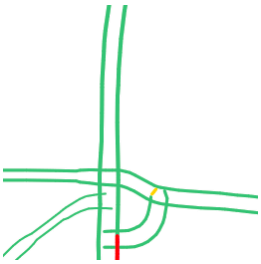


Figura 30 – Exemplo de imagem representativa do fluxo de tráfego.

Como é possível observar, apenas as linhas das estradas são representadas na imagem. Isto significa que estas imagens poderão assentar sobre qualquer mapa na localização correspondente. O nível de congestionamento é representado através de uma escala de cores:

Verde – Representa o fluxo livre, isto é, o volume de veículos é baixo, significando que a velocidade média é a espectável.

Amarelo - Representa trânsito condicionado, isto é, o volume de veículos é mais elevado, significando que a velocidade média é ligeiramente abaixo da espectável.

Vermelho – Representa a formação de filas de espera, causadas por um volume elevadíssimo de veículos, traduzindo-se numa velocidade média relativamente próxima de 0.

Preto – Representa estradas bloqueadas, devido a acidentes, construções, ou algo da mesma natureza.

Algo importante a destacar é que a natureza desta informação é volátil e, portanto, quando esta opção é selecionada, o fluxo de tráfego é atualizado com um período de um minuto.

***Toggle* das áreas de descanso –** Este botão está associado à visualização de marcadores representativos de áreas de descanso. As áreas de descanso encontram-se agrupadas numa camada *Leaflet*, tornando o controlo da sua visibilidade um processo simples. Estes pontos contêm ainda informação associada (nome, morada e horário de funcionamento, caso exista) que poderá ser vista através de um *popup* quando são clicados. A figura 31 representa o botão de controlo desta funcionalidade, estando o ícone representativo destes pontos apresentado na figura 32.

Figura 31 – Botão de *toggle* das áreas de descanso.

Figura 32 – Ícones representativo das áreas de descanso (fornecidos pela HERE *Places* API).

Para não sobrecarregar o mapa com marcadores e assim reduzir o tempo de processamento, estes pontos apenas são carregados para a área a visualizar do mapa e são desativados para níveis de *zoom* muito baixos.

***Toggle* dos postos de combustível –** De forma semelhante ao botão anterior, este botão está associado à visualização dos postos de combustível. Do mesmo modo, estes encontram-se agrupados através de uma camada *Leaflet*. Se algum destes pontos for acionado através de um clique, é mostrado um *popup* com a sua informação (nome, morada, horário de funcionamento (caso exista) e preço dos combustíveis (caso exista)). Na figura 33 poderá ser visto o botão de *toggle* desta funcionalidade. A figura 34 apresenta um marcador inserido no mapa através da referida funcionalidade.

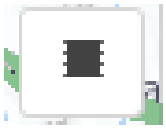


Figura 33 – Botão de *toggle* dos postos de combustível.

Figura 34 – Ícone representativo dos postos de combustível (fornecido pela HERE *Places* API).

Estes pontos apenas são carregados para a área a visualizar do mapa. Para níveis de *zoom* muito baixos (baixo detalhe do mapa) são desativados de forma a não sobrecarregar o mapa com marcadores.

***Toggle* dos veículos –** Este botão controla a visualização dos veículos no mapa. Estes correspondem a uma camada *Leaflet.* São representados através de um ícone definido pelo utilizador, pela matrícula correspondente e pelo seu estado (ligado ou desligado). Se estes ícones forem pressionados, é mostrado um *popup* com a sua informação (velocidade, data da recolha da informação e outras opções definidas). A figura 35 apresenta o botão de controlo descrito sendo a figura 36 representativa de um veículo marcado no mapa.

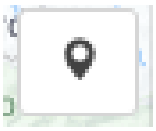


Figura 35 – Botão de *toggle* dos veículos representados no mapa.

Figura 36 – Exemplo de ícone utilizado para a representação dos veículos.

***Toggle* do centro do mapa –** Este botão serve apenas para o controlo de um ícone representativo do centro do mapa. A sua importância é evidenciada pela informação meteorológica e de fuso horário, dado que é gerada de acordo com o centro do mapa.

A figura 37 representa o controlador desta funcionalidade, estando o ícone de centro de mapa gerado na figura 38.

Figura 37 – Botão de *toggle* do marcador de centro do mapa.

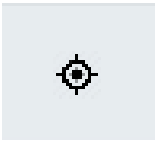


Figura 38 – Ícone para o marcador de centro do mapa.

***Toggle* de *fullscreen* –** Foi adicionado um *plugin Leaflet* (descrito com maior profundidade no ponto **4.2.3**) de modo a poder colocar o ecrã em tela cheia. Esta funcionalidade é controlada pelo botão apresentado na figura seguinte.

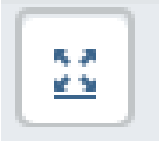


Figura 39 – Botão de *toggle* de *fullscreen*.

Limpar marcadores – Este botão apaga todos os marcadores representados no mapa. Isto significa que para reativar os marcadores apagados por este botão, é necessário utilizar os botões referidos anteriormente. A figura seguinte ilustra o botão utilizado para esta funcionalidade.

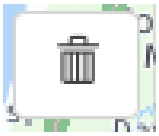


Figura 40 – Botão para limpar marcadores.

Mostrar menu lateral – De modo a não ocupar espaço desnecessário na página, o menu lateral encontra-se por defeito escondido. Este botão aciona a sua visibilidade, dando acesso às suas diferentes funcionalidades. Este botão desaparece quando é clicado, dando lugar ao botão "Esconder menu lateral" descrito de seguida. A figura 41 representa este botão.

Figura 41 – Botão para mostrar o menu lateral.

Esconder menu lateral – O uso deste botão oculta o menu lateral. Este botão desaparece quando é clicado, dando lugar ao botão "Mostrar menu lateral". A visibilidade deste botão corresponde à visibilidade do menu lateral. Na figura abaixo é possível ver este botão.

Figura 42 – Botão para esconder o menu lateral.

Menu lateral

O menu lateral contém opções que não funcionam diretamente com o mapa e requerem informação da base de dados. Por esta razão o uso destas funcionalidades ativa um pedido AJAX para obter a informação correspondente.

O menu encontra-se ainda dividido em 3 grupos: "Visualização", "Configuração" e "Eficiência". Estes grupos caracterizam o tipo de funcionalidade permitida para a informação listada. Para além destes grupos existem duas funcionalidades não agrupadas: "Sobre" que corresponde a uma página informativa da aplicação e "Sair" que corresponde à função de *logout*.

A maioria destas funcionalidades irá cobrir o mapa quando ativas (com a exceção de "Alterar password"). A figura seguinte apresenta o aspeto do menu lateral.

Figura 43 – Menu lateral.

Serviços - Esta funcionalidade pertence ao grupo "Visualização" e faz uma listagem dos serviços da empresa (até um mês atrás com base na data atual). Caso não haja serviços nas datas mencionadas é apresentada uma mensagem: "Não há serviços a apresentar".

É ainda associado um botão a cada serviço que quando clicado, gera todos os pontos relevantes do serviço (identificando com um ícone diferente a origem, destino e pontos intermédios). O mapa é então centrado no destino do serviço abrindo o *popup* que contém a nome do destino.

A figura 44 ilustra a página de listagem de serviços, estando na figura 45 a representação de um serviço no mapa.

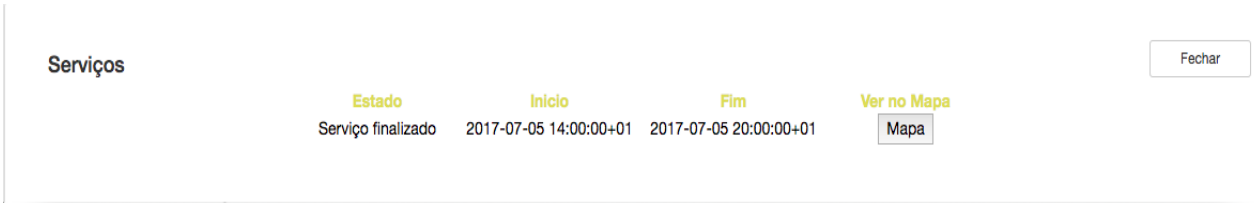


Figura 44 – Exemplo da página de serviços.

Figura 45 – Exemplo de um serviço marcado no mapa.

Alertas - Esta funcionalidade faz uma listagem dos alertas da empresa (até um mês atrás com base na data atual), pertencendo assim ao grupo "Visualização". No caso de não haver alertaş nas datas mencionadas é apresentada a mensagem: "Não há alertas a apresentar".

A lista de alertas descreve os alertas, associando cada um a um tipo, a uma data e a um veículo.

A figura seguinte ilustra a página de listagem dos alertas.

Alertas

Fechar

Viatura	Data	Tipo	Descrição
Blackberry	2013-08-16	insp	testedealerta2
Blackberry	2013-08-16	insp	testedealerta3
Blackberry	2013-08-16	insp	testedealerta4
Blackberry	2013-08-16	insp	testedealerta5
Blackberry	2013-08-16	insp	testedealerta6
Blackberry	2013-08-16	insp	testedealerta7
Blackberry	2013-08-16	insp	testedealerta8
Blackberry	2013-08-16	insp	testedealerta89

Figura 46 – Exemplo de página de alertas.

Definições – Esta opção pertence ao grupo "Configuração", correspondendo à página de configuração do funcionamento da aplicação.

O funcionamento desta página apenas é possível para utilizadores administradores, caso contrario servirá apenas para a visualização das definições.

O utilizador administrador pode definir o tipo de mapa a utilizar, podendo este ser alterado a qualquer momento do decorrer da aplicação. Pode também configurar a visualização ou não do fluxo de tráfego, das áreas de descanso e dos postos de combustível.

Pode ainda definir o número de dias a apresentar para a previsão meteorológica (atual até 6 dias mais tarde) e configurar a informação a mostrar por veículo (condutor, frota e rua).

Nesta página é possível configurar o tipo de ícone utilizado para representar os veículos. Este poderá ser um ícone já existente no servidor ou um ícone inserido pelo utilizador. Existem os 82 ícones do Geocar, podendo o utilizador administrador inserir até 10 novos ícones.

São também configuráveis as coordenadas geográficas iniciais através de um mini mapa ou pela sua inserção manual. Por defeito estas coordenadas correspondem ao distrito associado à empresa do utilizador.

A figura seguinte ilustra os tópicos até ao momento descritos, da página das definições.

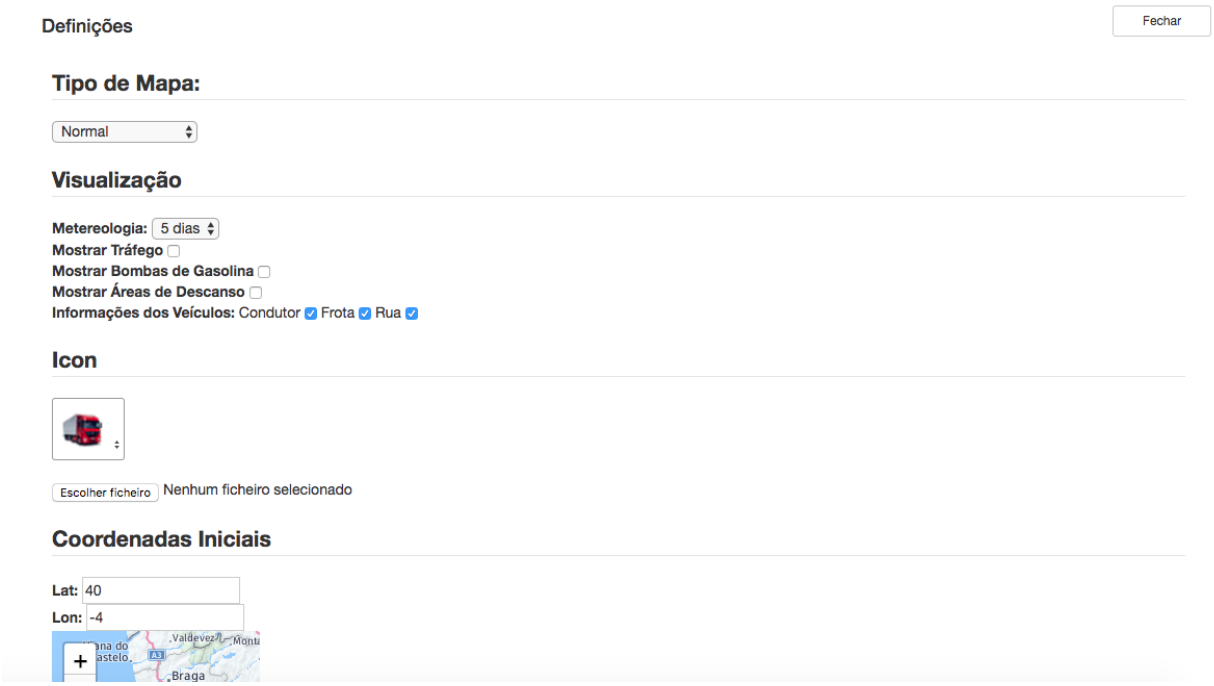


Figura 47 – Página das definições (Topo).

Finalmente, a aplicação poderá ser configurada no modo Livre ou em modo Animado.

Quando o modo Animado é selecionado, são geradas 2 cronologias, sendo possível criar grupos de animações na primeira cronologia e configurar cada animação num determinado grupo de animações na segunda cronologia.

