NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

DIOGO ALEXANDRE SILVA LOPES

Licenciado em Engenharia Informática

SISTEMA DE INFORMAÇÃO PARA MONITORIZAÇÃO DE FAIXAS DE GESTÃO DE COMBUSTÍVEL DE INCÊNDIOS

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa

Março, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENTO DE
INFORMÁTICA

SISTEMA DE INFORMAÇÃO PARA MONITORIZAÇÃO DE FAIXAS DE GESTÃO DE COMBUSTÍVEL DE INCÊNDIOS

DIOGO ALEXANDRE SILVA LOPES

Licenciado em Engenharia Informática

Orientador: Carlos Augusto Isaac Piló Viegas Damásio
Professor Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Coorientador: João Carlos Gomes Moura Pires
Professor Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente: Miguel Carlos Pacheco Afonso Goulão
Professor Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Arguentes: Luís Miguel Mendonça Rato
Professor Associado, Escola de Ciências e Tecnologia da Universidade de Évora

Carlos Augusto Isaac Piló Viegas Damásio
Professor Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Março, 2023

Sistema de Informação para Monitorização de Faixas de Gestão de Combustível de Incêndios

Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo NOVAthesis (v6.10.18) [33].

Agradecimentos

Gostaria de começar por agradecer ao meu orientador Carlos Damásio e ao meu co-orientador João Moura Pires por todo o apoio e ensinamentos que me transmitiram ao longo da elaboração desta dissertação. Desde o seu início, que senti que criaram as condições para que, com esforço e dedicação, pudesse alcançar as minhas capacidades. O meu sincero obrigado.

De seguida, queria agradecer a todos os restantes envolvidos no projeto Floresta Limpa, sobretudo ao Miguel Correia e ao Vasco Lucena, por todas as horas que perdemos a trocar ideias e a tentar que as coisas funcionassem, mesmo quando tal parecia pouco provável ou mesmo impossível.

Também não me posso esquecer dos amigos que fiz ao longo da faculdade, tornaram esta experiência melhor e ajudaram-me a manter o foco durante esta dissertação. Vou-vos estar sempre grato.

Agradecer também aos meus amigos fora da universidade, por todas as vezes que ouviram que não podia ir a algum lado porque tinha de trabalhar na tese, por me apoiarem e incentivarem a dar o meu melhor.

Por fim, agradecer às pessoas que mais tirei do sério durante esta dissertação, à minha mãe, ao meu pai e ao meu irmão. Obrigado por perderem o vosso tempo a ler esta dissertação comigo, por todas as vezes que me ouviram a falar dum assunto do qual não percebiam, por estarem lá para mim e por, sobretudo, nunca me deixarem ir a baixo. Sem vocês não tinha conseguido.

Resumo

Todos os anos, o território português é afetado por incêndios rurais e florestais, que causam problemas para a população e para o ambiente. Para minimizar as consequências destes, está atualmente em vigor o Decreto-Lei nº 82/2021, que estipula a criação das faixas de gestão de combustível de incêndios (FGCI), que são zonas com menor vegetação ou mesmo inexistente, ao redor de várias estruturas.

No entanto, existem fatores que dificultam a monitorização das faixas, tais como mudanças repentinas no estado da vegetação devido a efeitos climatéricos. Para isso no âmbito do projeto Floresta Limpa, estão a ser desenvolvidas técnicas de aprendizagem automática, através da deteção remota para ajudar a monitorização dessas faixas. No entanto, o projeto carecia dum sistema de informação que suportasse os diversos dados necessários para cumprir o seu propósito, funcionando da forma mais eficiente e genérica possível.

Esta dissertação centra-se, assim, na modelação e implementação de um sistema de informação que seja aplicável ao projeto Floresta Limpa. Este é um sistema de informação geográfica, uma vez que armazena um conjunto de dados disponibilizados pelas entidades competentes no âmbito do ordenamento do território e da conservação da natureza. Além disso, guarda dados de satélites, que servem de base para a deteção remota, e fornece suporte a uma aplicação móvel que permite a recolha de dados no campo. Por fim, disponibiliza, publicamente, um conjunto de estatísticas sobre os dados obtidos que são fornecidos às autoridades que atuam na área da proteção dos incêndios florestais.

Dessa forma, o sistema contribui para o desenvolvimento do projeto Floresta Limpa, acomodando as funcionalidades desenvolvidas nas restantes tarefas, facilitando o trabalho das entidades competentes na monitorização das FGCI.

Palavras-chave: Sistema de Informação, Sistema de Informação Geográfica, Dados recolhidos no campo, Dados de Satélite, Informação geográfica, Base de dados, Faixas de gestão de combustível de incêndios . . .

Abstract

Every year, the Portuguese territory is affected by rural and forest fires, which cause problems for the population and the environment. To minimize the consequences of these, the Decree-Law No. 82/2021 is currently in force, which stipulates the creation of fuel management zones (FMZ), which are areas with less vegetation or even nonexistent, around various structures.

However, there are factors that make monitoring these zones difficult, such as sudden changes in the state of the vegetation due to weather effects. To this end, under the Floresta Limpa project, machine learning techniques are being developed through remote sensing to help the monitoring of these zones. However, the project lacks an information system that supports the various data needed to fulfill its purpose, working as efficiently and generically as possible.

This dissertation focuses on the modeling and implementation of an information system applicable to the Floresta Limpa project. This will be a geographic information system, since it will store a set of data provided by the competent entities in the area of spatial planning and nature conservation. In addition, it will store satellite data, which will serve as the basis for the remote sensing, as well as providing support for a mobile application that will allow data collection in the field. Finally, it will make publicly available a set of statistics on the data obtained that will be provided to the authorities working in the area of forest fire protection.

In this way, the system will contribute to the development of the Floresta Limpa project, accommodating the functionalities developed in other tasks, facilitating the work of the competent entities in monitoring the FMZ.

Keywords: Information System, Geographic Information System, Field collected data, Satellite data, Geographic information, Database, Fuel management zones ...

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Siglas

CAOP	Carta Administrativa Oficial de Portugal
COS	Carta de Uso e Ocupação de Solo
DEM	Digital Elevation Model
DGT	Direção-Geral do Território
FGCI	Faixas de Gestão de Combustível de Incêndios
FIC	Faixas de Interrupção de Combustível
FRC	Faixas de Redução de Combustível
FWI	Fire Weather Index
GAE	Google App Engine
GCL	Google Cloud Logging
GCP	Google Cloud Platform
GCS	Google Cloud Storage
GDAL	Geospatial Data Abstraction Library
GEE	Google Earth Engine
GFE	Google Frontend Server
ICNF	Instituto da Conservação da Natureza e das Florestas
IFN	Inventário Florestal Nacional
OGC	Open Geospatial Consortium
PMDFCI	Plano Municipal de Defesa da Floresta contra Incêndios
SGIFR	Sistema de Gestão Integrada de Fogos Rurais no território continental
SNDFCI	Sistema Nacional de Defesa da Floresta Contra Incêndios
WFS	Web Feature Service
WMS	Web Map Service

Capítulo 1

Introdução

1.1 Contexto e Motivação

Os incêndios florestais representam um dos principais problemas na manutenção da sustentabilidade do ecossistema terrestre.

Anualmente, milhares de hectares de floresta são consumidos pelas chamas em Portugal, tornando os incêndios florestais, uma das piores catástrofes naturais a ocorrer no nosso país, causando inúmeras consequências de ordem material, pessoal e ambiental [12].

De forma a prevenir as consequências causadas pelos incêndios, bem como melhorar a resposta a estes cenários, foi necessário apresentar novas medidas. Assim no ano de 2006 entrou em vigor o Sistema Nacional de Defesa da Floresta Contra Incêndios (SNDFCI) [4], revogado atualmente pelo Decreto-Lei referente ao Sistema de Gestão Integrada de Fogos Rurais no território continental (SGIFR) [9], com exceção dos pontos referidos na sua norma transitória constante do artigo 79º que ainda se mantêm em vigor. Nestes Decretos-Lei são introduzidos, entre outros fatores de prevenção, os critérios referentes à criação das redes primárias, secundárias e terciárias de faixas de gestão de combustível de incêndios (FGCI).

As FGCI correspondem a faixas colocadas em locais estratégicos com o intuito de prevenir a propagação de incêndios, facilitar o seu combate, bem como proteger as infraestruturas perto das suas localizações. Para tal, mostra-se necessário proceder à remoção de grande parte da vegetação e dar cumprimento a uma série de normas acerca destas, estipuladas na legislação.

Para que as FGCI cumpram a sua função é necessário que estas sejam limpas regularmente. De forma a garantir estas condições surge a necessidade de haver uma contínua

monitorização do estado da vegetação ao longo das faixas, garantindo que todos os requisitos enumerados na lei continuam a ser cumpridos pelos responsáveis estipulados [9].

No entanto, apesar da existência de monitorização das FGCI existem fatores que dificultam este processo às entidades responsáveis, nomeadamente as condições climatéricas que podem provocar mudanças repentinas no estado da vegetação, trazendo como consequência a necessidade de tornar este processo contínuo. Além disso, o facto das FGCI apresentarem dimensões de grandes extensões dificulta a monitorização da sua área total, tornando este processo mais demorado [2].

Desse modo, surgiu o projeto Floresta Limpa [41] com o objetivo de arranjar novas ferramentas que facilitem a monitorização das FGCI, acelerando o processo de deteção das quais necessitam de intervenção, garantindo a sua manutenção atempada. Para isso, o projeto tem por base o recurso a processos automáticos ou semi-automáticos que permitem identificar o estado da faixa, isto é, se esta necessita ou não de intervenção. Estas condições podem ser inferidas através da obtenção do estado de vegetação das faixas, através de imagens de satélites, bem como, de imagens fornecidas por voluntários ou profissionais. Desta forma, os dados inferidos a partir destas funcionalidades podem ser utilizados no combate aos incêndios, permitindo o conhecimento das áreas que podem ser focos de contenção de incêndios.

Todavia, apesar de já existirem avanços no projeto na área da deteção remota, o projeto carecia de um sistema de informação que suportasse toda a informação obtida quer dos satélites, quer dos voluntários ou de outras fontes, bem como de toda a informação gerada a partir destes dados. Este sistema tem como objetivo, para além de armazenar os dados com respeito às FGCI, servir de suporte a uma aplicação móvel, que é essencial para a fiabilidade dos resultados obtidos, para o complemento da informação dos dados da deteção remota, para a criação duma consciência coletiva sobre a importância das FGCI na prevenção dos incêndios, bem como, para a disponibilização de estatísticas, que poderão ser utilizadas pelas autoridades competentes no âmbito da proteção dos incêndios rurais.

Assim sendo, nesta dissertação é realizado o desenho e implementação de um sistema de informação genérico que suporte projetos nesta área, nomeadamente para o projeto Floresta Limpa.

1.2 Problema

As FGCI necessitam de uma constante monitorização e devem ser alvo de intervenções periódicas de forma a garantir que todas as características estipuladas na legislação são cumpridas.

Para que as faixas cumpram o seu propósito, é necessário que uma série de informações sobre estas sejam disponibilizadas, tais como as datas do último levantamento de características do terreno, o tipo de intervenção a realizar num determinado ano, o responsável pela gestão de combustível, o tipo de faixa, o objetivo da faixa, entre outros.

Apesar da maior parte dos municípios conseguir manter atualizada esta informação, existem alguns onde não existe informação sobre as suas faixas e outros onde esta se encontra desatualizada, dificultando o trabalho da sua monitorização [15].

Quando estes dados são disponibilizados, não é garantido o estado de limpeza das faixas, uma vez que existe uma diversidade de fatores, nomeadamente, meteorológicos que podem levar à alteração do estado da biomassa numa faixa de forma, relativamente, repentina. Para isso, é necessário, uma monitorização praticamente constante de cada uma das faixas, o que não se verifica neste momento.

No entanto, graças ao projeto Floresta Limpa, o processo de monitorização das faixas é facilitado devido ao recurso a processos semi-automáticos e automáticos que podem dar origem a classificações do estado da vegetação em espaços reduzidos de tempo, indicando se essa faixa naquele momento necessita de intervenção ou não. Além disso, o facto de ter sido desenvolvida em paralelo uma aplicação móvel, com o intuito de aumentar a sensibilização dos utilizadores para a importância da manutenção do estado de limpeza destas faixas, origina uma nova fonte de dados que complementa os dados em falta ou desatualizados, partilhados pelos municípios.

Para isso é necessário, a incorporação destes dados, no sistema de informação desenvolvido, que fornece novas ferramentas de apoio à prevenção e ao combate dos incêndios.

1.3 Objetivos e Contribuições

Esta dissertação, tal como foi referido em 1.1, tem como objetivo a modelação e criação de um sistema de informação para suportar o projeto Floresta Limpa.

Este sistema é capaz de integrar todos os dados colecionados e processados no âmbito deste projeto, nomeadamente, dados provenientes de satélite, dados de previsão meteorológica, mapas de risco de incêndio, cartografias, dados provenientes do processamento semi-automático e automático, entre outros dados fornecidos por entidades externas.

Além do referido, o sistema incorpora dados multimédia georreferenciados obtidos *in-situ* por parte dos utilizadores de forma a integrar mais fontes de informação para gerir o estado das FGCI e tornar os resultados mais fidedignos. Para isso o sistema providencia uma API de suporte para uma aplicação móvel, que é o principal meio de disponibilização de informação voluntária.

Em relação à disponibilização de dados, o sistema de informação partilha dados georreferenciados sobre as FGCI, obtidos a partir do processamento dos algoritmos de aprendizagem automática na área da deteção remota realizados na componente privada do Sistema de Informação. Além disso, todos os dados obtidos são disponibilizados em formato *open-source* e através de informações estatísticas de forma a serem partilhados com as entidades competentes.

Devido à heterogeneidade do tipo de dados presentes, o sistema foi desenvolvido numa plataforma *cloud* pública e foram utilizados diversos tipos de sistemas de armazenamento de forma a garantir uma distribuição eficiente de cada tipo de dados.

Em suma, esta dissertação contribui para o desenvolvimento do projeto Floresta Limpa, facilitando o trabalho das entidades competentes na monitorização das faixas de gestão de combustível de forma a garantir a prevenção e a facilitar o combate a incêndios em Portugal.

O sistema, contribui para isso, sendo uma fonte de armazenamento de dados relevantes para outras tarefas do projeto, numa primeira fase.

Posteriormente, contribui, também, para a disponibilização da aplicação móvel aos utilizadores, de forma a que seja obtida informação suplementar sobre as FGCI.

Finalmente, após terem sido integrados estes dois aspetos, o sistema permite o desenvolvimento de informações estatísticas e fornece meios às entidades competentes para fiscalizarem o estado das FGCI com uma maior eficácia.

1.4 Estrutura do Documento

Este documento encontra-se divido em diversos capítulos:

- **Introdução:** O Capítulo 1 realiza uma breve introdução ao tema desta dissertação, destacando o problema geral, as motivações bem como as contribuições desta dissertação para o projeto.
- **Estado da Arte e Trabalho Relacionado:** No Capítulo 2 é apresentado o trabalho de pesquisa realizado para a dissertação, nomeadamente, introduz o conceito das FGCI, informação geográfica, bem como as características dos dados de satélites. Por fim, apresenta uma breve introdução a tecnologias consideradas relevantes, bem como a trabalhos relacionados com a área em questão.
- **Modelação do Sistema de Informação:** Relativamente ao capítulo 3, este descreve todo o processo de modelação do Sistema de Informação, nomeadamente, a sua arquitetura, modelo de dados e estrutura da API REST que o suporta.
- **Implementação:** No Capítulo 4 é apresentada a lógica relativa à implementação do Sistema de Informação, explicitando as tecnologias e os processos realizados.
- **Avaliação:** Quanto ao Capítulo 5, este procura realizar uma avaliação de desempenho às funcionalidades desenvolvidas na proposta apresentada nesta dissertação.
- **Trabalho Futuro:** Por último, em 6 pretende-se identificar um conjunto de funcionalidades a desenvolver no futuro de forma a melhorar a solução já desenvolvida, apresentando uma breve conclusão sobre o estado final desta.

Capítulo 2

Estado da Arte e Trabalhos Relacionados

Esta dissertação tem como objetivo, a criação de um sistema de informação para projetos de monitorização de faixas de gestão de combustível de incêndios e, por esse motivo, foi efetuado um estudo sobre o tipo de dados que foram necessários de ter em consideração.

Visto que o foco deste projeto são as FGCI foi necessário proceder à análise da legislação em vigor, de forma a compreender a informação que é necessária conter no sistema de informação.

Além disso, foram estudadas as formas de representação das FGCI, através da análise de sistemas de informação geográfica, bem como da informação que pode ser obtida sobre estas através de dados de satélites. Dessa forma foi, também, importante estudar as características dos satélites utilizados para perceber o tipo de informação que é útil para armazenar no sistema de informação.

Assim sendo, numa primeira fase deste capítulo é feita uma introdução sobre estes conceitos, sendo apresentadas, posteriormente, as tecnologias consideradas relevantes para a implementação do sistema de informação, bem como, trabalhos relacionados que servem de suporte a esta dissertação.

2.1 Faixas de Gestão de Combustível de Incêndios

O projeto, onde esta dissertação está inserida, tem por base a monitorização das FGCI. Por esse motivo, é necessário compreender o que estas representam e é necessário averiguar todos os critérios que lhes estão associados, de forma a saber o que é necessário representar no sistema de informação.

Em primeiro lugar, as FGCI correspondem a áreas estratégicas, com o objetivo de impedir a propagação dos incêndios, onde é realizada a modificação da estrutura e a

remoção da vegetação.

Para tal, é indispensável que estas cumpram as características descritas na lei. No entanto, estas características variam consoante o tipo de faixa. Assim sendo, é necessário compreender como é feito o processo de criação e manutenção de cada tipo de faixa, de forma a verificar que características são imprescindíveis no sistema de informação.

Em suma, nesta secção são abordadas as características que as FGCI devem cumprir, bem como descrito o processo de criação de cada tipo de faixa, tendo em conta o que está estipulado no Decreto-Lei 82/2021 [9].

2.1.1 Tipos de Faixas de Gestão de Combustível de Incêndios

De forma, a garantir que as FGCI cumprem com o seu objetivo foi necessário delinear dois tipos de faixas, classificadas de acordo com o tipo de intervenção que estas representam.

Desse modo, as intervenções realizadas às FGCI, permitem a divisão das faixas em: **Faixas de Interrupção de Combustível (FIC)** e **Faixas de Redução de Combustível (FRC)**. As FIC consistem em áreas onde ocorre a remoção total dos combustíveis, enquanto que as FRC são faixas onde se realiza a remoção do combustível de superfície, sendo introduzido, também, um espaçamento entre copas, suprimindo a sua parte inferior.

Em relação às FRC houve a necessidade de introduzir um conjunto de critérios de forma a definir a remoção que deve existir nestas faixas na rede secundária, com a sua versão mais recente estipulada no anexo do Decreto-Lei 10/2018 [29], que ainda se mantém em vigor devido à norma transitória presente na legislação atualmente em vigor, referente às faixas de gestão de combustível. Apesar destes critérios serem referentes à rede secundária, estes também serviram de ponto de partida para os critérios da rede primária e terciária, seguindo o estipulado pelo ICNF [11].

No entanto, visto que são introduzidas regras distintas entre os diversos tipos de rede, estas serão, posteriormente, identificadas nos tópicos 2.1.2.1 e 2.1.2.2.

2.1.2 Redes de Faixas de Gestão de Combustível de Incêndios

As FGCI formam uma rede dividida em três níveis, de acordo com a função que cada tipo de faixa representa:

a) Diminuição da superfície percorrida por grandes incêndios, facilitando uma intervenção direta de combate ao fogo.

b) Redução dos efeitos da passagem de incêndios, protegendo de forma passiva vias de comunicação, infraestruturas e equipamentos sociais, zonas edificadas e formações florestais e agrícolas de valor especial.

c) Isolamento de potenciais focos de ignição de incêndios.

Dessa forma, a organização por níveis permitiu definir: a **Rede Primária**, a **Rede Secundária** e a **Rede Terciária**. Em relação à rede primária, esta é definida a nível regional, estando a cargo do ICNF. Quanto às redes secundárias e terciárias, estas são definidas ao nível sub-regional, estando sob alçada dos municípios e das localidades.

A rede primária cumpre com a função descrita na alínea a) (deixou de ser necessário o cumprimento das outras alíneas no novo Decreto-Lei), enquanto que a rede secundária desempenha as funções referidas na alínea b) e c), ficando a rede terciária, apenas com a função da alínea c).

2.1.2.1 Rede Primária de Faixas de Gestão de Combustível

As FGCI pertencentes à rede primária têm como principal objetivo cumprir com a função descrita, previamente, na alínea a), sendo aplicadas em territórios rurais. Além disso, a sua criação ocorre em locais estratégicos, de condições favoráveis ao combate a incêndios, de forma a tornarem-se numa mais-valia para as entidades competentes.

Para tal, existe uma série de características que estas faixas têm de respeitar. As faixas da rede primária têm uma largura de pelo menos 125 metros, dividindo o território português em áreas entre 500 e 10 000 ha.

Como foi referido, em 2.1.1, o ICNF baseou-se nas regras referentes às faixas secundárias, estipulando a existência duma rede viária de 5 metros, seguida duma faixa de 10 metros do tipo FIC. Após esta faixa, seguem-se duas faixas do tipo FRC de 20 e 30 metros, que apresentam um espaçamento de copas de 4 e 2 metros, respetivamente. A estruturação destas faixas ocorre de forma semelhante para cada um dos lados da rede viária.

Além destas características, a quantidade de combustíveis não pode exceder os 2000 m3/ha, sendo que a altura máxima da vegetação varia em função da percentagem da cobertura do solo [11].

2.1.2.2 Rede Secundária e Terciária de Faixas de Gestão de Combustível

As FGCI pertencentes à rede secundária têm como objetivo cumprir com as funções descritas na alínea b) e na alínea c), enquanto que as faixas pertencentes à rede terciária procuram desempenhar a função referida na alínea c). Estas redes são de interesse sub-regional e são definidas no Plano Municipal de Defesa da Floresta Contra Incêndios (PMDFCI) [10].

Estas redes destacam-se por se desenvolverem em diversas envolventes, sendo que no caso da rede terciária, esta dedica-se, particularmente, aos instrumentos de gestão florestal, apresentando regras específicas para cada um dos casos.

No caso das faixas que rodeiam vias rodoviárias e ferroviárias, estas devem apresentar uma largura de pelo menos 10 metros em cada uma das margens.

No caso das linhas de transporte de rede elétrica é feita uma distinção entre linhas de alta, média e baixa tensão. No que se refere às linhas de alta tensão, é necessário

estabelecer uma faixa com a largura igual à dos cabos, acrescida de uma faixa de pelo menos 10 metros para cada um dos lados. Esta situação também ocorre para as linhas de média tensão, sendo que a largura das faixas adicionais podem ser apenas de 7 metros. Para as linhas de baixa tensão é necessário estabelecer uma faixa de largura não inferior a 3 metros a partir do centro dos cabos.

Em relação à rede de transporte de gás e produtos petrolíferos é definida a criação de uma faixa lateral de pelo menos 7 metros a partir do eixo da conduta.

Para estas estruturas, foi definido no Decreto-Lei 10/2018 [29] que estas faixas deveriam cumprir os seguintes critérios:

1. A distância entre copas deve ser no mínimo de 4 m, salvo em casos de presença de pinheiro bravo e eucalipto onde esta deve ser aumentada para 10 m.

2. As copas de árvores com altura de 8m ou menos devem ser alvo de desramações a partir do solo, até metade da sua altura, mantendo-se no mínimo dos 4m para árvores de maiores dimensões.

3. Em relação aos arbustos, estes não podem exceder o tamanho máximo de 50 cm, enquanto que no caso de plantas pertencentes ao estrato subarbustivo, a altura máxima da vegetação será de 20 cm.

4. A quantidade de combustíveis não pode exceder os 2000 m3/ha.

Além destes casos, em comum para ambos os tipos de rede, existem ainda medidas referentes à gestão de combustíveis em edifícios e aglomerados populacionais na rede secundária. Em situações de presença de edificações deve ser criada uma faixa com 50 metros de largura, caso esta faixa abranja territórios florestais ou de 10 metros no caso de abranger territórios agrícolas. Para os aglomerados populacionais, esta largura deverá ser de pelo menos 100 metros.

Para estas estruturas, adicionalmente aos critérios acima referidos, deve ser garantida a distância de 5 metros das copas das árvores para os edifícios.

2.2 Informação Geográfica

A informação geográfica consiste em dados que representam uma localização baseada na superfície terrestre, expressos, geralmente, através de coordenadas [20]. Esta representa uma componente fundamental no sistema de informação desenvolvido no âmbito desta dissertação e, por esse motivo, foi necessário realizar uma pesquisa sobre a modelação e a disponibilização de dados geográficos.

2.2.1 Modelação de Informação Geográfica

Os dados geográficos são, essencialmente, representados através de dois tipos de modelos: modelos raster e modelos vetoriais [25, 32], descritos em 2.2.1.1 e 2.2.1.2. Na figura 2.1 é representada uma comparação visual entre os dois modelos.

2.2.1.1 Modelos Raster

Nos modelos *rasters*, o mundo é representado através duma superfície que é dividida numa matriz de células, denominadas píxeis, que representam uma área da superfície terrestre.

Cada píxel possui, associado a si um valor que representa a característica pretendida dessa região. Estes valores podem ser de dois tipos, nomeadamente, categórico, onde o valor representa uma classe discreta, ou contínuo, representando um valor numérico que pode tomar qualquer valor dentro dum certo limite.

Através dos píxeis é possível representar pontos, onde cada píxel representa um ponto, linhas, representadas como uma sequência de células conectadas, espacialmente, com o mesmo valor, bem como polígonos, que são conjuntos de células contíguas com o mesmo valor que retratam com mais precisão a forma da área. Dessa forma é possível, manter a relação geográfica, com o valor da característica que se pretende representar em cada píxel.

Em relação a estes modelos, é preciso ter em consideração que os *rasters* podem apresentar diferentes níveis de precisão. Esta está diretamente relacionada com a resolução, que representa a área da superfície terrestre que cada píxel cobre. Quanto menor for o tamanho dum píxel, maior será a precisão, e por conseguinte, maior será o detalhe do mapa. No entanto, este não é o único fator a ter em conta, uma vez que quanto menor for o tamanho do píxel, maior será o tempo de processamento, uma vez que o conjunto de píxeis será mais numeroso.

O facto dos modelos *raster* permitirem a captura, armazenamento e análise de dados contínuos são fundamentais na obtenção de dados de satélites [17, 31].

Estes modelos são, geralmente, representados em formatos JPEG, GeoTIFF e PNG.

2.2.1.2 Modelos Vetoriais

Os modelos vetoriais baseiam-se na ideia de que é possível representar o mundo real através da sua divisão em elementos bem definidos. Para tal, cada objeto é representado pela sua geometria, nomeadamente, pontos, linhas ou polígonos.

No caso dos pontos, estes não possuem uma dimensão e são representados por um par de coordenadas (x,y), apresentando, opcionalmente, a coordenada z e têm como objetivo representar a localização de um determinado objeto na Terra.

Em relação às linhas, estas são constituídas por um conjunto de pontos ligados entre si, apresentando dimensão 1, uma vez que permitem ter a noção de comprimento, sendo

úteis na modelação de estradas, bem como na representação de rios.

Por fim, os polígonos são utilizados para representar áreas e incluem-se, portanto na dimensão 2, uma vez que constituem um plano, que permite a medição da área e do perímetro duma região.

Uma vez que os modelos vetoriais permitem a captura de dados discretos, estes tornam-se muito úteis na representação de dados geográficos num sistema de informação geográfico, permitindo, no âmbito desta dissertação, a representação das faixas de gestão de combustível de incêndios [17, 18].

Estes modelos são, normalmente, disponibilizados em formatos Shapefile e KML.

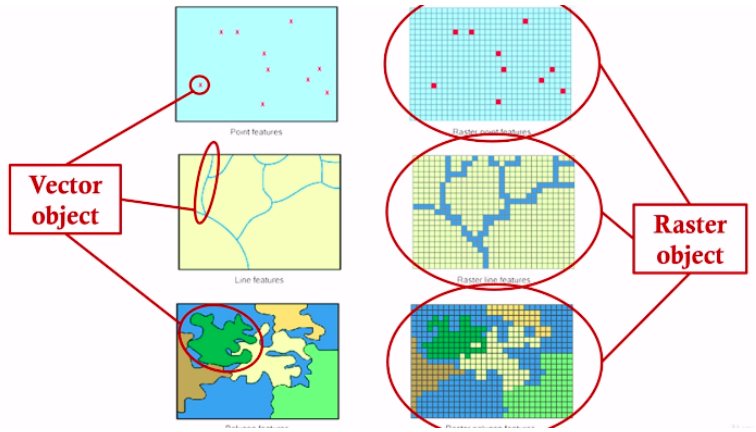


Figura 2.1: Comparação entre o modelo *raster* e o modelo vetorial [57]

2.2.2 Serviços de Disponibilização de Informação Geográfica

A *Open Geospatial Consortium* (OGC) é uma organização internacional criada com o objetivo de facilitar o acesso a informação geográfica, por parte de qualquer pessoa [28]. Para isso, o trabalho da OGC consiste no desenvolvimento de *standards* geoespaciais que pretendem que o acesso à informação geográfica seja *FAIR - Findable, Accessible, Interoperable and Reusable*.

Dessa forma, através da criação destes *standards* é permitida a disponibilização de interfaces abertas e de codificações para todos os serviços e produtos que recorram ao uso de informação geoespacial.

No âmbito desta dissertação foi necessário realizar o estudo de dois tipos de *standards*: *Web Map Service* e *Web Feature Service*.

2.2.2.1 Web Map Service (WMS)

O primeiro *standard* a ser abordado é o *Web Map Service* (WMS) [26] que tem por base a disponibilização de uma interface HTTP simples para solicitar pedidos de informação geográfica. Este serviço permite a produção dinâmica de mapas espacialmente referenciados a partir de informações geográficas. Segundo o WMS, um mapa é considerado um retrato de informação geográfica, disponibilizado num arquivo de imagem digital,

permitindo a sua exibição num ecrã de um computador. No âmbito do WMS, um mapa não é considerado como os dados georreferenciados em si. Ao invés disso, é definido como uma representação dos dados, renderizados em formatos PNG, JPEG ou GeoTIFF, ou ocasionalmente, em elementos gráficos vetoriais em formatos como o SVG ou, ainda, WebCGM.

Este *standard* possui três operações associadas:

- **GetCapabilities:** Operação obrigatória que tem por base a obtenção de serviços de metadata, que consistem na descrição da informação do conteúdo do servidor, bem como, dos valores aceitáveis para os parâmetros de cada pedido. A resposta a cada pedido desta operação é um ficheiro XML de acordo com o *schema* definido, contendo a metadata correspondente.

- **GetMap**: Operação obrigatória que tem por base a devolução de um mapa. Perante um pedido deste tipo, um serviço WMS deve avaliar a validade do pedido. No caso em que o pedido é válido, será retornado um mapa da camada de informação georreferenciada, representada no estilo designado no pedido, respeitando os parâmetros especificados no pedido, nomeadamente: *coordinate reference system*, *bounding box*, *size*, *format* e *transparency*. Se o pedido for inválido, é lançada uma exceção de serviço.

- **GetFeatureInfo**: Operação opcional, que é apenas suportada para camadas que aceitem *queries*. A operação tem como objetivo a disponibilização, para os clientes de um serviço WMS, de mais informação sobre os recursos nas imagens de mapas, devolvidos em pedidos prévios. A resposta a cada pedido é devolvida de acordo com o parâmetro *INFO_FORMAT*, se for um pedido válido. Caso o pedido seja inválido, é lançada uma exceção.

Em suma, um serviço WMS classifica a informação geográfica em camadas, oferecendo um número finito de estilos para as exibir.

2.2.2.2 Web Feature Service (WFS)

O outro *standard* abordado é o *Web Feature Service* (WFS) [27], que permite a troca de dados geográficos pela *Web*. Este serviço define regras para requisitar e recuperar informações geográficas através de protocolos HTTP, em vez de recorrer a protocolos de transferência ao nível do ficheiro. Dessa forma, o WFS fornece acesso direto a informação geográfica ao nível do recurso, permitindo que os clientes apenas recuperem ou modifiquem os dados que são do seu interesse, em vez de lhes providenciarem acesso a um ficheiro com informação excessiva.

Um serviço WFS consiste, então, na disponibilização duma interface que descreve as operações de manipulação de dados em características geográficas.

Este serviço possui onze operações associadas:

- **GetCapabilities:** Disponibiliza um documento de metadata do serviço, descrevendo um serviço WFS facultado pelo servidor.
- **DescribeFeatureType**: Retorna um esquema com uma descrição dos tipos de recursos disponibilizados por um serviço WFS. Esta descrição serve para definir como um WFS espera que cada *feature* seja codificada como *input* e como *output*.
- **GetPropertyValue**: Permite que o valor de uma propriedade duma *feature* seja recuperado, a partir duma *query expression*.
- **GetFeature**: Retorna uma coleção de *features*, que satisfazem uma determinada *query* especificadas pelo cliente no pedido.
- **LockFeature**: Tem como objetivo garantir a consistência dos dados, uma vez que é efetuado um *lock* sob uma *feature*, permitindo a realização de operações críticas, como por exemplo, atualizações.
- **GetFeatureWithLock**: Semelhante à *GetFeature*, mas para além de criar um documento com a coleção de *features*, irá, também, realizar um *lock* nas *features* retornados nesta coleção. Esta ação terá por objetivo garantir a atualização de *features* numa operação *Transaction* que irá ser chamada posteriormente.
- **Transaction**: Operação opcional que é utilizada para descrever a transformação de dados que irá ser aplicada a um conjunto de *features* de um WFS. Esta operação permite a criação, modificação, substituição e deleção de *features* da *data store* de um WFS.
- **CreateStoredQuery**: Permite a criação de uma *query* que irá ser armazenada no servidor.
- **DropStoredQuery**: Permite a remoção de *queries* armazenadas no servidor.
- **ListStoredQueries**: Disponibiliza a lista das *queries* existentes no servidor.
- **DescribeStoredQueries**: Providencia a metadata detalhada sobre cada *query* armazenada no servidor.

No âmbito do projeto estes serviços são utilizados para obter imagens georreferenciadas de algumas fontes de dados enumeradas em 2.3.

2.3 Fontes de Dados de Informação Geográfica

No âmbito deste projeto, foi fundamental a realização duma pesquisa das diversas fontes de dados e documentos disponibilizados pelas entidades competentes, de forma a identificar os vários formatos que o sistema de informação suporta.

2.3.1 Dados Georreferenciados das FGCI

O ICNF é o responsável pela disponibilização dos dados georreferenciados de todas as FGCI de cada concelho do país. O ICNF apenas produz o *shapefile* relativamente à rede primária, disponibilizando-o no seu geocatálogo, enquanto que a rede secundária e terciária são definidas e fornecidas ao ICNF por parte dos municípios, ficando o preenchimento de todos os dados a cargo de cada um.

Estes ficheiros representam modelos vetoriais, onde são retratados os polígonos correspondentes a cada uma destas faixas, com os atributos associados às FGCI em questão. Na tabela 2.1 são apresentados os atributos essenciais na identificação duma faixa.

Tabela 2.1: Atributos associados às FGCI [10]

Nome do Campo	Conteúdo	Tipo
ID_R_FGC	Número Natural que identifica a FGCI	Short Integer; 4
ID_S_FGC	Número Decimal com o algarismo das unidades a corresponder ao ID_R_FGC e o das décimas a um número que identifica a secção da faixa	Text; 12
DATA_ACCAO	Data de levantamento das características do terreno	Date
COD_INE	Código de Referenciação Territorial	Text; 6
DESC_FGC	Número natural que apresenta a descrição das faixas de gestão de combustível	Short Integer
TIPO_FGC	Tipo de faixa (FRC ou FIC)	Text; 3
OBJEC_FUNC	Objetivo/Função da Faixa	Short Integer (I.2)
AREA	Área de cada secção da faixa (ha)	Double; 8; 2
RESP_GC	Responsável pela gestão de combustível	Text; 75
INTER_AAAA	Tipo de intervenção a realizar nas faixas	Text; 3 (I.3)
EXEC_AAAA	Meio de execução das faixas	Short Integer (I.4)
FIN_AAAA	Meio de financiamento para execução das faixas	Short Integer (I.5)
FASE_AAAA	Fase do projeto das faixas	Short Integer (I.6)
OBSERV	Observações relevantes que complementam a informação dos campos anteriores	Text; 254

Nota: Os atributos com AAAA (correspondente ao ano) repetem-se para cada ano em que são atualizados.

2.3.2 Carta de Uso e Ocupação de Solo

A COS é uma carta produzida, inicialmente em 1990, por parte da Direção-Geral do Território (DGT), com o objetivo de representar as alterações que existiram na ocupação do solo entre cada um dos anos em que se verificaram atualizações (1995, 2007, 2010, 2015 e 2018).

Para tal, esta carta baseia-se numa cartografia de polígonos, onde são representadas para áreas com pelo menos 1 hectare, o tipo de vegetação presente no solo. Para isso, foi definida uma nomenclatura a seguir, que conta no momento com 83 classes. Estas classes encontram-se agrupadas por quatro níveis de detalhe, sendo que o primeiro nível conta com 9 classes de ocupação do solo (territórios artificializados, agricultura, pastagens,

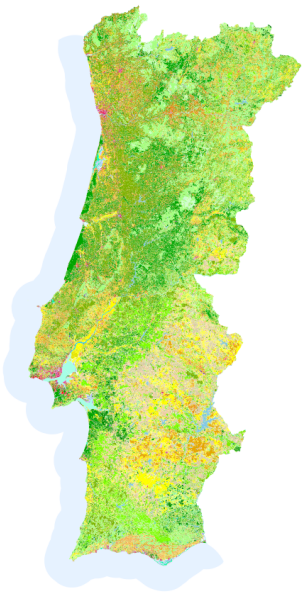


Figura 2.2: Carta de Uso e Ocupação do Solo 2018 [7]

superfícies agroflorestais, florestas, espaços descobertos, matos, zonas húmidas e massas de água superficiais) [53].

Na tabela 2.2 são apresentados os atributos associados ao mais recente ficheiro produzido pela COS, que é divulgado no formato *Shapefile*.

Tabela 2.2: Atributos associados ao COS2018 [53]

Nome do Campo	Conteúdo	Tipo
ID	Identificador Único da Zona	Long Integer
COS2015_n1	Código associado à classe do nível 1	String
COS2015_n4	Código associado à classe do nível 4	String
COS2018_Lg	Descrição das classes	String
AREA	Área de cada zona (ha)	Double

2.3.3 Mapas de Perigosidade e de Risco de Incêndio

Atualmente, existem dois tipos de mapas desenvolvidos para o modelo de risco de incêndio florestal, os mapas de perigosidade e os mapas de risco de incêndio [10].

Os mapas de perigosidade de incêndio dividem-se em dois tipos de cartas: conjuntural e estrutural que são definidos no tópico 2.3.3.1. Estes mapas resultam da combinação da probabilidade de ocorrer um incêndio numa determinada zona (geralmente representada por um píxel no espaço florestal) com a susceptibilidade, referente às condições que o terreno apresenta para a ocorrência de um fenómeno danoso, tais como, a topografia e a ocupação do solo. Assim, estes mapas são adequados para prevenir possíveis focos de incêndio.

Em relação ao mapa de risco, este baseia-se na obtenção da perigosidade nos mapas

acima referidos, combinando-a com as componentes de dano potencial, referentes aos prejuízos ambientais, materiais e pessoais que podem ser causados por incêndios nestes locais. O dano potencial está associado ao produto do seu valor económico, que representa o valor de mercado em euros dos elementos em risco, com a vulnerabilidade, que representa o grau de perda de um elemento, ou seja, face à ocorrência de um fenómeno classifica o quão afetado se espera que o elemento seja. Associados aos mapas de perigosidade, estes mapas apresentam um caráter preventivo, útil no combate aos incêndios, funcionando também para o planeamento de ações de supressão.

Na figura 2.3 são apresentados os tipos de mapa que são possíveis de obter à medida que se vão relacionando mais conceitos.

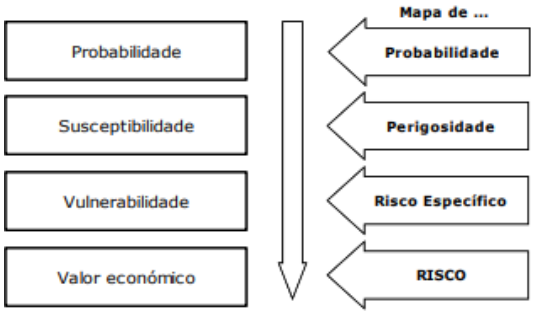


Figura 2.3: Mapas obtidos a partir dos conceitos do modelo de risco [10]

2.3.3.1 Cartas de Perigosidade Conjuntural e Estrutural

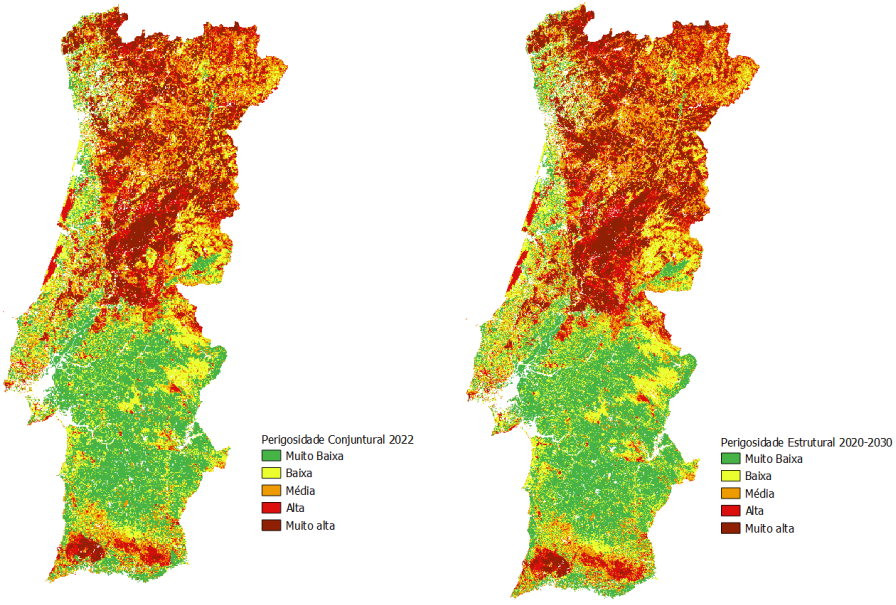


Figura 2.4: Mapas de perigosidade Conjuntural e Estrutural [35] [36]

O ICNF disponibiliza dois tipos de cartas: a carta de perigosidade estrutural e a carta de perigosidade conjuntural, também conhecida por perigosidade anual, ambas em

formatos GeoTIFF ou KML.

Em relação à perigosidade estrutural, esta foi desenvolvida com o objetivo de analisar as estruturas envolventes de forma a facilitar o ordenamento do território e é renovada de 10 em 10 anos. Desse modo, através desta carta é possível calcular um índice de perigosidade tendo em conta a informação estrutural duma zona, como por exemplo o seu declive, bem como o número de incêndios que ocorreram naquele local.

Por outro lado, a carta de perigosidade conjuntural pretende ajudar no planeamento anual dos combates aos incêndios, sendo por esse motivo desenvolvida a cada ano, tendo em conta as características temporais, tais como o tipo de floresta de cada zona [14].

2.3.3.2 Risco Conjuntural e Meteorológico e Fire Weather Index

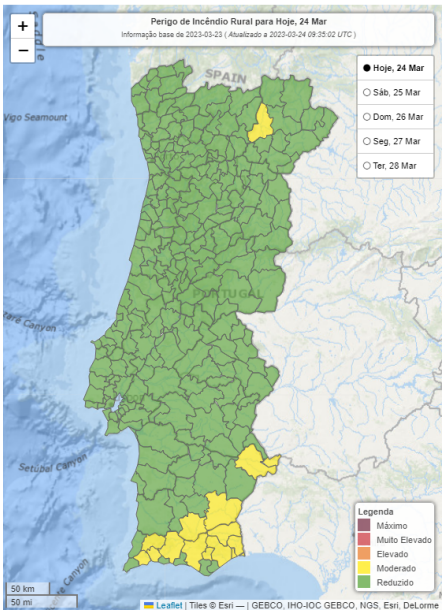


Figura 2.5: Mapa de Risco de Incêndio Rural obtido pelo IPMA [37]

O *Fire Weather Index* (FWI) é um índice utilizado pela maioria dos países do mundo, com o intuito de estimar o risco de incêndio a partir dos combustíveis presentes no solo. Dessa forma, o FWI é calculado, diariamente, a partir de elementos, tais como a temperatura, a humidade, a intensidade do vento e a precipitação do último dia.

Com base nas cartas de Perigosidade disponibilizadas pelo ICNF, bem como ao FWI, o IPMA produz, diariamente, o mapa do Risco Conjuntural Meteorológico. Para isso, o IPMA reclassifica o FWI em valores entre 1 e 5, de acordo com a função da intensidade do potencial incêndio. Posteriormente, os valores das cartas de perigosidade, bem como do FWI são combinados através duma matriz de ponderação de risco, dando origem ao RCM, que estima o valor do risco de incêndio florestal [38].

2.3.4 Territórios Ardidos

A informação sobre áreas ardidas apresentam um grande relevo, pois servem de base ao planeamento do sistema de combate aos incêndios. Este tipo de cartografia permite estipular as cartas de perigosidade estrutural, bem como identificar possíveis locais de faixas que foram intervencionadas de forma natural. Para este tipo de ficheiros o ICNF disponibiliza, anualmente, um *shapefile* com os territórios consumidos pelas chamas [16].

Tabela 2.3: Atributos associados aos Territórios Ardidos obtidos a partir do QGIS

Nome do Campo	Conteúdo	Tipo
cod_sgif	Identificador do código SGIF	String; 254
cod_ncco	Identificador do código NCCO	Real; 32; 15
data_inicio	Data do início do incêndio	String; 254
data_fim	Data do fim do incêndio	String; 254
tipo	Tipo de incêndio	String; 254
tipo_causa	Causa do incêndio	String; 254
Ano	Ano em que a área ardeu (ha)	Integer; 9
area_ha	Área ardida (ha)	Real; 32; 15

2.3.5 Inventário Florestal Nacional

O Inventário Florestal Nacional (IFN) é um documento disponibilizado pelo ICNF, em formato *shapefile*, que tem como objetivo fazer uma recolha de informação sobre as florestas portuguesas, avaliando a condição, estado e abundância dos seus recursos. O método de elaboração deste documento baseia-se numa recolha de amostras *in situ* realizada em períodos de 10 anos. Através deste documento é possível elaborar uma série temporal dos recursos florestais desde 1965 [13].

2.3.6 Digital Elevation Model

Os *Digital Elevation Model* (DEM) baseiam-se em representações digitais de superfícies topográficas sob a forma de pontos georreferenciados e são, geralmente, utilizados para representar o relevo de terrenos na Terra ou de qualquer outro astro [24]. Estes têm o principal objetivo de descrever a elevação do solo, privado de vegetação, estruturas ou quaisquer outros objetos situados em cima deste.

Os DEM são modelos baseados em píxeis (modelos raster) que apresentam duas principais funções. Estes permitem, que a superfície da terra seja dividida em píxeis onde a cada um é associado um valor de elevação. Além disso, são utilizados para modelação e análise da topografia tridimensional.

O termo DEM é muitas vezes associado a um termo genérico que pode ser dividido em dois tipos, particularmente em *Digital Terrain Model* (DTM) e *Digital Surface Model* (DSM). Um DTM apresenta uma definição semelhante à de um DEM, onde o terreno representa o solo, excluindo qualquer outro tipo de superfície, sendo associado um valor de elevação

para cada píxel. No caso dos DSM, estes representam a superfície como a zona mais alta para uma área (a zona que é iluminada pela radiação), não excluindo outros tipos de objetos. Desta forma, a representação da elevação do solo só é possível quando não existe nenhum objeto acima deste.

Na figura 2.6 é apresentada a comparação visual entre o DTM e o DSM.

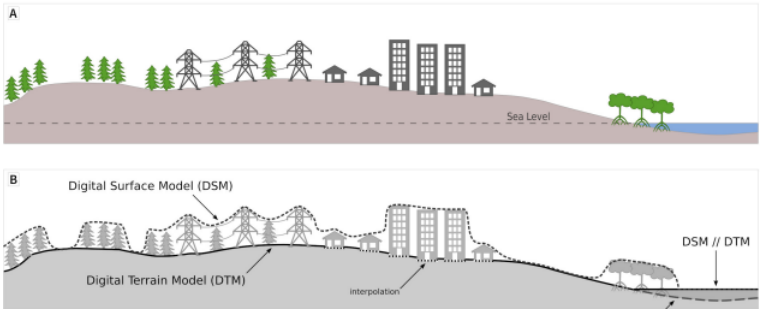


Figura 2.6: Comparação entre o DTM e o DSM [24]

2.3.7 Ortofotos

As ortofotos baseiam-se em fotografias tiradas com câmaras digitais em voos aerofotogramétricos. A disponibilização deste tipo de fotos é feita por parte da Direção-Geral do Território, através de serviços WMS, apresentando diversos tipos de resoluções espaciais: 25cm, 50cm e 1m, com quatro bandas espetrais. [54].

2.3.8 Dados Georreferenciados recolhidos por utilizadores

No âmbito do projeto Floresta Limpa, foi desenvolvida uma aplicação móvel onde os utilizadores têm a possibilidade de fornecer vídeos, imagens, coordenadas, avaliações sobre as coordenadas do terreno juntamente com imagens e informação textual ou recolhida por voz sobre as faixas. Dessa forma, o sistema de informação está pronto a responder a este aspeto, suportando os respetivos formatos, bem como disponibilizando *endpoints* que são especificados numa fase posterior desta dissertação.

2.4 Dados de satélite

O foco do projeto onde esta dissertação está inserida é a monitorização das FGCI, a partir da deteção remota. Para tal, são utilizados os satélites como principal fonte de dados para as observações realizadas nesta área.

De facto, uma série de programas espaciais disponibilizam os dados dos seus satélites de forma aberta, como é o caso dos dos programas *Landsat*, *Copernicus*, bem como dos satélites *MODIS*. No entanto, estes dados não são disponibilizados aos utilizadores de forma direta, mas sim, através de vários níveis de processamento que variam para satélites

distintos, de forma a permitir a extração de novas informações e a correção de erros das imagens originais.

A realização do pré-processamento de imagens é assim necessária, de forma, a que os erros que ocorrem nas mesmas sejam corrigidos. Dentro destes destacam-se erros associados a interferências atmosféricas, tais como o aparecimento de nuvens que podem cobrir na totalidade uma imagem, bem como erros de georreferenciação, onde o mesmo ponto representa duas localizações distintas em duas imagens.

Após a realização do pré-processamento das imagens, a partir dos dados é possível calcular índices espectrais como o NDVI, o NDWI, o SAVI, entre outros, que permitem a estimação de características da vegetação. Desta forma, é possível obter informação sobre o estado de limpeza das FGCI, olhando para as características da vegetação ao longo do tempo.

Visto que a maior parte dos trabalhos realizados no âmbito deste projeto se baseiam nas imagens fornecidas pelos Sentinel 1 e 2, estes são abordados com alguma profundidade, sendo realizada apenas uma breve introdução aos satélites Landsat 8 e MODIS que também poderão ser úteis em trabalhos posteriores.

2.4.1 Sentinel-1

Sentinel-1 é uma missão do programa *Copernicus* constituído por dois tipos de satélites, os Sentinel-1A e os 1B, com o intuito de obter informações sobre o ambiente e a segurança do planeta Terra.

A órbita realizada pelos satélites desta missão é praticamente polar, com um ciclo de repetição a cada 12 dias. O plano de órbita dos dois Sentinel-1 é semelhante, separados apenas por 180º, garantindo um ciclo de repetição de 6 dias, caso ambos se encontrem a funcionar.

Além disso, estes satélites destacam-se por apresentarem um radar denominado *Synthetic Aperture Radar* (SAR) que é capaz de atuar em comprimentos de onda capazes de penetrar as nuvens, permitindo obter imagens mesmo em situações atmosféricas adversas e quer seja de dia ou noite.

Os satélites Sentinel-1 incluem imagens de banda C que são disponibilizadas em quatros modos de operação, destacando-se como principal, o modo *Interferometric Wide swath* (IW), que realiza uma recolha de dados com uma resolução espacial de 5 m por 20 m. Este tipo de modo disponibiliza dados com polarizações simples, VV e HH ou dupla, nomeadamente, VV+VH e HH+HV. No âmbito do projeto, o modo de operação utilizado é este, uma vez que se apresenta como um modo primário sem conflito, através da polarização VV+VH, realizada sobre a terra, enquanto os outros modos de operação dedicam-se sobretudo à monitorização sobre o mar.

Tal como foi referido, previamente, os dados dos satélites são disponibilizados em diferentes níveis de processamento. No caso dos Sentinel-1 são oferecidos 3 níveis. O nível 0 que fornece dados brutos SAR comprimidos e não focados, que servem como base da

obtenção de produtos de maior nível de processamento. Relativamente ao nível 1, que é o nível mais acedido pelos utilizadores, este disponibiliza dados com imagens focadas a partir do nível 0 que podem ser processadas em produtos *Single Look Complex* (SLC), que consistem em dados processados no espaço natural dos píxeis, ou *Ground Range Detected* (GRD), que permite a obtenção de imagens de maior resolução. Quanto ao nível 2, fornece produtos derivados a partir do nível 1, obtendo informações sobre o vento, as ondas dos oceanos e as correntes.

2.4.2 Sentinel-2

Sentinel-2 é uma missão desenvolvida no âmbito do programa *Copernicus*, composta por satélites com sensores multi-espectrais, o Sentinel-2A e 2B.

Estes satélites realizam uma órbita idêntica, síncrona com o sol, permitindo que o ângulo de incidência da radiação solar seja mantido, minimizando o potencial impacto das sombras e do nível de iluminação no solo. As órbitas dos Sentinel-2 apresentam um ciclo de repetição de 10 dias e encontram-se separadas por 180 graus, reduzindo o tempo do ciclo em metade caso ambos os satélites se encontrem a funcionar.

Os sensores multi-espectrais presentes nos Sentinel-2 captam informação em 13 bandas, que são divididas em 2 níveis: bandas *Visible and Near-Infra-Red* (VNIR) e bandas *Short Wave Infra-Red* (SWIR), que apresentam resoluções espaciais de 10 metros, 20 metros ou 60 metros.

Apesar de possuírem três níveis de processamento distintos, com sub-níveis associados apenas são fornecidos aos utilizadores dois destes, nomeadamente produtos do nível 1C e do nível 2A. Relativamente aos produtos do nível 1C, estes apresentam correções radiométricas e geométricas, permitindo a disponibilização de imagens com refletância no topo da atmosfera. Quanto aos produtos 2A, estes apresentam uma correção atmosférica aplicada aos produtos 1C, permitindo a obtenção de imagens com refletância na base da atmosfera.

Quer o Sentinel 1 quer o 2 disponibilizam um ficheiro que contém a informação do produto em formato XML, bem como subpastas com *measurement datasets*, disponibilizando imagens em vários formatos binários, tais como GeoTIFF (Sentinel 1 - nível 1C), netCDF (Sentinel 1 - nível 2A) e JPEG2000 e GML - JPEG2000 (Sentinel 2). Também apresentam algumas pré-visualizações em HTML, PNG, KML, bem como a metadata do produto em XML e a especificação dos seus esquemas XML em XSD. Na figura 2.7 são apresentados os ficheiros e os formatos disponibilizados pelo Sentinel-1 e 2 que o sistema de informação é capaz de suportar.

2.4.3 Landsat 8 e 9

O Landsat 8 e o 9 são os satélites mais recentes do programa *Landsat* desenvolvido por parte da NASA, em conjunto com a USGS. O *Landsat 8* destaca-se por apresentar dois sensores, *Operational Land Imager* (OLI) e *Thermal Infrared Sensor* (TIRS), que fornecem

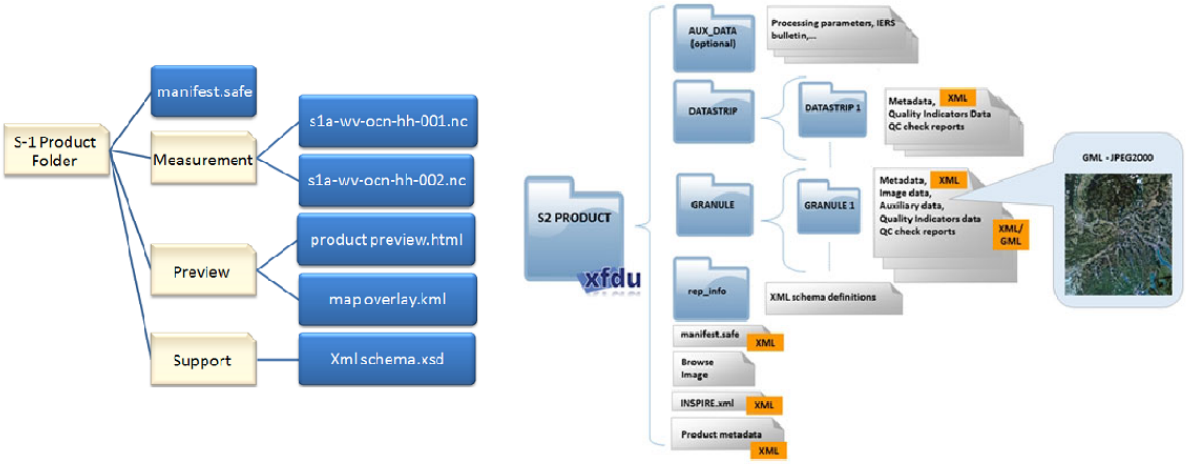


Figura 2.7: Ficheiros de dados disponibilizados pelo Sentinel 1 e 2

uma cobertura da massa terrestre através de 11 bandas com uma resolução espacial de 30 metros na zona do visível, NIR e SWIR, 100 metros para o térmico (TIR) e 15 metros para o pancromático, que são adequadas para a análise das características da vegetação [30].

Mais recentemente foi lançado o *Landsat 9* que é uma cópia aprimorada do 8, apresentando os sensores TIRS-2 e OLI-2 para as mesmas 11 bandas. Este satélite foi colocado com uma distância de 8 dias, relativamente, à posição do *Landsat 8*, de forma a reduzir o ciclo de repetição para metade (8 dias). Os dados de ambos os satélites são disponibilizados em ficheiros de formato GeoTIFF.

2.4.4 MODIS

O MODIS consiste num instrumento científico criado pela NASA, lançado nos satélites Aqua e Terra. Os satélites com este instrumento realizam uma observação da superfície terrestre a cada 1 ou 2 dias, obtendo imagens com alta sensibilidade radiométrica em 36 bandas espectrais. Destas 36 bandas, 2 têm uma resolução de 250m, 5 apresentam 500m, enquanto as restantes 29 estão limitadas a 1km [40].

O facto destes satélites apresentarem uma resolução temporal muito elevada, permite a sua combinação com imagens com alta resolução espacial, como é o caso dos Sentinel 2, de forma a obter melhores resultados nos diversos parâmetros. Os dados do satélite Modis são fornecidos no formato HDF.

2.5 Tecnologias Relevantes

No âmbito desta dissertação ficou definido que o sistema de informação iria ser desenvolvido com recurso à *Google Cloud Platform*. Dessa forma, neste tópico são abordados os serviços escolhidos para a arquitetura do sistema, que é apresentada em 3.3. Adicionalmente, é realizada a introdução a algumas ferramentas e bibliotecas úteis na manipulação

de dados de informação geográfica, importantes para o desenvolvimento do projeto.

2.5.1 QGIS

O QGIS [46] é uma aplicação *open-source* de um sistema de informação geográfica de *Desktop* que permite gerir dados georreferenciados.

De facto, o QGIS é uma ferramenta bastante útil para os projetos que tenham por base dados geográficos, fornecendo aos utilizadores ferramentas para visualizar, analisar e editar a informação geográfica. O QGIS suporta os dois tipos de modelos definidos em 2.2.1.1 e 2.2.1.2: modelos *raster* e modelos vetoriais, agrupando-os por *layers*, permitindo a criação de novos mapas.

Além disso, o QGIS suporta vários tipos de ficheiros, nomeadamente, *shapefiles*, *coverages*, *dxf*, *MapInfo*, interface *PostGIS*, entre outros. Também é possível fornecer dados georreferenciados através dos serviços de disponibilização (WFS e WMS). O QGIS pode, também, atuar como um serviço de WMS e WFS, funcionando como uma fonte de dados georreferenciados.

Outra funcionalidade do QGIS é a disponibilização de plugins de forma a estender as funcionalidades da sua aplicação. Desse modo é possível criar novos plugins escritos em *Python* ou em C++, com o intuito de desenvolver novas ferramentas para o QGIS. Por outro lado, o QGIS permite, ainda, a integração de outras aplicações *open-source* para estender as suas funcionalidades.

No âmbito deste projeto, o QGIS é uma ferramenta essencial, visto que a maior parte dos dados são recebidos em formatos suportados por este e, dessa forma, permite a análise e manipulação dos dados, de acordo com as pretensões do projeto.

2.5.2 GDAL

A *Geospatial Data Abstraction Library* (GDAL) [19] é uma biblioteca dedicada à manipulação de dados geoespaciais, quer sejam *raster* ou vetoriais, lançada pela *Open Source Geospatial Foundation*. O seu desenvolvimento foi realizado através da escrita em C++, mas no entanto, apresenta uma API para o seu uso em *Python*, bem como em *Java*.

Para o projeto, esta ferramenta é útil uma vez que disponibiliza um conjunto de operações, tais como, obtenção de informação sobre um conjunto de dados, conversões entre ficheiros geoespaciais de formatos distintos, bem como o recorte de um conjunto de dados para serem aplicados noutro conjunto.

2.5.3 Google Cloud Platform

A *Google Cloud Platform* (GCP) representa um conjunto de serviços de computação em *cloud* disponibilizados pela Google, tais como armazenamento de dados, análise de dados e serviços de aprendizagem automática. Para tal a Google fornece ambientes do tipo *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *serverless*. [23].

De forma a disponibilizar estes ambientes, a GCP fornece serviços de computação como a *Google Compute Engine* (GCE) que é uma IaaS que permite o uso de máquinas virtuais (VMs). Através das VMs, a Google cede parte do seu processamento, armazenamento e tráfego da rede a um cliente, dando-lhe a possibilidade de desenvolver aplicações com base no hardware físico da Google. Neste tipo de serviço, o cliente fica responsável pela configuração do ambiente, bem como da segurança da VM.

Por outro lado, fornece um serviço denominado *Google App Engine* (GAE) que consiste numa PaaS que permite a criação e desenvolvimento de aplicações sem que o utilizador tenha de se preocupar com o software onde a aplicação é lançada, bem como com toda a segurança e manutenção do sistema operacional e hardware associados. Neste serviço podem ser desenvolvidas aplicações de diversos ambientes de linguagem, nomeadamente, Go, Java, Python, PHP, Node.js, .NET e Ruby.