Ao adicionar um novo grupo de animações, este terá o nome de “Nova animação x” (onde x é o índice de grupos de animações), que é editável. A duração dos grupos de animações não

é ajustável, sendo calculada com base nos elementos do grupo (somatório das durações das animações correspondentes ao grupo).

A seleção deste modo apresenta também uma tabela com todas as animações disponíveis (Veículos Ligados, Veículos Desligados, Percursos, Serviços e Indicadores de Eficiência), em que cada elemento dessa tabela é arrastável para a segunda cronologia. Quando é clicado um elemento da segunda cronologia é mostrado um *slider* de modo a ajustar o tempo de funcionamento de cada animação.

A figura seguinte ilustra a segunda parte das definições com o modo Animado selecionado, utilizado para definir o modo de funcionamento da aplicação.

Figura 48 – Página das definições (Fundo).

Quando o modo Livre é selecionado as cronologias são apagadas, bem como a tabela de animações.

Alterar *password* – Esta funcionalidade corresponde a um pedido de alteração de password. A página gera um alerta e executa um pedido AJAX para o envio de um email de confirmação de alteração de password (o conteúdo deste email será abordado na subsecção de desenvolvimento de *back-end*).

A figura abaixo ilustra o alerta gerado quando esta opção é selecionada.

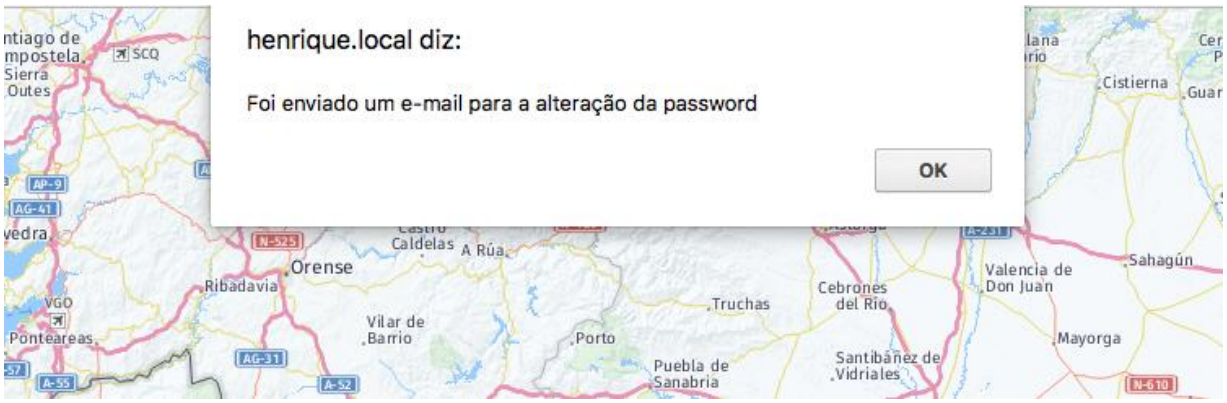


Figura 49 – Notificação de envio de email para alteração de *password*.

Agendamento de conteúdo publicitário – Esta funcionalidade pertence ao grupo "Configuração" e corresponde à página de configuração do conteúdo publicitário a apresentar durante o funcionamento da aplicação em modo Animado.

Esta página lista o conteúdo já inserido e permite a edição da mesma (reordenar, eliminar e adicionar conteúdo). É ainda possível escolher uma frequência de repetição para a apresentação do conteúdo da lista, esta é rodada cada vez que é apresentado um elemento. Quando se passa o cursor por cima de cada link da lista é apresentada uma pré-visualização do elemento da lista.

A figura seguinte ilustra uma lista de conteúdo publicitário com 2 elementos.

Figura 50 – Exemplo de lista de conteúdo publicitário.

O conteúdo é adicionado através de um *link* para o mesmo, que tem que terminar em 'PNG/JPG/GIF/JPEG' de modo a garantir que se trata de uma imagem, ou terá que ser um *link* válido para um vídeo do *Youtube*. A figura seguinte apresenta a mensagem de erro, após uma tentativa inválida de inserção de um *link*.

Figura 51 – Mensagem de erro na inserção de um *link* inválido.

Sublinha-se que a visualização do conteúdo publicitário é completamente dependente da inserção de um *link* válido. A integridade do *link* é verificada pelo programa, mas existência do mesmo não é. Nesse sentido, o utilizador deverá confirmar que o *link* existe. Tal poderá ser facilmente feito na configuração do conteúdo publicitário, dado que caso o *link* não exista, a pré-visualização do mesmo não será possível.

Kilometros – Esta funcionalidade pertence ao grupo "Eficiência" e correspondendo à página de apresentação de gráficos relativos à eficiência dos quilómetros percorridos.

Esta página gera um gráfico, onde cada barra corresponde a um dia, apresentando valores para os últimos 7 dias. O valor de cada barra corresponde à média para o correspondente dia dos quilómetros percorridos por cada veículo.

A figura seguinte mostra um gráfico gerado por esta funcionalidade.

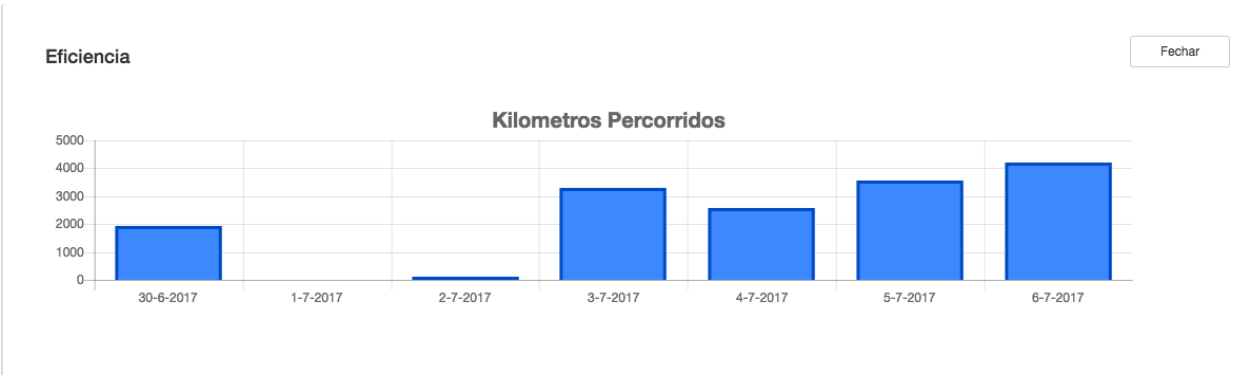


Figura 52 – Exemplo de gráfico gerado para eficiência dos quilómetros percorridos.

Combustível – Esta opção é semelhante à opção anterior, mas com foco no consumo de combustível.

Esta página gera 3 gráficos de barras, cada um correspondendo a um tipo de combustível (Gasolina 95, Gasolina 98, Gasóleo). São apresentados valores correspondentes ao consumo

dos últimos 7 dias, correspondendo cada barra do gráfico a cada dia. Este cálculo é a média de consumos dos veículos de cada tipo de combustível.

Esta funcionalidade só é aplicável para veículos com CAN-Bus (apenas veículos com CAN-Bus contêm informação relativa ao combustível), apenas sendo possível se todos os veículos de um determinado tipo de combustível já tiverem CAN-Bus instalado (mostrando o gráfico correspondente).

A figura seguinte corresponde à mensagem de erro quando os veículos da empresa em questão não têm CAN-Bus instalado.

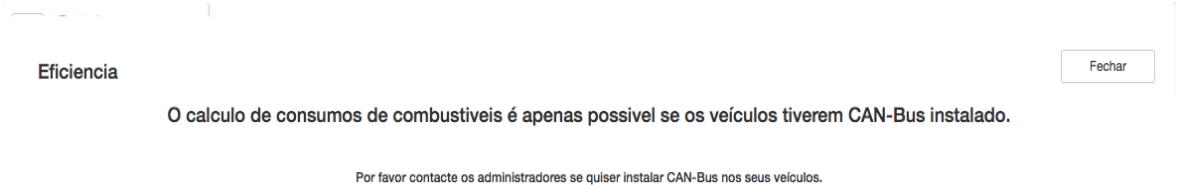


Figura 53 – Mensagem de erro para o cálculo da eficiência dos consumos de combustível.

Os gráficos gerados pelas duas funcionalidades descritas são baseados na biblioteca Chart.js (secção **3.8.1**). A escolha desta biblioteca foi feita por duas principais razões:

- Esta biblioteca apresenta todas as funcionalidades necessárias para os efeitos do projeto.
- O menor número de funcionalidades presentes nesta biblioteca, conduz a uma maior simplicidade no desenvolvimento, dado que menos opções resultam numa menor configuração dos gráficos.

Refere-se ainda que a sua natureza *open-source* foi um fator determinante na escolha.

Sobre – Trata-se de uma pequena página informativa e introdutória para a aplicação.

Esta página contém um *link* para o Geocar e um botão para ver as licenças *open-source* utilizadas. Na figura seguinte é possível ver o aspeto desta página.

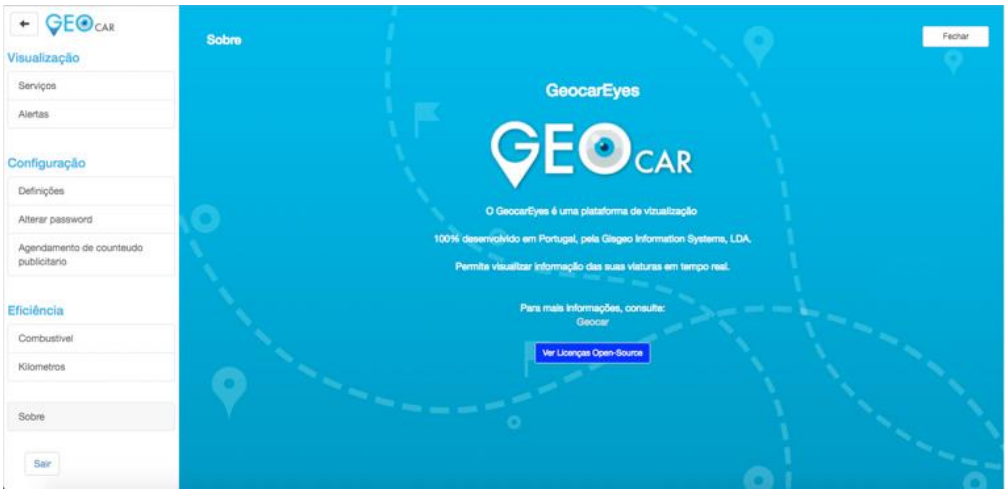


Figura 54 – Página "Sobre".

Clicando no botão, é gerada a página seguinte com as licenças *open-source*.

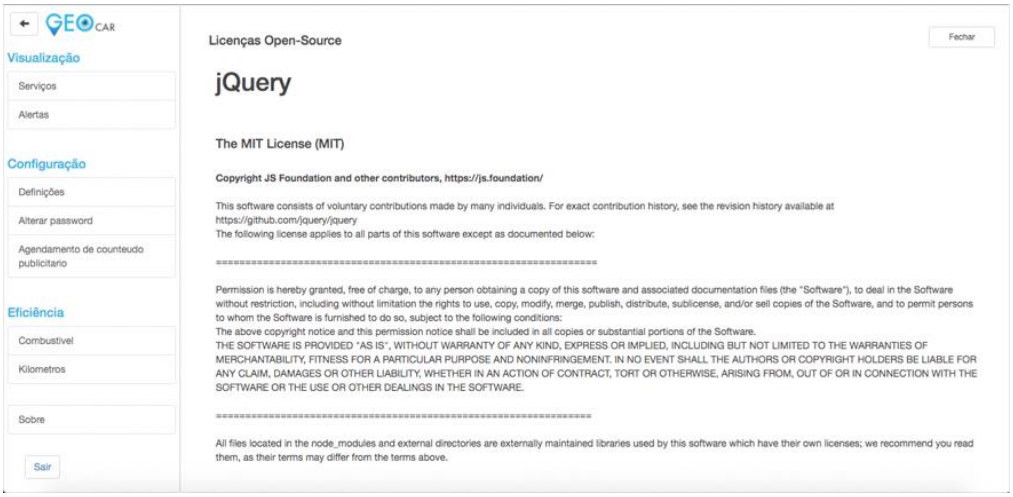


Figura 55 – Página de listagem das licenças *open-source.*

Sair – Esta funcionalidade corresponde ao processo de término de sessão e apenas redireciona o utilizador para a página "*logout.php*" descrita posteriormente.

Dashboard

O *dashboard* (que apenas existe para o modo Livre) ´é utilizado para a apresentação de informação relativa a um veículo bem como para a funcionalidade "Pesquisa Veículos" descrita mais abaixo.

Veículo - Quando é clicado um veículo no modo Livre, o *dashboard* é aberto é apresentada informação relativa ao mesmo (matricula, estado, condutor, frota, rua e data da recolha de

informação). A figura abaixo apresenta um exemplo da informação que poderá aparecer no *dashboard*.

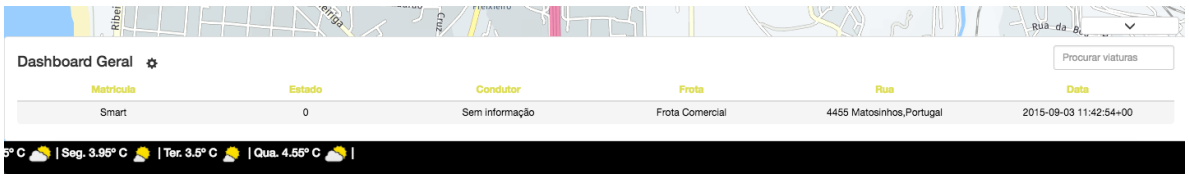


Figura 56 - *Dashboard* com informação de um veículo.