A GCP faculta um número vasto de zonas de disponibilidade, bem como de regiões onde as aplicações podem ser desenvolvidas, garantindo todas as propriedades de consistência, segurança e replicação.

O desenvolvimento do sistema de informação poderia ter sido realizado de forma remota, no entanto, no projeto Floresta Limpa, optou-se por apenas realizar a atividade de aprendizagem automática em *clusters* privados, ficando o restante serviço de informação desenvolvido na GCP. A decisão tem por base o facto dos servidores de *cloud* fornecerem inúmeras garantias, nomeadamente:

- **Disponibilidade**: Uma vez que possui um grande número de zonas de disponibilidade e regiões.

- **Custo Eficiente**: Apenas é necessário pagar pelos serviços que estão a ser utilizados, evitando custos desnecessários para as organizações.

- **Escalabilidade e Elasticidade**: Os sistemas de *cloud* providenciam a uma aplicação os requisitos adequados num dado momento, permitindo realizar técnicas de *scale up*, que se baseiam no aumento da quantidade de CPUs duma máquina e *scale out*, que consiste na disponibilização de mais componentes com requisitos semelhantes de forma a permitir uma melhor distribuição da carga.

- **Sistemas Operativos**: A existência de imagens de sistemas de operativos, com todas as bibliotecas necessárias para o desenvolvimento de aplicações.

- **Facilidade no desenvolvimento de aplicações**: Através da disponibilização dos serviços da *cloud*, os programadores têm a opção de apenas se preocupar com o desenvolvimento da aplicação, não sendo necessário ter em conta o ambiente onde esta é inserida.

2.5.4 Google App Engine

Como foi referido em 2.5.3, a plataforma *Google App Engine* (GAE) representa uma PaaS disponibilizada pela Google que permite o desenvolvimento e alocação de aplicações, onde a manutenção do software é realizada por parte da Google. Cada aplicação desenvolvida na GAE baseia-se num *container* que possui um conjunto de serviços.

Estes serviços representam os componentes lógicos duma aplicação, apresentando várias versões que mostram as alterações que lhe são feitas. Cada um destes serviços corre por fim em diversas instâncias de acordo com as necessidades de tráfego exigidas pela aplicação [5]. Na figura 2.8 é apresentada a da estrutura duma aplicação desenvolvida no GAE.

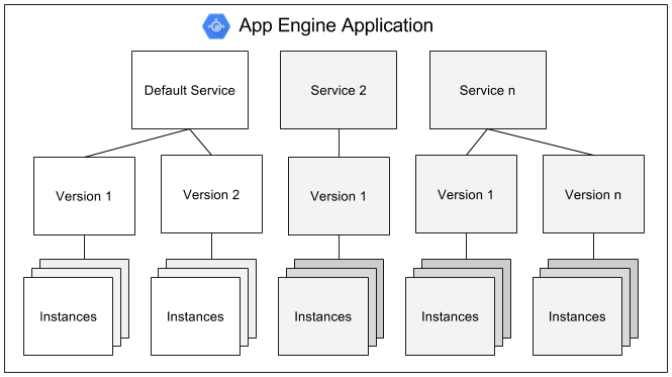


Figura 2.8: Estrutura genérica duma aplicação do GAE [52]

Assim sendo, os serviços consistem nas funcionalidades que a aplicação disponibiliza e que lida com diversos tipos de pedidos por partes do utilizador. O facto destes serviços estarem divididos por versões permite que seja fácil recuperar versões anteriores em caso de erros ou de testes. Além disso, é possível existir uma migração de carga entre versões, permitindo que quando uma nova versão seja lançada, as versões anteriores continuem a correr e os pedidos sejam transferidos progressivamente para a nova versão. Dessa forma, evita-se uma perda de ligação à aplicação, quando ocorrem atualizações.

Ademais, a GAE permite o lançamento de várias instâncias, que permitem que esta escale automaticamente os recursos da aplicação de forma a responder à carga de trabalho exigida. Assim sendo, torna-se possível o desenvolvimento de mais instâncias em caso de sobrecarga, bem como, a diminuição de instâncias quando estas não se mostrem necessárias, permitindo assim, que os custos sejam apenas relativos ao que a aplicação está realmente a consumir.

O facto de ser uma plataforma paga implica que existam limites diferentes consoante o plano utilizado. Dessa forma para uma aplicação que siga o plano grátis, apenas podem ser utilizados no máximo 5 serviços, 15 versões e 20 instâncias. No caso do plano pago, podem ser utilizados cerca de 105 serviços, 210 versões e 25 instâncias, que podem ser 200, no caso em que o servidor se situa na região *us-central*.

2.5.5 Google Cloud Storage

A *Google Cloud Storage* é um serviço disponibilizado pela *Google Cloud* que permite o armazenamento de dados imutáveis, em qualquer tipo de formato, denominados por objetos [56]. O facto destes objetos serem imutáveis significa que não podem ser realizadas alterações a estes ficheiros. No entanto, os ficheiros podem ser substituídos de forma atómica, ou seja, até que o novo ficheiro seja carregado é mostrado o ficheiro antigo, procedendo-se à sua substituição assim que o carregamento esteja concluído.

Na *Cloud Storage* existe ainda a noção de *buckets* que representam contentores, onde os objetos são armazenados. Estes têm o objetivo de organizar e controlar o acesso aos dados, de forma a definir uma localização geográfica, onde estes são armazenados.

Outra característica da *Google Cloud Storage* é que permite a georreplicação dos dados por diferentes regiões. Dessa forma, os dados podem ser acedidos em diversas regiões com menores taxas de latência, uma vez que podem estar armazenados em *Content Delivery Networks* (CDN), que são acedidas consoante a localização do utilizador, isto é, o pedido do utilizador é reencaminhado para uma CDN perto da sua localização, permitindo uma resposta mais rápida do que no caso em que o pedido fosse direcionado para o servidor. Dessa forma é possível garantir uma maior disponibilidade dos dados, uma vez que em situações de falhas, estes estão armazenados em máquinas de outras regiões ou até mesmo em diferentes zonas de disponibilidade dentro duma região.

A *Google Cloud Storage* apresenta, ainda, diversos tipos de classes de armazenamento consoante as características dos dados que pretendemos utilizar, nomeadamente, *Standard Storage*, *Nearline Storage*, *Coldline Storage* e *Archive Storage*, com diferentes custos e performances associadas. Estes tipos de armazenamento estão enumerados de acordo com a frequência de acesso aos dados, sendo maior no caso da *Standard Storage* e menor para o caso da *Archive Storage*. A cada um destes tipos está associado um custo de armazenamento, bem como um custo de acesso aos dados. Quando os dados são frequentemente acedidos, os custos de armazenamento são superiores e os de acesso são inferiores, alterando de forma inversa à medida que se tratam de dados com menores frequências de acesso.

Na figura 2.9 são apresentados estes tipos de classes de armazenamento com uma explicação associada a cada uma.

No âmbito do projeto, este serviço pode ser utilizado para armazenar imagens de satélites, imagens e vídeos disponibilizados pelos utilizadores, entre outros ficheiros estáticos. Neste caso particular, podem ser criados dois *buckets* de forma a efetuar uma separação entre ficheiros que representem imagens e ficheiros que sejam vídeos.

2.5.6 Google Cloud Datastore

A *Google Cloud Datastore* é um serviço disponibilizado pela Google que se baseia numa base de dados de documentos NoSQL com replicação automática, que facilita o desenvolvimento de aplicações. Este tipo de armazenamento divide-se em dois mecanismos que são implementados sobre o *Firebase*: *Native Mode* e *Datastore mode* [8].

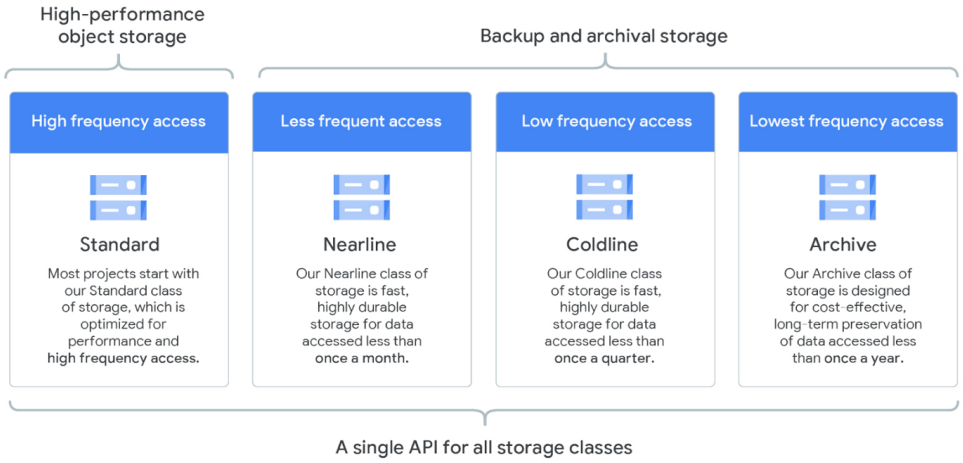


Figura 2.9: Tipos de classes de armazenamento disponibilizadas pela GCS [45]

Relativamente ao *Datastore Mode*, este mecanismo representa um comportamento semelhante ao de uma *Datastore*, apresentando entidades organizadas em tipos e grupos de entidade. Além disso, pode aceder a funcionalidades disponibilizadas pelo *Firestore*, removendo portanto algumas limitações que as *Datastores* apresentam, por exemplo, o facto das *queries* passarem a utilizar consistência forte, em vez da eventual. O recurso a este modo é aconselhado pela Google para novos projetos de servidor, ou seja para aplicações integradas com o *App Engine* de forma a existir uma interligação entre este e a aplicação.

Por outro lado, o *Native Mode* consiste numa base de dados organizada em documentos e coleções (JSON). O *Firestore* apresenta uma relação de compatibilidade com as *Datastores* e introduz ainda, para além de consistência forte, novas funcionalidades, tais como bibliotecas para clientes web e móveis que não são suportadas pela *Datastore*. Desta forma, a Google recomenda a utilização deste modo para o desenvolvimento de aplicações web e móveis, visto que permite o acesso a um conjunto vasto de funcionalidades em tempo real, bem como em *offline*.

No âmbito do projeto, este tipo de serviço é utilizado para armazenar informação dos utilizadores, disponibilizadas através da aplicação móvel, no modo *Firestore*.

2.5.7 Google Cloud SQL

A *Google Cloud SQL* é um mecanismo de armazenamento da Google que tem como objetivo a disponibilização de bases de dados SQL localizadas em *cloud*. Este é um serviço que disponibiliza uma série de funcionalidades que são geridas automaticamente pela *Cloud SQL*, tais como, *backups*, alta disponibilidade e tratamento de falhas, monitorização, *logging*, entre outros. As bases de dados são criadas dentro dum disco persistente, que se encontra conectado à VM, que possui um endereço IP estático a partir do qual será possível à aplicação, desenvolvida no *App Engine*, aceder aos recursos desta base de dados [55].

Para tal a *Cloud SQL* disponibiliza instâncias que correm numa máquina virtual alocada nos servidores da Google. Estas máquinas virtuais disponibilizam serviços de bases de dados, tais como *MySQL*, *PostgreSQL* ou *SQL server*, sem que seja necessário a sua configuração por parte do programador. Estas instâncias serão, posteriormente, replicadas por diversas zonas e regiões, de forma a permitir um elevado nível de disponibilidade, bem como a distribuição da carga pelas várias instâncias. Para isso, a *Google Cloud SQL* cria instâncias de leitura na mesma zona ou noutra distinta, que são réplicas da instância primária. No entanto, isto permite uma divisão de pedidos, onde os pedidos de leitura são dirigidos para estas réplicas, enquanto que os pedidos de atualização são direcionados para a instância primária. Uma vez que as atualizações feitas na instância primária são reproduzidas, praticamente, em tempo real nas instâncias de leitura, verifica-se uma diminuição de tráfego na instância primária. Além disso, para cada região, é criada também uma instância *standby* numa zona diferente da zona da instância primária, que tem como objetivo a prevenção de falha nessa instância. Assim, quando esta instância falha, os pedidos de atualizações são redirecionados para esta instância que deixa de estar em *standby*. Na figura 2.10 é representado um esquema genérico da distribuição destas instâncias.

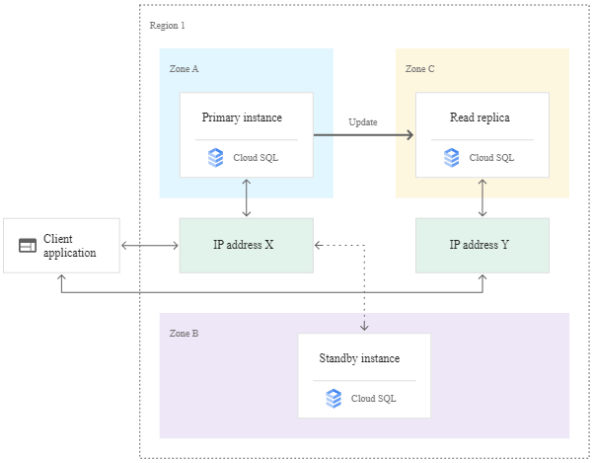


Figura 2.10: Configuração genérica da *Google Cloud SQL* numa região [1]

Este tipo de serviço será útil para o projeto, uma vez que será possível armazenar informações acerca das faixas de gestão de combustível, tais como os seus *shapefiles* e classificações associadas, através duma base de dados *PostgreSQL*, que apresenta uma extensão de forma a suportar dados georreferenciados como é explicado em 2.5.8 e 2.5.9. Para além disso, o facto de apresentar mecanismos automáticos de replicação facilita o seu uso, uma vez que estes são tratados pela *Google Cloud*.

2.5.8 PostgreSQL

O *PostgreSQL* [43] é um sistema de gestão de base de dados relacional *open-source* que se baseia no uso da linguagem SQL. Atualmente, esta é uma das base de dados

mais conceituada do mundo, visto que fornece uma arquitetura confiável com várias ferramentas para a elaboração de novas aplicações.

De facto, este sistema dá inúmeras garantias aos programadores devido aos recursos que apresenta. Trata-se dum sistema que permite o uso de inúmeros tipos de dados, dos quais se destacam os primitivos, estruturados, documentos, geométricos e os tipos compostos. Permite, ainda, a criação de novos tipos de dados definidos pelos utilizadores, como por exemplo dados georreferenciados obtidos a partir da extensão *PostGIS* que será referida posteriormente.

Além disso, garante a integridade dos dados, através da inclusão de funcionalidades como as chaves primárias, estrangeiras e sistemas de *locks*.

Também permite a criação de ambientes tolerantes a falhas, visto que apresenta um sistema de replicação de dados, que lhes fornece uma maior segurança e disponibilidade. Para tal, o *PostgreSQL* baseia-se em *Write-ahead logs* (WAL), que são enviados para as réplicas de forma assíncrona, permitindo assim, a capacidade de realização de leituras nas réplicas.

Ademais, apresenta ainda mecanismos, destinados a garantir a consistência dos dados, tais como, o uso de transações, com a possibilidade de escolha de qualquer nível de isolamento definidos nos standards do SQL, bem como um controlo de multi-versões, dando a cada transação um *snapshot* do estado da base de dados, permitindo que alterações dentro duma transação não afetem outras e, ainda, recursos de indexação, tais como a *B-tree* ou os índices de *Hash Table*.

Por outro lado, é um sistema que apresenta robustez, uma vez que está pronto a lidar com bases de dados de quaisquer dimensões, o que se verifica útil para este projeto, visto que é necessário lidar com enormes quantidades de dados de diferentes tipos.

Apresenta, também, características extensíveis, podendo ser adaptado, consoante os avanços tecnológicos e as suas necessidades.

No âmbito deste projeto, este sistema apresenta, também uma funcionalidade útil, nomeadamente efetuando notificações assíncronas que se baseiam num sistema de mensagens assíncronas que são lançadas através de cada sessão. Dessa forma, é possível a criação de eventos que podem ser utilizados para, por exemplo, notificar os utilizadores da ocorrência dum novo incêndio.

Na área da informação geográfica, o *PostgreSQL* acaba por ser uma base de dados de eleição, visto que possui uma extensão, chamada *PostGIS*, que será descrita em 2.5.9, que permite a representação de dados georreferenciados.

2.5.9 PostGIS

O *PostGIS* [42] é uma extensão geoespacial referente à base de dados relacional *PostgreSQL*, que surge com o objetivo de satisfazer as necessidades dos sistemas de informação geográficas, no processamento de dados complexos e massivos. Através desta

extensão torna-se possível a inclusão de dados de informação geográfica na base de dados, representados tanto em 2D como em 3D.

De facto, esta extensão permite a adição de novos tipos de dados, nomeadamente, dados geométricos, geográficos, *rasters*, entre outros. Assim, o *PostGIS* tem como objetivo o armazenamento dos dados espaciais, bem como a sua gestão, permitindo a realização de *queries* de localização em linguagem SQL, pois segue os *standards* de SQL definidos pela OGC.

Dessa forma, o *PostGIS* fornece um conjunto de características tais como a disponibilização de todo o tipo de dados geoespaciais, isto é, a representação de geometria através de tipos que são pontos, multi-pontos, linhas, multi-linhas, polígonos e multi-polígonos.

Além disso, são disponibilizadas funções analíticas e de processamento para dados vetoriais e de *raster*, bem como, operadores espaciais de forma a permitir a medição de elementos geoespaciais, tais como área, perímetro e distância entre dois locais e, também, operadores de união, diferença e simetria entre os objetos geoespaciais [58].

Ademais, apresenta suporte SQL para vários formatos de dados vetoriais, tais como KML, GML, GeoJSON, GeoHash e WKT, para formatos de dados raster, nomeadamente, GeoTIFF, PNG, e JPG.

Outra vantagem do *PostGIS* consiste no facto dos seus dados poderem ser incluídos em ferramentas de visualização e edição de dados geográficos tais como o QGIS, referido em 2.5.1 e apresenta ainda suporte para serviços externos, como WFS e WMS.

2.5.10 Google Earth Engine

O *Google Earth Engine* (GEE) é uma plataforma em *cloud* disponibilizada pela Google para análise e visualização de grandes conjuntos de dados geoespaciais, permitindo o seu processamento através dos algoritmos de classificação presentes nas suas bibliotecas, bem como, através da integração de algoritmos desenvolvidos pelos utilizadores [22].

De facto, o GEE providencia um vasto catálogo de dados, com imagens de satélite atualizadas diariamente, imagens históricas da Terra, previsões meteorológicas e climáticas, bem como um conjunto de dados topográficos, armazenando-as num repositório público que pode ser acedido por qualquer utilizador. Estes dados são alvo de pré-processamento, preservando a sua informação, para estarem prontos a serem acedidos para análise dos utilizadores. Neste catálogo destacam-se produtos dos satélites *Landsat* 4, 5, 7 e 8, *MODIS* e também imagens do *Sentinel* 1 e 2 [39]. Para além destes dados, o GEE ainda permite que os utilizadores insiram conjuntos de dados em formato raster ou vetorial, para poderem trabalhar sobre estes através duma interface REST.

O GEE pode ser acedido de duas formas, nomeadamente, através de uma interface (API) Javascript ou Python e através dum ambiente de desenvolvimento interativo (IDE), disponibilizando um conjunto de ferramentas, como por exemplo, processamento de imagem através da manipulação de bandas disponibilizadas pelos satélites, algoritmos de classificação com *Support Vector Machines* ou o *Random Forests*, análises de séries temporais,

extrações de estatísticas de imagens, entre outras. Além disso, permite ainda a exportação de gráficos, imagens e tabelas dos dados que podem ser utilizadas noutras ferramentas, como por exemplo no QGIS, de forma a serem processados manualmente. Dessa forma, o GEE permite a análise de várias características da Terra, tais como a cobertura florestal ou hídrica, a alteração do uso do solo e avaliação do estado da vegetação.

A ferramenta do GEE apresenta o benefício de poder ser executada em paralelo com a GAE, de forma a permitir a criação de aplicações geoespaciais escaláveis. Para isso, o código da GAE inclui a biblioteca python/javascript do GEE, de forma a ser possível a realização de pedidos ao GEE, sem que um utilizador esteja especificamente registado no GEE. Por outro lado, existe a possibilidade de um utilizador realizar a autenticação de cliente no GEE, o que beneficiaria a aplicação, uma vez que os limites de recursos estariam associados à conta do utilizador em vez de estarem relacionados com a aplicação do GAE.

Visto que no contexto do projeto, necessitamos da análise da maior parte dos produtos de satélites disponibilizados pelo GEE e este pode ser integrado com o GAE, esta ferramenta é fundamental, uma vez que permite o acesso a um conjunto de dados de grandes dimensões sem a necessidade de fazer a sua transferência localmente.

2.5.11 Node.js

No âmbito da implementação da API REST e da lógica do servidor foi escolhido o Node.js com recurso à *framework* Express.js, bem como à ORM Sequelize [48].

A escolha sobre o Node.js ocorre visto que permite o desenvolvimento de aplicações de forma rápida e eficaz, graças ao conjunto de *frameworks* que fornece através do *Node Package Manager* (npm).

Dessa forma, foi identificada a *framework* Express.js uma vez que esta disponibiliza operações essenciais no suporte à criação duma API REST, tais como o uso de *routing* e da integração dos controladores, responsáveis pela lógica de cada *endpoint*, que irão ser descritos numa fase posterior.

Por fim, foi identificada a ORM Sequelize, visto que é a ORM que apresenta os melhores resultados na comunicação do Node.js com o *PostgreSQL* e devido à facilidade de mapeamento das tabelas da base de dados para as classes do servidor (modelos) através dos scripts já existentes, bem como da manipulação dos dados destas.

2.5.12 Python

Dado que foi necessário estabelecer *scripts* no âmbito do processamento de dados a serem integrados no Sistema de Informação foi necessário escolher uma linguagem para este processo.

Tendo em conta estas necessidades foi escolhida a linguagem *Python*, que é uma linguagem orientada a objetos e com tipos de dados dinâmicos, uma vez que se apresenta como uma linguagem fácil de utilizar e com um bom suporte na sua documentação. Além disso o *Python* é reconhecido pela facilidade de uso na componente de manipulação

de dados sobretudo devido a bibliotecas como o *pandas* e *geopandas* que permitem a importação dos mais variados tipos de dados, permitindo que estes sejam transformados em novos tipos de dados.

No âmbito da integração das fontes de dados geográficos, um dos processos utilizados foi totalmente desenvolvido em *Python* devido a estas características, pelo que a sua utilização foi um processo fundamental para o processamento de dados a serem inseridos no Sistema de Informação. Este processo será posteriormente descrito em 4.3.1.

2.5.13 Raster2pgsql

No âmbito da integração de dados *raster* foi necessário identificar ferramentas adicionais às referidas previamente de forma a que fosse possível integrar este tipo de dados na base de dados *PostgreSQL*.

Para isso foi identificado, o raster2pgsql que consiste num executável capaz de carregar dados raster cujos formatos são suportados pelo GDAL, transformando-os em dados que podem ser armazenados e processados pelas tabelas do *PostGIS*. A escolha desta ferramenta ocorre devido à sua compatibilidade com a base de dados escolhida, tornando a inserção de dados raster um processo mais simples.

2.5.14 Firebase Cloud Messaging

Apesar de já referida, previamente, o *Firebase* apresenta um conjunto de serviços que serão úteis no âmbito desta dissertação. Face à necessidade de existir comunicação a tempo real entre o Sistema de Informação e a Aplicação Móvel, nomeadamente devido ao envio de alertas foi necessário identificar um serviço que fosse responsável por este efeito.

Uma vez que o servidor está alojado na *Google App Engine*, a escolha do *Firebase Cloud Messaging* para este efeito surge como uma escolha óbvia, uma vez que disponibiliza o envio de mensagens entre plataformas distintas com elevado grau de facilidade e sem custo.

O processo de envio de notificações e alertas, para qual o FCM será descrito, posteriormente em detalhe, em 4.4.

2.5.15 Google Cloud Logging

Com vista a garantir que todas as funcionalidades do Sistema de Informação funcionam como o previsto foi necessário utilizar uma ferramenta de *Logging* de forma a rastrear o comportamento do Sistema perante as interações com os outros serviços da Google, bem como com os pedidos dos utilizadores.

Assim sendo, foi escolhida a utilização da ferramenta *Google Cloud Logging*[21], que é um serviço que permite realizar a monitorização de eventos que ocorrem no *App Engine*, registando *logs* sobre os pedidos dos utilizadores e também, outros que podem ser especificados na implementação da lógica do servidor. Dessa forma é possível obter

informação sobre certos aspetos que possam vir ser importantes a monitorizar de forma a garantir o correto funcionamento do sistema, como por exemplo, a informação dum erro que é devolvido caso a lógica dum *endpoint* falhe.

Além disso, a utilização do *Google Cloud Logging* surge como escolha óbvia devido a todas as ferramentas de análise e de visualização oferecidas por esta, permitindo distinguir com facilidade os variados tipos de *logs* que podem ser emitidos, facilitando a análise da informação sobre cada um destes.

2.5.16 Postman

No âmbito do processo de elaboração do Sistema de Informação foi importante arranjar uma forma acessível de testar os vários *endpoints* disponibilizados pela API REST. Dessa forma, recorreu-se ao Postman [44] para este efeito, que consiste numa plataforma API, com o objetivo de facilitar a criação e a utilização de APIs.

O Postman disponibiliza ferramentas que simplificam a execução de pedidos ao servidor com que se pretende interagir, através da definição do método do pedido, do seu *endpoint*, bem como dos mais diversos parâmetros, nomeadamente, parâmetros de *query*, parâmetros de autorização, bem como a definição dum *body*, caso isso se aplique.

Além disso, o Postman permite a importação de coleções através dum documento JSON, onde está especificada a informação relativa aos *endpoints* definidos na API REST. Dessa forma, foi possível garantir que todos os *endpoints* eram testados, com o intuito de garantir a sua correta execução. Ademais foi possível gerar exemplos de pedidos e de respostas que podem ser utilizados na criação da documentação, que também poderá ser suportada pelo Postman, uma vez que este disponibiliza ferramentas para a criação de documentações para as coleções definidas.

2.6 Trabalhos Relacionados

Para a realização desta dissertação foi necessário dividir a pesquisa sobre trabalhos realizados em duas partes, nomeadamente, na pesquisa de trabalhos que consistiam na criação de sistemas de informação, bem como na aplicação de alguns serviços da Google Cloud Platform e na pesquisa de trabalhos que já tinham sido realizados para o projeto Floresta Limpa, de forma a perceber que dados e funcionalidades o sistema de informação deve, especificamente, conter.

2.6.1 Projeto Floresta Limpa

Relacionado com o projeto Floresta Limpa, o trabalho realizado por Ricardo Afonso [3] serviu de ponto de partida para inferir dados que foram e que continuam a ser utilizados pelos membros do projeto no âmbito da deteção remota.

De facto, neste trabalho as principais fontes de dados consistiram em *shapefiles* das FGCI, nomeadamente, os ficheiros vetoriais disponibilizados pelo ICNF, bem como os

disponibilizados pela Câmara Municipal de Mação e imagens de satélite obtidas a partir dos satélites das missões Sentinel-1 e Sentinel-2.

Relativamente aos ficheiros vetoriais das FGCI, foi realizada uma separação das faixas pelo seu tipo, que é necessária indicar, uma vez que diferentes tipos de faixa apresentam características diferentes que têm de respeitar. Além disso, esta divisão foi realizada para proceder à criação de *buffers*, que consistem em secções exteriores com 20 metros de largura, adjacentes às faixas. A criação dos *buffers* tem como objetivo estabelecer uma análise da comparação com o estado da vegetação exterior à faixa com o do interior da faixa de forma a detetar possíveis intervenções. Assim sendo, o sistema de informação está pronto para armazenar estes *buffers* associados a cada faixa.

Além disso, para cada faixa foi ainda realizada a divisão das FGCI bem como dos seus *buffers* associados por segmentos, de forma a facilitar a comparação entre o estado exterior da vegetação com o interior à faixa.

Quanto aos dados por satélites, foram utilizadas as imagens dos Sentinel-1 e Sentinel-2 e foram inferidas características a partir destes. Dessa forma, foram criados 4 conjuntos de dados, nomeadamente **CD_TUDO** que contém todas as bandas do Sentinel-2, índices de vegetação e polarizações, **CD_BANDAS** com as bandas do Sentinel-2, **CD_ÍNDICES** com os índices de vegetação apenas, **CD_FUSÃO**, com as bandas do vermelho, verde, *red-edge* e infravermelho, bem como os índices de vegetação NDVI e NDWI. Dessa forma, o sistema de informação tem de estar pronto para disponibilizar os dados dos satélites, bem como das características inferidas a partir deles.

2.6.2 Projeto FitoAgro

Com vista a compreender informações essenciais sobre a modelação e implementação de um sistema de informação foi consultado o trabalho realizado por André Malafaia [34], realizado no âmbito do Projeto FitoAgro, que tem como objetivo o estudo e a monitorização das pragas emergentes que afetam as pomóideas do Oeste.

Apesar dos dados deste projeto não estarem relacionados com o projeto onde esta dissertação se insere, existem características semelhantes, uma vez que este projeto é um sistema de informação geográfica, que lida com dados provenientes de estações meteorológicas, interagindo, também, com utilizadores para processos de recolha e consulta de dados de campo.

Dessa forma, a partir deste trabalho foram consultados algumas regras de boa prática na modelação, bem como na identificação de requisitos gerais de um sistema de informação. Além disso foi possível identificar pedidos da API REST em comum aos dois projetos, facilitando a identificação das operações genéricas. O facto deste projeto se dedicar a um sistema de informação, em conjunto com a pesquisa realizada no trabalho do Ricardo Afonso, foi um passo fundamental no estabelecimento de um ponto de partida no desenvolvimento desta dissertação.

2.6.3 Outros trabalhos

Embora os projetos enumerados apresentem as características principais que esta dissertação tem de cobrir, nenhum deles apresenta características sobre o desenvolvimento dum sistema de informação em *cloud*, especificamente, na *Google Cloud Platform*.

Assim sendo, foi necessário realizar uma pesquisa sobre trabalhos que utilizam os serviços disponibilizados pela GCP, de forma a compreender como estes podem ser utilizados de forma eficiente de acordo com as intenções deste projeto.

Em *Deploying an Application using Google Cloud Platform* [23] é realizada uma introdução às características que a GCP fornece, são apresentados os benefícios do alojamento de um sistema na GCP por comparação a um servidor remoto, permitindo inferir características que podem ser vantajosas em cada um dos casos. Além disso, são apresentados os requisitos para desenvolver uma aplicação no GCP.

Em *Large scale crop classification using Google earth engine platform* [50] é demonstrada a aplicação do serviço GEE na classificação de culturas em territórios de grandes dimensões, através de imagens obtidas a partir do Landsat 8 disponível no catálogo da GEE. O facto deste artigo demonstrar sucesso com territórios vastos permite inferir que o GEE pode apresentar características úteis no âmbito das FGCI, visto que estas, também, têm dimensão nacional.

2.7 Conclusões

Este capítulo permitiu adquirir conhecimentos na área onde esta dissertação se insere, através do estudo da sua temática, bem como das tecnologias e dados necessários.

De facto, o estudo da informação geográfica foi fundamental, uma vez que permitiu perceber como esta se representa e como é partilhada na Internet, nomeadamente através dos serviços WMS e WFS. Relacionada com a informação geográfica, foi necessário averiguar as principais fontes de dados que necessitam de estar contidas no sistema, permitindo adquirir conhecimentos sobre que tipos de dados o sistema tem de suportar. O estudo dos dados de satélite permitiu compreender de que forma estes são úteis para o projeto, uma vez que fornecem de livre acesso dados dos quais se pode inferir um conjunto de características sobre as FGCI.