Pesquisa Veículo - Outra funcionalidade do *dashboard* é a pesquisa de veículos (através da matrícula). Para tal, será necessário clicar no botão (situado no canto inferior direito) para abrir o *dashboard*. O botão encontra-se representado na figura seguinte.

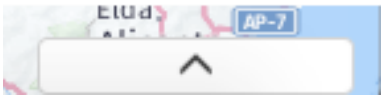


Figura 57 - Botão de mostrar *dashboard.*

Através dele poderá ser feita uma pesquisa no campo "Procurar viaturas". À medida que o utilizador preenche este campo, serão apresentadas sugestões de acordo com a frotado cliente No caso de o utilizador clicar numa sugestão, ou premir "*Enter*" quando apenas existe uma sugestão, a aplicação comportar-se-á do mesmo modo que caso o utilizador clicasse no veículo pesquisado. A figura 58 apresenta uma representação de sugestões.

Figura 58 - Exemplo de pesquisa no *dashboard.*

Rodapé

Foi criado um rodapé para a visualização de informação de forma compacta. No modo Livre o rodapé apenas é utilizado para a representação do fuso horário e da meteorologia. A informação visível no rodapé da página e encontra-se em constante movimento, da esquerda para a direita.

Meteorologia – A informação relativa às condições climatéricas foi estabelecida como uma das funcionalidades essenciais a implementar. A sua importância advém da possibilidade de previsão de condições adversas, podendo ser melhor geridos os eventuais atrasados causados pelo estado do tempo.

O número de dias a apresentar é configurável apenas na página das definições e varia entre o dia atual e 7 dias. A representação desta informação tem o seguinte formato:

[Dia da semana] [Temperatura] [Ícone meteorológico]

Neste rodapé, o dia da semana corresponde à abreviatura de 3 letras do mesmo, a temperatura corresponde à temperatura do dia representado. No dia atual é apresentada a informação da temperatura atual, nos restantes dias é feita a média aritmética do máximo e mínimo da temperatura. O ícone meterológico representa o estado do tempo (sol, chuva, nuvens, neve, etc.). Finalmente, caso sejam selecionados vários dias, o formato de cada dia mantém-se e cada dia se encontra separado por uma barra vertical (|). A figura seguinte ilustra um exemplo desta informação para 4 dias.

Figura 59 – Exemplo de informação meteorológica para 4 dias.

Se a posição do mapa for alterada, esta informação é alterada de modo a corresponder ao centro do mapa.

Fuso horário – A representação de um fuso horário é uma funcionalidade importante para situar o utilizador não só no espaço, mas também no tempo. Isto poderá ter especial importância em casos onde o utilizador da aplicação se encontra numa localização diferente dos veículos.

Assim, foi implementada a visualização de um fuso horário associado a uma localização. O fuso horário será apresentado no rodapé, precedendo a informação meteorológica. O formato representativo desta informação tem o seguinte aspeto:

[Ano]-[Mês]-[Dia] [Hora]:[Minuto] – [Localidade]:

Os dois pontos sucedem a designação da localidade devido à concatenação desta informação com a informação meteorológica. A figura seguinte mostra como aparecerá no rodapé esta informação.

Figura 60 – Exemplo de fuso horário para Formigosa.

Esta informação é alterada com movimentos do mapa para corresponder ao centro do mapa.

4.2.2. MODO ANIMADO

Neste modo, é suposto não haver interação do utilizador. Quando este modo está ativo, são carregadas as animações previamente definidas e certas funcionalidades são desativadas (escondidos determinados botões).

Botões do mapa

A maioria dos botões do mapa presentes no modo Livre são removidos no modo Animado. No entanto, são listadas as seguintes exceções:

- "Alterar tipo de mapa";
- "*Toggle* de centro do mapa";
- "*Toggle* de *fullscreen*";
- "Mostrar menu lateral";
- "Esconder menu lateral";

Os restantes botões são escondidos e as suas funcionalidades são carregadas de acordo com as definições, não podendo ser alteradas no decorrer do programa.

Menu lateral

No modo Animado, as funcionalidades do menu lateral são utilizadas durante a animação. Por essa razão, quando o menu lateral é aberto (utilizando o botão "Mostrar menu lateral") todas as animações são terminadas e o utilizador terá de recarregar a página para voltar ao modo Animado.

Caso o menu lateral seja aberto, as funcionalidades do mesmo ficarão todas disponíveis, mas os botões do mapa permanecerão escondidos. Na figura abaixo pode visualizar-se, no topo da página, a mensagem que é mostrada quando o utilizador abre o menu lateral em modo de animação (retângulo vermelho).

Figura 61 – Mensagem de aviso de interrupção do modo Animado.

Alertas Periódicos

No modo Animado existe ainda uma funcionalidade não disponível no modo Livre, a emissão de alertas periodicamente.

Esta funcionalidade é caracterizada por, com um intervalo de tempo um de minuto, verificar a existência de novos alertas. Esta verificação é feita com base na hora atual numa primeira instância e posteriormente através da hora do último alerta. Caso existam alertas a apresentar, é emitido um som de notificação e aparece uma mensagem no ecrã a informação relativa ao alerta. Na figura seguinte, no topo da página, é possível visualizar um alerta gerado.

Figura 62 – Exemplo de alerta criado.

Sublinhe-se ainda que a verificação e emissão destes alertas termina com abertura do menu lateral.

Visualização de conteúdo publicitário

Outra funcionalidade exclusiva do modo Animado é a visualização do conteúdo publicitário. Esta funcionalidade faz uso da lista de conteúdos publicitários previamente definida através da página "Alteração do conteúdo publicitário". De acordo com a frequência estipulada, abre uma *modal box* com o primeiro elemento da lista.

Caso se trate de uma imagem, esta estará presente no ecrã durante 10 segundos. No caso de um vídeo, o tempo é avaliado pela aplicação e este é mantido no ecrã na totalidade da sua duração.

Em ambos os casos, depois da sua visualização, a *modal box* é fechada. A lista é então rodada, passando o primeiro elemento para último, o segundo para primeiro e assim sucessivamente.

A figura 63 apresenta um exemplo de visualização do conteúdo publicitário (neste caso uma imagem).

Figura 63 – Exemplo de conteúdo publicitário.

Rodapé

O rodapé do modo Animado é composto por duas componentes, um rodapé igual ao representado no modo Livre e um rodapé informativo sobre o elemento representado na página.

Este segundo rodapé foi utilizado para o acesso rápido à informação que não consta no mapa. A informação contida neste rodapé é adaptada à animação que está a decorrer.

No caso das animações de veículos (ligados ou desligados) ou dos seus percursos, o formato é o seguinte:

[ícone de estado]:[matricula]:[condutor*]:[frota*]:[rua*][1]

Em animações de serviços é utilizado o seguinte formato:

[Estado do serviço]: [[Origem do serviço]->[Pontos intermédios do serviço]->[Destino do serviço]] – [Data de agendamento do serviço]

[1] Os elementos marcados com asterisco são opcionais. A sua visualização deverá ser definida previamente pelo utilizador.

A imagem seguinte apresenta um exemplo deste último rodapé.

Figura 64 – Exemplo do segundo rodapé.

Todas as outras animações escondem o mapa e apresentam uma página nova. Por essa razão o rodapé não é visível durante essas animações. É de notar ainda que o *dashboard* está completamente desativado para este modo.

4.2.3. LEAFLET PLUGINS

No sentido de otimizar a experiencia do utilizador, foram incluídos alguns *plugins* para o *Leaflet*. Estes acrescentam algumas das funcionalidades gráficas descritas anteriormente e serão descritos nesta subsecção.

Leaflet.fullscreen

Este *plugin* é uma adaptação da subclasse *Leaflet.Control* que permite apresentar a aplicação em tela cheia. O *plugin* foi desenvolvido para aplicar este efeito apenas relativamente ao mapa e aos seus constituintes. No desenvolvimento deste projeto este *plugin* foi alterado, de maneira a incluir o rodapé (que contém informação sobre os constituintes do mapa) no modo de tela cheia.

Para dispositivos que não suportem o modo tela cheia, o *plugin* permite a criação de um efeito de "*pseudo* tela cheia", que é semelhante aquela, mas em que o conteúdo da página estará emolderado por uma janela.

Leaflet.Markercluster

O *Leaflet.Markercluster* é um *plugin* utilizado na criação de aglomerados de marcadores, de modo a economizar espaço no mapa.

Funciona através do agrupamento de marcadores que se encontrem relativamente próximos, criando um círculo com um número, representando os marcadores que se encontram nesse grupo. Apresenta diversos parâmetros para uma melhor customização dos efeitos pretendidos, entre as quais se destacam:

- **zoomToBoundsOnClick** – Quando um grupo é clicado, permite fazer *zoom* até os limites do mesmo, desfazendo-o e mostrando os marcadores originais;

- **animate** – Esta opção cria uma pequena animação para o agrupamento e desagrupamento dos marcadores.

- **maxClusterRadius –** Este parâmetro define o máximo raio de cada grupo em pixéis.

- **disableClusteringAtZoom –** Este parâmetro define o valor de *zoom* a partir do qual os grupos automaticamente se irão desfazer.

No contexto desta aplicação, este *plugin* foi utilizado no agrupamento dos veículos, o que é particularmente importante em empresas com um elevado número de veículos, que evita assim a confusão de marcadores para níveis de detalhe de mapa mais baixos.

4.3. BASES DE DADOS

Para o funcionamento correto da aplicação, é necessário estabelecer comunicação com as bases de dados. Como descrito no capítulo anterior, as bases de dados são PostgreSQL utilizando o módulo PostGIS (Ver **1.6** – Análise de Requisitos).

Assim, são necessárias duas conexões, uma às tabelas do Geocar (que por motivos de confidencialidade não serão apresentadas neste documento) e outra às tabelas do GeocarEyes. É importante ainda referir que são efetuadas duas conexões, pois durante o desenvolvimento do projet, as tabelas criadas para a aplicação GeocarEyes se encontravam numa base de dados de teste. Apesar de à data de conclusão deste projeto se terem a efetuado duas conexões, é possível e provável que no momento de comercialização deste produto, seja feita a passagem das tabelas GeocarEyes para a base de dados Geocar resultando na necessidade de apenas uma conexão.

De modo a garantir uma base de funcionamento autónomo da aplicação, foram criadas as seguintes tabelas.

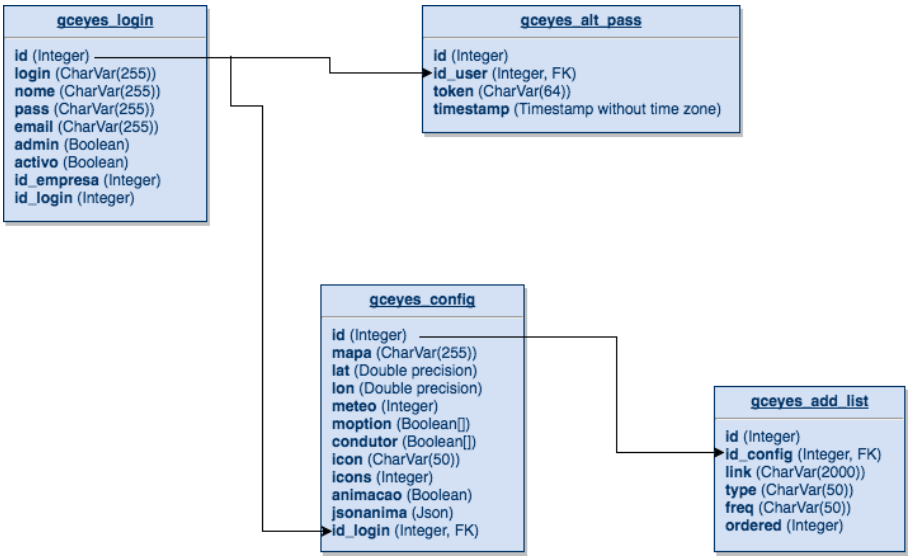


Figura 65 - Esquema representativo das tabelas criadas para o GeocarEyes

É de notar que os campos que contêm "(tipo de variável, FK)" correspondem a "*Foreign Keys*" de outras tabelas. *Foreign Keys* poderão ser definidas como chaves que indica uma dependência (de um campo) entre duas tabelas. Estas dependências são descritas nos subcapítulos seguintes.

O uso das tabelas do Geocar é imprescindível para esta aplicação, todavia, afim de evitar sobrecarga das mesmas, alguma da informação foi exportada para as tabelas da base de dados GeocarEyes.

4.3.1. GCEYES_LOGIN

Esta tabela armazena toda a informação relativa ao *login* do utilizador. Nela, existem ainda campos de referência às tabelas do Geocar (campos **'id_empresa'** e **'id_login'**) de modo a formar uma ligação entre um utilizador GeocarEyes e um utilizador Geocar.

Os campos desta tabela são descritos da seguinte maneira:

- **Id –** campo indentificador único, utilizado para distinguir cada entrada nesta tabela, bem como para fazer referência a outras tabelas. É do tipo *integer* e é a *primary key* desta tabela.

- **Login –** campo para o nome de utilizador, utilizado para efetuar o *login* e correspondente ao *login* (nome de utilizador) presente nas tabelas do Geocar. Esta

coluna será preenchida com valores encriptados. É do tipo *character varying* e tem uma dimensão máxima de 255 caracteres.