No âmbito da parte tecnológica foi fundamental realizar um estudo sobre a GCP, permitindo ter uma melhor noção dos serviços que seriam necessários de implementar, garantindo a melhor eficácia possível ao sistema. Além disso, foram também consultadas bibliotecas e ferramentas úteis para a manipulação de dados georreferenciados.

Por fim, os trabalhos relacionados permitiram compreender o tipo de dados que o sistema tem de conter, bem como as melhores práticas na sua implementação para o projeto Floresta Limpa.

Capítulo 3

Modelação do Sistema de Informação

Esta dissertação, tal como foi referido no capítulo Introdução, tem como objetivo a implementação de um sistema de informação para o projeto Floresta Limpa, tendo por base a disponibilização de dados que permitam a realização de algoritmos de aprendizagem automática destinados a classificar o estado da vegetação das FGCI, bem como do armazenamento dos produtos gerados a partir destes. Além disso, irá incorporar dados fornecidos por voluntários que se irão deslocar ao local para obter ou confirmar informação sobre as faixas, bem como disponibilizar estatísticas sobre a informação produzida.

Dessa forma, numa primeira fase foi necessário identificar os atores do sistema de informação e elaborar uma lista de requisitos funcionais e informacionais que o sistema deve respeitar.

Posteriormente, face aos requisitos do sistema, foi necessário definir a arquitetura do sistema, escolhendo como os dados iriam ser armazenados e que serviços disponibilizados pela GCP iriam ser utilizados.

Na fase seguinte, procedeu-se à construção dum modelo de dados genérico, definindo algumas regras de nomenclatura, bem como, identificando as principais entidades que o sistema tem de possuir e a sua distribuição ao longo dos diversos módulos.

Por fim, são apresentadas algumas considerações finais sobre o estado do projeto e o seu desenvolvimento.

3.1 Atores do Sistema de Informação

Os atores dum sistema de informação consistem nos utilizadores que interagem com o sistema, quer seja através de consulta ou fornecimento de dados. No âmbito do projeto Floresta Limpa foram identificados os seguintes atores:

- **Organizações**: As organizações representam as instituições que têm associações

com os utilizadores do sistema ou que têm interesse na utilização do sistema, tais como empresas florestais, câmaras municipais, autoridades de fiscalização, como a GNR e outras organizações parceiras.

- **Pessoas**: As pessoas representam, sobretudo, os membros das organizações ou utilizadores destinados a consumir informação do sistema, tais como os membros do projeto ou os bombeiros.

- **Voluntários**: Os voluntários representam os utilizadores cuja função primária é o fornecimento de dados ao sistema, nomeadamente, informação sobre as FGCI.

- **Aplicação Móvel**: A aplicação móvel é o dispositivo primário que os outros atores referidos irão utilizar de forma a interagir com o sistema, no entanto, a própria aplicação irá interagir autonomamente com o sistema, sendo portanto, mais um ator do sistema.

3.2 Requisitos Funcionais e Informacionais

O processo de identificação dos requisitos funcionais e informacionais foi realizado em simultâneo, através da enumeração de *user stories*, com base nas necessidades que cada ator iria ter na interação com o sistema de informação.

Visto que o sistema de informação tem de cumprir um conjunto vasto de requisitos funcionais e informacionais, estes são apresentados na totalidade no apêndice A e no apêndice B, respetivamente.

3.3 Arquitetura

A arquitetura deste sistema de informação consiste num modelo cliente-servidor e encontra-se dividida em três camadas principais, a camada do servidor ou *business logic*, que irá ser desenvolvida no conceito desta dissertação, tal como a camada de armazenamento e a camada de cliente ou de apresentação que se encontra a ser produzida noutro trabalho que irá interagir com o servidor através da API REST pública que permitirá a consulta e a manipulação de informação, pelo que a interação entre as duas tem de ser respeitada.

Relativamente à camada do servidor, esta irá lidar com duas componentes, nomeadamente um *Cluster* privado alocado no DI da FCT NOVA para a realização de atividades de aprendizagem automática no âmbito da classificação das FGCI e uma componente que irá ser desenvolvida na GCP, de forma a que também será necessário criar uma API REST privada para a comunicação entre estes dois componentes. Na figura 3.1 é possível consultar uma representação visual da arquitetura escolhida.

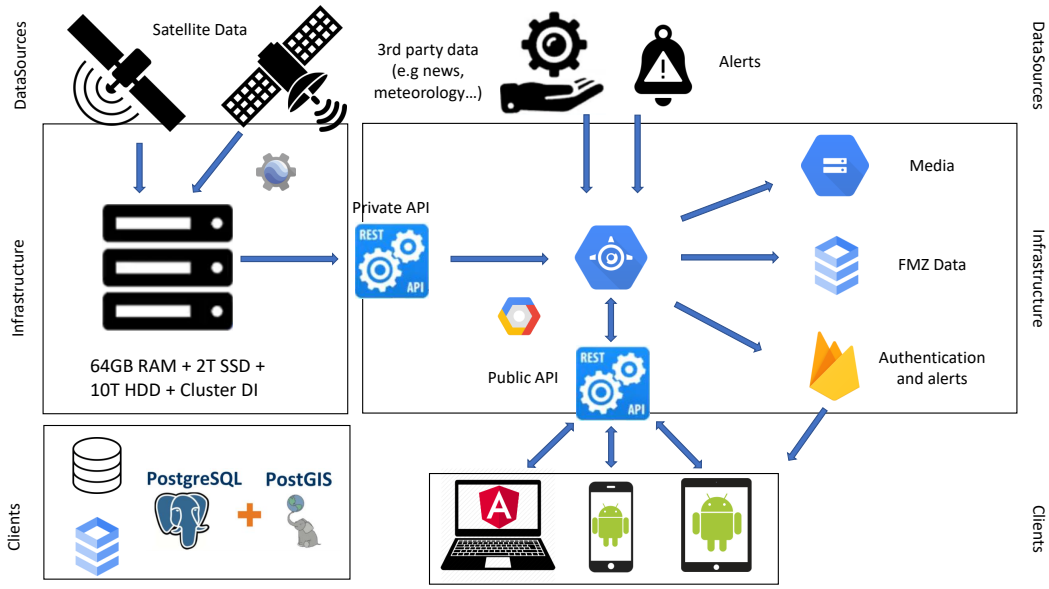


Figura 3.1: Arquitetura do sistema de informação do Floresta Limpa

O desenvolvimento da aplicação na GCP envolve uma escolha de produtos da Google, de forma a garantir que as propriedades que o sistema deve cumprir são respeitadas com uma elevada performance e custos adequados.

Dessa forma, para desenvolver a camada do servidor, esta vai ser elaborada através do uso da *Google App Engine*, uma vez que este produto permite a elaboração de aplicações sem ser necessário ter em conta a manutenção da infraestrutura, que é tratada pela Google, disponibiliza um mecanismo de *logging*, que permite consultar todas as chamadas da aplicação, bem como um mecanismo automático de versões, que permite o controlo das várias versões desenvolvidas. Este serviço irá fornecer a API REST, conter toda a lógica do sistema e processar os pedidos que chegam pela API REST, acedendo aos dados contidos nas bases de dados presentes na camada de armazenamento, de forma a devolvê-los ou a alterá-los.

De seguida no âmbito da camada de armazenamento foram escolhidas diversos produtos disponibilizados pela Google, de acordo com o tipo de dados que se querem tratar, através dos quais a GAE consegue aceder diretamente sem ser necessário configuração específica destes serviços. Assim, optou-se por utilizar o serviço da *Google Cloud Datastore* para armazenar informações sobre os utilizadores, tais como os dados das suas contas, bem como a informação recolhida pelos utilizadores na aplicação móvel (à exceção das imagens e vídeos), uma vez que este produto destina-se a armazenar informações de aplicações móveis e encontra-se desenvolvida sob o *Firebase*. Ademais, o *Firebase* permite também o envio de notificações para as aplicações, útil para o processo do envio de alertas. Posteriormente, foi escolhida a utilização da *Google Cloud Storage* destinada a armazenar informação estática e persistente, tal como as imagens e vídeos disponibilizados pelo utilizador. Por fim, irá ser utilizada a *Google Cloud SQL* para armazenar informação das

FGCI, bem como dos seus ficheiros vetoriais e classificações, visto que este produto permite a disponibilização duma base de dados *PostgreSQL* que apresenta a extensão *PostGIS*, que fornece suporte a dados georreferenciados, que será também utilizada para a criação da base de dados da camada de armazenamento do Cluster do DI, que irá receber dados de satélites provenientes do catálogo do *Google Earth Engine*.

3.4 Modelo de Dados

De forma a planear a implementação do sistema, foi necessário proceder a um conjunto de tarefas para a construção de um modelo de dados. Para isso, primeiramente, foram definidas as meta regras em 3.4.1 que foram seguidas no processo de modelação, com o objetivo de facilitar a compreensão do modelo de dados do sistema.

De seguida, prosseguiu-se com a identificação das entidades necessárias para o sistema, de forma a que os requisitos funcionais enumerados no apêndice A fossem respeitados. Após esta fase, procedeu-se à distribuição das diversas entidades pelos módulos do modelo de dados. Estes módulos pretendem retratar cada um dos focos do modelo de dados e são identificados pela sua respetiva cor: Agentes e Contas 3.4.2 (módulo amarelo), Espaço-Temporal 3.4.3 (módulo verde), Dados Recolhidos no Terreno 3.4.4 (módulo azul), Séries Temporais 3.4.5 (módulo vermelho), e por fim, Configurações 3.4.6 (módulo cinzento). Estes módulos são apresentados, posteriormente, seguidos duma explicação sobre as suas entidades principais, fundamentando as opções tomadas.

Por fim é apresentada a visão geral do modelo de dados, representado na imagem 3.2.

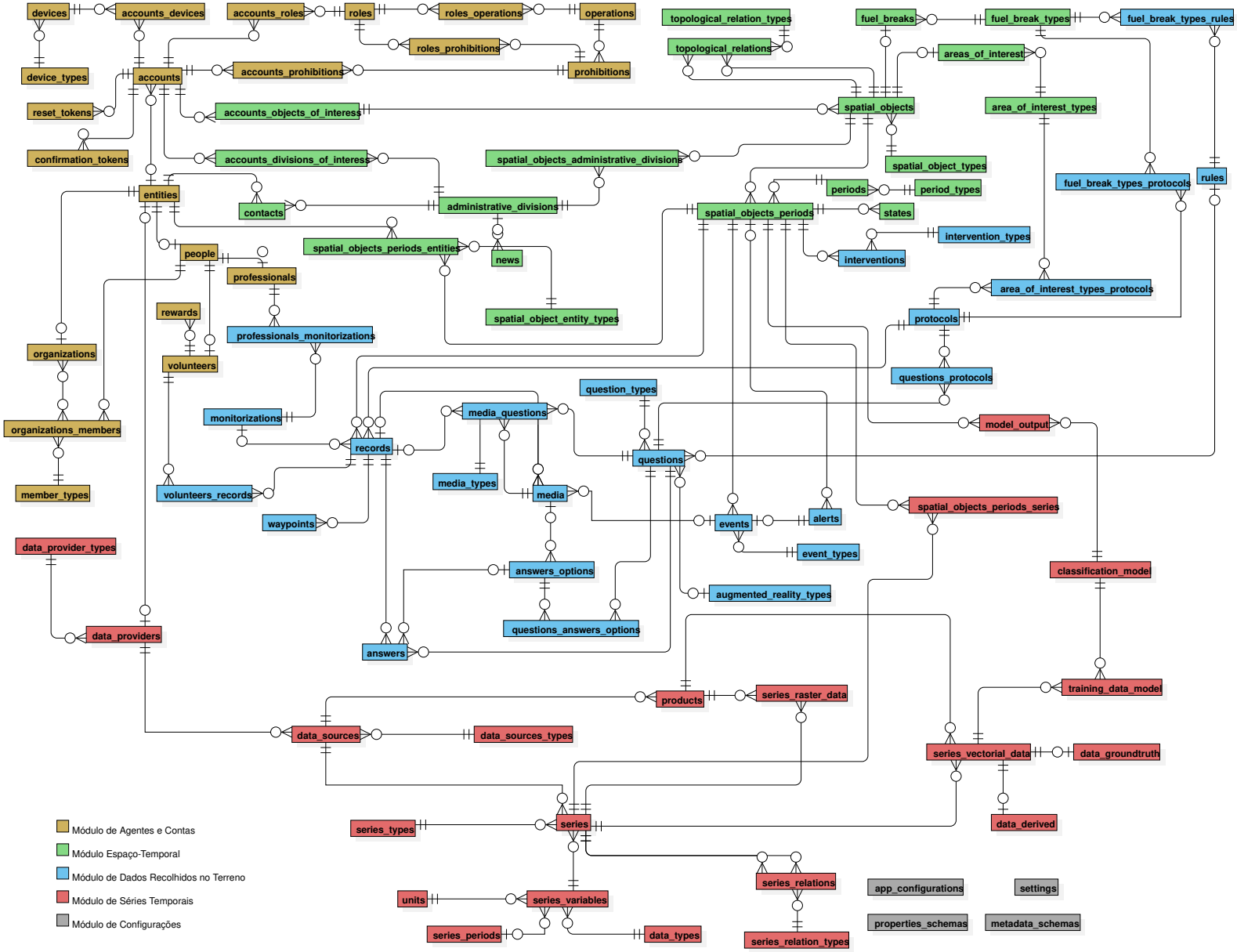


Figura 3.2: Visão Geral do Modelo de Dados

3.4.1 Meta Regras

Para facilitar a criação do modelo de dados foram definidas algumas meta regras a seguir ao longo do processo de modelação, que serão enumeradas, de seguida.

- **Nomenclatura Geral**: O nome das tabelas e dos atributos deverá seguir o padrão *snake case*, ou seja cada nome deverá começar com letra minúscula e os espaços deverão ser ocupados com *underscore*.
- **Nomenclatura de Atributos**: Todos os atributos deverão apresentar o seu nome escrito no singular.
- **Relação entre Tabelas**: Numa relação de vários para vários, a tabela intermédia deve conter o nome de ambas as tabelas (podendo haver exceções desde que estes sejam auto explicativos).
- **Chaves primárias**: Todas as tabelas deverão apresentar um atributo com id que irá ser a chave primária de cada tabela, incrementando automaticamente após a criação de cada objeto dessa tabela.
- **Chaves estrangeiras**: Todas as chaves estrangeiras deverão seguir o padrão <nome_tabela_singular_atributo>, para facilitar a compreensão de qual é a tabela de onde surge a relação bem como da identificação que é uma chave estrangeira, visto que não segue a nomenclatura dos atributos dessa tabela.
- **Timestamps**: Todas as tabelas deverão apresentar os seguintes timestamps: created_at, updated_at, deleted_at. No caso do created_at, este irá registar o tempo a que um dado objeto de cada tabela foi criado. O updated_at irá ser alterado sempre que houver qualquer alteração ao objeto em questão. Quanto ao deleted_at este surge, visto que o sistema de informação não irá permitir que ocorram remoções diretas de objetos, de forma a permitir uma fácil recuperação em caso de erro dos utilizadores.
- **Extensão**: Todas as tabelas deverão ter um atributo que representa um objeto JSON de forma a permitir a adição de novos atributos a cada tabela sem alterar a sua estrutura, garantindo um caráter extensível ao modelo.
- **Metadados**: Todas as tabelas deverão possuir um atributo Metadata, que é um objeto JSON de forma a guardar todas as informações sobre os metadados dos dados.

3.4.2 Módulo de Agentes e Contas

Este módulo pretende representar toda a informação relevante para a autenticação e autorização de todos os agentes presentes no sistema. Para tal, neste módulo será armazenada toda a informação sobre as contas (pessoais ou institucionais) dos utilizadores para que estes possam aceder ao sistema.

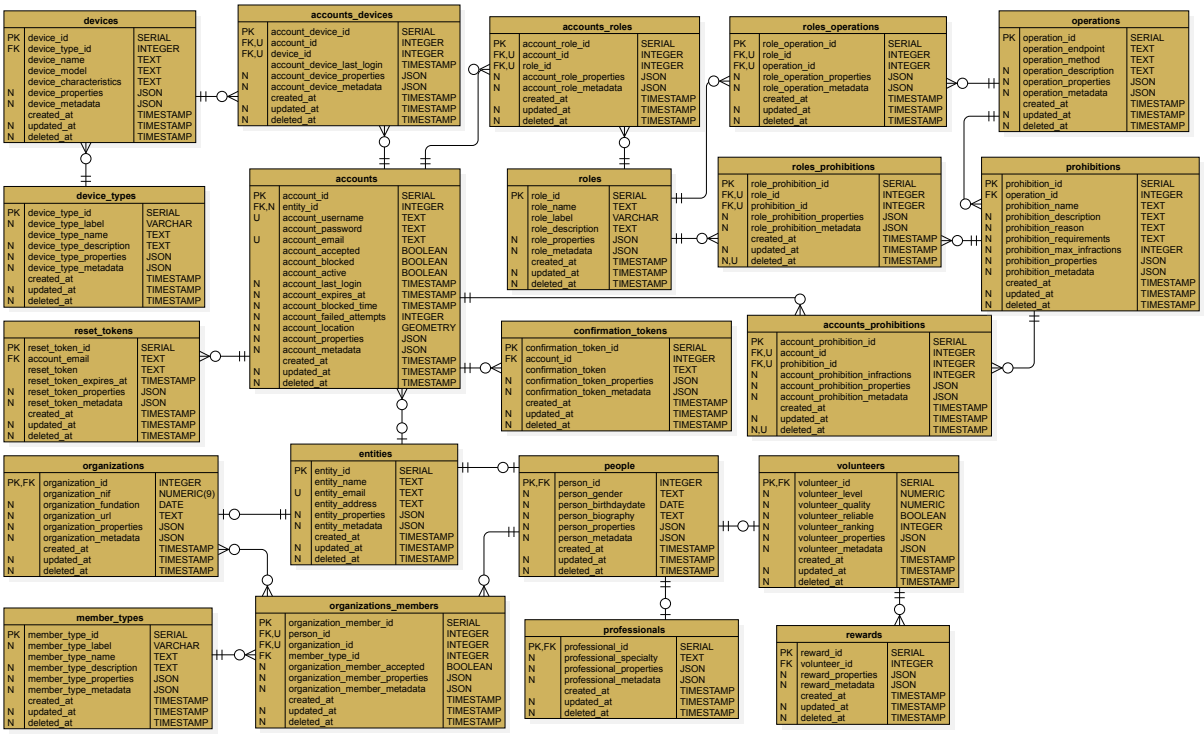


Figura 3.3: Módulo de Agentes e Contas

3.4.2.1 Entidades, Organizações, Pessoas e Voluntários

No âmbito deste módulo foram criadas entidades com o intuito de representar, os agentes do sistema, tendo sido identificados dois principais grupos, as organizações e as pessoas, sendo que estas últimas também pode adquirir a função de voluntário.

Dessa forma, as organizações têm como objetivo representar as instituições que podem ter interesse ou responsabilidade em acompanhar informação sobre as faixas de gestão de combustível. Por outro, lado, o sistema permite que cada utilizador seja apenas um indivíduo com interesse no âmbito das faixas, pelo que permite o registo da sua informação pessoal. Caso o utilizador queira fazer parte da partilha de dados recolhidos no terreno sobre as faixas, este pode especificar-se como um voluntário, possuindo, assim, um conjunto de características adicionais que permitam avaliar o seu trabalho de voluntariado.

Por fim, as entidades correspondem a uma entidade genérica que poderá ser especificada como uma pessoa ou como uma organização. Esta tabela existe com o intuito de guardar atributos comuns entre as tabelas referidas.

3.4.2.2 Contas, Funções, Operações e Proibições

As contas representam a entidade base deste módulo, estas têm como objetivo permitir a interação com o sistema a cada uma das entidades acima referidas. Para isso, cada conta disponibiliza a opção de associar a si a sua entidade correspondente.

Para que as contas possam interagir com os sistemas foi necessário definir um conjunto

de funções que cada conta pode desempenhar. Cada uma destas funções tem como objetivo possuir o conjunto de operações que podem ser realizadas pelo sistema, isto é, o conjunto de pedidos REST que podem ser requeridos ao sistema.

Além disso, foram também introduzidas proibições, de forma a limitar as funcionalidades que alguns utilizadores podem executar, devido ao uso incorreto das mesmas, permitindo assim manter um histórico dos motivos que levaram uma conta a ter certos acessos revogados. Dessa forma, é possível controlar as ações realizadas pelos utilizadores no sistema de informação.

3.4.3 Módulo Espácio-Temporal

Este módulo tem como objetivo modelar os aspetos espácio-temporais. Dessa forma, neste módulo são contidas as zonas (FGCI ou áreas de interesse) associadas a períodos, que visam destacar o estado destas em intervalos de tempo.

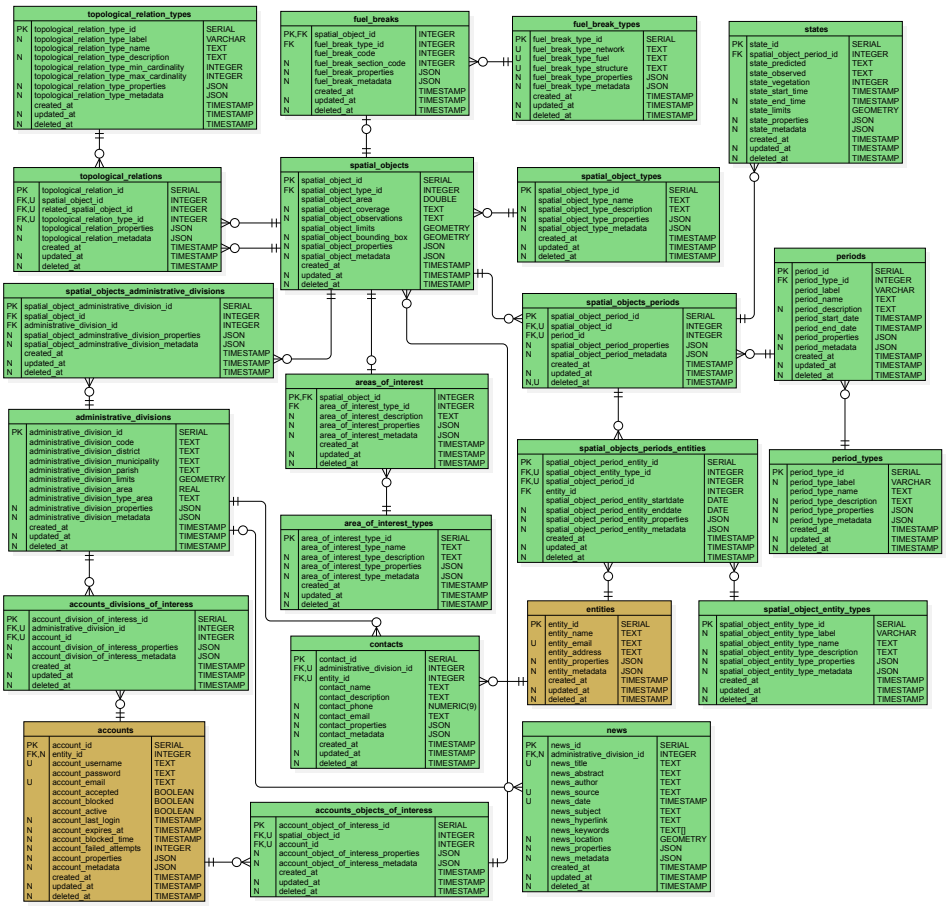


Figura 3.4: Módulo Espaço-Temporal

3.4.3.1 Períodos

Os períodos pretendem modelar os intervalos temporais, sendo, por isso, uma entidade que apresenta uma data inicial e final definidas. Além disso, disponibiliza informação

como o seu nome e uma descrição opcional de forma a facilitar a identificação destes. Estes períodos estão, por fim, relacionados com tipos de período que, por exemplo, podem tomar valores trimestrais, anuais, entre outros.

3.4.3.2 Objetos espaciais, Faixas de Gestão de Combustível e Áreas de Interesse a Monitorizar

As entidades referidas nesta secção representam-se como as entidades essenciais deste módulo. No âmbito do projeto será importante avaliar várias zonas, nomeadamente, faixas de gestão de combustível, bem como áreas de interesse a monitorizar.

De forma a permitir a avaliação das FGCI e das áreas de interesse, foi criada uma entidade denominada objetos espaciais que consistem em polígonos ou multi-polígonos definidos através duma componente geográfica vetorial. Além disso, esta entidade pretende guardar a informação sobre os atributos em comum das outras duas entidades referidas. Dessa forma, é possível especificar estes objetos espaciais, como uma FGCI ou como uma área de interesse, fornecendo a estes objetos espaciais, atributos adicionais.

Para cada um destes dois tipos é, ainda, associada uma tabela de tipos, permitindo inferir informações genéricas sobre o tipo de FGCI, bem como o tipo de área de interesse a monitorizar.

Além destas tabelas foi, ainda, criada uma tabela de relações topológicas permitindo registar no sistema qualquer tipo de relação topológica entre os objetos espaciais. Além disso, como forma de suporte a esta tabela está associada uma tabela de tipos de relações topológica, onde é possível definir qual é o tipo de relação em questão, bem como definir a cardinalidade entre estas relações. No caso do projeto da Floresta Limpa um dos exemplos de relações topológica possível é a representação hierárquica de faixas, onde uma faixa pode ser constituída por diversos segmentos de faixa.

3.4.3.3 Objetos espaciais em períodos

Os objetos espaciais em períodos representam o conceito principal deste módulo, uma vez que associa as entidades que são tema principal do projeto com a entidade períodos, permitindo representar cada um destes objetos espaciais numa determinada janela temporal. Esta associação é obtida através duma relação N:M entre os objetos espaciais e os períodos, sendo fundamental para disponibilizar a representação, através doutras tabelas, de tudo o que ocorre nestes objetos espaciais num determinado período temporal. No âmbito do projeto, uma faixa de gestão de combustível ao longo do ano de 2022 é um exemplo de um objeto espacial num determinado período.

3.4.3.4 Estados

Os estados têm como objetivo representar a informação sobre o estado dum objeto espacial, num dado espaço ou instante temporal. Apresentam características sobre a vegetação, nomeadamente, o seu estado previsto, observado, bem como o seu tipo. Disponibiliza,

ainda uma componente geográfica vetorial, de forma a poder identificar diferentes estados em diferentes zonas dos objetos espaciais. No âmbito do projeto, um exemplo dum estado é o estado duma faixa de gestão de combustível num período temporal.

3.4.3.5 Divisões administrativas

De forma a facilitar o agrupamento dos objetos espaciais foi introduzida, neste módulo, a tabela das divisões administrativas. As divisões administrativas têm como objetivo limitar as regiões dum país, atribuindo a essas regiões a responsabilidade sobre o que acontece na sua área, nomeadamente, o controlo das faixas da rede secundária. Para este projeto, um exemplo duma divisão administrativa é um dos concelhos existentes em Portugal, definidos na Carta Administrativa Oficial de Portugal (CAOP)

3.4.4 Módulo de Dados Recolhidos no Terreno

Este módulo visa retratar os dados obtidos no terreno para uma determinada zona num dado período de tempo, por parte dos utilizadores do sistema. Assim sendo, são disponibilizadas as intervenções realizadas, os eventos associados, os registos de voluntários, bem como os registos de monitorizações realizadas pelas entidades competentes.

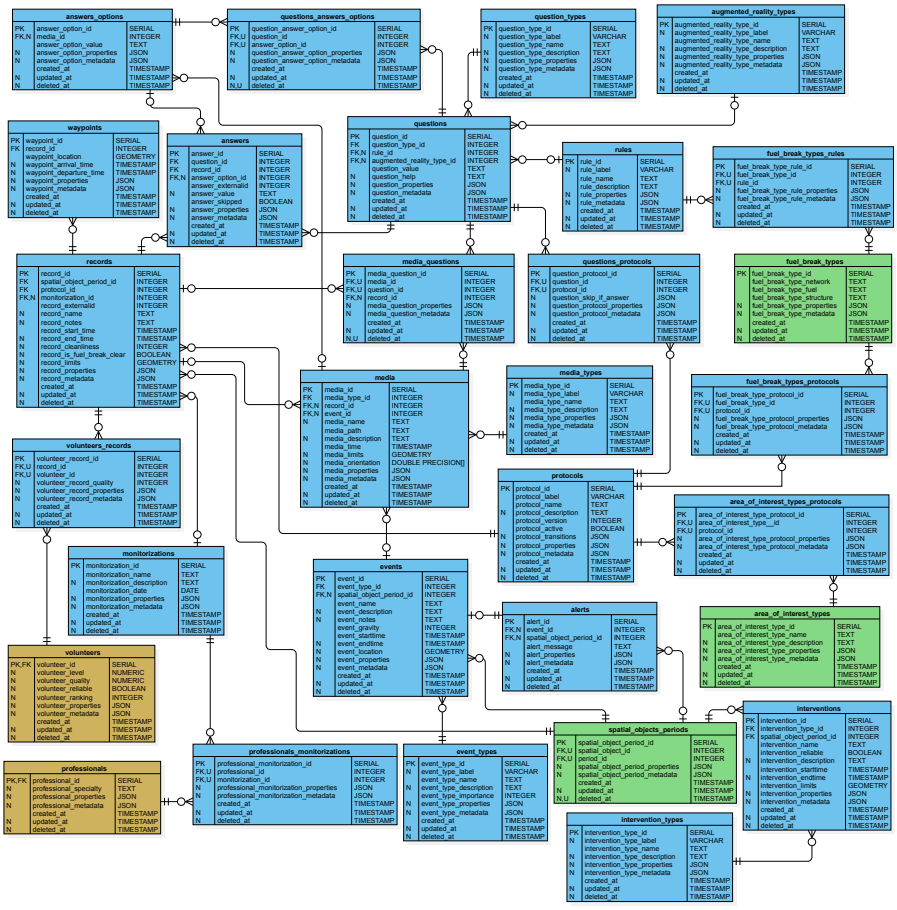


Figura 3.5: Módulo de Dados Recolhidos no Terreno

3.4.4.1 Protocolos

Os protocolos de recolha de dados consistem num conjunto de questões, definidas pelos membros do projeto, que são colocadas aos utilizadores sobre a faixa ou área de interesse onde se encontram, baseadas nas condições e regras, definidas por especialistas, que cada uma destas zonas deve respeitar.

Cada uma destas questões pode ter associada a si fotos para ajudar a compreensão do utilizador, um conjunto de opções de resposta de forma a obter a informação pretendida sobre estas zonas, bem como técnicas de realidade aumentada de forma a facilitar a experiência do utilizador, por exemplo em termos de cálculo de distâncias e alturas. No âmbito deste projeto, um dos protocolos que pode ser colocado aos utilizadores baseia-se num conjunto de questões sobre aspetos que todas as faixas devem cumprir.

3.4.4.2 Registos e Monitorizações

Face à resposta, aos protocolos referidos em cima sobre uma determinada faixa ou área de interesse num determinado espaço temporal, serão criados registos que poderão ser realizados quer por voluntários ou por especialistas. No caso destes serem realizados por especialistas, vai ser possível registar uma monitorização de forma a poder distinguir com uma maior facilidade os registos que apresentam uma informação mais fidedigna, uma vez que foi realizada por peritos na matéria.

Estes registos vão ter associados a si as diversas respostas que os utilizadores deram face às perguntas dos protocolos, bem como fotos que os utilizadores tenham tirado de forma a identificar aspetos úteis sobre as faixas e irão processar a informação de forma a obter respostas genéricas face ao estado dessas zonas naquele período de tempo.