- **Nome -** campo para o nome do cliente, utilizado para efeitos de visualização (identificação do cliente durante a aplicação). É do tipo *character varying* e tem uma dimensão máxima de 255 caracteres.

- **Pass -** campo para a *password*, utilizado para efetuar o *login* e correspondente à *password* presente nas tabelas do Geocar. Esta coluna será preenchida com valores encriptados. É do tipo *character varying* e tem uma dimensão máxima de 255 caracteres.

- **Email -** campo para o email do cliente, utilizado para efeitos de visualização (identificação do cliente durante a aplicação), bem como para o envio de *emails* (de alteração de *password*). É do tipo *character varying* e tem uma dimensão máxima de 255 caracteres.

- **Admin -** campo identificador de utilizadores administradores (gestores da aplicação), referente a um campo análogo ao existente nas tabelas do Geocar. Utilizado para alterar as configurações da aplicação para a empresa do utilizador. É do tipo *boolean* e tem o valor *default* de *false*.

- **Activo -** campo identificador de utilizadores ativos referente a um campo análogo ao existente nas tabelas do Geocar. Utilizado para bloquear utilizadores que não estejam ativos. É do tipo *boolean* e tem o valor *default* de *false*.

- **Id_empresa -** campo identificador da empresa do utilizador referente a um campo análogo ao existente nas tabelas do Geocar. Utilizado para carregar toda a informação relativa à empresa. É do tipo *integer* e é vital para o funcionamento da aplicação.

- **Id_login -** campo identificador do utilizador, referente a um campo análogo ao existente nas tabelas do Geocar. Utilizado para fazer a referência a utilizadores específicos da empresa (por exemplo para enviar um email) e é do tipo *integer*.

4.3.2. GCEYES_CONFIG

Esta tabela armazena toda a informação relativa às configurações do utilizador. É nela que o utilizador vai definir as opções de visualização da aplicação.

Os campos desta tabela estão apresentados abaixo:

- **Id –** campo indentificador único, utilizado para distinguir cada entrada nesta tabela, bem como para fazer referência a outras tabelas. É do tipo *integer* e é a *primary key* desta tabela.

- **Mapa –** campo para o tipo de mapa a carregar, utilizado após o *login* para definir o mapa a ser carregado por defeito. Este campo contém um valor do género "bl_normal_day". É do tipo *character varying* e tem uma dimensão máxima de 255 caracteres.

- **Lat -** campo para a coordenada geográfica de latitude, utilizado para definir o ponto do mapa onde a aplicação irá começar. É do tipo *double precision*.

- **Lon -** campo para a coordenada geográfica de longitude, utilizado para definir o ponto do mapa onde a aplicação irá começar. É do tipo *double precision*.

- **Meteo -** campo para definir o número de dias a apresentar de previsão meteorológica. O seu valor varia entre 1 (meteorologia atual) e 7 (previsão meteorológica de até 6 dias). É do tipo *integer*.

- **Moption -** campo para ativação de opções, utilizado para definir opções de visualização, nomeadamente a visualização do fluxo tráfego, postos de combustível e áreas de descanso. É do tipo *array boolean* e tem de armazenar 3 elementos (3 *boolean*).

- **Condutor -** campo para ativação de opções, utilizado para definir opções de visualização de informação do veículo, nomeadamente a visualização do condutor, frota e rua atual do veículo. É do tipo *array boolean* e tem de armazenar 3 elementos (3 *boolean*).

- **Icon -** campo identificador do *icon* a ser utilizado para a representação dos veículos no mapa. Corresponde ao URL da imagem no servidor. É do tipo *character varying* e tem uma dimensão máxima de 50 caracteres.

- **Icons -** campo que indica o número de *icons* presentes possíveis para escolha do utilizador. Utilizado para saber se o utilizador já inseriu algum *icon* personalizado. Contém um valor que varia entre 82 (82 é o número de *icons default* na data de escrita deste documento) e 92 (número de *icons default* mais um máximo de 10 *uploads*). É do tipo *integer*.

- **Animacao –** campo para a mudança do modo de funcionamento da aplicação. Os valores possíveis deste campo são *true* (modo Animado) e *false* (modo Livre). É do tipo *boolean.*

- **Jsonanima -** campo relativo à animação a ser executada. Utilizado para carregar todas as animações configuradas. Quando o campo 'Animacao' contém o valor *false* este campo está a *null*. É do tipo JSON.

- **Id_login -** campo identificador do utilizador referente a um campo análogo ao existente nas tabelas do Geocar. Utilizado para associar uma configuração a um utilizador**.** É do tipo *integer* e corresponde a uma *foreign key* para a coluna 'id_login' da tabela 'gceyes_login'.

4.3.3. GCEYES_ALT_PASS

Esta tabela armazena os *tokens* gerados pelos pedidos de alteração de *password*, permitindo a gestão dos mesmos.

Os campos desta tabela são descritos da seguinte maneira:

- **Id –** campo indentificador único, utilizado para distinguir cada entrada nesta tabela, bem como para fazer referência a outras tabelas. É do tipo *integer* e é a *primary key* desta tabela.

- **Id_user -** campo identificador do utilizador referente a um campo análogo ao existente nas tabelas do Geocar. Utilizado para associar uma *token* a um utilizador**.**

É do tipo *integer* e corresponde a uma *foreign key* para a coluna ‘id_login’ da tabela ‘gceyes_login’.

- **Token –** campo que contém chaves únicas de identificação de pedidos. Utilizado após um pedido de alteração de *password*, o valor deste campo é gerado e é-lhe associado uma validade (ver campo ‘Timestamp’). Esta coluna será preenchida com valores encriptados. É do tipo *character varying* e tem uma dimensão máxima de 64 caracteres.

- **Timestamp -** campo para indicação da validade do *token* associado, utilizado na validação do *token*, estabelecendo uma data limite para a utilização do mesmo. É do tipo *timestamp without time zone*.

- **Valid -** campo para a validação dos *tokens*, utilizado para bloquear um segundo pedido com o mesmo *token* que ainda se encontre dentro de uma data válida. Cada vez que é alterada uma *password* através de um *token*, a linha correspondente irá ser atualizada com este campo como valor *false*. É do tipo *boolean*.

4.3.4. GCEYES_ADD_LIST

Esta tabela armazena toda a informação relativa ao conteúdo publicitário inserido pelo o utilizador. Este conteúdo apenas está disponível no modo Animado da aplicação.

Os campos desta tabela são descritos da seguinte maneira:

- **Id –** campo indentificador único, utilizado para distinguir cada entrada nesta tabela, bem como para fazer referência a outras tabelas. É do tipo *integer* e é a *primary key* desta tabela.

- **Id_config –** campo de identificação da configuração do cliente, utilizado para associar esta configuração à lista de conteúdo publicitário correspondente. É do tipo *integer* e corresponde a uma *foreign key* para a coluna ‘id’ da tabela ‘gceyes_config’.

- **Link -** campo para todos os *links* do conteúdo publicitário, sendo cada *link* associado a uma entrada nesta tabela. Estes podem ser imagens (*Portable Network Graphics* (PNG), *Graphics Interchange Form* (GIF) e JPEG) ou vídeos (Apenas *links* válidos

do Youtube). É do tipo *character varying* e tem uma dimensão máxima de 2000 caracteres, permitindo assim *links* muito extensos.

- **Type –** campo identificador do tipo de *link* inserido, utilizado para diferenciar o tipo de *tag* HTML a usar. Pode ter o valor 'img' que corresponde a uma imagem ou 'vid' que na sua vez corresponde a um vídeo. É do tipo *character varying* e tem uma dimensão máxima de 50 caracteres.

- **Freq -** campo para definir a frequência de repetição dos *links* da lista. No final do tempo definido neste campo, é apresentado o conteúdo publicitário e a lista é rodada (1º elemento -> último elemento, 2º elemento -> 1º elemento, etc). O seu valor corresponde a um número *float* concatenado com um identificador ('H' – horas, 'M' - minutos). É do tipo *integer*.

- **Ordered –** campo que mantém informação relativa à ordem dos elementos da lista, utilizado na construção da lista em tempo real. É do tipo *integer* e começa em 1.

4.4. API

O desenvolvimento deste projeto está bastante dependente de um conjunto de fontes de informação. Estas podem ser internas (informação da base de dados do Geocar) ou externas (informação de diversas APIs).

Este subcapítulo pretende enunciar e caracterizar as APIs utilizadas neste projeto e a sua utilidade, fazendo de seguida uma descrição dos pedidos e respostas possíveis.

Na aplicação desenvolvida, todos os pedidos efetuados às APIs são do tipo REST descritos na secção **2.5**.

No caso das APIs pertencentes à HERE, o uso da versão REST deve-se ao facto de um dos requisitos do projeto ser o uso do *layout* versão 2 do Geocar, que faz uso da biblioteca *Leaflet*, não permitindo a utilização da API *JavaScript*. No caso da API GeoNames, o único modo de acesso é através de pedidos REST.

Refere-se ainda que as APIs utilizadas refletiram a escolha da empresa GISGEO, quer pela sua natureza *open-source*, quer pela pré-existência de um contrato com os prestadores destes serviços.

4.4.1. REGISTO

Como referido anteriormente, as APIs utilizadas na aplicação necessitam de um registo de modo a obter uma chave de API.

Assim, abaixo é apresentada a lista de APIs utilizadas, sublinhando que em cada uma delas é necessário um registo para o seu uso:

- HERE *Map Tile* API (https://developer.here.com/authenticationpage);
- HERE *Places* API (https://developer.here.com/authenticationpage);
- HERE *Weather* API (https://developer.here.com/authenticationpage);
- GeoNames API (http://www.geonames.org/login);

Destaca-se ainda que a API GeoNames não necessita de uma chave, bastando o nome do utilizador para o seu acesso.

4.4.2. PEDIDOS

Nas APIs o tipo de informação da resposta é dependente dos parâmetros passados no pedido, note-se que para os propósitos da aplicação desenvolvida, nem toda a informação passível de ser obtida é necessária.

Em seguida será feita uma apresentação dos tipos de pedidos efetuados às diferentes APIs bem como das respostas obtidas.

HERE Map Tile API

Esta API é utilizada na aplicação para o carregamento de diversos tipos de mapas.

O URL de base utilizado para estes pedidos é:

```
https://1.{base}.maps.cit.api.here.com/maptile/2.1/{type}/newest/{scheme}/{z}/{c}/{l}/256/png8?app_id={app_id}&app_code={app_code}
```

Os tipos de mapa implementados na aplicação são:

- Vista normal do mapa durante o dia (base -> “base”, type -> “maptile”, scheme -> “normal.day”);

- Vista normal do mapa durante o dia com um esquema dos transportes públicos (base -> “base”, type -> “maptile”, scheme -> “normal.day.transit”);

- Vista satélite do mapa durante o dia (base -> “base”, type -> “maptile”, scheme -> “hybrid.day”);

- Vista do relevo do mapa durante o dia (base -> “base”, type -> “maptile”, scheme -> “terrain.day”);

- Vista normal do mapa durante o dia, assinalando os sinais de transito específicos para camiões. (base -> “base”, type -> “trucktile”, scheme -> “normal.day”);

- Vista do fluxo de tráfego no mapa, sem carregar o próprio mapa (base -> “traffic”, type -> “flowtile”, scheme -> “normal.day”);

Note-se que esta API funciona através de um mapeamento total do mundo em linhas e colunas para diferentes níveis de aproximação (*zoom*).

Assim, é fácil perceber que os parâmetros {z}, {c}e {l} correspondem ao nível de *zoom*, coluna e linha da imagem. Por sua vez os parâmetros {app_id} e {app_code} correspondem às chaves obtidas no momento de registo da aplicação.

Como o carregamento dos mapas no *browser* é feito através da biblioteca *Leaflet*, estes parâmetros não são preenchidos e, de acordo com as interações do utilizador, o *Leaflet* carrega as imagens correspondentes à porção do mapa pretendida.

HERE Places API

Esta API é utilizada na representação de diversos marcadores com a informação relativa a pontos de interesse.

O URL de base utilizado para estes pedidos é:

```
https://places.api.here.com/places/v1/discover/explore?in={lng_sw},{lat_sw},{lng_ne},{lat_ne}&cat={type}&{options}&app_id={app_id}&app_code={app_code}
```

A API disponibiliza um número elevado de tipos de pontos de interesse (cerca de 17), no contexto do trabalho apenas se tornam relevantes os seguintes:

- Postos de combustível (type -> “petrol-station”);
- Áreas de descanso (type -> “toilet-rest-areas”);

O URL utilizado para estes pedidos está em forma de *bounding box*, o que significa que os pedidos são feitos com base numa área retangular formada por dois pares de coordenadas geográficas (no caso da aplicação desenvolvida são as coordenadas geográficas correspondentes ao canto inferior esquerdo e canto superior direito do ecrã).

Assim, os parâmetros {lng_sw} e {lat_sw} correspondem ao par de longitude e latitude do canto inferior esquerdo (Sudoeste) e analogamente {lng_ne} e {lat_ne} correspondem ao par de longitude e latitude do canto superior direito (Nordeste). Por sua vez, {type} será o tipo de ponto de interesse (enunciadas acima) a obter na área selecionada.

As {options} são um conjunto de opções que podem ser adicionadas à informação pedida, e que por vezes dependem do tipo de ponto de interesse selecionado. As mais relevantes para a aplicação desenvolvida são:

- Accept-Language=pt-PT -> Linguagem da informação na resposta (Neste caso Português de Portugal). Isto torna-se relevante, pois muitas vezes é necessário utilizar nomes de ruas para cada ponto de interesse.
- Show_content=fuel -> Opção específica para pedidos de postos de combustível. Mostra os preços dos combustíveis em cada posto, caso essa informação esteja disponível.