3.4.4.3 Eventos

Os eventos pretendem modelar qualquer atividade fora do normal que aconteça num objeto espacial num determinado período de tempo, com relevância para o projeto. Nestes eventos será possível registar o tipo de evento, o período de tempo em que este evento ocorreu, bem como a sua gravidade. No contexto do projeto Floresta Limpa é exemplo dum evento, um incêndio que ocorra numa faixa de gestão de combustível.

3.4.4.4 Multimédia

Associadas aos registos, às questões, às respostas e respetivas opções, bem como aos eventos, existe a possibilidade de guardar conteúdo multimédia, nomeadamente, fotografias, vídeos, imagens georreferenciadas. Este conteúdo será armazenado num *container* no âmbito do *Google Cloud Storage*, apresentando a base de dados um apontador para a *path* do ficheiro multimédia no *container*, bem como o identificador deste na *storage*, de forma a serem criados as hiperligações de *resumable uploads*, quando um utilizador pretende aceder às suas fotos.

3.4.4.5 Intervenções

As intervenções têm como objetivo registar atividades realizadas num determinado objeto espacial, tais como a limpeza dessa zona. Para isso, é possível registar o tipo de intervenção que ocorreu, o período em que esta foi realizada, bem como identificar a localização geográfica dentro dessa zona onde ocorreu a intervenção, uma vez que esta pode não ocorrer em toda a sua extensão.

3.4.5 Módulo de Séries Temporais

Este módulo pretende modelar todo o conjunto de dados disponibilizado pelas fontes de dados de informação geográfica, que estão, normalmente, associados a séries temporais, que consistem num conjunto de observações sobre uma determinada variável ao longo dum período de tempo. Dessa forma, toda a informação acerca dos fornecedores, das fontes, das séries, das relações entre séries, bem como dos formatos dos dados fornecidos estará ao cargo deste módulo.

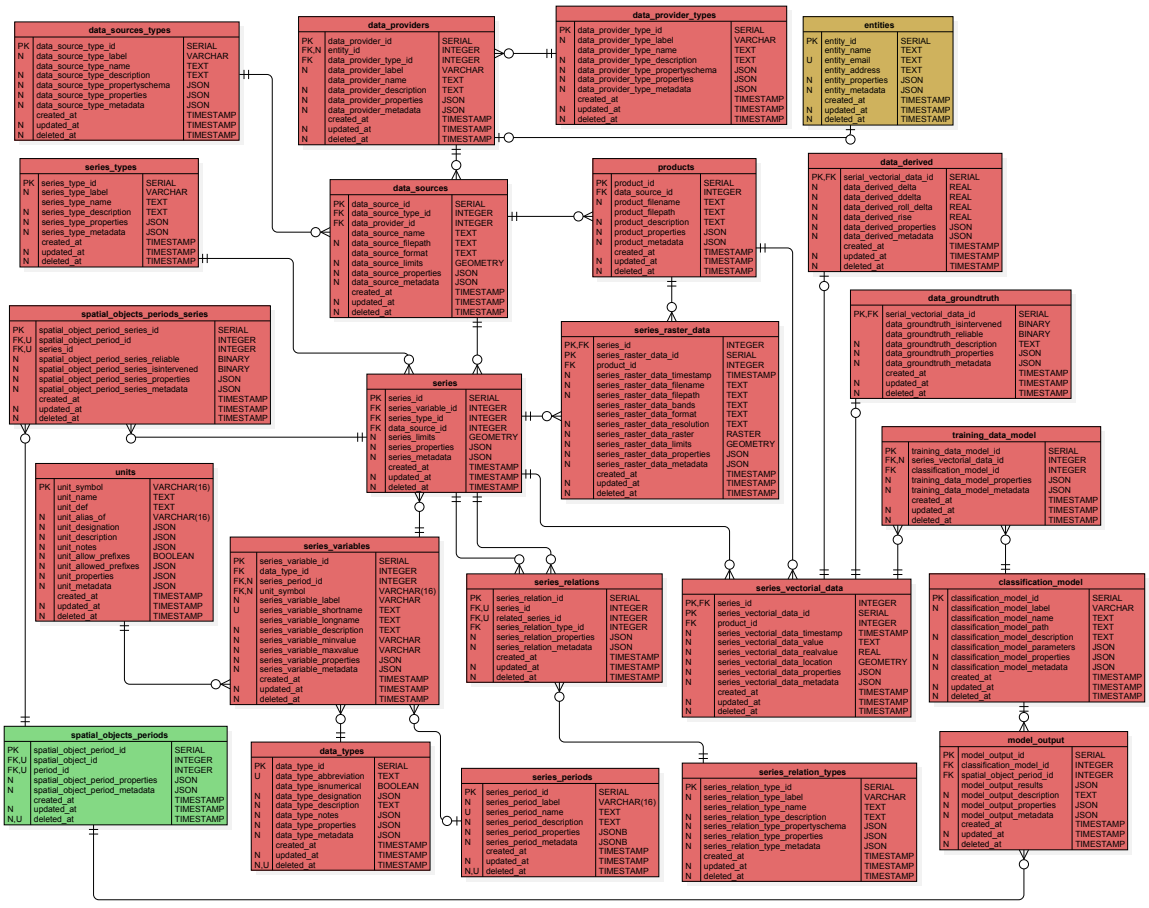


Figura 3.6: Módulo de Séries Temporais

3.4.5.1 Fornecedores de Dados

Os fornecedores de dados representam as entidades que disponibilizam as fontes de dados que serão utilizadas para guardar séries temporais no sistema de informação. No âmbito do projeto Floresta Limpa são exemplos de fornecedores o ICNF, a Navigator, a DGT, as diversas câmaras municipais, entre outras.

3.4.5.2 Fontes de Dados e Produtos

As fontes de dados correspondem aos ficheiros disponibilizados pelos fornecedores, de onde vão ser inferidas as diversas séries temporais. Estas fontes têm associadas a si, o seu tipo (por exemplo o satélite Sentinel-2), as suas coordenadas geográficas, caso seja uma fonte com uma localização constante. Em alguns casos, sobretudo na situação dos satélites, associados à fonte de dados, estão os diversos produtos disponibilizados por esta, uma vez que não será apenas um ficheiro.

3.4.5.3 Variáveis das Séries

As variáveis das séries dizem respeito à variável que está a ser observada num dada série temporal. Nestas podem ser registadas o seu tipo de dados, o período em que esta é obtida, caso seja um período uniforme, bem como a unidade utilizada para registar os valores desta. Além disso disponibiliza ainda valores dos valores máximos e mínimos que esta variável pode tomar. No contexto do projeto Floresta Limpa um exemplo duma variável a ser registada é o *Normalized Difference Vegetation Index* (NDVI) obtido a partir dos dados de satélites através do cruzamento de bandas.

3.4.5.4 Séries, Dados Raster e Vetoriais de Séries

As séries representam as séries temporais que estão a ser registadas, nestas vai ser possível obter a variável que está a registada, a fonte de dados que a está a gerar, bem como o tipo de série (se é *raw* ou derivada a partir doutra série temporal). Caso sejam séries derivadas é possível registar a relação entre a série *raw* e a série derivada na tabela dedicada às relações entre séries. Além disso, caso a fonte de dados não seja uma fonte fixa, é possível associar na série temporal uma componente geográfica para identificar a localização da zona que a série está a monitorizar.

Associadas a estas séries são apresentadas no módulo, duas tabelas: dados *raster* das séries e dados vetoriais das séries, onde serão registados os valores sobre cada série temporal. A distinção entre estas duas tabelas surge uma vez que o tipo de dados de cada série poderá ser um vetor ou um *raster*. Para tal, nos dados vetoriais serão registados os valores em texto ou numérico que a variável da série temporal pode tomar. Relativamente aos dados *raster*, estes pretendem apresentar a informação sobre os ficheiros *raster* que dão origem a esta série temporal, bem como as bandas utilizadas para o obter, guardando por fim a o *raster* em si associado.

3.4.5.5 Dados Derivados, Groundtruth, Dados de treino, Modelos de Classificação e Resultados dos Modelos

No âmbito dos trabalhos de aprendizagem automática que têm sido realizados no projeto surgiu a necessidade de criar tabelas para armazenar as informações geradas para estes. Através da interpretação dos dados obtidos das séries temporais são identificados instantes temporais nas séries que podem ser registados como pontos onde ocorreram intervenções, dando origem a dados de *groundtruth*. Ademais, é possível identificar estes mesmos instantes em pontos onde existem certeza que não ocorreram intervenções, bem como identificar casos onde existem esta incerteza.

Além disso, de forma a treinar os modelos de classificação é necessário obter variáveis extra, que irão ser armazenadas nos dados derivados. Assim sendo, será, também armazenada a informação sobre quais foram os dados utilizados para treinar cada modelo.

Por fim, irão ser armazenados os resultados obtidos por parte dos algoritmos de aprendizagem automática tais como, o número de intervenções identificadas, mantendo informação sobre quais foram os modelos que os geraram e sobre que fragmento duma faixa de gestão de combustível dizem respeito.

3.4.6 Módulo de Configurações

Para concluir, o módulo de configurações disponibiliza a informação sobre as configurações base do modelo de dados, do Sistema de Informação bem como da aplicação móvel.

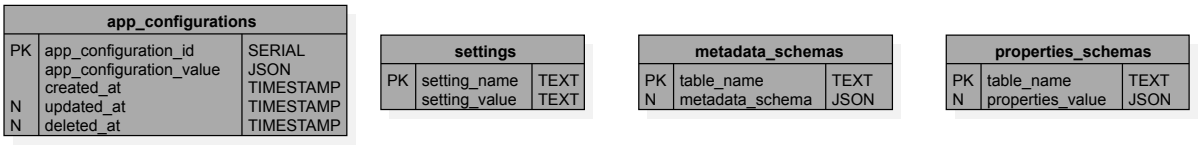


Figura 3.7: Módulo de Configurações

3.4.6.1 Esquemas de Metadados

A tabela de esquemas de metadados será responsável por definir qual é o esquema JSON que irá ser utilizado, por todas as tabelas, no campo "*metadata*".

3.4.6.2 Esquemas de Propriedades

A tabela de esquemas de propriedades será responsável por definir qual é o esquema JSON que irá ser utilizado, por todas as tabelas, no campo "*properties*".

3.4.6.3 Definições

Na tabela de definições será armazenada a informação geral sobre o modelo de dados.

3.4.6.4 Configurações da App

Na tabela de configurações da aplicação irão ser armazenadas em formato JSON, as variáveis de configuração necessárias à inicialização da aplicação móvel.

3.5 API REST

3.5.1 Regras de Desenho

Nesta secção são apresentadas as regras de desenho adotadas para o desenvolvimento da API REST:

- **Modelo de Maturidade Richardson**: A API desenvolvida situa-se no nível 2 do modelo de maturidade Richardson [47]. Assim sendo, esta faz uso de múltiplos *endpoints* para disponibilizar a informação. Cada um dos *endpoints* faz então uso dum método HTTP (POST, GET, PATCH e DELETE) consoante a operação que se pretenda realizar. Todas as respostas do servidor usam códigos de resposta HTTP de forma a permitir à camada de cliente identificar o estado do pedido que esta realizou.

- **Operações CRUD**: Cada um dos recursos da API apresenta *endpoints* para cada uma das operações CRUD (Create, Read, Update, Delete) utilizando para isso os métodos HTTP POST, GET, PATCH e DELETE, respetivamente.

- **Autorização**: O acesso à API é disponibilizado através de um *Bearer Token* passado no *Authorization Header* de cada pedido. Para tal, o *token* de acesso pode ser gerado e enviado ao utilizador de duas formas distintas: através do *endpoint* que permite a identificação do utilizador com recurso ao email e à password da sua conta ou através do *endpoint* da autenticação *third party* com a sua conta Google.

- **Querying**: As operações GET que devolvem coleções de objetos de cada recurso na API podem recorrer a *querying* de modo a que os resultados devolvidos sejam filtrados. Dessa forma, é possível filtrar este resultados consoante o atributo de qualquer um dos valores que um campo do recurso possa tomar. Além disso, foram introduzidas ferramentas de paginação com recurso às palavras-chave *limit* e *offset* (onde o *limit* representa o número de objetos que irá aparecer numa página e *offset* a página que queremos devolver). Foi, ainda introduzido o recurso à ordenação, através das palavras-chave *sort_by* e *order_by* (onde *sort_by* diz respeito ao campo pelo qual queremos ordenar os objetos e *order_by* especifica o sentido de ordenação ou seja decrescente ou crescente). Nos campos *limit* e *offset* são utilizados valores *default* caso estes não sejam introduzidos, 3 e 0 respetivamente. Além disso é ainda disponibilizada a opção *all* no *limit* para que sejam devolvidos toda a coleção em casos que isso seja o pretendido.

- **Definição do tipo de recurso**: De forma a facilitar a criação dos *endpoints*, as entidades da base de dados foram divididas em 3 tipos de recursos:

  a. **Recursos Principais**: Representam os recursos que são considerados independentes, isto é, não dependem do contexto doutro recurso para existir.

  b. **Recursos de Associação**: Representam os recursos gerados a partir da associação entre dois recursos (derivados de tabelas que apresentariam a relação M:M)

  c. **Recursos Subordinados**: Representam os recursos cuja sua existência só faz sentido na presença doutro recurso (por exemplo, o recurso estado, só faz sentido existir dado o contexto dum objeto espacial)

3.5.2 Estrutura

A estrutura da API REST tem como objetivo cobrir cada operação CRUD para cada recurso identificado. No entanto, dependendo dos requisitos da camada de cliente, nomeadamente da aplicação móvel, foi necessário criar *endpoints* adicionais.

No entanto, numa primeira fase foi necessário identificar o tipo de cada uma das entidades do modelo de dados. Dessa forma, deu-se origem ao seguinte mapeamento:

Recurso	**Tipo**	**Subordinado de / Associação entre:**
spatial objects	Principal	-
spatial object types	Principal	-
fuel breaks	Principal	-
fuel break types	Principal	-
areas of interest	Principal	-
area of interest types	Principal	-
topological relations	Associação	spatial objects
topological relation types	Principal	-
periods	Principal	-
period types	Principal	-
spatial objects periods	Associação	spatial objects, periods
states	Subordinado	spatial objects periods
administrative divisions	Principal	-
spatial objects administrative divisions	Associação	spatial objects, administrative divisions
accounts	Principal	-
accounts divisions of interess	Associação	accounts, administrative divisions
account objects of interess	Associação	accounts, spatial objects

entities	Principal	-
contacts	Associação	administrative divisions, entities
spatial objects periods entitites	Associação	spatial objects periods, entities
spatial object entity types	Principal	-
devices	Principal	-
device types	Principal	-
accounts devices	Associação	accounts, devices
roles	Principal	-
accounts roles	Associação	accounts, roles
operations	Principal	-
roles operations	Associação	roles, operations
confirmation tokens	Subordinado	accounts
reset tokens	Subordinado	accounts
organizations	Principal	-
people	Principal	-
volunteers	Principal	-
professionals	Principal	-
organization members	Associação	organizations, people
member types	Principal	-
rewards	Subordinado	volunteers
prohibitions	Subordinado	operations
roles prohibitions	Associação	roles, prohibitions
accounts prohibitions	Associação	accounts, prohibitions
protocols	Principal	-
fuel break types protocols	Associação	fuel break types, protocols
area of interest types protocols	Associação	area of interest types, protocols
questions	Subordinado	protocols
rules	Principal	-
fuel break types rules	Associação	fuel break types, rules
question types	Principal	-
augmented reality types	Principal	-
answers options	Principal	-
questions answers options	Associação	questions, answers options
records	Subordinado	spatial objects periods
answers	Subordinado	records
waypoints	Subordinado	records
media	Principal	-
media types	Principal	-
media questions	Associação	media, questions

questions protocols	Associação	questions, protocols
records protocols	Associação	records, protocols
volunteers records	Associação	volunteers, records
monitorizations	Principal	-
professionals monitorizations	Associação	professionals, monitorizations
events	Principal	-
event types	Principal	-
alerts	Principal	-
interventions	Subordinado	spatial objects periods
intervention types	Principal	-
news	Principal	-
data providers	Principal	-
data provider types	Principal	-
data sources	Subordinado	data providers
data sources types	Principal	-
series	Subordinado	data sources
series types	Principal	-
series raster data	Subordinado	series
series vectorial data	Subordinado	series
products	Subordinado	data sources
spatial objects periods series	Associação	spatial objects periods, series
series relations	Associação	series
series relations types	Principal	-
series variables	Principal	-
units	Principal	-
data types	Principal	-
series periods	Principal	-
app configurations	Principal	-
settings	Principal	-
properties schemas	Principal	-
metadata schemas	Principal	-
data derived	Subordinado	series vectorial data
data groundtruth	Subordinado	series vectorial data
classification model	Principal	-
training data model	Associação	series vectorial data, classification model
model output	Associação	classification model, spatial objects periods

Tabela 3.1: Mapeamento dos Recursos da API

Após associar cada entidade ao seu tipo correspondente, foi definido um modelo a seguir para a criação dos respetivos *endpoints*, tendo em conta cada um dos três tipos de recurso. Além disso, foram ainda implementados outros *endpoints*, com vista a suportarem certos serviços, tais como o serviço de autenticação, bem como a disponibilização de alguns dados estatísticos requeridos pela camada do cliente. De seguida, são enumerados os modelos correspondentes para cada tipo de recurso já apresentado.

Recursos Principais:

- **GET** /*recurso* – Devolve a coleção de objetos do recurso correspondente.
- **GET** /*recurso*/{recurso_id} – Devolve o objeto do respetivo recurso com o identificador recurso_id.
- **POST** /*recurso* – Cria um ou vários objetos para o recurso correspondente.
- **PATCH** /*recurso*/{recurso_id} – Edita o objeto do recurso correspondente com o identificador recurso_id.
- **DELETE** /*recurso*/{recurso_id} – Elimina o objeto do recurso correspondente com o identificador recurso_id.

Recursos de Associação:

- **GET** /*recurso*/{recurso_id}/*associação* – Devolve a coleção de objetos da associação identificada do recurso correspondente com o identificador recurso_id.
- **GET** /*associação*/{associação_id} – Devolve o objeto da respetiva associação com o identificador associação_id.
- **POST** /*associação* – Cria um ou vários objetos da respetiva associação.
- **PATCH** /*associação*/{associação_id} – Edita o objeto da respetiva associação com o identificador associação_id.
- **DELETE** /*associação*/{associação_id} – Elimina o objeto da respetiva associação com o identificador associação_id

Recursos Subordinados:

- **GET** /*recurso*/{recurso_id}/*subordinado* – Devolve a coleção de objetos do respetivo subordinado do recurso correspondente com o identificador recurso_id.
- **GET** recurso/{recurso_id}/*subordinado*/{subordinado_id} – Devolve o objeto do respetivo subordinado com o identificador subordinado_id do recurso correspondente com o identificador recurso_id.

- **POST** */recurso*/{recurso_id}/*subordinado* – Cria um ou vários objetos do respetivo subordinado do recurso correspondente com o identificador recurso_id.

- **PATCH** */recurso*/{recurso_id}/*subordinado*/{subordinado_id} – Edita o objeto do respetivo subordinado com o identificador subordinado_id do recurso correspondente com o identificador recurso_id.

- **DELETE** */recurso*/{recurso_id}/*subordinado*/{subordinado_id} – Elimina o objeto do respetivo subordinado com o identificador subordinado_id do recurso correspondente com o identificador recurso_id.

3.5.3 Integração com a Camada de Cliente

Visto que no âmbito do projeto Floresta Limpa, o Sistema de Informação funcionará de suporte a uma aplicação móvel foi necessário identificar *endpoints* específicos (além dos genéricos já introduzidos) para satisfazer as necessidades da camada do cliente.

Com vista a facilitar o processo de recolha de dados de terrenos, foram criados *endpoints* cuja lógica permite a adição de dados em múltiplas tabelas, reduzindo o número de pedidos que a aplicação móvel faz ao sistema, permitindo melhorar a performance quer da camada do cliente quer da camada do servidor.

Além disso, foram também identificadas situações em que o sistema teria de devolver vários recursos à aplicação, como no caso da obtenção de todas as perguntas e imagens associadas a um protocolo. Dessa forma, em vez de ser devolvido cada recurso separadamente, tornou-se necessária a criação de novos *endpoints*, permitindo assim que vários recursos sejam retornados à aplicação através dum só pedido.

Por fim, foram identificadas funcionalidades associadas à aplicação móvel, que poderão estar disponíveis para outras camadas do cliente, tais como o envio de alertas, bem como do armazenamento de multimédia nos serviços da *Google Cloud Platform*. Assim sendo, foi necessário criar *endpoints*, onde estas operações são realizadas, disponibilizando estas funções para a aplicação móvel e evitando erros noutras camadas do cliente que não as suportem.

Os *endpoints* associados à aplicação móvel procuram seguir a mesma lógica que os anteriormente especificados, focando-se no recurso principal nos casos em que devolve ou realiza operações sobre vários. De forma a facilitar a identificação dos recursos da aplicação móvel estes possuem **/mobileapp** como caminho inicial. Visto que a lista de *endpoints* dedicados à aplicação móvel já apresenta um número considerável de operações, estas são apresentadas e abordadas, brevemente, no apêndice D.

3.5.4 Casos de Uso

Com o objetivo de que o Sistema de Informação possa vir a ser utilizado por diversas camadas de cliente distintas foi necessário, como já foi referido, a disponibilização de

endpoints genéricos para os métodos CRUD. Dessa forma, os casos de uso do Sistema de Informação apresentam uma listagem genérica, onde um agente interage com o sistema:

- O agente consulta um ou vários objetos de um recurso.
- O agente cria um ou mais objetos de um recurso.
- O agente edita um objeto de um recurso.
- O agente elimina um objeto de um recurso.

No entanto, para além disso, foi necessário ter em conta os casos de uso requeridos pela aplicação móvel. Dessa forma, visto que a lista de todos os casos de uso é extensa, estes são listados no apêndice C e neste tópico apenas são demonstrados dois exemplos de casos uso, nomeadamente um dos casos genéricos e outro de um caso específico requerido pela aplicação móvel.

Caso de uso genérico:

Caso de uso: C220
Descrição: Um agente consulta o objeto espacial com o id 1.
Método: GET
Authorization Header: Bearer Token
Endpoint: /spatial_objects/1
Código: 200
Resposta:

```
{
    "spatial_object_id": 1,
    "spatial_object_type_id": 1,
    "spatial_object_area": 1.98280976084,
    "spatial_object_coverage": null,
    "spatial_object_observations": null,
    "spatial_object_limits": {
        "crs": {
            "type": "name",
            "properties": {
                "name": "EPSG:3763"
            }
        },
        "type": "Polygon",
        "coordinates": [
            [
                [
                    -13381.258899999782,
                    52680.664499999955
                ],
                [
```

```
                    -13381.490000000224,
                    52680.61999999918
                ],
                [
                    -13371.620000000112,
                    52649.289999999106
                ],
                (...)
            ]
        ]
    },
    "spatial_object_bounding_box": null,
    "spatial_object_properties": null,
    "spatial_object_metadata": {
        "shapefile": "FGC_0607.shp"
    },
    "created_at": "2022-11-02T12:19:44.054Z",
    "updated_at": "2023-01-09T16:54:29.668Z",
    "deleted_at": null
}
```


Caso de uso específico da aplicação móvel:

Caso de uso: C375
Descrição: A aplicação móvel consulta quais são os protocolos ativos.
Método: GET
Authorization Header: Bearer Token
Endpoint: /mobileapp/protocols
Parâmetros de *Query*: ?protocol_active=true
Código: 200
Resposta:


```
[
    {
        "protocol_id": 3,
        "protocol_label": "RVF",
        "protocol_name": "Protocolo de Rede Viária Florestal",
        "protocol_description": "Protocolo com perguntas sobre a rede viá
    ria florestal",
        "protocol_version": 1,
        "protocol_active": true,
        "protocol_transitions": [
            {
                "id": 30,
                "question_id": 11,
                "navigation": {
```

```
                    "-1": 3
                }
            },
        (...)
        ],
         "protocol_properties": {
            "first_question": 11,
            "last_question": 29
        },
        "protocol_metadata": null,
        "created_at": "2022-12-07T18:41:44.825Z",
        "updated_at": "2022-12-12T11:18:41.955Z",
        "deleted_at": null,
        "questions_protocols": [
        {
            "question_id": 3,
            "question_type_id": 1,
            "rule_id": null,
            "augmented_reality_type_id": null,
            "question_value": "A faixa tem árvores?",
            "question_help": "Veja se observa uma árvore no seu redor",
            "question_skip_if_answer": [
                2,
                9
            ],
            "question_properties": null,
            "question_metadata": null,
            "created_at": "2022-11-02T12:20:34.058Z",
            "updated_at": null,
            "deleted_at": null,
            "questions_answers_options": [
                {
                    "answer_option_id": 2
                },
            (...)
            ],
            "media_questions": [
                {
                    "media_id": 246,
                    "media_type_id": 1,
                    "record_id": null,
                    "event_id": null,
                    "question_id": 3,
                    "media_name": "/protocols/faixa_arvores.jfif",
                    "media_path": "https://storage.googleapis.com/
florestalimpa-374214-imagens/protocols/faixa_arvores.jfif",
                    "media_description": "",
                    "media_time": "2022-12-02T12:04:48.465Z",
                    "media_limits": null,
                    "media_orientation": null,
```

```
                    "media_properties": null,
                    "media_metadata": null,
                    "created_at": "2022-12-02T12:04:47.878Z",
                    "updated_at": null,
                    "deleted_at": null,
                    "media_question_id": 42
                },
                (...)
            ],
        "fuel_break_types_protocols": [
            {
                "fuel_break_type_id": 10
            }
        ]
    },
    (...)]
```


3.6 Conclusões

Este capítulo foi essencial no desenvolvimento desta dissertação, uma vez que permitiu identificar os casos com que o Sistema de Informação se irá deparar, permitindo assim a elaboração dum conjunto de regras a seguir de forma a facilitar o armazenamento de dados bem como a sua consulta, culminando numa resposta mais eficaz por parte do Sistema.

De facto, numa fase inicial a identificação dos atores e dos requisitos funcionais e informacionais permitiu estabelecer as necessidades a que o Sistema teria de corresponder. Dessa forma, foi possível identificar uma arquitetura adequada para a resposta a estas funcionalidades.

De seguida, foi construído um modelo de dados extenso e robusto, que procura corresponder a todas as necessidades dum projeto de monitorização de faixas de gestão de combustível de incêndios. A maior preocupação na elaboração deste modelo recaiu sobre três fases distintas, nomeadamente, o armazenamento dos dados associados às faixas, dos dados recolhidos por utilizadores, bem como dos dados provenientes das mais variadas séries temporais. Uma vez que a heterogeneidade dos dados é enorme, foi necessário preparar o modelo de dados para estes casos, permitindo assim o armazenamento das várias fontes de dados. Além disso, foi fornecido ao modelo um caráter extensível através da criação dum esquema de propriedades em todas as tabelas, permitindo que este corresponda a novas funcionalidades que possam vir a ser posteriormente identificadas associadas a este projeto.

Relativamente à API REST, esta foi desenhada de forma a ser possível que qualquer camada do cliente interaja com esta, consultando e manipulando os dados de cada recurso identificado no modelo de dados. No entanto, uma vez que à medida que esta dissertação

estava a ser elaborada, encontrava-se a ser realizada em paralelo uma camada do cliente, nomeadamente, uma aplicação móvel, foram identificadas funcionalidades que teriam de ser disponibilizadas pela API REST do sistema para interagir com esta. Assim sendo, a API REST apresenta um caráter genérico, possibilitando que qualquer nova camada de cliente desenvolvida interaja rapidamente com esta, bem como um caráter específico, uma vez que está preparada para responder à aplicação móvel duma forma mais rápida e eficaz, melhorando a performance quer da aplicação móvel quer do Sistema de Informação.

Capítulo 4

Implementação

Finalizado o processo de modelação do Sistema de Informação prosseguiu-se para a implementação das funcionalidades referidas. Desta forma, neste capítulo serão detalhados todos os passos relacionados com a implementação do Sistema, que correspondem à versão do mesmo na data da entrega desta dissertação, podendo este sofrer alterações posteriores.

O Sistema de Informação do projeto Floresta Limpa foi desenvolvido recorrendo à *Google Cloud Platform*, sendo que toda a sua lógica foi implementada na *Google App Engine*, alocada nos servidores da Google. Esta camada irá posteriormente comunicar com as camadas de armazenamento situadas na *Google Cloud SQL* e no Cluster situado no Departamento de Informática da FCT NOVA.

Assim sendo, serão demonstradas as técnicas de implementação associadas à lógica da arquitetura do Sistema de Informação, dando destaque à interação do sistema com as camadas de armazenamento, bem como de toda a lógica associada à criação da API Rest.

Seguidamente, são apresentados os vários processos de integração de fontes de dados de informação geográfica, bem como funcionalidades relacionadas com o sistema de alertas, *logging* e, ainda, o recurso ao *Google Earth Engine*.

Por fim, o capítulo encerra com uma descrição da documentação elaborada para o Sistema de Informação, apresentando uma breve conclusão final sobre todo o processo de implementação realizado.

4.1 Base de Dados

De forma a iniciar a implementação do Sistema de Informação foi necessário desenvolver as camadas de armazenamento que foram desenvolvidas sob o PostgreSQL com recurso à sua extensão PostGIS.

Para isso nesta secção são abordados os processos de desenvolvimento do Sistema de Informação com base na interação com as suas bases de dados.

4.1.1 Ligação à base de dados

A ligação à base de dados PostgreSQL é realizada através da ORM Sequelize. O Sequelize disponibiliza a configuração duma *pool* de conexões personalizável, que procura reutilizar conexões sempre que possível. No caso do projeto Floresta Limpa, uma vez que foi necessário repartir a arquitetura pela *Google Cloud Platform* e pelo Cluster privado, foi necessário proceder à inicialização de duas bases de dados conforme a localização destas (Google Cloud SQL ou Cluster privado). Na listagem 4.1 é demonstrada a inicialização associada à Google Cloud SQL.

```
1  db = new Sequelize(process.env.POSTGRES_DB_NAME, process.env.POSTGRES_USER,
       process.env.POSTGRES_PASSWORD, {
2          host: '/cloudsql/' + process.env.POSTGRES_DSN,
3          dialect: 'postgres',
4          minifyAliases: true,
5          operatorsAliases: 0,
6          pool: {
7              max: 5,
8              min: 0,
9              idle: 10000,
10             handleDisconnects:true,
11         },
12         dialectOptions: {
13             socketPath: '/cloudsql/' + process.env.POSTGRES_DSN
14         },
15     });
```

Listagem 4.1: Ligação à base de dados da Google Cloud SQL.

4.1.2 Modelos

Após uma primeira fase de modelação do modelo de dados, foi necessário proceder à sua implementação através da criação de tabelas na base de dados, bem como do respetivo mapeamento entre esta e o servidor. Para tal, este processo dividiu-se em duas fases.

Em primeiro lugar procedeu-se à geração do código SQL através da ferramenta *PostgreSQL DDL* fornecida pelo *StarUML* [51] que permite converter o modelo de dados já apresentado em código SQL, de forma a criar as tabelas e as suas restrições na base de dados PostgreSQL.