Novamente, {app_id} e {app_code} representam as chaves obtidas no momento de registo da aplicação.

Finalmente, a resposta é um objeto com formato JSON com a seguinte organização:

```
results
{
      items
      {
            [
                  0
                  {
                        position
                        [
                              0: Latitude;
                              1: Longitude;
```

```
                        ]
                        vicinity: Morada completa;
                        title: Título (e.g. "GALP");
                        openingHours
                        {
                              label: Identificador na
                                    linguagem do pedido
                                (e.g. "Horário de Abertura");
                              text: Descrição do horário de
                                    abertura;
                        }
                        fuelPrices
                        {
                              label: Identificador na
                                    linguagem do pedido
                                     (e.g. "Preços dos
                                    Combustíveis");
                              text: Descrição dos preços dos
                                    combustíveis;
                        }
                  },
                  1
                  {
                        position
                        [
                              0: Latitude;
                              1: Longitude;
                        ]
                        vicinity: Morada completa;
                        title: Título (e.g. "Repsol");
                  }
            ]
            length: 2;
      }
}
```

Os campos assinalados a verde correspondem a informação que nem sempre está disponível. O campo "fuelPrices" apenas se aplica a pedidos de postos de combustível (informação correspondente ao parâmetro "show_content=fuel" descrito acima).

HERE Weather API

A HERE *Weather* API foi utilizada no contexto deste trabalho para representar previsões meteorológicas, bem como o estado atual do tempo, nas diversas partes do mapa onde o utilizador interage. Esta funcionalidade permite aos gestores da frota de veículos antecipar condições adversas, podendo prever atrasos ou desvios.

O URL de base utilizado para estes pedidos é:

https://weather.cit.api.here.com/weather/1.0/report.json?app_id={app_id}&app_code={app_code}&product={type)&latitude={lat}&longitude={lng}

Facilmente se observa que {lat} e{lng} correspondem aos pares de latitude e longitude do pedido. Assim como {app_id} e {app_code} representam as chaves obtidas no momento de registo da aplicação.

A API disponibiliza um leque de tipos de previsões (correspondentes ao parâmetro {type}) listadas de seguida:

- **observation** – Condições atuais do estado do tempo, para as localizações mais próximas das coordenadas especificadas.
- **forecast_7days** – Previsão meteorológica dos próximos 7 dias, considerando quatro diferentes períodos ao longo de cada dia: manhã, tarde, fim-de-tarde, noite.
- **forecast_7days_simple** – Previsão meteorológica diária dos próximos 7 dias.
- **forecast_hourly** – Previsão meteorológica a cada hora dos próximos 7 dias.
- **forecast_astronomy** – Informação relativa ao nascer e por do sol e da lua, bem como a fase da lua nos próximos 7 dias.
- **alerts** – Alertas de condições severas previstas para as próximas 24 horas.
- **nws_alerts** – Todos os alertas relativos aos potenciais riscos meteorológicos nos Estados Unidos e Canadá.

Neste trabalho apenas foram apenas utilizados os tipos “observation” e “forecast_7days_simple” que se consideram adequados aos propósitos do projeto. Estes tipos são os que apresentam informação mais simplificada não consumindo uma quantidade excessiva de espaço na página.

O um exemplo de resposta a um pedido do tipo “forecast_7days_simple” é apresentado abaixo (em forma de objeto JSON):

```
{
  "dailyForecasts": {
  "forecastLocation": {
    "forecast": [
    {
      "daylight": "D",
```

```
      "description": "Drizzle. Overcast. Cool.",
      "skyInfo": "18",
      "skyDescription": "Overcast",
      "temperatureDesc": "Cool",
      "comfort": "9.21",
      "highTemperature": "12.00",
      "lowTemperature": "7.80",
      "humidity": "94",
      "dewPoint": "10.43",
      "precipitationProbability": "28",
      "precipitationDesc": "Drizzle",
      "rainFall": "0.00",
      "snowFall": "*",
      "airInfo": "32",
      "airDescription": "Damp",
      "windSpeed": "18.26",
      "windDirection": "163",
      "windDesc": "South",
      "windDescShort": "S",
      "uvIndex": "0",
      "uvDesc": "Minimal",
      "barometerPressure": "1007.17",
      "icon": "18",
      "iconName": "sprinkles",
      "iconLink":
"https://weather.cit.api.here.com/static/weather/icon/27.
png",
      "dayOfWeek": "4",
      "weekday": "Wednesday",
      "utcTime": "2013-12-04T00:00:00.00-06:00"
    },
    {
      "daylight": "D",
      "description": "Sprinkles late. High level clouds.
Cool.",
      "skyInfo": "12",
      "skyDescription": "High level clouds",
      "temperatureDesc": "Cool",
      "comfort": "-6.32",
      "highTemperature": "8.90",
      "lowTemperature": "-3.40",
      "humidity": "55",
      "dewPoint": "-8.74",
      "precipitationProbability": "18",
      "precipitationDesc": "Sprinkles late",
      "rainFall": "*",
      "snowFall": "*",
      "airInfo": "*",
      "airDescription": "",
      "windSpeed": "21.96",
      "windDirection": "276",
      "windDesc": "West",
      "windDescShort": "W",
      "uvIndex": "0",
      "uvDesc": "Minimal",
      "barometerPressure": "1012.06",
```

```
      "icon": "18",
      "iconName": "sprinkles",
      "iconLink":
"https://weather.cit.api.here.com/static/weather/icon/27.
png",
      "dayOfWeek": "5",
      "weekday": "Thursday",
      "utcTime": "2013-12-05T00:00:00.00-06:00"
    },
    {
      ---remaining observations removed to shorten document
    }
    ],
    "country": "United States",
    "state": "Illinois",
    "city": "Chicago",
    "latitude": 41.83,
    "longitude": -87.68,
    "distance": 0,
    "timezone": -6
  }
  },
  "feedCreation": "2013-12-04T09:44:13.574",
  "metric": true
}
```


Por motivos que se prendem com a facilidade de representação gráfica (enunciados anteriormente), nem toda a informação apresentada neste objeto JSON é utilizada na aplicação. Assim, com vista a uma economia de espaço, apenas são utilizados (para cada dia) os parâmetros: "dayOfWeek" que representa o índice do dia da semana (0 a 6, onde 0 é segunda-feira), "lowTemperature" e "highTemperature". Com os dois últimos a aplicação calcula a média aritmética das temperaturas, que é associada ao "iconLink" que contém um *link* para uma imagem representativa da condição meteorológica.

GeoNames API

Esta API é utilizada na conversão de coordenadas geográficas em fusos horários. É importante referir que esta API não funciona para coordenadas nos oceanos.

O URL utilizado para os pedidos a esta API é:

http://api.geonames.org/timezoneJSON?lat={lat}&lng={lng}&username={user}

Neste caso os parâmetros são bastante diretos: {lat} e {lng} correspondem aos pares de latitude e longitude, respetivamente, e {user} será o nome de utilizador escolhido no registo.

A resposta é um Objeto com formato JSON apresentando o seguinte formato:

```
{
  "sunrise":"2017-09-24 08:24",
  "lat":43.43
  "lng":-8.8231,
  "countryCode":"ES",
  "gmtOffset":1,
  "rawOffset":1,
  "sunset":"2017-09-24 20:29",
  "timezoneId":"Europe/Madrid",
  "dstOffset":2,
  "countryName":"Spain",
  "time":"2017-09-24 19:43",
}
```


4.5. DESENVOLVIMENTO LÓGICO

O desenvolvimento lógico da aplicação corresponde à componente de *back-end* da aplicação. Representa a interface de ligação às bases de dados e o processamento da informação obtida.

De acordo com os requisitos definidos na secção **1.6**, a linguagem utilizada para o desenvolvimento de funções de *back-end* da aplicação foi o PHP. Assim, as funções de *back-end* estão divididas em diferentes ficheiros de PHP.

À exceção de alguns ficheiros (login2.php, logout.php, index.php e altpass.php), estes são acedidos através de pedidos AJAX de modo a garantir o funcionamento dinâmico da aplicação.

A figura seguinte ilustra os ficheiros PHP utilizados na aplicação.

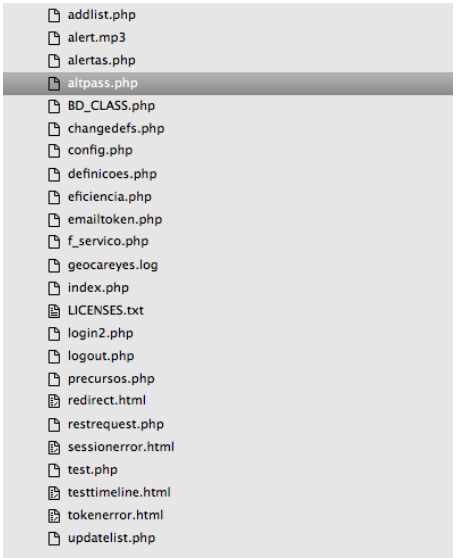


Figura 66 – Ficheiros utilizados no desenvolvimento *back-end* (Extensão .php)

Login2.php

Este ficheiro conecta-se às bases de dados para efetuar o *login* de um utilizador. Numa primeira fase cria uma página HTML (ver imagem abaixo) com um formulário para o *login* (*user* e *password*). Quando é feita uma tentativa de *login*:

- Primeiro, faz-se o *login* através da base de dados do GeocarEyes;
- Caso não existam registos é feita uma conexão à base de dados Geocar;
- Caso exista um registo na base de dados Geocar, é criada uma nova conta no GeocarEyes, e é efetuado o *login*;
- O utilizador é então redirecionado para a página 'index.php'.

Na figura seguinte é possível observar esta página.

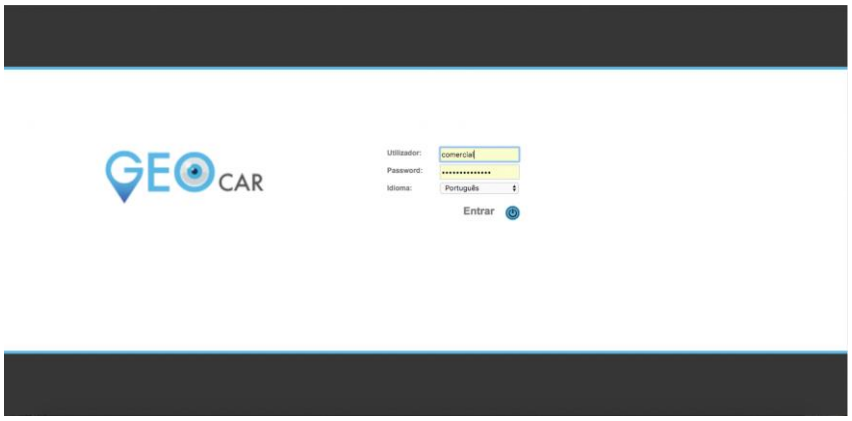


Figura 67 – Página de *login* GeocarEyes.

As variáveis de sessão criadas são:

- 'email' - Para enviar email de alteração de *password*.
- 'empresa' - Id de empresa, usada em diversos ficheiros.
- 'admin' - 'S' ou 'N' para determinar se o utilizador pode alterar as configurações.
- 'id' - *Id login*, usado para identificar o *login* (utilizador) em diversos ficheiros.

O utilizador e *password* são encriptados de acordo com a encriptação utilizada para as credenciais Geocar. Por razões de segurança e confidencialidade, a sequência de encriptação não será descrita neste documento.

Index.php

Este ficheiro representa a base da aplicação. Sobre ele irão decorrer as diversas funções (*JavaScript*) definidas no programa. São carregadas todas as bibliotecas utilizadas, bem como ficheiros *JavaScript* e CSS. Finalmente, quando o 'body' deste ficheiro é carregado, é executada a função 'init()' que dá início ao decorrer do programa. Este ficheiro apenas é acedível por utilizadores com sessão iniciada.

A figura abaixo ilustra esta página.

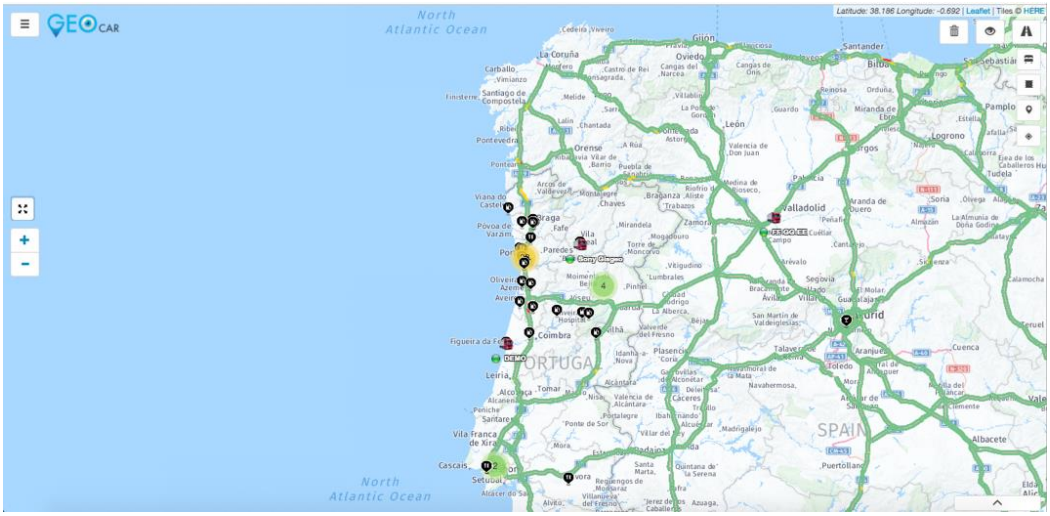


Figura 68 – Página index.php.