De seguida, foi necessário criar os modelos no servidor, que correspondem a classes definidas, com recurso ao Sequelize, que representam uma tabela na base de dados. Estes modelos são responsáveis pelo mapeamento entre a tabela e o servidor, sendo que para tal é necessário definir todos os atributos, bem como as validações do seu *input*,

como por exemplo, a impossibilidade dum atributo vir com valor *null*. Uma vez que a base de dados é constituída por um vasto número de tabelas que necessitam de ser mapeadas individualmente para um modelo, foi identificada e utilizada a ferramenta *sequelize-auto* [49], que consiste num *script* que percorre uma base de dados, identifica as tabelas existentes e realiza o mapeamento das tabelas para os modelos, bem como das associações existentes entre estas.

```
module.exports = function(sequelize, DataTypes) {
  return sequelize.define('spatial_objects', {
    spatial_object_id: {
      autoIncrement: true,
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true
    },
    spatial_object_type_id: {
      type: DataTypes.INTEGER,
      allowNull: false,
      references: {
        model: 'spatial_object_types',
        key: 'spatial_object_type_id'
      }
    },
    (...)
```

Listagem 4.2: Mapeamento da tabela spatial_objects

Além disso, através dos modelos definidos no Sequelize, é possível mapear as relações que existem entre as tabelas da base de dados (N:M, 1:N, 1:1), permitindo assim que o Sequelize realize *queries* de *eager-loading*, indicando que modelo queremos associar ao modelo em que estamos a realizar a *query*. Sendo assim, todas as relações presentes na base de dados foram definidas também pela ferramenta *sequelize-auto-master*, gerando origem a um ficheiro designado *init-models.js*. Na listagem 4.3 são apresentados exemplos das definições dos três tipos de associação referidos.

```
spatial_objects.belongsTo(spatial_object_types,
    { foreignKey: "spatial_object_type_id"});

spatial_objects.hasMany(accounts_objects_of_interest,
    { foreignKey: "spatial_object_id"});

spatial_objects.hasOne(areas_of_interest,
    { foreignKey: "spatial_object_id"});
```

Listagem 4.3: Exemplo de associações da tabela spatial_objects.

Através do mapeamento dos modelos, bem como das associações, é então possível utilizar as funções nativas do Sequelize referente a cada um deles para realizar as operações sobre a base de dados, das quais se destacam:

- **create** - Cria um tuplo na tabela desse modelo.
- **bulkCreate** - Cria múltiplos tuplos na tabela desse modelo.
- **update** - Atualiza um ou mais tuplos na tabela desse modelo.
- **destroy** - Elimina um ou mais tuplos na tabela desse modelo.
- **findAll** - Devolve vários tuplos na tabela desse modelo.
- **findByPk** - Devolve o tuplo correspondente à chave primária indicada na tabela desse modelo.

No entanto, existem casos onde não existem associações pela chave estrangeira, e o *eager-loading* é realizado através das funções disponibilizadas pelo PostGIS, como por exemplo a interseção da localização dos utilizadores com a localização dos eventos. Dessa forma, uma vez que o Sequelize não possui uma relação definida por elas não é possível realizar a operação com uma das funções nativas do modelo. Para isso é realizada uma *raw query* que consiste numa *query* diretamente à base de dados, através da variável **db** (função db.query), obtida, previamente, na ligação à base de dados, tal como exemplificado em 4.1.1.

4.1.3 Índices

No âmbito dos algoritmos de aprendizagem automática do projeto são inseridas no sistema porções de segmentos de faixa denominados *clusters*. Para manter uma relação entre estes e a faixa de gestão de combustível principal é necessário dados estes fragmentos descobrir qual é o fragmento que estes intersetam. Além disso, ao serem registados eventos é preciso descobrir que utilizadores estão perto dessa zona ou têm locais que lhes interessam perto dessa zona de forma a serem notificados.

Uma vez que o sistema de informação irá possuir milhões de objetos espaciais que representam faixas é necessário a criação de índices de forma a lidar com este processo, de forma a apresentar maior performance. Dessa forma, um exemplo de um índice criado no sistema de informação é o seguinte:

```
1 CREATE INDEX ON spatial_objects USING GIST (spatial_object_limits);
```

Listagem 4.4: Índice espacial criado sobre o atributo spatial_object_limits.

4.1.4 Vistas Materializadas

Como já foi descrito ao longo do documento, o projeto Floresta Limpa irá precisar de fornecer uma componente analítica disponibilizando estatísticas sobre os dados das faixas, como por exemplo através da disponibilização do número de faixas intervencionadas ou

não intervencionadas, número de registos por faixas, entre outros que se considerem pertinentes.

Visto que a listagem de todos estes requisitos carece ainda duma decisão final, até ao momento não foi, ainda, criado um modelo de *data warehousing* de forma a armazenar cópias de dados inseridos na base de dados que serão utilizados para gerar estes dados analíticos.

No entanto, visto que poderá ser necessário disponibilizar este modelo de *data warehouse*, começou-se a preparar a base de dados para gerar estes dados. Assim sendo, para os casos já identificados de necessidade analítica foram criadas vistas materializadas de forma a dar a resposta necessária pelo sistema. A criação de vistas materializadas deveu-se ao facto das *queries* a estas serem mais rápidas que as que são realizadas diretamente às tabelas da base de dados e ao facto de não ser necessário disponibilizar os dados estatísticos em tempo real. Dessa forma, definiu-se que os dados destas vistas serão povoados e atualizados diariamente, através duma função definida na lógica do servidor, sendo disponibilizadas assim estatísticas diárias sobre as faixas do sistema.

Na listagem 4.5 é possível consultar a criação duma vista materializada relacionada com as faixas, registos e seus pontos de recolha, bem como protocolos, sendo possível obter dados como o número de registos associados a uma faixa, os pontos de recolha dum registo de forma a gerar um *heatmap*, bem como informações sobre os protocolos utilizados por faixas.

```
1  CREATE MATERIALIZED VIEW datawarehouse.records
2   AS
3   SELECT r.record_id, sop.spatial_object_id, r.protocol_id, w.waypoint_id,
4      r.record_externalid, r.record_name, r.record_notes, r.record_start_time
       ,  r.record_end_time, r.record_cleanliness, r.
       record_is_fuel_break_clear, r.record_limits
5   FROM records r
6      INNER JOIN spatial_objects_periods sop ON r.spatial_object_period_id =
7          sop.spatial_object_period_id
8      LEFT JOIN waypoints w ON w.record_id = r.record_id
9   WHERE r.deleted_at IS NULL AND sop.deleted_at IS NULL AND w.deleted_at IS
       NULL;
```

Listagem 4.5: Exemplo de vista materializada

4.2 Camada do Servidor

A secção seguinte pretende retratar toda a lógica relacionada com a implementação da camada do servidor, nomeadamente, explicando como o processo de recebimento e redirecionamento de pedidos é tratado pelo *App Engine*, bem como todo o processo de elaboração da API REST.

Desta forma, é exemplificado todo o processo de criação de *endpoints*, bem como da lógica para onde estes pedidos são redirecionados, demonstrando os vários passos por

onde um pedido passa até a sua resposta ser devolvida à camada do cliente.

4.2.1 App Engine

Um pedido enviado ao servidor cuja lógica está alocada no *Google App Engine* passa por diversas fases. Assim sendo, na figura 4.1 podem ser visualizadas estas fases, que serão descritas de seguida.

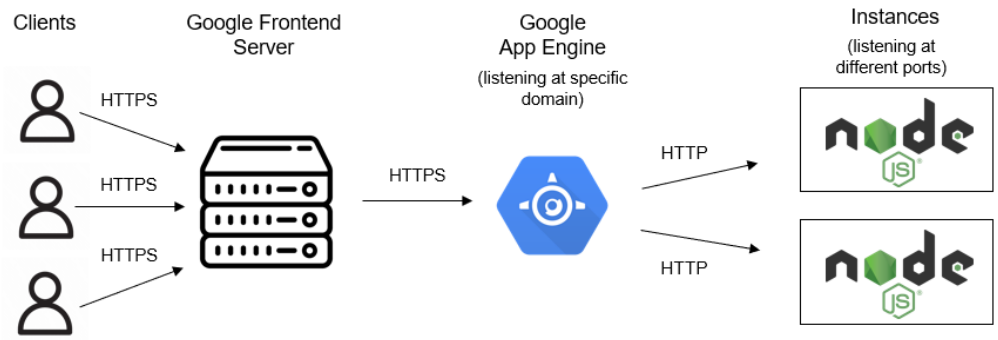


Figura 4.1: Lógica do Envio de Pedidos com HTTPS.

De facto, quando um pedido é enviado para o Sistema de Informação, este é numa primeira fase recebido pelo *Google Frontend Server* (GFE) que é um servidor da Google que atua como *reverse proxy*. Este servidor permite que os pedidos realizados pelos utilizadores sejam recebidos pelo centro de dados da Google mais perto da sua localização.

Após esta fase, o servidor *proxy* irá reencaminhar o pedido para o centro de dados onde a aplicação implementada no *Google App Engine* está localizada. No entanto, a Google dá a liberdade que os pedidos enviados para o Google App Engine sejam realizados quer em HTTP quer em HTTPS, pois fornece um certificado SSL grátis. Visto que no caso do nosso servidor pretende-se que apenas pedidos HTTPS cheguem ao servidor, foi necessário definir no ficheiro de configuração app.yaml que todos os pedidos que fossem realizados pelo cliente com HTTP fossem redirecionados para HTTPS de forma a garantir encriptação dos dados enviados nos pedidos. Assim sendo, caso o pedido do cliente seja no protocolo HTTP, este será intercetado pelo GFE e será devolvido ao cliente, uma resposta redirecionada a dizer que o pedido deverá ser realizado na forma de HTTPS. Ao receber esta resposta, o pedido é automaticamente retornado pelo cliente na forma de HTTPS, sendo enviado para o GFE que o recebe na porta 443, prosseguindo, assim, o restante fluxo. Na listagem 4.6 pode ser verificada a alteração realizada ao ficheiro para que os pedidos HTTPS fossem forçados.

```
handlers:
  - url: /.*
    secure: always
    redirect_http_response_code: 307
    script: auto
```

Listagem 4.6: Redirecionamento HTTPS no ficheiro app.yaml

De seguida, os pedidos chegam, portanto, ao *Google App Engine*, que irá dividir o processo em duas fases, tendo em conta a arquitetura que foi explicada em 2.5.4. Primeiramente, o pedido chega ao serviço correspondente, que no caso do servidor do projeto Floresta Limpa é apenas um denominado *default service*, uma vez que a lógica da aplicação foi toda desenvolvida em apenas um serviço. Este serviço irá então possuir as várias versões que já foram implementadas desta aplicação. No caso do Sistema de Informação do Floresta Limpa apenas é acedida uma destas versões que é considerada a versão atual do servidor. Esta versão atual do serviço *default* irá dar origem a várias instâncias onde a lógica da aplicação está situada. Desta forma o GAE *Frontend* atua como balanceador de carga, uma vez que permite que os pedidos sejam distribuídos por várias instâncias. Assim sendo, face a um aumento do número de pedidos pode ser necessário disponibilizar mais instâncias, cuja Google trata automaticamente, fornecendo assim escalabilidade horizontal sem que nos tenhamos que preocupar com isso. Além disso, caso ocorra uma diminuição do número de pedidos, o processo inverso também ocorre garantindo que não estamos a utilizar mais recursos do que os que necessitamos.

Por fim, os pedidos chegam então à última fase do GAE, denominado *Google App Engine Backend*, onde estão situadas as instâncias, onde como já foi dito, os pedidos irão ser recebidos e processados pela lógica implementada na aplicação, podendo aceder ao serviços da camada de armazenamento. Estes pedidos são enviados pelo *Frontend* do GAE através HTTP, uma vez que se trata dum envio de pedidos na rede da Google, onde não acontece exposição para a rede pública, não sendo, assim, necessário o envio segundo HTTPS. Após toda a lógica aplicada, a resposta é enviada ao *Google Frontend*, onde será encriptada através de HTTPS, e posteriormente reencaminhada para o cliente.

4.2.2 Routing

Todos os pedidos realizados ao servidor foram definidos recorrendo à função *app.use()* e à classe Router disponibilizadas pelo Express. Numa primeira fase foram definidos os caminhos iniciais para cada um dos tipos de recurso enumerados em 3.5.1, redirecionando-os para o *middleware* correspondente, onde a classe Router irá tratar do que irá ser realizado em cada *endpoint*. Em cada um dos endpoints, o servidor irá começar por verificar o *token* de autorização (explicado posteriormente em 4.2.3, seguindo-se das permissões do utilizador, de forma a verificar se o utilizador está autorizado a realizar o pedido que deseja. Além disso, em alguns casos, verifica também se o utilizador foi o criador ou tem permissões de posse sobre o recurso que deseja alterar/obter. Caso estas condições se verifiquem

o servidor irá então realizar toda a lógica associada ao pedido. Para além dos *endpoints* relativamente aos diversos tipos de recurso foi também realizado o mapeamento através do caminho "/mobileapp"para todos os *endpoints* definidos em exclusivo para a aplicação móvel.

Nas seguintes listagens está representado um exemplo do código da lógica de *routing* associado ao recurso *accounts*.

```
1 app.use('/accounts', require('./routes/accounts_routes'))
```

Listagem 4.7: Redirecionamento do routing para o middleware correspondente.

```
1 router.get('/:id', authToken, checkRole, checkOwnership, controller.getById
      )
```

Listagem 4.8: Lógica do routing realizado pela classe Router.

4.2.3 Autenticação, Autorização e Permissões

O acesso à lógica dos *endpoints* da API REST é dividido em duas fases, nomeadamente, autenticação e autorização.

Para isso numa primeira fase, um utilizador tem de se autenticar perante o Sistema de Informação. Para tal, são disponibilizados três *endpoints* com esse efeito, nomeadamente, /token e /auth para o acesso genérico de qualquer camada do cliente ao sistema e /mobileapp/authentication para a autenticação específica da app móvel. O primeiro *endpoint*, aceita a criação de qualquer conta, através da definição da password e email do utilizador, no entanto, os outros dois *endpoint*s utilizam a Google como provedor de autenticação, recorrendo portanto ao *OAuth*, sendo recomendado que as contas que acedem ao sistema sejam do domínio da Google de forma a terem disponíveis todas as funcionalidades disponibilizadas pelos seus serviços.

Quanto ao método chamado em /token, este para além de processar o email e a password, é o método responsável pela criação do *token* de acesso ao Sistema de Informação, devolvendo-o após uma autenticação bem sucedida. Relativamente aos *endpoints* que utilizam a Google como provedor, no caso genérico o utilizador é redirecionado para uma página browser onde pode fazer o login com a sua conta Google, enquanto que no *endpoint* da aplicação móvel, esta lógica é tratada pela aplicação e apenas é enviado ao sistema o código de autorização retornado pelo login com sucesso à sua conta Google, permitindo portanto identificar esta conta perante o Sistema. De seguida, ambos os *endpoints* recorrem ao método que é executado pelo *endpoint* /token, onde a lógica de autenticação do utilizador é ignorada, uma vez que já foi realizada pelos *endpoints* que se autenticam pela Google e é realizada toda a lógica associada à obtenção do *token* de acesso ao Sistema de Informação. Na listagem 4.9 está representada a lógica da autenticação dum utilizador perante o *firebase*, após obter o código de autorização.

```
const authApp = async (req, res) => {
    const authorization_token = req.body.code;
    //Verificação da versão mínima da aplicação
    (...)

    try {
        //Obtém o payload com o access token através do authorization token
        const response = await utils.get_access_token(authorization_token);

        //Obtém o access token através do payload
        const {access_token, id_token, refresh_token} = response.data;

        //Autenticação no firebase (regista o utilizador, ou insere o seu login mais recente)
        await firebaseAuthentication(res, id_token)

        //Obtém informação sobre o perfil do utilizador na Google.
        const user = await utils.get_profile_data(access_token);

        //Define variáveis a serem enviadas no req para a função handleOauth
        req.user_data = user.data
        (...)

        //Função responsável por processar um utilizador após autenticação num provedor da Google
        //Verifica se já existe na base de dados e cria informação sobre este se for o primeiro login.
        await handleOauth(req, res)
    } catch (error) {
        console.log (error);
        res.sendStatus (500);
    }
};
```

Listagem 4.9: Lógica do *endpoint* /mobileapp/authentication.

Após a autenticação com a Google é ainda processada informação sobre se se trata dum utilizador novo ou dum utilizador antigo adicionando informação sobre a conta, bem como sobre o dispositivo que o está a aceder ao sistema.

De seguida, após a fase de *OAuth*, de forma a que a autenticação perante a Google não tenha de ser constantemente realizada, foi definido que cada pedido terá de ter no seu *header* um *token* de autorização (*Bearer token*), gerado pelo Sistema de Informação após um *login* com sucesso de forma a garantir a autenticação perante o Sistema.

Assim sendo, para esse efeito foi criada uma função que se dedica à criação do *token* que será enviado após um *login* com sucesso. Este é um JSON Web Token, gerado a partir do algoritmo HS256, válido por 24h, cuja lógica da sua criação pode ser consultada na listagem 4.10.

```
1  const token = async (req, res) => {
2      // Parse dos parâmetros e verifica se a conta já existe na base de
       dados.
3      (...)
4
5      // Realiza a autenticação caso o utilizador não tenha recorrido a um mé
       todo Oauth
6      if(req.oauth !== true) {
7          const is_password_valid = await bcrypt.compare(user.
       account_password, account.account_password)
8          if (!is_password_valid)
9              res.status(401).send('Credenciais inválidas')
10     }
11
12     //Obtenção da role e proibições do utilizador.
13     (...)
14
15     jwt.sign({
16             account: {
17                 account_id: account.account_id,
18                 account_username: account.account_username,
19                 entity_id : account.entity_id
20             },
21             roles: roles_acc,
22             access_token: req.access_token,
23             prohibitions: prohibitions
24         }, secretKey, {expiresIn: '24h'}, (err, token) => {
25             res.json({
26                 token,
27                 prohibitions
28             })
29         })
30
31     //Obtenção e armazenamento de informação do dispositivo que está a
       aceder ao Sistema.
32     (...)
33 }
```

Listagem 4.10: Lógica da criação do token

Por fim, tal como demonstrado em 4.2.2, são utilizadas as funções *authenticateToken*, *checkRole* e em alguns casos *checkOwnership* de forma a verificar se um utilizador pode realizar um pedido. No primeiro caso, é efetuada a validação do *token*, sendo obtidas as permissões e proibições do utilizador. De seguida, estas permissões são verificadas na função seguinte, destinada a ver se o utilizador tem uma função que lhe permite aceder àquele *endpoint*. Opcionalmente, em *endpoints* que apenas podem ser acedidos se o utilizador tem alguma posse sobre o recurso, é efetuada a validação dessa posse. Além disso na lógica dos *endpoints* com funcionalidades restritas são, posteriormente, verificadas as proibições do utilizador, de forma a verificar este se encontra impossibilitado de realizar

alguma operação.

Na listagem 4.11 é apresentada a lógica associada à validação do token.

```
function authenticateToken(req, res, next) {
    const authHeader = req.headers['authorization']
    (...)

    if (token == null) return res.sendStatus(401)

    // Verifica a validade do token
    jwt.verify(token, secretKey, (err, user) => {
        if (err) return res.sendStatus(403)
        else {
            //Verificação com sucesso e definição das variáveis úteis para
    os pedidos.
            req.user_id = user.account.account_id
            (...)
            next()
        }
    })
}
```

Listagem 4.11: Lógica da validação do token

4.2.4 Controladores

Uma vez criados os modelos e o respetivo *routing* foi necessário implementar a lógica responsável pelas operações executadas sobre os diversos recursos, dando origem aos controladores.

Os controladores têm, assim, a função de manipular as operações que serão realizadas através de cada *endpoint* da API REST na base de dados. Para isso, servem-se dos modelos definidos pelo Sequelize sobre cada uma das tabelas e aplicam as funções, disponibilizadas pela ORM, sobre estes modelos, dando origem às queries SQL que serão aplicadas na base de dados.

Quando um pedido chega ao servidor, este é reencaminhado pelo serviço de routing (explicado em 4.2.2) para o controlador correspondente, bem como para a função especificada. Após isso, o controlador interpreta os parâmetros recebidos, valida-os, executa a lógica referente à função e devolve, por fim, a resposta para o cliente.

Visto que o servidor terá de ter a capacidade de integrar uma aplicação móvel, mas também outras camadas de cliente que possam vir a ser implementadas, foi tomada a decisão de dividir o processo de criação dos controladores em duas fases.

Numa primeira fase, foram criados três controladores tipo, para cada um dos tipos de recurso identificados em 3.5.1, disponibilizando, em cada um, as respetivas operações de CRUD. No caso das operações de consulta optou-se pela filosofia de *lazy-loading*, de forma a reduzir o *payload* desnecessário devolvido por estas respostas. Desta forma, caso

uma camada de cliente nova seja implementada, numa primeira fase pode usufruir destes *endpoints* de forma eficaz. De seguida, foi gerado um *script* em python com o objetivo de a partir destes três controladores dar origem aos restantes para cada recurso. Para isso, foi feito um mapeamento entre cada recurso e o seu tipo e foram gerados ficheiros a partir dos três controladores tipo, onde o modelo utilizado era substituído pelo modelo a que cada controlador se referia.

No entanto, tendo em conta que no decorrer da implementação do sistema de informação, estava a ocorrer em paralelo o desenvolvimento da camada de cliente referente à aplicação móvel foram identificadas situações que levaram à criação de novas funções em alguns controladores específicos. Estas funções foram implementadas com o intuito de reduzir o número de pedidos que a aplicação móvel teria de realizar ao servidor, recorrendo ao *eager-loading* em algumas funções de consulta, como por exemplo, na obtenção de questões e imagens associadas a um protocolo específico cuja lógica é apresenta na listagem 4.12.

```
1  const getAllApp = async (req,res) => {
2    //Parse dos parâmetros do pedido
3    (...)
4      await protocols.findAll({
5      //Parâmetros responsáveis pelo eager-loading
6          include: [{
7              model: models.questions_protocols,
8              attributes: ["question_id"],
9              required: false,
10             include: [{
11                 model: models.questions,
12                 required: false,
13                 (...)
```

Listagem 4.12: Exemplo de eager-loading no controlador dos protocolos

Ademais foram também criadas funções para permitir o controlo de situações de inserção de tuplos nas tabelas através do uso de transações, uma vez que poderiam surgir estados inconsistentes caso as inserções fossem realizadas de forma individual, como por exemplo, na inserção das respostas, imagens e informações relativas a um registo. Desta forma, garante-se que se ocorre um erro numa inserção destas tabelas, as restantes são também invalidadas, exemplificadas em 4.13.

```
const addFullApp = async (req,res) => {
  //Parse dos parâmetros do pedido
  (...)
        // Criação da transação
        const t = await db.transaction()
            try {
                const record = await records.create(obj_create,
                    {
                        silent: true,
                        //Definição da transação onde a query corre.
                        transaction: t
                    })
                    waypoints_added = models.waypoints.bulkCreate(waypoints
    , {
                        transaction: t
                    })
                }
                //Restante lógica de inserção nas tabelas
                (...)

                await t.commit();
            } catch (error) {
                await t.rollback()
            }
            (...)
```

Listagem 4.13: Exemplo de uma transação no controlador dos records

4.3 Integração de Fontes de Dados de Informação Geográfica

Tal como tem vindo a ser referido ao longo desta dissertação, o Sistema de Informação foi concebido de forma a receber dados de vários formatos distintos, pelo que foi necessário criar processos que processem os dados das variadas fontes, de forma a ser possível a sua inclusão na base de dados. Para isso, neste tópico serão abordados os processos elaborados, tendo em conta o tipo de dados inserido no sistema.

4.3.1 Scripts Python

Um dos processos adoptados para a integração de fontes de dados de informação geográfica foi a utilização de *Scripts* em Python com o objetivo de realizar o pré-processamento dos dados de forma a que fosse possível a sua inclusão genérica na base de dados. Este processo destaca-se pela correção de erros nos dados *raw*, alterando tipos de certos atributos que vêm incorretos, fazendo a seleção dos atributos considerados importantes para o modelo. Além disso, mapeia estes dados para as tabelas do modelo de dados, realizando posteriormente a sua integração na base de dados. Estes *scripts* estão programados para

lidar, quer com dados vetoriais, quer com dados *rasters*, tendo originado vários processos que irão ser descritos de seguida.

4.3.1.1 Faixas de Gestão de Combustível

As faixas de gestão de combustível são a fonte de dados mais importante deste projeto e também a mais numerosa, pelo que tornou-se necessário facilitar a sua integração no Sistema de Informação para disponibilizar toda a informação que necessita de ser consultada para estas. Este processo é exemplificado no diagrama de atividades representado em 4.2, que será especificado em baixo.

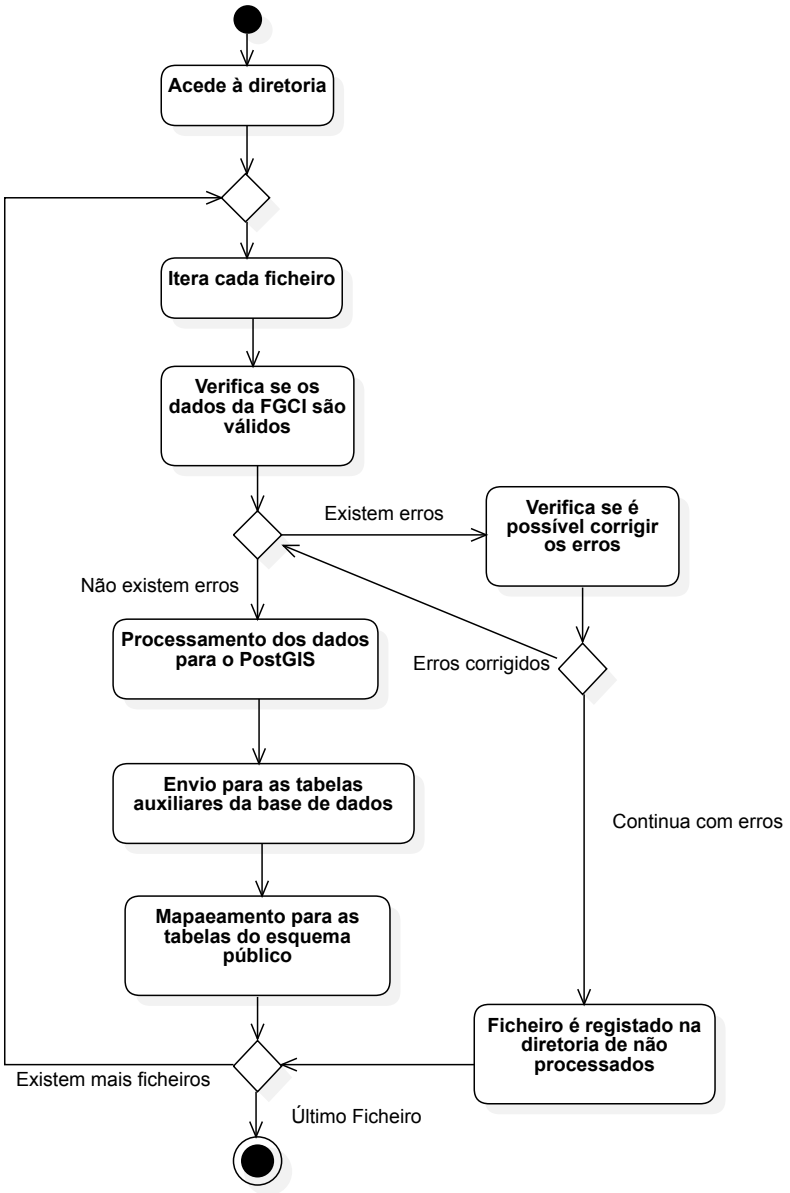


Figura 4.2: Diagrama de atividades da inserção das FGCI.

Os dados das faixas de gestão de combustível são fornecidos, sobretudo, a partir do

Plano Municipal de Defesa da Floresta contra Incêndios (PMDFCI) através de ficheiros vetoriais (*Shapefiles*), cuja sua inserção está dependente de todas as Câmaras Municipais do país, sendo que adicionalmente podem ser fornecidas informações sobre outros locais de importância (Áreas de Interesse a Monitorizar) que cada uma pretende monitorizar.

Ao longo do processo de integração foram identificados dados em alguns ficheiros vetoriais, cujos atributos não seguiam o definido no PMDFCI, apresentando tipos distintos do que os esperados. Assim sendo, após os ficheiros com dados *raw* sobre as faixas a inserir serem colocados numa diretoria, são inicializados os *scripts* com vista ao pré-processamento dos dados. Numa primeira fase são analisados os ficheiros vetoriais, convertidos para um *Geopandas DataFrame* de forma a verificar se algum destes possui erros em relação aos formatos esperados. Os ficheiros são assim divididos entre ficheiros prontos a serem enviados à próxima fase e ficheiros com erros.

Desta forma, os ficheiros com erros vão passar por uma fase extra onde o *script* irá tentar identificar os erros e corrigi-los. Esta correção baseia-se nos erros que foram apontados ao longo do processo de integração e visualização dos dados, nomeadamente, nomes de atributos incorretos, bem como o seu conteúdo, que serão alterados para os definidos no PMDFCI, caso seja possível através dos dados fornecidos e tipos associados aos atributos incorretos, que serão convertidos para os tipos indicados no plano. No entanto, caso não seja possível identificar as correções a partir dos dados *raw*, estes serão separados do processo, sendo necessário uma análise manual ou o contacto com a Câmara Municipal correspondente para que seja realizada a posterior correção dos dados com vista à sua inserção. Após esta fase, os dados são considerados corretamente processados e segue-se para o próximo passo.

De seguida, os dados sofrem ainda uma outra fase de processamento onde as sua informação georreferenciada é transformada numa coluna de geometria pronta a ser armazenada pelo PostgreSQL com recurso ao PostGIS. Seguidamente, a informação sobre cada faixa é enviada para um esquema de tabelas auxiliares, da base de dados situada na *Google Cloud SQL* de forma a facilitar e aumentar a eficácia do processo de inserção na base de dados. Após a inserção nas tabelas auxiliares, é realizado o processo de mapeamento da informação de cada ficheiro vetorial para as tabelas do esquema público do modelo. Caso ocorra algum erro neste processo, o processo de inserção neste esquema sofre *rollback*, podendo ser iniciado a partir do ponto anterior quando o erro identificado estiver resolvido.

Na listagem apresentada em 4.14 é possível consultar uma porção do código que é executado para a realização da importação das faixas, onde é possível verificar a utilização de *multiprocessing*, uma vez que permite realizar a inserção das várias faixas em processos que atuam paralelamente, mostrando um aumento de desempenho na sua importação, como irá ser comprovado, posteriormente, no capítulo das avaliações.

```
def import_fuels_thread(dir, filename):
    # Conexão à base de dados da Google Cloud SQL
    conn, cur = connect_gcp()
    try:
        # Transformação do shapefile em geopandas
        path = os.path.join(dir, filename)
        file = gpd.read_file(path)

        table_name = filename.split(".shp")[0].replace(" ", "_").lower()

        # Alterações aos dados dos atributos das faixas
        changes_to_fgc(file)

        # Envio dos dados para a tabela auxiliar da base de dados
        to_postgis_using_psycopg2(file, conn, cur, table_name, False,
    schema, geom_name="geom")

        # Criação dos objetos espaciais e associações à sua divisão
    administrativa
        import_fuel_breaks_model(conn, cur, schema + "." + table_name)

        # Importação da informação sobre as faixas associadas aos objetos
    espaciais na tabela das faixas.
        import_fuel_breaks_specs(conn, cur, schema + "." + table_name,
    filename)

        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        conn.rollback()
    close_connection(conn, cur)

# Função que lança um sub-processo para cada ficheiro shapefile da faixa,
    permitindo paralelismo
def import_fuel_shapefiles(dir):
    # Itera os ficheiros na diretoria das faixas
    for filename in os.listdir(dir):
        if filename.endswith('.shp'):
            multiprocessing.Process(target=import_fuels_thread, args=(dir,
    filename,)).start()
```

Listagem 4.14: Script de inserção de FGCI

Finalizado o processo de inserção das faixas no Sistema de Informação, estas têm que ser geradas e enviadas para a aplicação móvel. Para isso, visto que a aplicação móvel utiliza como ferramenta de apresentação de dados georreferenciados, o MapBox, as faixas inseridas no sistema são, novamente, processadas para ficheiros *Shapefiles* através do QGIS, sendo posteriormente enviados para o MapBox realizar a sua inserção na aplicação móvel.