Addlist.php

Este ficheiro vai buscar a informação relativa à lista de conteúdo publicitário (já definida pelo utilizador), enviando depois a resposta como um objeto JSON.

Abaixo está representado um exemplo de resposta deste ficheiro:

```
response{[
   0:{
     unique: 2,
     order: 3,
     link: 'http://www.youtube.com/watch?v=DoTdTtGZcJk',
     type: 'vid',
     freq: '0.08333'
    },
    1:{
     unique: 2,
     order: 1,
     link:'http://1.bp.blogspot.com/-
DR5nWw8Tii4/Td_wbavsTCI/AAAAAAAAAV8/8-
mJMG0_3es/s1600/Fixe.jpg',
     type: 'img',
     freq: '0.08333'
```

```
   }
]}
```

A resposta contém a informação relativa à tabela **gceyes_add_list** descrita anteriormente, em que as designações **unique** e **order** correspondem aos campos da tabela ‘id’ e ‘ordered’, respetivamente.

Altpass.php

Este ficheiro processa os pedidos de alteração de palavra-passe e apenas deve ser acedido através de um *link* gerado e enviado para o *email* do utilizador.

Esse *link* deverá conter o *token* gerado no momento do pedido de alteração de *password* através do ficheiro ‘emailtoken.php’ descrito adiante nesta subsecção.

Assim, este ficheiro conecta-se às bases de dados confirmando o *token* recebido. Caso o *token*, o *id* de utilizador ou a data do *token* sejam inválidas o utilizador é redirecionado para a página de *login*. Caso contrário, é carregada uma página de alteração de *password*.

Esta página pede a *password* antiga e uma *password* nova que deverá ser confirmada e têm que se verificar as seguintes condições:

- A *password* antiga tem que corresponder à que está guardada na base de dados,
- A *password* nova tem que ser válida, contendo pelo menos 8 caracteres, uma maiúscula, uma minúscula e 1 dígito,
- A repetição da *password* nova tem que corresponder à inserida no campo anterior.

A password é alterada com sucesso e o utilizador é redirecionado para o *login*, marcando o *token* usado como inválido.

A figura 69 ilustra a página altpass.php no seu primeiro acesso e as figuras 70 e 71 mostram as possíveis mensagens de erro desta página.

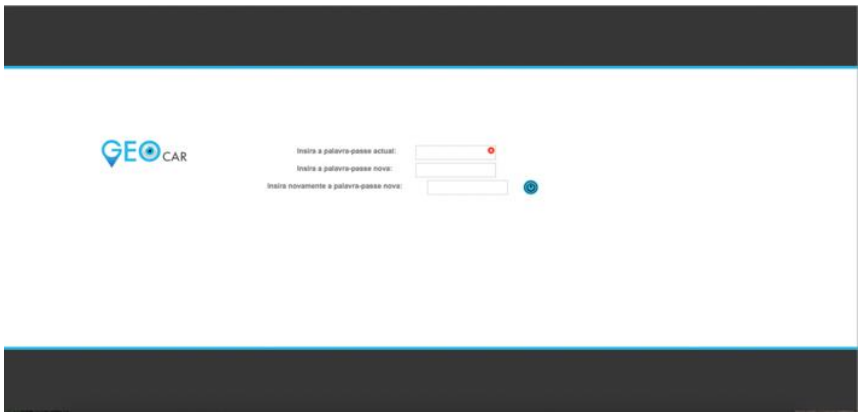


Figura 69 – GeocarEyes alteração da *password*.

Figura 70 – GeocarEyes alteração da *password*: 'As passwords não correspondem'.

Figura 71 – GeocarEyes alteração da *password*: 'A password deverá conter pelo menos 8 letras, 1 Maiúscula, 1 minúscula e 1 dígito'.

É ainda importante referir que este ficheiro apenas efetua a alteração da *password* para os utilizadores do GeocarEyes e que a nova *password* não será atualizada na aplicação Geocar.

Changedefs.php

Este ficheiro é utilizado para fazer a alteração das definições do utilizador. Deve ser acedido através do botão 'Alterar definições' na página das Definições. Recebe então os diversos parâmetros POST correspondentes às escolhas do utilizador.

Tabela 1 – Parâmetros POST recebidos no ficheiro 'changedefs.php'

Nome	Tipo	Descrição
anima	*Boolean*	Modo Animado ou modo Livre
jsontime	*Object*	Objeto das animações (No modo Livre, este parâmetro está a *null*)
icon	*Integer*	Índice do *icon* a ser utilizado
totalicons	*Integer*	Número total de *icons* associados a essa empresa
mapa	*String*	Tipo de mapa a ser apresentado
meteo	*Integer*	Número de dias a mostrar de previsão meteorológica
traf	*Boolean*	Mostrar o fluxo de tráfego no mapa
gas	*Boolean*	Mostrar os postos de combustível no mapa
rest	*Boolean*	Mostrar as áreas de descanso no mapa
cond	*Boolean*	Mostrar o condutor nas informações de veículo
frot	*Boolean*	Mostrar a frota nas informações de veículo
rua	*Boolean*	Mostrar a rua nas informações de veículo
lat	*Float*	Coordenada geográfica de latitude do inicio da aplicação
lon	*Float*	Coordenada geográfica de longitude do inicio da aplicação

Estes parâmetros são formatados em forma de *query* e atualizados na tabela correspondente (gceyes_config).

Destaca-se que se o parâmetro *icon* for recebido com o valor -1, significa que o utilizador pretende fazer o *upload* do seu próprio ícone. Este é guardado no servidor, numa pasta associada ao utilizador. Considerando que não seria aconselhável deixar o utilizador inserir uma quantidade ilimitada de imagens no servidor, é utilizado o parâmetro *totalicons*. Este contém o número total de ícones associados ao utilizador e se a este valor corresponder a número total de 10 *uploads*, o novo *upload* não será feito e o ícone selecionado será o anterior às alterações. Neste caso será enviada uma mensagem de erro para notificar o utilizador. Finalmente o utilizador é redirecionado para a página inicial 'index.php'.

Config.php

Ficheiro utlizado para carregar as configurações base do utilizador. Estas definições são formatadas em forma de objeto JSON e enviadas na resposta. Caso ainda não existam definições para este utilizador (no caso do primeiro *login*), são carregadas definições *default* (ver exemplo de resposta).

Exemplo de resposta:

```
response{[
0: 'bl_normal_day', //Tipo de mapa
1: 42.235, //Latitude
2: -8.8632, //Longitude
3: 5, //Dias de previsão meteorológica
4: [true,true,true], //Fluxo de tráfego/Postos de
Combustível/Áreas de Descanso
5: [true,true,true], //Condutor/Frota/Rua
6: 'veicicon4.png', //Link do icon a utilizar
7: true, //Modo Animado
8: {
   "Full": //Nome do grupo de animações
   {
      "VL":5, //Veículos Ligados - 5 minutos
      "VD":5, //Veículos Desligados - 5 minutos
      "P":10, //Percursos - 10 minutos
      "S":5, //Serviços - 5 minutos
      "IE":5, //Indicadores de eficiência - 5 minutos
      "diff":"30M" //Total de tempo do grupo de
animação
   },"Veiculos":
   {
      "VL":15,
      "VD":15,
      "diff":"0.5H"
   }
},
9: 200190 //Id empresa
]}
```

Definicoes.php

Este ficheiro está encarregue do carregamento das definições guardadas na base de dados, formatando-as num formulário HTML, de modo a criar uma página para a configuração das mesmas.

Sublinha-se que esta página é visível para todos os utilizadores, mas a edição das definições está bloqueada. Deste modo, o botão para submeter o formulário apenas é gerado para utilizadores administradores.

A tabela seguinte descreve cada *tag* HTML utilizada nesta página para a alteração das definições.

Tabela 2 – Todos os *inputs* criados para o formulário das definições.

Nome	HTML *Tag*	Descrição
mapa	*select*	Tipo de mapa (Normal, Satélite, Transportes, Terreno, Camião, *Open Street Maps*)
meteo	*select*	Dias de previsão meteorológica (1-7 dias)
traf	*input checkbox*	Ativar/Desativar fluxo de tráfego
gas	*input checkbox*	Ativar/Desativar postos de combustível
rest	*input checkbox*	Ativar/Desativar áreas de descanso
cond	*input checkbox*	Ativar/Desativar condutor
frot	*input checkbox*	Ativar/Desativar frota
rua	*input checkbox*	Ativar/Desativar rua
#empsess*	*input hidden*	*Id* de empresa
fileToUpload	*input file*	Ficheiro do *icon* a carregar (PNG)
lat	*input number*	Coordenada geográfica de latitude do início da aplicação
lon	*input number*	Coordenada geográfica de longitude do início da aplicação
jsonanima	*input hidden*	Objeto das animações em forma de *String* (No modo Livre este elemento tem o valor *'{}'*)
Submit	*input submit*	Botão de *submit* -> Ação -> 'changedefs.php'

* O nome apresentado para esta *tag* HTML encontra-se precedido de um # para simbolizar que esse nome representa o atributo 'id'.

Emailtoken.php

Este é o ficheiro utilizado na criação de um pedido de alteração de *password*. Começa por ser gerado um *token* encriptado que é guardado na base de dados (tabela gceyes_alt_pass) associado a um prazo (*timestamp*).

De seguida, é enviado um *email* ao utilizador, notificando-o do pedido feito e apresentando um *link* correspondente a um pedido GET ao ficheiro 'altpass.php', já apresentado, gerando-se um *token* como parâmetro. A figura seguinte ilustra um *email* gerado através deste ficheiro.

Figura 72 – Exemplo de *email* de alteração de *password.*

Updatelist.php

Este ficheiro foi criado para efetuar alterações na lista de conteúdos publicitários do utilizador. Recebe como parâmetro (POST) um objeto JSON com um *array* para cada campo correspondente às colunas da tabela gceyes_add_list - **link**, **type** e **freq,** em que o campo **ordered** não é replicado pois o *array* já se encontra ordenado. Esse objeto contém ainda um *array* adicional que lista o tipo de *query* a utilizar para cada elemento do objeto (0 – *UPDATE*, 1 – *INSERT*, -1 – *DELETE*, 2 – Nenhuma *query* a executar).

Um exemplo de uma lista guardada neste objeto pode ser visto abaixo.

```
'newlist':{
  'link':
  {
    0:'https://www.youtube.com/watch?v=Sw-ko6aINI4',
    1:'http://recipp.ipp.pt/retrieve/31671',
  },
  'type':
  {
    0:'vid',
    1:'img'
  },
  'freq':
  {
    0:'30M',
    1:'1H'
  },
  'new':
  {
    0: 0,
    1: 1
  }
}
```

Pela natureza confidencial da informação manipulada nestes ficheiros, os restantes que fazem uso das tabelas Geocar não são apresentados com grande detalhe. Será apenas feita uma breve descrição da sua funcionalidade.

- **Test.php –** Ficheiro que carrega toda a informação dos veículos associados à empresa do *login*. Este ficheiro é chamado com uma frequência de um minuto para a atualizar as posições e estados dos veículos.

- **Restrequest.php –** Ficheiro utilitário para executar pedidos do tipo REST através do comando ”curl”. Apesar de este tipo de pedidos poderem ser executados na parte de *front-end* através do *JavaScript*, afim de proteger chaves de API, foi utilizado este ficheiro para pedidos às APIs.

- **Precursos.php –** Ficheiro que responde com o percurso de um determinado veículo. A pesquisa de percursos é desde o dia anterior até ao. Se o veículo estiver parado nesse período, a informação apenas é obtida desde esse momento até ao momento atual.

- **Eficiencia.php –** Ficheiro que faz o cálculo das médias do combustível gasto e do número de quilómetros percorridos, nos últimos 7 dias.

- **Alertas.php –** Ficheiro que responde com a informação relativa aos alertas gerados pela empresa do utilizador desde a data atual até 30 dias atrás.

- **F_servico.php –** Ficheiro usado para receber os serviços efetuados pela empresa do utilizador desde a data atual até 30 dias atrás.

4.6. CONSIDERAÇÕES SOBRE O DESENVOLVIMENTO

Esta subsecção inclui um conjunto de considerações e justificações sobre algumas escolhas/dificuldades que surgiram ao longo da elaboração do projeto.

setTimeout vs setInterval

A aplicação desenvolvida implicou o uso de diversas animações. Estas foram feitas em *JavaScript* e desenvolvidas através das funções “*setTimeout*” e “*setInterval*”.

A função *setTimeout(func,x)* executa o código definido (parâmetro *func*), após a espera de um determinado intervalo de tempo (parâmetro *x*). O excerto de código abaixo transcrito ilustra o uso desta função.

```
alert(“Olá no inicio do programa”);
var id=setTimeout(
```

```
  function()
  {
          alert("Olá passados 5 segundos");
  },5000);
```

Este código irá criar 2 alertas, um no início do programa e um outro passado 5 segundos (5000 milissegundos). Destaca-se ainda o uso da variável "id", que irá conter um índice identificativo do *setTimeout.* Este poderá ser utilizado para interromper o seu funcionamento através da função *clearTimeout(id)*

A função *setInterval(func,x)* estabelece um ciclo infinito[1] de código a executar (parâmetro *func*) com um intervalo de tempo (parâmetro *x*) definido pelo utilizador.

No excerto de código gora transcrito o seguinte pode notar-se o uso desta função.

```
alert("Olá no inicio do programa");
var i=0;
var id=setInterval(
  function()
  {
     i++;
     var seg=i*5;
          alert("Passaram "+seg+" segundos");
},5000);
```

Neste caso o código gera um número infinito de alertas de 5 em 5 segundos, identificando a totalidade do tempo decorrido.

Dado que a aplicação desenvolvida deverá correr em ciclo infinito, poderia ter sido escolhida a função "*setInterval*". No entanto tal não é o caso, foi recorrido ao uso da função "*setTimeout*" de forma recursiva.

Esta escolha deve-se ao facto de já se usarem pedidos AJAX, que, dependendo da informação da resposta, podem introduzir um compasso de espera, alterando assim os ciclos de funcionamento da aplicação. Assim, o uso da função "*setInterval*" iria dessincronizar a aplicação, uma vez que esta não contabiliza o tempo de espera dos pedidos AJAX.

[1] O ciclo apenas é infinito se o programador assim o decidir. Através do índice do intervalo é possível parar este ciclo com a função clearInterval(id).

Por outro lado, o "*setTimeout*" usado de forma recursiva permite a contabilização deste tempo, independentemente da sua variação. Tal é possível através da chamada desta função após o sucesso do pedido AJAX.

Um exemplo de uma implementação recursiva do "*setTimeout*" que contabiliza o tempo dos pedidos AJAX poderá ser visto no excerto de código abaixo transcrito.

```
function ajaxrequest()
{
  $.ajax({
    url: "pedidoajax.php",
    dataType: "json",
    type: "POST",
    success: function(result){
      //Fazer alguma coisa com o resultado
      process_information(result);
      setTimeout(
      function()
      {
            //Função recursiva
            ajaxrequest();
        },5000); //5 segundos
    },
    error: function(e){
      console.log(e);
    }
  });
}
```

No exemplo dado, mesmo que o pedido AJAX consuma muito tempo, o intervalo até à próxima repetição apenas começa a ser contabilizado quando o pedido termina. Por exemplo, se neste caso o pedido tiver uma duração de 2 segundos, só ao final de 7 é que a função será executada novamente.

Destaca-se também que o exemplo aqui exposto salvaguarda a ocorrência de erros, uma vez que um erro interrompe a recursividade desta função.

Console.log para objetos/arrays JSON variáveis

Uma boa pratica no desenvolvimento de programas é a utilização de métodos de *debbuging*, isto é, métodos que permitem a identificação e correção de erros. Um desses métodos é a impressão das variáveis utilizadas no programa, normalmente numa consola, para verificar a validade dos seus valores. No caso do *JavaScript*, esta funcionalidade é acedida pela função "*console.log*", através da consola do *browser*.

Assim, durante o trabalho esta função foi utilizada enumeras vezes, mas nem sempre com o sucesso desejado. A questão reside no uso das animações e na forma como *browser* faz a impressão de variáveis do tipo de *arrays* JSON ou de *arrays* contidos num objeto JSON.Um "*console.log*" de um *array* contido num objeto poderá ser visto nas figuras 73 e 74.

Figura 73 – Exemplo de um *array* contido num objeto JSON, impresso na consola de *JavaScript*.

Figura 74 – Exemplo de um *array* contido num objeto JSON, impresso na consola de *JavaScript* (expandido).

Como é possível observar, o *browser* não imprime à partida o conteúdo do *array*, mostrando apenas a sua existência e dando a possibilidade de expansão para a visualização dos seus conteúdos. Tal poderá ser problemático no caso de os valores contidos no *array* serem variáveis, pois estes apenas serão avaliados pelo *browser* no momento da expansão do *array*.

No caso da aplicação desenvolvida, esta questão tornou-se uma grande dificuldade na a identificação de erros. Como o programa está em contínuas animações, o valor dos objetos a avaliar está em constante mudança.

Solucionou-se este problema pela transformação dos objetos JSON em *Strings*. Esta solução é efetiva pois obriga o *browser* a interpretar os valores do objeto/*array* no momento da impressão.

Cronologia

O modo Animado desta aplicação é dependente de uma configuração prévia (discutida na página das definições). De modo a criar uma interface intuitiva e de fácil uso, foi feita a sugestão da criação de uma cronologia para a configuração das animações.

Na verdade, foram criadas duas cronologias, uma para os grupos de animações e outra para cada animação. A figura 49, correspondente à página das definições, contém a representação destas cronologias. Para a sua criação, foi considerado vantajoso o uso de uma biblioteca de *JavaScript*, de forma a:

- Não permitir a sobreposição de elementos num determinado momento da cronologia.
- Mostrar a escala horizontal sem datas, apenas representando o intervalo temporal.

No entanto, não foram encontradas bibliotecas que disponibilizassem as funcionalidades enunciadas. Algumas das bibliotecas estudadas foram:

- **Timeline.js –** Biblioteca direcionada para a criação de cronologias encastradas, nomeadamente para páginas *WordPress*. Apresenta a possibilidade de ser carregada através de um objeto JSON, não permitindo, no entanto a adição de elementos dinamicamente. Tal significa que cada elemento novo a adicionar obriga ao recarregamento da totalidade da cronologia.
- **Vis.js** – Apresenta um elevado número de funcionalidades, das quais se destacam, a inclusão de eventos específicos para elementos da cronologia. O facto desta biblioteca apresentar uma cronologia horizontal é também uma mais-valia dado que está conforme a apresentação gráfica pretendida.
- **Timemaps.js** – Esta é uma biblioteca para criação de cronologias interligadas com um mapa. Trabalha com os diversos fornecedores de serviços e bibliotecas de carregamento de mapas. A cronologia criada permite despoletar eventos que se traduzem em ações no mapa, dando a perceção de uma escala temporal do mesmo. Esta biblioteca não é, no entanto, a mais recomendada para a funcionalidade pretendida, uma vez que as configurações da aplicação não deverão fazer uso do mapa. A isto acresce o facto de não cumprir as funcionalidades listadas anteriormente.

Foi descartada a biblioteca Timemaps.js pela dificuldade de implementação que representa sem um contributo significativo no desenvolvimento deste trabalho. Assim, foi escolhida a biblioteca com maior funcionalidade (Vis.js), procedendo-se à sua alteração de acordo com as funcionalidades necessárias.

A sobreposição de elementos foi corrigida arrastando os elementos para os lados quando acontece uma sobreposição.

Para escala horizontal, a solução encontrada foi esconder a data (ano, mês e dia), deixando a hora visível e forçando a cronologia a começar no início de um minuto (por exemplo se a cronologia foi acedida às 12:42:48, apenas é mostrada a hora a partir das 12:43:00).

Dashboard

Uma das funcionalidades implementadas no projeto desenvolvido é o *dashboard*. Esta funcionalidade é aparentemente redundante, dado que a informação aí contida também se encontra repetida noutros elementos.

No entanto, o desenvolvimento do *dashboard* deveu-se 4 condicionantes:

- O modelo para esta funcionalidade já se encontrava no *layout* utilizado e, portanto, foi sugerida a sua utilização.

- A utilização do *dashboard* para a exposição de informação redundante cria múltiplas opções de visualização da aplicação. Assim, no momento de comercialização do produto, será escolhida uma das opções de visualização, pelo cliente ou pela empresa GISGEO.

- O *dashboard* adiciona a funcionalidade de pesquisa. Esta opção é de especial importância para empresas com um elevado número de veículos. Destaca-se que esta funcionalidade se encontra no *dashboard* pelas características do *layout* utilizado.

- O *dashboard* fará parte da próxima versão do Geocar, sendo que o seu desenvolvimento neste projeto constitui uma base para o mesmo.

Assim, apesar da sua redundância, esta funcionalidade foi mantida.

5. TESTES E IMPLEMENTAÇÕES FUTURAS

Neste capítulo são abordados os diferentes testes efetuados para garantir a integridade da aplicação, analisando posteriormente possíveis melhorias e implementações futuras à aplicação.

5.1. TESTES

A realização de testes é um ponto fulcral no desenvolvimento de uma aplicação. Estes permitem evidenciar o nível de integridade da aplicação. Assim, esta secção dedica-se a descrever os testes efetuados e as principais conclusões que estes produziram.

Bases de dados de produção

Durante o desenvolvimento do trabalho, por razões de segurança, foram utilizadas as bases de dados de teste para obter a informação do Geocar, informação nem sempre completa e por vezes datada.

Assim, num momento final foi feita a ligação do projeto desenvolvido às bases de dados de produção. Tal permitiu analisar o comportamento da aplicação com dados reais, verificando-se então na necessidade de proceder a pequenos ajustes à aplicação.

A ligação às bases de dados de produção permitiu ainda um teste adicional, a análise do comportamento da aplicação para uma empresa com um elevado número de veículos.

Este teste foi feito com sucesso, apenas havendo a assinalar um maior compasso de espera para cálculo dos índices de eficiência (o processo mais pesado da aplicação).

Finalmente destaca-se uma funcionalidade que não pode ser testada em ambiente de produção: a emissão de alertas periódicos. Estes não foram testados com dados reais pois para tal implicariam a inserção de dados em contas de clientes. No entanto, através da inserção de valores em ambiente de desenvolvimento (base de dados de teste) confirmou-se o seu funcionamento.

WebTV

O desenvolvimento deste projeto baseou-se na construção de uma aplicação automática para a visualização de informação, direcionada para monitores panorâmicos (*WebTV*).

Por esta razão, um parâmetro importante a ser testado é o funcionamento em ecrãs de diferentes tamanhos, nomeadamente nos panorâmicos. Assim, foi utilizado um ecrã panorâmico da empresa, associado a um computador, para o teste gráfico da aplicação.

Após este teste realizaram-se ajustes em algumas componentes do *layout* que perdiam a formatação com o ajuste do tamanho. Destaca-se que a perda de formatação apenas acontecia para componentes estilizados dinamicamente (estilizados através de *JavaScript*).

Deste modo foram efetuadas as devidas correções e a aplicação encontra-se agora ajustável aos diferentes tamanhos de ecrã possíveis.

Infelizmente, durante o período de testes não foram tiradas fotografias da representação em ecrã panorâmico. No entanto, para ser percetível o aspeto da aplicação num ecrã, a figura 75 apresenta a aplicação iniciada num Macbook Air (13.3 polegadas de ecrã).

Figura 75 – Representação da aplicação num Macbook Air 13.3''.

SQL Injection

De modo a testar a segurança da aplicação foram efetuados testes de SQL *Injection* (secção **2.8**). A não execução das *queries* injetadas refletiu a segurança da aplicação, evidenciando a separação lógica das bases de dados, dado que o acesso às mesmas apenas é feito através de controladores, os quais adotam medidas de segurança para os diferentes tipos de *queries* a executar.

Nestes testes sublinha-se que não foram utilizadas *queries* maliciosas, tendo sido apenas testada a execução de *queries* não desejadas.

5.2. IMPLEMENTAÇÕES FUTURAS

Nesta secção são discutidas as eventuais implementações e progressos futuros afim de melhorar a qualidade do produto. O desenvolvimento das funcionalidades agora avançadas não foi efetuado no decorrer do trabalho por diversas razões: limitações da empresa, limitações dos fornecedores de serviços ou ainda pela sua sugestão tardia dentro do calendário de trabalhos previsto.

Postos de combustível

A localização dos postos de combustível encontra-se implementada na aplicação desenvolvida. Uma das informações importantes deste serviço é preço dos combustíveis, dado que poderá ser determinante na seleção do ponto de abastecimento mais vantajoso. Porém, o preço dos combustíveis encontra-se muitas vezes indisponível. Isto deve-se à falta

de área de cobertura desta informação pelo fornecedor de serviço (HERE *Places*) e em países como Portugal e Angola a sua existência é escassa ou nula.

É expectável que esta informação se expanda com a evolução do fornecedor de serviço, constituindo assim uma melhoria da aplicação desenvolvida.

Custo das portagens

O cálculo do custo de portagens é uma funcionalidade interessante no contexto deste projeto. O seu uso permite uma melhor avaliação dos custos associados a um serviço, representando uma mais-valia para as empresas. A implementação deste serviço foi estudada, tendo sido planeada a sua implementação. Porém, chegou-se à conclusão que este serviço acarreta um custo elevado, independentemente do fornecedor de serviço, não existindo soluções *open-source.* A melhor opção encontrada foi a HERE *Toll Cost* API, que para além do volume de informação e opções que disponibiliza, é desenvolvida pela empresa HERE, que já fornece outros serviços à empresa GISGEO. No entanto, o uso deste novo serviço um reajuste do plano contratado, implicando o pagamento de um valor adicional pelos serviços da HERE. Assim, a empresa considerou que, neste momento, a implementação desta funcionalidade não seria uma mais-valia para o custo que esta representa.

De qualquer modo, foi desenvolvido um modelo de acesso a este recurso (baseado no exemplo de resposta da HERE *Toll Cost* API), que poderá ser posteriormente implementado, na eventualidade deste serviço vir a ser contratado.