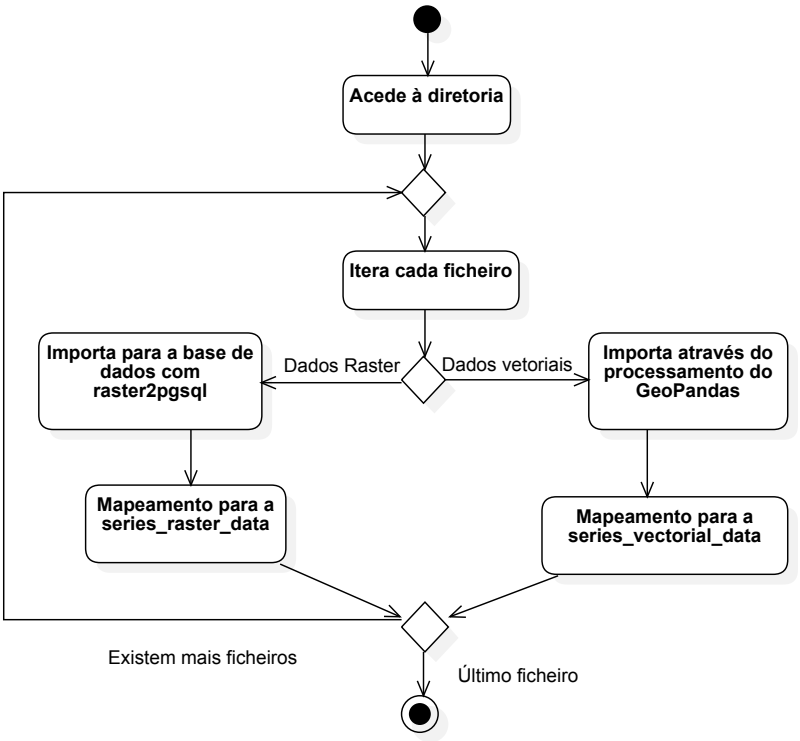


Figura 4.3: Diagrama de atividades da integração dos dados de séries temporais.

4.3.1.2 Séries Temporais

As séries temporais correspondem a informações obtidas ao longo do tempo, a partir doutras fontes de dados, que irão ser utilizadas, sobretudo para a classificação das faixas com recurso aos algoritmos de aprendizagem automática, sendo portanto armazenadas na base de dados presente no Cluster do DI. Estas fontes de dados podem tanto ser ficheiros vetoriais, tais como *shapefiles*, como ficheiros *raster*, donde se destacam os formatos GeoTiff, JPEG2000, NetCDF, bem como HDF. Devido à heterogeneidade dos dados foi, então, necessário estabelecer *scripts* que processassem os dados dos mais variados tipos. O processo de inserção de dados das séries temporais divide-se então em dois sub-processos consoante o tipo de dados que se deseja inserir (vetorial ou *raster*). Estes processos são exemplificados através do diagrama de atividades representado em 4.3 e serão descritos de seguida.

No caso dos dados vetoriais, o processo é semelhante ao especificado em 4.3.1.1, onde os dados do ficheiro vetorial são enviados para um esquema de tabela auxiliares e são, posteriormente, mapeados para os atributos especificados no modelo de dados, sendo que neste caso são inseridos na tabela series_vectorial_data. No caso desta tabela, apenas é registado um valor principal que pode ser um número ou um valor texto, no entanto, é possível registar outros valores que sejam igualmente considerados importantes, através do atributo series_vectorial_data_properties.

```
# Função que é lançada pelo processo responsável pelas áreas ardidas
def import_burn_areas(filename, product_name, filepath, year, data_variable
    ):
    conn, cur = connect_gcp()
    try:
        # Importa os dados vetoriais para a tabela auxiliar
        import_shapefiles(conn, cur, filename)

        # Mapeia a informação dos produtos e dos dados vetoriais para as
    respetivas tabelas
        import_products_general_with_data(cur, product_name, filename,
    filepath, year, data_variable, schema + '.' + filename)
        conn.commit()
    except (Exception, psycopg2.DatabaseError):
        conn.rollback()
    close_connection(conn, cur)
```

Listagem 4.15: Script de inserção de Dados Vetoriais (Áreas Ardidas)

Relativamente, à integração dos dados raster, esta difere na forma em que os dados são processados, uma vez que já não podem ser interpretados pelo Python com o Geopandas. Assim sendo, para realizar a integração dos *rasters* na base de dados utilizou-se a ferramenta *raster2pgsql* disponibilizada pelo PostgreSQL + PostGIS. Esta ferramenta realiza a transformação dos dados *raster* em dados suportados pelas tabelas do PostGIS, no entanto a nova versão desta ferramenta apenas é utilizada pela Shell, pelo que o *script* executa o comando da shell, de forma a que esta se encarregue do processo de envio dos *rasters* para o esquema de tabelas auxiliares.

No âmbito do sistema de Informação desenvolvido nesta dissertação, os *rasters* podem ser inseridos no sistema de duas formas, nomeadamente, através da inserção total do *raster* nestas tabelas auxiliares (apenas recomendado para *rasters* de dimensões reduzidas), ou então com recurso a *out-db rasters*. Estes últimos consistem na inserção do conteúdo dos *rasters* no sistema de ficheiros, como por exemplo a GCS, sendo que o que vai ser introduzido das tabelas é apenas a meta-informação sobre o *raster* e um apontador para a sua localização nesse sistema de ficheiros.

Dessa forma, no caso da inserção do *raster* completo na tabela, os dados podem, então, ser mapeados da tabela auxiliar para a tabela series_raster_data, colocando toda a informação relativa ao *raster* nesta, sendo possível este seja dividido em vários sub-rasters integrados em vários tuplos nesta tabela, uma vez que o PostGIS tem um tamanho máximo de suporte de um *raster* por tuplo. Quanto aos *out-db rasters*, estes são, previamente, enviados para um *bucket* no GCS ou outro sistema de ficheiros escolhido, sendo depois mapeada toda a metainformação gerada nas tabelas auxiliares para a tabela series_raster_data.

```
def call_process_raster(file):
    # Obtém as credenciais da conta de serviço do sistema
    (...)

    #Acede ao bucket dos rasters onde irá criar um blob para o raster a adicionar
    client = storage.Client(credentials=credentials)
    bucket = client.get_bucket(os.getenv('RASTER_BUCKET'))
    blob = bucket.blob(file)
    blob.upload_from_filename(file)

    # Obtém um URL assinado do blob criado para enviar para o raster2pgsql
    url = blob.generate_signed_url(
        # This URL is valid for 15 minutes
        expiration=timedelta(minutes=15),
        # Allow GET requests using this URL.
        method="GET",
    )

    #Realiza o processo de enviar a metainformação (out-db raster) para a base de dados do Google Cloud SQL
    subprocess.call(f'raster2pgsql -s 4326 -I -R -F "/vsicurl/{url}" aux_tables.{file[:-4]} | psql -h {os.getenv("POSTGRES_HOST")} -U {os.getenv("POSTGRES_USER")} -d {os.getenv("POSTGRES_DB_NAME")} -p 5432',
    shell=True, stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)

    conn, cur = connect_gcp()
    try:
        #Converte os dados da tabela auxiliar para a tabela series_raster_data
        handle_raster_info(file, cur)

        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        conn.rollback()
    close_connection(conn, cur)

def import_raster(dir_rasters):
    rasters = os.listdir(dir_rasters)
    for file in enumerate(rasters):
        # Lança uma thread por cada raster que necessita de enviar para o Sistema
        t = threading.Thread(target=call_process_raster, args=(file,)).start()
```

Listagem 4.16: Script de inserção de Dados Raster

Posteriormente, os dados vetoriais e raster das séries temporais podem ser consultados em ferramentas de visualização de dados, tais como o QGIS, onde é possível realizar a filtragem das fontes de dados presentes nesta tabela, quer através da série que se pretenda

consultar, quer do produto que deseja observar.

4.3.1.3 Inserção da Informação sobre as Fontes

De forma a garantir que não existam erros de integração de novos dados nestas tabelas que irão ter milhões de tuplos, foi necessário desenvolver um processo de criação das novas fontes de dados.

Para isso, este processo dividiu-se nas seguintes fases:

1. Inserção da informação dos fornecedores (caso este não esteja já registado), fontes de dados, séries temporais e seus produtos através dos *endpoints* da API REST para esse efeito.

2. Implementar o *script* referente ao mapeamento dos atributos da fonte de dados para os atributos do modelo do Sistema de Informação.

3. Adicionar este *script* ao *script* principal responsável por executar os processos de integração de dados.

4.3.1.4 Modelos e Dados de Aprendizagem Automática

No âmbito da classificação das FGCI, tal como já foi referido, são utilizados mecanismos de aprendizagem automática. Para isso, tornou-se necessário realizar a integração dos dados associados aos modelos de classificação.

Os algoritmos de aprendizagem automática utilizam os dados oriundos das séries temporais, cujo processo de integração foi descrito em 4.3.1.2. Através destes dados são originados, para os processos de classificação, dados derivados que irão ser necessários para treinar o modelo. Além disso, dentro destes dados são selecionados conjuntos que irão ser marcados como *groundtruth*.

Assim sendo, o processo de integração dos dados de aprendizagem automática tem como objetivo armazenar os modelos utilizados para classificar as faixas, bem como a informação sobre os dados utilizados para o seu desenvolvimento e, também, os seus resultados.

Por fim, associados a estes modelos serão criados novos objetos espaciais denominados *clusters* com forma a registar os segmentos das faixas que estão a ser avaliados. Dessa forma, no processo de integração destes dados, para além de serem armazenados estes segmentos de faixa, é ainda realizado um processo de criação das relações topológicas destes com as faixas já inseridas no sistema. Dessa forma, ao obter informação sobre estes segmentos de faixas é possível descobrir com facilidade informações sobre a faixa em si.

Em suma, este processo pode ser dividido em três sub-processos, nomeadamente, a inserção da informação do modelo e dos dados derivados necessários ao seu treino, a criação dos *clusters* e suas relações topológicas, bem como a integração dos resultados originados pela classificação dos modelos.

4.3.1.5 Protocolos

A inserção de protocolos foi um processo trabalhoso e essencial para a disponibilização das perguntas e opções de respostas para a realização dos registos por parte dos utilizadores, tendo sido dividido em várias fases.

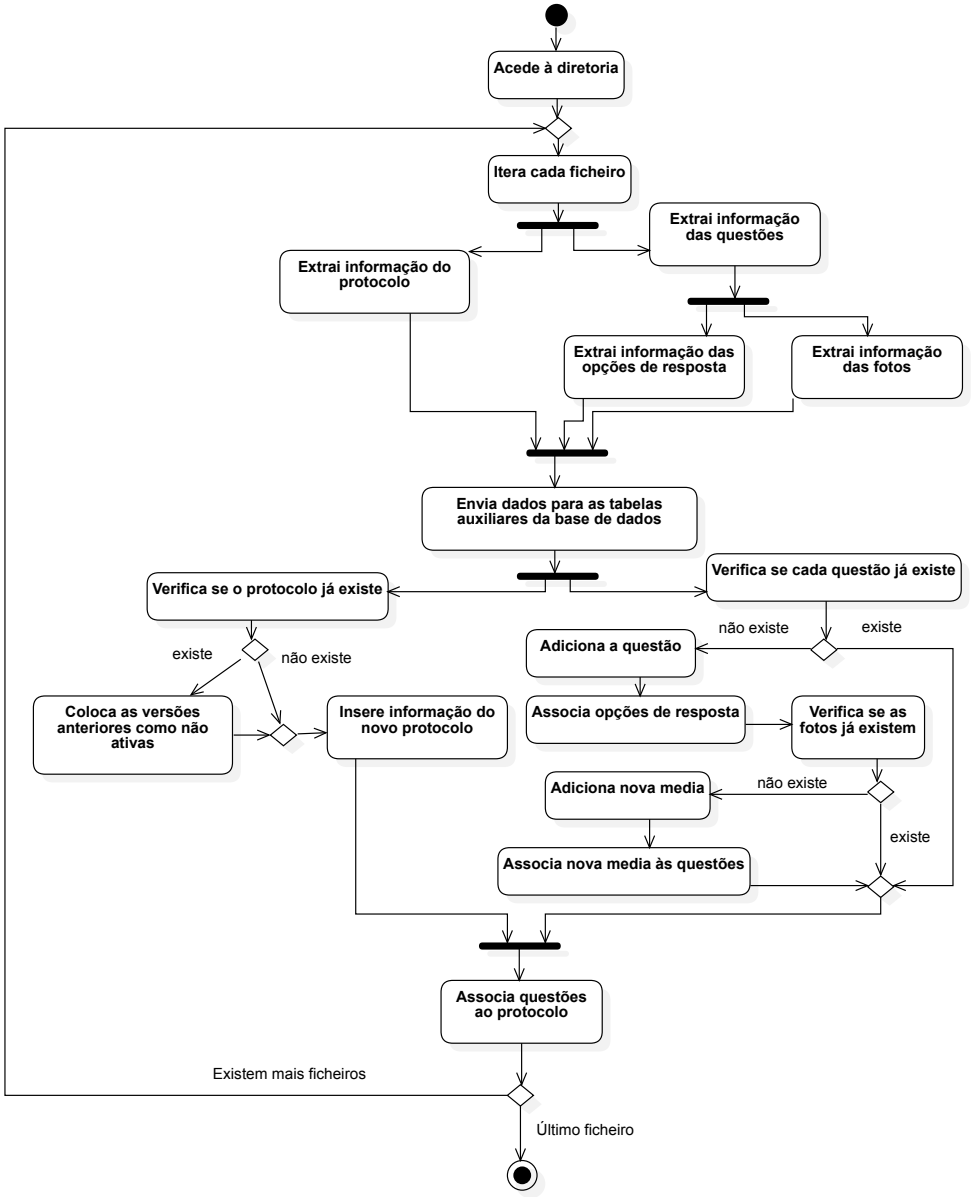


Figura 4.4: Diagrama de atividades da inserção dos protocolos.

A integração dos protocolos no Sistema de Informação é mais uma vez realizada com recurso a *scripts python*. Para isso, o script responsável por esta integração, acede à diretoria correspondente aos protocolos novos, que são ficheiros JSON, devidamente formatados para que o *script* os consiga processar. Para cada um destes protocolos, numa primeira fase irão ser processados com recurso ao *pandas*, dando o ficheiro principal origem a quatro *dataframes* distintos, nomeadamente, informação dos protocolos, multimédia, questões e

opções de resposta. Após a extração destes dados, cada um destes *dataframes* é enviado para o esquema de tabelas auxiliares na base de dados de forma a que se realize as suas inserções. Na listagem 4.17 está representada parte do código de processamento realizado para estes efeitos:

```
def process_protocol_thread(filepath):
    conn, cur = connect_gcp()
    try:
        # Processa a informação do JSON
        with open(filepath, encoding='utf-8') as f:
            data = json.loads(f.read())

        # Realiza a normalização do JSON seguindo o path das questões.
        df = pd.json_normalize(data, record_path=['questions'], sep='_')

        df_media = pd.DataFrame()
        df_answers_ops = pd.DataFrame()

        # Obtém a multimédia associada às questões
        for x in range(0, len(df.index)):
            aux_df = pd.DataFrame(df['attached_media'][x])
            aux_df['question_id'] = df['id'][x]
            df_media = pd.concat([df_media, aux_df], ignore_index=True)

        # Obtém as opções de resposta associadas a cada pergunta
        for x in df:
            if 'answer_options' in x:
                aux_df = pd.DataFrame(df[x])
                aux_df['question_id'] = df['id']
                aux_df = aux_df.rename(columns={x: 'answer_option'})
                df_answers_ops = pd.concat([df_answers_ops, aux_df],
    ignore_index=True)
                df = df.drop([x], axis=1)
        df_answers_ops['colFromIndex'] = df_answers_ops.index
        df_answers_ops = df_answers_ops.sort_values(by=['question_id', '
    colFromIndex'])

        # Extrai informação do protocolo
        df_protocols = pd.DataFrame()
        df_protocols = assign_protocol(data)

        # Cria tabelas auxiliares na base de dados
        create_table_from_pd(cur, conn, df_protocols, "protocols","
    aux_tables")
        to_postgis_without_geom(conn, cur, df_protocols, "aux_tables.
    protocols")
        (...)

        //Criação das queries e execução das mesmas para inserir nas
    tabelas principais
```

```
41          (...)
42
43          conn.commit()
44      except (Exception, psycopg2.DatabaseError) as error:
45          conn.rollback()
46      close_connection(conn, cur)
47
48  # Função que lança várias threads para cada protocolo aumentando a sua
        performance
49  def process_protocol(dir):
50      #Itera cada ficheiro da diretoria dos protocolos
51      for filepath in dir:
52          threading.Thread(target=process_protocol_thread, args=(filepath,)).
        start()
```

Listagem 4.17: Processamento dum protocolo

Na fase seguinte, na base de dados serão realizadas em paralelo várias ações, com vista a inserir os novos dados provenientes destas tabelas. Assim sendo, de forma a facilitar a compreensão destes passos foi criado o diagrama de atividades representando em 4.4, onde é demonstrado o comportamento do *script* de inserção de protocolos, desde a fase de extrações, até à fase onde são realizadas as ações na base dados.

4.3.2 Aplicação Móvel

A aplicação móvel traz ao projeto do Floresta Limpa uma maior interatividade com o utilizador no que diz respeito à visualização e recolha de dados. Dessa forma, esta apresenta-se como uma ferramenta de integração de fontes de dados, uma vez que envia para o Sistema de Informação a informação obtida pelos voluntários ao longo dos seus registos sobre as faixas de gestão de combustível.

Assim sendo, foi necessário estabelecer um processo de integração e processamento destes dados de forma a que sejam mapeados de forma adequada para as tabelas e serviços do Sistema de Informação, que será descrito no tópico seguinte.

4.3.2.1 Registos sobre as Faixas de Gestão de Combustível

O processo de registos sobre as faixas de gestão de combustível consiste na resposta a um conjunto de perguntas presentes em protocolos sobre estas. Numa primeira fase são apresentadas ao utilizador, por parte da aplicação móvel o mapa das FGCI. O utilizador, perante as faixas, escolhe a faixa sobre a qual pretende começar um registo, sendo-lhe apresentadas várias questões a que o utilizador deve responder, tendo ferramentas para adicionar multimédia que suportem as suas respostas, bem como o envio de notas, permitindo enviar informação extra que o utilizador considere de especial relevância. Na figura 4.5 é demonstrada uma ilustração visual deste processo.

Tendo em conta, que o processo de criação do Sistema de Informação e da Aplicação Móvel têm vindo a ser desenvolvidos ao mesmo tempo, a modelação da informação

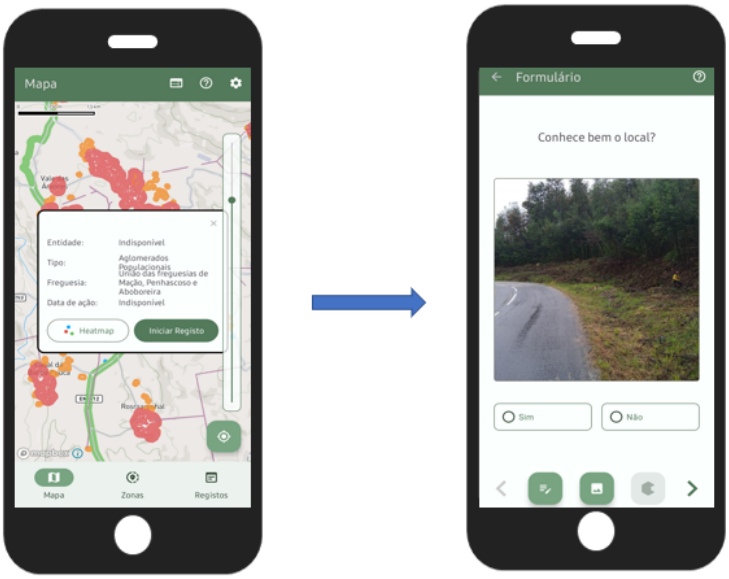


Figura 4.5: Processo de recolha de dados da Aplicação Móvel

enviada pelo utilizador foi desde logo realizada de forma a facilitar os processos quer da Aplicação Móvel quer do Sistema de Informação. Para isso, após o utilizador realizar um registo, a aplicação móvel irá dar origem aos recursos que o Sistema de Informação está à espera de receber, nomeadamente, a informação do registo, o seu conjunto de respostas, de pontos de recolha, bem como de multimédia associada. A informação dos registos é, posteriormente, enviada ao Sistema de Informação aquando do processo de sincronização da Aplicação Móvel com o Sistema. Desta forma, poderão ser enviado um conjunto de registos em simultâneo, pelo que se teve de preparar o Sistema para esta opção. A decisão deste envio múltiplo tem por base a intenção de reduzir o número de pedidos entre a Aplicação e o Sistema.

Assim sendo, numa segunda fase do processo, irá ser processada, pela lógica do *endpoint* de receção de registos, a informação enviada pela aplicação móvel. Este processo encontra-se demonstrado na figura 4.6 e será, brevemente, descrito.

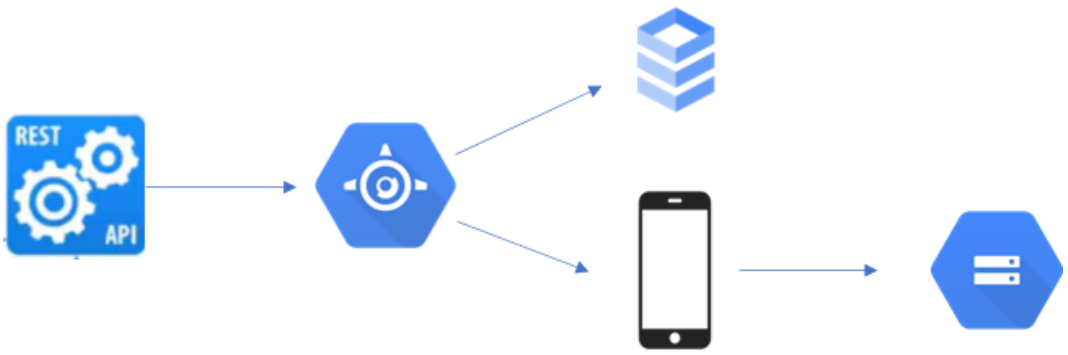


Figura 4.6: Esquema da integração dos registos por parte do Sistema de Informação

O processo por parte do Sistema de Informação, passa pelo processamento dos dados enviados pela Aplicação Móvel. Para isso, o sistema irá colocar na sua base de dados PostgreSQL + PostGIS a informação sobre os registos, respostas às perguntas, pontos de recolha, bem como a informação sobre os ficheiros multimédia. Além disso irá encarregar-se de criar os tuplos nas tabelas de associações, nomeadamente os registos dum utilizador bem como a multimédia associada a uma ou várias perguntas. O Sistema está também preparado para receber pedidos que pedem a eliminação de registos, bem como pedidos de inserção de registos que foram eliminados pelo utilizador antes de ser realizada a sincronização entre a aplicação móvel e o Sistema, garantindo assim, que não existe informação perdida.

Após o processo de inserção dos dados, o Sistema de Informação irá enviar uma resposta à aplicação móvel com o id dos registos que inseriu, de forma a que seja possível detetar erros, quer por falha de internet, quer por dados mal formatos, durante o processo de sincronização, permitindo à Aplicação Móvel saber se necessita de reenviar algum registo. Além disso, o Sistema de Informação irá enviar à Aplicação Móvel uma lista de URL, denominados *Resumable Uploads* que têm como objetivo permitir à Aplicação Móvel aceder diretamente a certos *buckets* da *Google Cloud Storage*. Estes *buckets* foram, previamente, criados durante o processamento por parte da lógica presente na *App Engine* e estão assim disponíveis para serem acedidos por parte da Aplicação Móvel para armazenarem o conteúdo multimédia. A escolha deste processo deveu-se ao facto de não ser possível enviar enormes quantidades de bytes pelos pedidos HTTPS, uma vez que existe tamanho máximo no *payload* destes. Caso existe um erro no envio do conteúdo multimédia foi, ainda, criado outro *endpoint* que pode ser requisitado pela Aplicação Móvel com vista a que lhe seja fornecido um novo URL de *Resumable Upload*.

4.3.3 Comunicação com APIs Externas

Além dos casos acima referidos, é possível ainda adicionar dados ao Sistema de Informação através da comunicação com APIs externas, cuja informação pode ser relevante para o projeto.

A comunicação com a API em causa é então realizada, periodicamente, por parte do Sistema de Informação, através do *endpoint* disponibilizado pela mesma, recebendo novos dados para inserir no Sistema. No âmbito do projeto, um exemplo deste caso é a recolha de notícias relacionadas com os temas principais do projeto Floresta Limpa, que serão requisitadas a uma API denominada NewsData.io, que irá devolver para o Sistema um conjunto de notícias a serem inseridas.

De facto, a NewsData.io API devolve um conjunto de dados em JSON que pode ser diretamente mapeado por parte da lógica inserida no *App Engine*, para os atributos da tabela de notícias dedicados a esse efeito. No entanto, a comunicação com APIs pode ainda ser realizada por parte do Sistema de Informação para receber fontes de dados associadas a séries temporais em certos intervalos de tempo, tais como a informação meteorológica,

obtida a partir da API do IPMA. Nesse caso, a inserção dos dados obtidos pela comunicação tem com a API externa tem ainda de passar pelos casos de processamento de dados definidos em 4.3.1.

4.4 Notificações e Alertas

As notificações e alertas são uma das principais inovações do projeto Floresta Limpa, uma vez que permitem ao utilizador um maior controlo sobre os eventos que se passam em seu redor, bem como dos locais que este definiu como seu interesse.

O processo de envio de alertas e notificações baseia-se numa interação entre o Sistema de Informação e a Aplicação Móvel. Para esse efeito, foi utilizado o serviço do *Firebase* que apresenta suporte para esta funcionalidade, sendo possível verificar o esquema utilizado através da figura 4.7

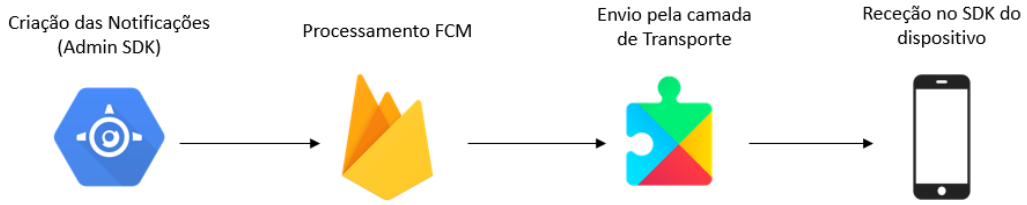


Figura 4.7: Processo de envio de alertas/notificações.

Primeiramente, o processo de criação de alertas e notificações no Sistema de Informação dividiu-se em quatro tipos:

1. **Gerais** - Consistem em notificações que devem ser enviadas para todos os utilizadores.
2. **Dependentes da Localização** - Baseiam-se em alertas enviados aos utilizadores por proximidade de eventos da sua zona de localização.
3. **Zonas de Interesse** - Alertas que informam o utilizador da existência dum evento próximo duma das suas zonas de interesse.
4. **Reboots Obrigatórios** - Notificações sobre a necessidade de reiniciar a aplicação devido à atualização de configurações.

Os alertas e notificações são assim distinguidos pela lógica presente na *App Engine*, sendo que são configurados os tipos de alertas a serem enviados, consoante as situações já descritas. Além disso, no caso em que o Sistema de Informação tem de enviar uma notificação a um utilizador específico, é necessário realizar a identificação dos utilizadores a quem o alerta tem de ser enviado, uma vez que o envio dos alertas é realizado por tópicos.

O envio das notificações por tópicos tem por objetivo enviar notificações apenas para utilizadores que subscreveram estes tópicos, sendo a lógica da subscrição realizada por

parte da Aplicação Móvel. Cada utilizador quando utiliza a aplicação móvel está subscrito a três tópicos, nomeadamente, gerais, reboots e a um tópico com o seu email. Os primeiros dois tópicos dizem respeito às notificações gerais e de reboots obrigatórios, respetivamente, sendo que o tópico com o seu email destina-se a receber as notificações direcionadas apenas ao utilizador em questão. Na listagem 4.18 é possível ver uma porção do código utilizado para enviar os alertas por tópicos.

```
const createAlert = async (title, text, topic, notification_type, adminSDK)
    => {
    //Lógica de criação dum alerta na base de dados
    (...)

    for(let x in topic) {
        let message = {
            data: {
                    alert_id: alert_id,
                    notification_type: notification_type,
                    title: title,
                    body: text
            },
            topic: topic
        };
        adminSDK.messaging().send(message)
            .then((response) => {
                console.log('Successfully sent message:', response);
            })
            .catch((error) => {
                console.log('Error sending message:', error);
            });
    }
};
```

Listagem 4.18: Criação dum alerta.

Dessa forma, após a criação da notificação/alerta por parte do Sistema de Informação este é enviado por tópicos para o *Firebase Cloud Messaging* que possui toda a lógica necessária para o processamento desta. Desta forma, será o FCM a tratar da lógica de envio dos alertas realizando a distribuição dos alertas através dos tópicos que recebeu.

Finalmente, os alertas são enviados para serem distribuídos pela camada de transporte, que realiza o encaminhamento da notificação através da camada de transporte do Android (ATL) para dispositivos Android com o Google Play Services, sendo finalmente entregues ao SDK do dispositivo onde a aplicação se encontra que irá ser responsável por tratar da forma como a notificação é exibida tendo em conta se a aplicação se encontra em primeiro ou segundo plano.

Na figura 4.8 é demonstrado a receção dum alerta por parte de um utilizador.

Figura 4.8: Exemplo de um alerta recebido por um utilizador.

4.5 Logging

De forma a garantir o bom funcionamento de todas as funcionalidades do projeto Floresta Limpa é, como já foi referido, necessário realizar uma monitorização constante de todos os serviços e aplicações que interagem com o Sistema de Informação. Para isso. foi necessário estabelecer lógicas nos processos de *logging*.

De facto, tal como foi abordado em 2.5.15, o *Google Cloud Logging* (GCL) regista automaticamente informação sobre os pedidos que são realizados à *App Engine*, permitindo saber qual foi o *endpoint* requisitado neste pedido, o código da resposta, bem como o tempo que demorou a ser executado. Além disso permite ainda que sejam analisadas informações adicionais sobre o pedido, tais como o protocolo de comunicação utilizado, o endereço IP de onde o pedido foi feito, entre outras informações que possam ser pertinentes.

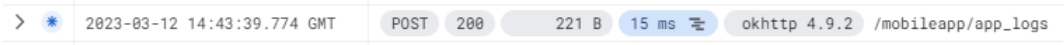


Figura 4.9: Log dum pedido ao *endpoint* do envio de logs da App Móvel.

No entanto, além da monitorização à realização dos pedidos, foi necessário desenvolver outros casos onde o *logging* se apresenta como uma ferramenta essencial. Na informação registada automaticamente pelo GCL, em caso de erro durante a lógica do pedido pode vir a ser necessário obter dados extra, de forma a compreender o porquê do mal funcionamento do pedido. Assim sendo, foi definido um *logger* que também irá registar *logs* no GCL, para situações que possam vir a ser úteis no âmbito do Sistema de Informação.