Vídeos

Um dos objetivos do trabalho era a possibilidade de inserção de conteúdo publicitário, designadamente imagens estáticas ou animadas e vídeos. Esta opção foi implementada e funciona através de *links* para o conteúdo a inserir. No entanto, a funcionalidade está limitada a determinados formatos para as imagens (PNG, JPG, GIF, JPEG) e a vídeos do *Youtube*. Os formatos das imagens encontram-se limitados por uma questão de segurança, sendo que as extensões definidas são usadas de modo a garantir que se trata de uma imagem. Isto é efeito através de uma verificação da terminação do *link.*

Como é necessário o uso de uma API para cada prestador de serviço, apenas foram considerados os vídeos pertencentes ao *Youtube* (pela sua popularidade). Teoricamente, seria possível implementar os vídeos de forma encastrada. Infelizmente, dado o carácter

automático da página, a duração do vídeo tem de ser contabilizada, necessitando assim do uso de uma API específica.

Assim, uma melhoria possível é a inserção de vídeos provenientes de outros servidores (*Vimeo*, *Dailymotion*, entre outros).

Melhorias externas

As melhorias/implementações futuras até aqui abordadas têm todos um carácter interno, isto é, dizem respeito apenas à própria aplicação e a consequentes alterações da mesma. Contudo, existem também possíveis melhorias de carácter externo, nomeadamente as possíveis melhorias dos fornecedores dos serviços de informação.

Destaca-se o peso que o processamento das operações de cálculo de índices de eficácia representa, considerando o número elevado de dados manipulados. Por essa razão, foi sugerido que a informação utilizada para este processo fosse compactada numa nova tabela da base de dados Geocar, reduzindo o número de dados a processar. A passagem entre tabelas poderá ser feita através de *triggers* periódicos (operações SQL despoletadas por determinada condição) periódicos. Esta operação está marcada para uma implementação futura da empresa e sem dúvida irá melhorar o desempenho da aplicação.

6. CONCLUSÕES

Ao longo deste relatório foram apresentadas diversas justificações para a sustentação das opções de desenvolvimento efetuadas no trabalho. Pretendeu-se aqui fazer uma síntese que compreenda as principais conclusões, consequências e relevância do projeto realizado.

A concretização do projeto implicou a utilização de diferentes tipos informação disponibilizada por diversos fornecedores. O estudo e análise das diferentes opções oferecidas permitiu uma seleção consciente e ponderada das mais adequadas em cada caso. Simultaneamente permitiu um conhecimento detalhado do leque de opções utilizáveis na implementação de outros produtos pela empresa.

Estes serviços são todos disponibilizados em formato de API. As APIs dos serviços disponíveis foram estudadas ao longo do trabalho, permitindo uma análise das mais-valias existentes em cada opção. Destaca-se a seleção das APIs HERE, pelas suas inúmeras funcionalidades e dado que a HERE já tinha um contrato com a empresa onde foi desenvolvido este projeto

Evidencia-se ainda o uso de *Bootstrap e jQuery* neste projeto, que, independentemente de serem requisitos da empresa, foram também uma mais-valia dado que, permitiram uma maior facilidade no desenvolvimento gráfico, nomeadamente na sua configuração e

simplificação. Assim, estas bibliotecas de *JavaScript* permitiram uma implementação fácil do *layout*, mantendo a sua responsividade em diferentes dispositivos.

Sublinha-se ainda as vantagens do uso de PostgreSQL, que sendo uma base de dados objeto-relacional, permitiu o armazenamento de objetos num formato tradicional de bases de dados relacionais. O uso desta funcionalidade torna-se particularmente útil, nomeadamente no armazenamento do objeto JSON utilizado para as animações.

Destaca-se a elaboração de funcionalidades redundantes, caracterizadas por estilos ou posições diferentes. Estas permitirão, no momento de comercialização do produto, a escolha (do cliente ou da empresa), aumentando assim o leque de possibilidade gráficas.

Foram cumpridos vários requisitos impostos pela empresa, como o uso do *layout* proposto (Geocar v2), bem como o desenvolvimento nas linguagens previamente definidas (HTML, CSS, *JavaScript* e PHP). A lei de proteção de dados é também cumprida através do bloqueio do acesso a informação fora do horário laboral.

O projeto desenvolveu-se com o total cumprimento dos objetivos previamente definidos pela empresa, a saber:

- A aplicação contempla a listagem diária dos serviços agendados, evidenciando o seu estado.

- A solução desenvolvida apresenta informação do fuso horário, de meteorologia e do tráfego rodoviário referente à localização do cliente no mapa. É feita ainda a representação de um logótipo, criado com base no logótipo do Geocar.

- É feita a divulgação de informação interna em diferentes formatos (*flash*, vídeo, imagem) para a disponibilização de conteúdos publicitários de parceiros comerciais.

- Foi criada a página de gestão de conteúdos representada pela página de configuração das definições.

- Foi criada uma página para inclusão temporária de conteúdo representada pela página de alteração de conteúdo publicitário.

O sucesso desta aplicação já se refletiu na divulgação do produto a clientes, pelo responsável comercial. De igual modo, o responsável pelo desenvolvimento de uma nova versão do

Geocar considera que o projeto desenvolvido poderá ser utilizado como modelo para esta nova versão.

A título pessoal considero que o desenvolvimento deste trabalho teve um balanço francamente positivo. Nas tarefas executadas ampliei o meu leque de competências e amadureci também as minhas capacidades na área de desenvolvimento *web*. A inserção no contexto empresarial (em formato de Estágio Curricular) para a elaboração do projeto foi também importante. A inserção neste ambiente motivou o aperfeiçoamento das minhas competências de relacionamento interpessoal e alargou a minha perspetiva sobre a estruturação dos negócios.

Finalmente sublinho a mais-valia curricular que o desenvolvimento de *software* geográfico representa, tendo em conta que as funcionalidades geográficas são cada vez mais requeridas para o desenvolvimento de aplicações *web*.

Referências Documentais

[1] Bourgeois, D. T. (2004). Chapter 1: What Is an Information System? | Information Systems for Business and Beyond. In *Information Systems for Business and Beyond*. Retrieved from https://bus206.pressbooks.com/chapter/chapter-1/

[2] Alter, S. (1999). A general, yet useful theory of information systems. *Communications of the Association for Information Systems*, *1*(Article 13), 1–70. http://doi.org/0579

[3] "Gisgeo - information systems." Accessed April 9,2017. Retrieved from http://www.gisgeo.pt/index.php

[4] Decreto Lei no 67/98 de 26 de Outubro - Lei de proteção de Dados Pessoais, Pub. L. No. Diário da República º 247/1998, Série I-A (1998).

[5] Pinto, Inês (2009). "Introdução Aos Sistemas de Informação Geográfica (SIG)." Retrieved from http://www2.iict.pt/archive/doc/georrefIntroducaoSIG_InesPinto.pdf.

[6] Ângela, P., Da, A., & Santos, S. (n.d.). Os Sistemas de Informação Geográfica no ensino da geografia: aplicação a uma turma do 3º ciclo do ensino básico. Retrieved from https://run.unl.pt/bitstream/10362/5365/1/TEGI0270.pdf

[7] Alesheikh, AA., Helali, H., Behroz, HA. (2002). Web GIS: Technologies and Its Applications.

[8] P., Kumar Sampath, Gautam, V. K. (2016). Open source tools facilitate image georeferencing over internet. In *Geospacial World*. Retrieved from https://www.geospatialworld.net/article/open-source-tools-facilitate-image-georeferencing-over-internet/

[9] De, B. (2017). *API Management*. *An Architect's Guide to Developing and Managing APIs for Your Organization*. http://doi.org/10.1007/978-1-4842-1305-6_2

[10] Simple Object Access Protocol (SOAP) 1.1. (n.d.). Retrieved July 27, 2017, from https://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383491

[11] Understanding SOAP. (2003). Retrieved July 27, 2017, from https://msdn.microsoft.com/en-us/library/ms995800.aspx

[12] Leach, P. J., Berners-Lee, T., Mogul, J. C., Masinter, L., Fielding, R. T., & Gettys, J. (n.d.). Hypertext Transfer Protocol -- HTTP/1.1. Retrieved from https://tools.ietf.org/html/rfc2616#page-51

[13] Fielding, R. T. (2000). *Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)*. Retrieved from https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[14] Extensible Markup Language (XML). (2006). Retrieved August 1, 2017, from https://www.w3.org/XML/

[15] XML Tutorial. (n.d.). Retrieved August 1, 2017, from https://www.w3schools.com/xml/default.asp

[16] JSON. (n.d.). Retrieved July 12, 2017, from http://json.org/

[17] Using SQL bind variables for application performance and security. (n.d.). Retrieved September 25, 2017, from https://www.ibm.com/developerworks/library/se-bindvariables/index.html

[18] Introduction · Bootstrap. (n.d.). Retrieved August 1, 2017, from https://getbootstrap.com/docs/4.0/getting-started/introduction/

[19] Bootstrap Resources and Plugins List - Start Bootstrap. (n.d.). Retrieved August 1, 2017, from https://startbootstrap.com/bootstrap-resources/#plugins

[20] PostgreSQL: About. (n.d.). Retrieved August 2, 2017, from https://www.postgresql.org/about/

[21] Bousnina, F. E., Elmi, S., Tobji, M. A. B., Chebbah, M., Hadjali, A., & Ben Yaghlane, B. (2016). Object-relational implementation of evidential databases. *Proceedings - 2016 International Conference on Digital Economy: Emerging Technologies and Business Innovation, ICDEc 2016*, 80–87. http://doi.org/10.1109/ICDEC.2016.7563149

[22] Relational Algebra. (n.d.). Retrieved August 2, 2017, from http://cnx.org/contents/NAA4OPSt@1/Relational-Algebra

[23] Object Oriented Databases. (n.d.). Retrieved August 2, 2017, from http://www.comptechdoc.org/independent/database/basicdb/dataobject.html

[24] PostGIS — PostGIS Feature List. (n.d.). Retrieved August 2, 2017, from http://postgis.net/features/

[25] OpenLayers - Basic Concepts. (n.d.). Retrieved August 2, 2017, from http://openlayers.org/en/latest/doc/tutorials/concepts.html

[26] Documentation - Leaflet - a JavaScript library for interactive maps. (n.d.). Retrieved August 2, 2017, from http://leafletjs.com/reference-1.2.0.html

[27] API - OpenStreetMap Wiki. (n.d.). Retrieved August 26, 2017, from http://wiki.openstreetmap.org/wiki/API

[28] Getting Started with the Bing Maps REST Services. (n.d.). Retrieved August 27, 2017, from https://msdn.microsoft.com/en-us/library/ff701702.aspx

[29] Bing Maps Tile System. (n.d.). Retrieved August 27, 2017, from https://msdn.microsoft.com/en-us/library/bb259689.aspx

[30] Google Maps APIs | Google Developers. (n.d.). Retrieved August 27, 2017, from https://developers.google.com/maps/

[31] Google Maps Web Service APIs | Google Developers. (n.d.). Retrieved August 28, 2017, from https://developers.google.com/maps/web-services/

[32] MapQuest Development and Documentation - Getting Started - MapQuest API Documentation. (n.d.). Retrieved August 28, 2017, from https://developer.mapquest.com/documentation/

[33] Yandex Technologies - Maps API. (n.d.). Retrieved August 28, 2017, from https://tech.yandex.com/maps/doc/jsapi/

[34] Build apps with HERE Maps API and SDK Platform Access - HERE Developer. (n.d.). Retrieved August 28, 2017, from https://developer.here.com/documentation

[35] Primeiros passos | Google Maps Time Zone API | Google Developers. (n.d.). Retrieved September 6, 2017, from https://developers.google.com/maps/documentation/timezone/start

[36] Moment Timezone | Docs. (n.d.). Retrieved September 6, 2017, from https://momentjs.com/timezone/docs/

[37] GeoNames Web Service Documentation. (n.d.). Retrieved September 6, 2017, from http://www.geonames.org/export/web-services.html#timezone

[38] API Access for Developers - TimeZoneDB. (n.d.). Retrieved September 6, 2017, from https://timezonedb.com/api

[39] Weather API - OpenWeatherMap. (n.d.). Retrieved September 8, 2017, from https://openweathermap.org/api

[40] AccuWeather Enterprise API - AccuWeather Enterprise API Documentation. (n.d.). Retrieved September 8, 2017, from http://apidev.accuweather.com/developers/

[41] API | Weather Underground. (n.d.). Retrieved September 8, 2017, from https://www.wunderground.com/weather/api/d/docs

[42] simpleWeather.js - Pretty much the best jQuery plugin in the world. (n.d.). Retrieved September 8, 2017, from http://simpleweatherjs.com/

[43] Overview - Weather API - HERE Developer. (n.d.). Retrieved September 8, 2017, from https://developer.here.com/documentation/weather/

[44] Weather API | Local Weather API | World Weather Online. (n.d.). Retrieved September 8, 2017, from https://developer.worldweatheronline.com/api/

[45] Google Places API Web Service | Google Developers. (n.d.). Retrieved September 9, 2017, from https://developers.google.com/places/web-service/

[46] Introduction - Places (Search) API - HERE Developer. (n.d.). Retrieved September 9, 2017, from https://developer.here.com/documentation/places/

[47] Locations API. (n.d.). Retrieved September 9, 2017, from https://msdn.microsoft.com/en-us/library/ff701715.aspx

[48] Places API — Yandex Technologies. (n.d.). Retrieved September 9, 2017, from https://tech.yandex.com/maps/doc/tariffs/geosearch/index-docpage/

[49] Chart.js · GitBook. (n.d.). Retrieved September 15, 2017, from http://www.chartjs.org/docs/latest/

[50] JavaScript Charts - amCharts. (n.d.). Retrieved September 15, 2017, from https://www.amcharts.com/JavaScript-charts/

[51] Using Google Charts | Charts | Google Developers. (n.d.). Retrieved September 15, 2017, from https://developers.google.com/chart/interactive/docs/