Além disso, foram identificados casos onde seria útil manter informação sobre a execução dos processos da aplicação móvel, de forma a obter informação sobre o consumo de bateria, bem como a execução dos processos. A criação destes *logs* da aplicação móvel

foi desenvolvida no âmbito do projeto responsável pela criação desta camada de cliente, no entanto, foi necessário realizar um local onde estes possam ser armazenados. Com vista a esse fim, quando a aplicação móvel realiza *logs* envia estes para o Sistema de Informação que os irá receber no respetivo *logger* dedicado a este fim e irá registá-los no GCL.

Assim sendo, para esse efeito, na lógica do Sistema de Informação, desenvolvido em Node.js, foram instanciados dois *loggers*, um responsável por registar os *logs* do Sistema de Informação e outro responsável por registar os da aplicação móvel. Para isso, foi necessário então escolher uma biblioteca do Node.js de *logging*, tendo a escolha recaído sobre a biblioteca Winston, que suporta múltiplos modos de transporte, bem como múltiplos níveis, que permitem diferenciar o tipo de *log* registado. O processo de criação destes *loggers* é exemplificado na listagem 4.19.

```
let lw = require('@google-cloud/logging-winston');

//Logger SI
let loggingWinston = new lw.LoggingWinston({
    logName: 'si_log',
    prefix: 'SI'
});
let logger = winston.createLogger({
    level: 'debug',
    transports: [loggingWinston]
});

//Logger App
let loggingWinstonApp = new lw.LoggingWinston({
    logName: 'app_log',
    prefix: 'APP'
});
let loggerApp = winston.createLogger({
    level: 'debug',
    transports: [loggingWinstonApp]
});
```

Listagem 4.19: Inicialização dos loggers

Os *logs* registados no Sistema de Informação, apresentam assim, a seguinte estrutura base:

- **Nível do *Logging*** - Apresenta o nível de importância da mensagem de *logging*.
- **Estampilha Temporal** - Regista a altura em que o *log* foi criado.
- **Identificador do *Logger*** - Apresenta a identificação do *Logger* que gerou esta mensagem (SI ou APP).
- **Mensagem** - Regista a mensagem de *logging*.

No caso dos pedidos registados no *logger* do Sistema de Informação, este apresenta, ainda, a informação sobre o *trace*, que permite identificar o pedido onde este *log* foi criado. No caso da aplicação móvel, este não é registado uma vez que só existe um *endpoint* que regista os *logs* da aplicação móvel. Além disso, no caso da aplicação móvel, a estrutura do *log* é enviada no corpo da mensagem, uma vez que pode diferir da utilizada pelo SI.

No que diz respeito aos níveis de mensagem, foram identificados cinco níveis, nomeadamente, ordenados por ordem crescente de nível:

- **Debug** - Regista *logs* durante o ambiente de desenvolvimento com informações importantes para a tarefa de *debug*
- **Info** - Nível com o intuito de registar informação de rotina, tal como o sucesso da realização dos pedidos.
- **Notice** - Utilizado para registar *logs* de eventos normais, tais como inicialização de serviços, ou alterações de configuração.
- **Warning** - *Logs* que pretendem registar anormalidades na execução dum evento, mas que não impede a realização do processo.
- **Error** - Nível que pretende indicar eventos que foram responsáveis pela quebra da execução dum processo.

Além destes níveis que são utilizados pelos *loggers*, o GCL pode ainda dar origem a outro tipo de *logs*, nomeadamente:

- **Default** - *Log* sem nível de severidade associado, como por exemplo no registo dos acessos à base de dados (nível mínimo).
- **Critical** - Eventos que provocam sérios problemas no Sistema de Informação (nível acima do *Error*).
- **Alert** - Alertas que levaram à inoperacionalidade do Sistema de Informação, e que necessitam da intervenção humana para serem corrigidos imediatamente.
- **Emergency** - *Logs* que indicam a não disponibilidade de sistemas da Google.

Na figura 4.10 é possível observar a apresentação dos mais diversos níveis de *log* no GCL, bem como a sua estrutura.

Ademais, uma vez que os *logs* no GCL apenas são mantidos durante 30 dias foi decidido que iriam ser armazenados num *bucket* da *Google Cloud Storage*, os *logs* cujo nível de gravidade fossem iguais ou superiores a *Error*.

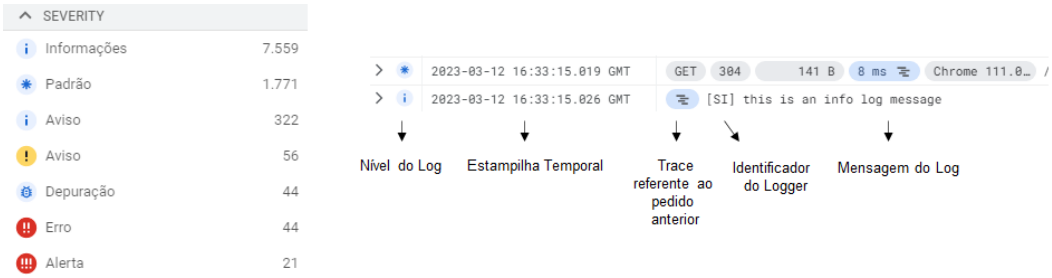


Figura 4.10: Níveis e Estrutura dum *Log* no GCL.

4.6 Google Earth Engine

O *Google Earth Engine* fornece ao Sistema de Informação a possibilidade de aceder a um vasto catálogo de dados de satélite. Uma vez que a obtenção destes dados é essencial para a avaliação de características das séries temporais das faixas, tais como o NDVI e outros índices foi essencial a sua integração no Sistema de Informação.

Assim sendo, após uma análise das bibliotecas disponibilizadas pela GCP, foi identificada a biblioteca javascript (@google/earthengine) que poderia ser integrada com facilidade na lógica do Sistema, uma vez que este é desenvolvido em Node.js. Esta biblioteca fornece duas possibilidades de autenticação e uso do GEE, nomeadamente, a utilização da conta de serviço do Sistema de Informação para realizar pedidos ao GEE, bem como a autenticação usando as contas de cada utilizador. De forma a disponibilizar ao utilizador as várias opções foram elaborados pedidos com as duas formas de abordagem.

Dessa maneira, no primeiro caso, o utilizador ao aceder aos pedidos específicos para o GEE, irá fazer com que as imagens de satélite que pretende obter sejam enviadas para o *Google Cloud Storage*, uma vez que a autenticação perante o GEE é feita com a conta de serviço do projeto, podendo, posteriormente, aceder a estas e realizar novas operações com recurso a outras ferramentas do GEE. No outro caso, o utilizador que deseja utilizar os métodos dedicados ao GEE, deve ter, previamente, a sua conta do GEE criada para poder executar a lógica dos *endpoints* com esta, uma vez que a autenticação é realizada apenas com recurso ao código de acesso adquirido aquando da autenticação do utilizador perante o Sistema de Informação.

De seguida, uma vez que o objetivo principal do Sistema de Informação é a obtenção de dados de satélite, foram, então, desenvolvidos os *endpoints* para esse efeito, acedendo ao catálogo do GEE, permitindo obter as imagens de satélites das coleções especificadas. No entanto, para além disso, é desde logo possível manipular as bandas que são devolvidas na imagem obtida, tendo sido criadas funções para processarem as imagens de forma a que seja possível obter os índices que serão úteis para as séries temporais. Na listagem 4.20 está exemplificado o código genérico criado para a obtenção das imagens do Sentinel relacionadas com um específico índice.

Esta função irá ser chamada após a lógica de autenticação dum pedido ao GEE, onde também é fornecida a geometria relacionada com a região que se pretende observar na

imagem. Após isso é obtida a coleção de imagens do produto especificado do Sentinel-2, sendo disponibilizados um filtro de datas, bem como de percentagem de píxeis identificados como nuvens, sendo no entanto fornecida uma máscara de nuvens de forma a tornar as imagens mais fiáveis para a avaliação do estado das faixas. De seguida, a coleção de imagens é processada inserindo uma banda única relacionada com o índice que se pretende obter.

```
async function handleSentinel(ee, collection, geometry, addIndex, index,
    req, res) {

    //Obtém uma coleção de imagens do Sentinel-2 com os filtros definidos
    let S2_SR = ee.ImageCollection(collection).filterBounds(geometry)
        .filterDate(req.body.startDate, req.body.endDate)
        .filter(ee.Filter.lt('CLOUDY_píxel_PERCENTAGE', req.body.cloudy_pí
    xel_percentage))
        .map(maskcloud1);

    //Aplica o indíce definido (NDVI, NDI45, IRECI, etc...) nas imagens
    let S2_index = S2_SR.map(addIndex);

    //Transforma a coleção de imagens numa lista de imagens
    let list_images = S2_index.toList(S2_index.size());

    //Itera cada imagem da lista, obtendo-a com o índice aplicado
    anteriormente
    list_images.size().getInfo(function (size) {
        for (let i = 0; i < size; i++) {
            let recent_S2 = ee.Image(list_images.get(i))
            let img = recent_S2.select(index)

            //Começa uma task que irá realizar o envio da imagem para a
    Drive do Utilizador ou para o GCS dependendo do acesso escolhido
            startTask(req, ee, img, geometry, res, index, 'PRODUCT_ID')
        }
        res.status(200).json()
    })
```

Listagem 4.20: Obtenção de dados do catálogo do GEE referente ao Sentinel 2

Após isso, são processadas todas as imagens obtidas, sendo selecionado o índice criado, dando origem a uma *task* que se baseia na chamada à função **ee.batch.Export.image.toDrive** ou **ee.batch.Export.image.toCloudStorage** que tem como objetivo agendar o envio desta imagem para a *Google Drive* do utilizador que fez o pedido, ou para a GCS do projeto, respetivamente. Na função **startTask**, é também realizado, imediatamente o processamento deste envio através da chamada à função **start** do objeto gerado pela função responsável pelo envio para a *drive* ou para o GCS.

Embora, o GEE forneça muitas mais operações, estas estão a ser desenvolvidas no projeto no âmbito nos processos de aprendizagem automática para a classificação das

FGCI, pelo que o Sistema de Informação, por enquanto ainda só apresenta funções cujo objetivo principal é a obtenção dos dados de satélite a partir do catálogo do GEE. No entanto, caso os *scripts* de aprendizagem automática venham a ser desenvolvidos com recurso ao JavaScript estes podem ser incluídos e disponibilizados com relativa facilidade em *endpoints* no Sistema de Informação. Por outro lado, é ainda possível integrar os *scripts* desenvolvidos em Python, para este efeito, uma vez que o Node.js disponibiliza processos de forma a que código em Python seja corrido pela aplicação.

4.7 Documentação

A elaboração de documentação é um processo essencial no desenvolvimento dum Sistema de Informação.

De facto, a criação da documentação facilita a interação dos utilizadores com os vários *endpoints* do Sistema de Informação, facilitando o seu processo de aprendizagem através da explicação das respostas e dos parâmetros esperados em cada pedido. Além disso, permite, ainda, explicar cada tabela criada nas bases de dados, bem como os seus atributos, atribuindo a estas um contexto no âmbito do projeto.

Dessa forma, foram elaboradas a documentação da API REST, bem como um dicionário de dados sobre o modelo implementado.

4.7.1 Documentação da API REST

O processo de criação da documentação da API REST consistiu na exportação e especificação de todos os *endpoints* do Sistema de Informação para uma ferramenta dedicada à criação de documentação de API RESTs, nomeadamente, o SwaggerHub.

A especificação duma documentação com o SwaggerHub, tem a vantagem de permitir ao utilizador interagir com todos os *endpoints* do Sistema, através dum ficheiro HTML navegável gerado por este.

4.8 Conclusões

Este capítulo teve como objetivo abordar e descrever todas as funcionalidades que foram implementadas no Sistema de Informação desenvolvido no âmbito desta dissertação.

De facto, a implementação das funcionalidades do Sistema de Informação corresponderam com sucesso ao processo de modelação planeado em 3, mostrando a eficácia que esse teve para esta fase.

Numa primeira fase deste capítulo foram apresentadas as tecnologias escolhidas durante a implementação, que não tinham sido identificadas anteriormente, apresentando uma breve descrição sobre estas e do porquê da sua escolha. A integração destas tecnologias demonstrou-se como uma grande mais-valia no desenvolvimento deste projeto, uma

vez que permitiu que funcionalidades que já tinham sido, anteriormente referidas fossem implementadas com relativa facilidade.

De seguida, prosseguiu-se com uma explicação dos processos de ligação à base de dados e de interação com esta através da camada do servidor. Estas secções foram importantes para compreender a extensibilidade de ligações que o servidor está preparado para realizar, podendo aceder a base de dados alocadas em sítios diferentes. Além disso, permitiu, também, demonstrar a importância da utilização da ORM Sequelize no processo de criação da API REST, uma vez que permitiu o mapeamento das tabelas presentes nas bases de dados para classes que o Sistema consegue identificar, facilitando todo o processo de interação deste com a camada de armazenamento, através dos *endpoints* definidos na API REST.

Ademais, foram apresentados os vários processos de integração de dados de fontes de informação geográfica, demonstrando que o Sistema está preparado para o caráter heterogéneo dos dados que vai integrar. A criação dos processos facilita, ainda, a inserção de novos dados que possam vir a ser identificados, podendo utilizar os processos já definidos ou adaptá-los a novos casos.

Foram, ainda, apresentadas outras funcionalidades desenvolvidas ao longo do processo de implementação, que fornecem características únicas ao Sistema de Informação, nomeadamente, o aviso em tempo real de eventos perto das suas localizações/zonas de interesse através do uso de notificações e alertas, bem como da disponibilização do catálogo do *Google Earth Engine*, de forma a facilitar o processo de obtenção de dados para aprendizagem automática. Além disso, foi ainda desenvolvido um sistema de *logging* encarregue por garantir o bom funcionamento de todas estas funcionalidades produzidas.

Por fim, descreveu-se como foi realizado o processo de documentação do modelo de dados e da API REST, criados com o intuito de facilitar a compreensão dos utilizadores do Sistema de Informação, bem como de todos os interessados no projeto relativos à parte de elaboração do Sistema de Informação.

Capítulo 5

Avaliação

Neste capítulo serão descritos os testes que foram realizados para avaliar o desempenho do Sistema de Informação. Este capítulo foi, sobretudo, útil para testar o limite do servidor com diversos utilizadores concorrentes, procurando diminuir o número de erros que poderiam ser devolvidos em grandes cargas de trabalho para um grande conjunto de utilizadores. Além disso, são primeiramente apresentados os testes realizados aos *scripts* de integração de dados, de forma a garantir a correta execução destes e verificar o impacto do comportamento nos mais diversos componentes do Sistema de Informação.

5.1 Características do Hardware

Todas as experiências e avaliações realizadas ao longo deste capítulo foram feitas com recurso a um computador Desktop com o Sistema Operativo Windows 10 Home e com as seguintes especificações:

- **CPU**: AMD Ryzen 5 5600X 6-Core Processor 3.70 GHz
- **Memória**: 32GB 2x16GB DDR4 3600 MHZ
- **GPU**: GeForce RTX 3070 LHR 8GB GDDR6
- **Disco**: 1.81TB SSD com 1,71TB livres

Relativamente aos serviços da Google foram escolhidas as seguintes arquiteturas:

- **Google Cloud SQL**:
  - **vCPUS**: 2
  - **Memória**: 4 GB

  - **Armazenamento Máximo**: 200 GB SSD

- **Google App Engine**:

  - **Número Máximo de Instâncias** : 20
  - **Classes de Instância**: F1 / F4

5.2 Integração de Dados sobre as FGCI

As FGCI são o recurso central do projeto Floresta Limpa, pelo que é importante que os dados das faixas sejam inseridos com eficácia. No âmbito dos dados disponibilizados por municípios através do PMDFCI foi possível observar uma discrepância no número de faixas disponibilizadas por cada ficheiro de diferentes municípios. Dessa forma, foi necessário preparar os *scripts* para lidar com esta divergência de tamanhos, recorrendo assim ao paralelismo da inserção de dados de faixas, onde vários ficheiros são processados paralelamente, aumentando a velocidade de integração no Sistema.

No entanto, as integrações concorrentes de dados numa base de dados podem levar a diversos problemas, nomeadamente *deadlocks* e inconsistência dos dados em caso de conflitos, bem como problemas de performance, nomeadamente através da sobrecarga no CPU e na memória das máquinas, bem como de operações de escrita e leitura nos discos. No processo de elaboração dos *scripts* foi tido em conta os possíveis *deadlocks* e inconsistência de dados e foram criadas *queries* que garantissem que estas situações não pudessem acontecer.

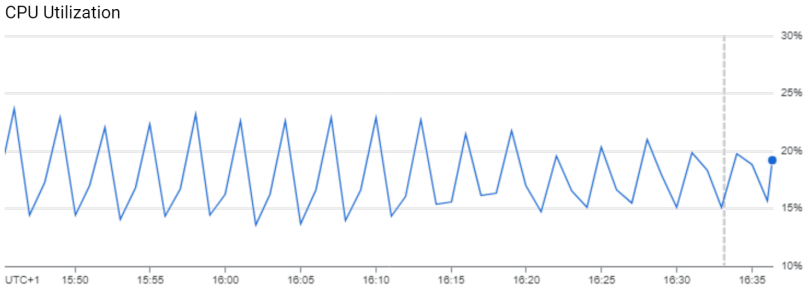
Todavia, é necessário avaliar o comportamento destes dados através da performance da máquina do *Google Cloud SQL* e para isso procedeu-se à execução da inserção de 10 ficheiros de faixas que representavam no total cerca de 73852 faixas, correspondentes a 151.4 MB, onde o menor número de faixas de um ficheiro é 257 e o maior apresenta 13210. Uma vez que os dados das faixas são divididos em duas tabelas no sistema de informação, isto representa a inserção de cerca de 147704 tuplos, tendo sido o processo repetido cerca de 50 vezes de forma a encontrar um valor estável para o tempo de execução. Na tabela 5.1 é possível observar os resultados do tempo médio de execução obtidos na execução deste *script*.

De facto, o tempo médio de execução do processo responsável por integrar os dados foi de cerca de 23,073 segundos, apresentando um desvio-padrão de cerca de 0.803 segundos, o que foi satisfatório perante a carga que a instância do *Google Cloud SQL* foi sujeita. De salientar que o tempo de inserção de alguns destes ficheiros seria menor caso a quantidade de dados a serem adicionados, em simultâneo, fosse mais reduzida. Além disso, para casos em que só se pretende executar a inserção dum ficheiro, o paralelismo do *multiprocessing* deixa de ser necessário, uma vez que o tempo de lançar o sub-processo deixa de ser compensatório perante a execução sequencial.

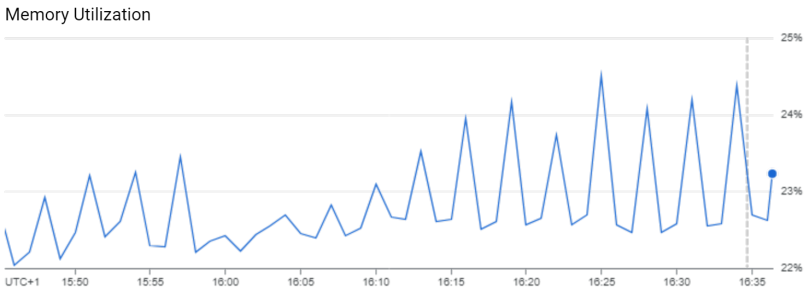
Tabela 5.1: Tempo médio de execução de cada ficheiro de FGCI (50 execuções)

Nome do Ficheiro	Número de Faixas	Tamanho (MB)	Tempo Médio de Execução (s)	Min (s)	Max (s)	Std. Dev (s)
FGC_0607.shp	6552	29.7	11.554	9.697	15.743	1.338
FGC_0612.shp	13210	17.4	20.974	18.638	23.707	1.110
FGC_0804.shp	11414	17.4	20.389	17.829	23.105	1.339
FGC_0809.shp	12415	14.1	19.761	16.573	23.753	1.603
FGC_0814.shp	1688	17.3	16.133	12.144	23.084	2.472
FGC_0816.shp	4313	18.8	8.357	6.171	13.539	1.393
FGC_1212.shp	10901	18.2	20.572	17.999	22.769	0.995
FGC_1413.shp	2056	7.25	7.352	5.355	10.787	1.113
FGC_1417.shp	11046	10.6	15.771	12.591	19.704	1.768
FGC_1503.shp	257	0.656	1.023	0.862	1.917	0.136
Total	73852	151.4	23.073	22.012	25.392	0.803

De seguida, de forma a analisar como o *script* atuou no hardware da máquina da Google foram ainda observados os seguintes gráficos em 5.1.



(a) Utilização do CPU



(b) Utilização de memória

Figura 5.1: Análise de recursos da *Cloud SQL* extraídos do *Metrics Explorer* - Faixas

Quer os níveis de utilização do CPU, quer os níveis de utilização de memória mantiveram-se estáveis durante a execução do *script*, ocupando cerca de 23% no caso do CPU e 24% no caso da memória, ao longo da execução do *script*. Ademais, é possível verificar umas oscilações nos gráficos que ocorrem, visto que nos testes após a inserção das faixas, estas foram, imediatamente, eliminadas da base de dados para proceder à repetição do processo. Assim sendo, entre cada processo foram esperados 45 segundos, de forma a permitir que a máquina libertasse alguns recursos da utilização anterior. De notar que em certos casos, os recursos podem apresentar ligeiros aumentos, uma vez que a instância possui replicação,

sendo necessário alocar mais recursos para realizar.

Apesar do tempo de execução e comportamento do *script* se ter revelado dentro dos limites esperados é importante que se mantenha uma contínua monitorização das suas próximas utilizações, de forma a garantir que a sua performance continua a ser a esperada.

5.3 Integração de Protocolos

A integração dos protocolos é um processo mais leve por comparação com a integração das faixas, uma vez que se baseia em ficheiros JSON, que não contêm informação georreferenciada e a quantidade de tuplos a inserir é também reduzida. Além disso, este processo foi dividido em dois sub-processos que devem ser executados sequencialmente de forma a não criar inconsistências nos dados, nomeadamente, deve ser realizada, em primeiro lugar, a inserção de todas as fotos que vão ser utilizadas pelos protocolos na *Google Cloud Storage* e só, posteriormente, realizada a inserção dos dados referentes aos protocolos na *Google Cloud SQL*. Dessa forma, para testar este *script* foram introduzidos 11 novos protocolos no Sistema juntamente com 36 fotografias.

Primeiramente, são enviadas as fotografias para a *Google Cloud Storage*, através da criação do *blob* na diretoria dos protocolos presente no *bucket* das imagens. Este processo obteve tempos de execução que variaram entre os 1s e 2,124 segundos para ficheiros distintos no envio dos seus *bytes*, tendo originado uma média 2,689 segundos na execução do *script* ao fim de 50 execuções para um total de 0,244 MB.

Relativamente à inserção na *Google Cloud SQL*, no caso dos protocolos, escolheu-se o *threading* em vez do *multiprocessing*, visto que os tempos de execução de ambos eram praticamente semelhantes, uma vez que o processo de transformação de dados dos protocolos é menos pesado que o das faixas, não compensando o lançamento de vários sub-processos que é mais demorado e mais pesado que o lançamento de *threads*. Os tamanhos dos ficheiros a serem processados eram praticamente semelhantes, não levando a grandes diferenças de tempo de execução entre estes (cerca de 0,9s - 1s), sendo que o processo global teve um tempo médio de 1.267s após ter sido executado cerca de 50 vezes. Desta integração resultaram a inserção de 584 tuplos divididos por 7 tabelas.

A diferença entre a integração mais demorada, quer do ficheiro inserido na *Google Cloud Storage* quer no ficheiro inserido na *Google Cloud SQL* com o tempo total, justifica-se uma vez que o *script* espera pela execução de todas as suas *threads*.

5.4 Integração de Séries Temporais

A integração das Séries Temporais e dos dados das suas fontes e produtos foi outro dos *scripts* trabalhados no âmbito desta dissertação. De facto, a arquitetura e lógica do Sistema está pronta para estabelecer a ligação com a camada pública da GCP, bem como uma camada privada alocada numa máquina.

No entanto, durante o processo da realização de testes foi verificada a incompatibilidade de acesso ao *cluster* do DI da FCT, uma vez que não é possível abrir portas públicas nesta máquina para que o *Google App Engine* aceda a esta. Assim sendo, apesar da arquitetura estar pronta para este efeito na disponibilidade duma máquina onde a abertura de uma porta pública seja possível, foi adotada outra estratégia no armazenamento dos dados das séries temporais.

Para este efeito, optou-se por armazenar os dados das séries temporais no *Google Cloud Storage* e apenas registar nas tabelas da *Google Cloud SQL*, os apontadores para o caminho destes ficheiros. De notar que este processo é semelhante ao já descrito para os *rasters* em 4.3.1.2, atuando o *Cloud Storage* como sistema de ficheiros, sendo que foi tomada a decisão de realizar o mesmo para os dados vetoriais uma vez que estes são de grandes dimensões e poderiam provocar sobrecarga na máquina que aloca a instância da *Cloud SQL*.

Assim sendo, para testar os *scripts* correspondentes às séries temporais foi escolhida a inserção de 13 *rasters* de diversas dimensões correspondendo a cerca de 821MB. Na tabela 5.2 é possível consultar o tempo médio de execução para estes ficheiros e *script* após 50 execuções.

Tabela 5.2: Tempo médio de execução de cada ficheiro raster (50 execuções)

Nome do Ficheiro	Tamanho (MB)	Tempo Médio de Execução (s)	Min (s)	Max (s)	Std. Dev (s)
LC08_L1TP_T1_B1.TIF	78.0	46.634	42.293	50.060	2.241
LC08_L1TP_T1_B3.TIF	83.6	49.648	44.704	53.746	2.232
LC08_L1TP_T1_B5.TIF	94.7	52.421	47.586	55.709	1.760
LC08_L1TP_T1_B6.TIF	93.5	52.796	50.249	54.962	1.171
LC08_L1TP_T1_B8.TIF	338.0	96.701	92.291	100.771	1.841
LC08_L1TP_T1_B9.TIF	46.1	34.403	30.713	40.208	2.185
T22UDD_AOT_10m.jp2	0.083	1.410	1.148	2.160	0.222
T22UDD_B02_10m.jp2	17.3	16.722	10.281	20.331	1.814
T22UDD_B03_10m.jp2	16.8	16.842	12.071	19.615	1.713
T22UDD_B04_10m.jp2	16.6	16.057	11.880	19.751	1.941
T22UDD_B08_10m.jp2	16.8	16.358	6.445	19.966	2.266
T22UDD_TCI_10m.jp2	19.0	18.248	12.887	22.659	1.818
T22UDD_WVP_10m.jp2	0.012	1.328	1.014	2.527	0.292
Total	821	96.750	92.338	100.800	1.840

Como pode ser observado, o tempo médio de execução tem por base o tamanho da transferência de *bytes* que é necessário fazer para o GCS, sendo o *script* limitado pelo tempo de execução do maior ficheiro enviado. Além disso, no gráfico em 5.2 é possível consultar o tráfego de dados recebidos pela rede ao longo do período das 50 execuções deste *script*.

De modo geral, os resultados deste *script* mostraram-se satisfatórios, mostrando a eficácia de inserção dos dados dos *rasters* consoante o seu tamanho, ou seja caso o ficheiro seja de dimensões reduzidas a sua inserção é relativamente rápida, aumentando progressivamente para casos onde o número de *bytes* transferidos é maior. No gráfico

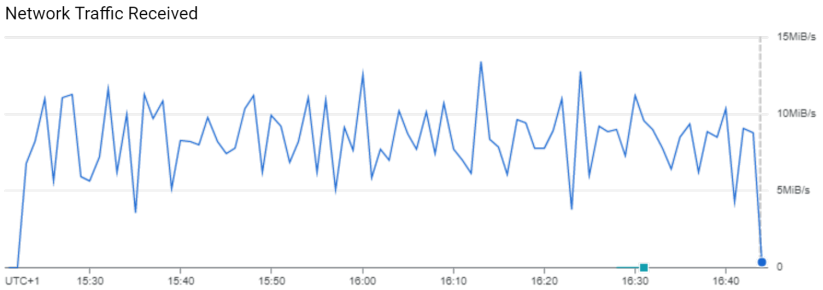


Figura 5.2: *Network Traffic Received* - Rasters

5.2 foi possível observar que os *bytes* recebidos pela *Google Cloud Storage* por segundo se ajustam conforme a quantidade de dados que estão a ser enviados para o Sistema nesse momento, permitindo assim, ajustar-se a pedidos de diversas ordens de grandeza. Relativamente ao acesso à máquina da *Google Cloud SQL*, o número de inserções de tuplos foi reduzida pelo que o estado da CPU e da memória se manteve praticamente inalterado face ao estado de mínima utilização.

5.5 Consultas à base de dados

Com o intuito de testar a eficácia das *queries* desenvolvidas para realizar consultas à base de dados, bem como o tempo necessário pelo servidor para a obter foram realizadas avaliações à *query* feita diretamente na base de dados do *Cloud SQL*, bem como ao pedido responsável por originar esta mesma *query* na API REST do Sistema de Informação. Assim sendo, foi escolhida a consulta que retorna os dados dos primeiros 15000 objetos espaciais registados no sistema.

Assim sendo, após realizar esta consulta 20 vezes diretamente à base de dados foi obtido o tempo médio de cerca de 0,0112s para a consulta em questão, demonstrando a eficácia na base de dados no planeamento e execução desta consulta que devolve 15000 tuplos.

De seguida, foi realizado o pedido responsável por esta consulta na API REST o mesmo número de vezes e foi possível verificar uma média de tempo de resposta de 15s, contabilizando a latência entre o cliente e o servidor (sendo que nos logs do *Google Cloud Logging* este valor decresce para 14,2s). Como seria de esperar, este pedido demora muito mais tempo a ser executado pelo servidor Node.js, uma vez que realiza o mapeamento da consulta realizada na base de dados para JSON, de forma a que seja um formato compreendido pela camada de cliente. Além disso, de notar que este pedido gera cerca de 22 MB de dados, uma vez que apresenta objetos espaciais com multi-polígonos, dando estes origem a um conjunto enorme de vetores, sendo portanto ainda mais dispendioso mapear estes valores.

Em suma, este teste serviu para garantir o bom funcionamento da consulta na base de dados SQL, bem como para identificar possíveis limites do lado da camada do servidor, tendo sido importante para este contexto o desenvolvimento de paginação e limitação de

dados, uma vez que permite que os dados duma tabela sejam divididos em fragmentos mais pequenos até ser obtida toda a informação pelo cliente.

5.6 Inserção de Registos

Depois de ter sido verificado o comportamento do Sistema de Informação por parte de consultas realizadas a partir do API REST foi necessário compreender o efeito das inserções através desta. Assim sendo, foi testado o *endpoint* utilizado pela aplicação móvel para a inserção de registos, uma vez que este representa a maior quantidade de dados que pode ser enviada por um utilizador da aplicação móvel ao Sistema de Informação.

Dessa forma, para testar o comportamento deste pedido, foi simulado o caso em que um utilizador perdeu a ligação à internet e continua a realizar registos. Nestas condições a aplicação móvel irá guardar todos os registos feitos pelo utilizador e enviá-los em massa para o Sistema de Informação quando a aplicação recuperar a ligação à internet. Para este caso particular foram realizados 3 tipos de teste, nomeadamente o envio dos 50 registos por parte de um utilizador, por parte de 5 e de 10 em simultâneo, tendo cada teste sido repetido 10 vezes de forma a garantir que as suas execuções são constantes.

Na tabela 5.3 é possível verificar o tempo de execução médio da inserção de registos em cada um destes casos. Além disso, de forma a suportar estes dados é possível observar os valores de latência obtidos por parte destes envios, diretamente do *Metrics Explorer* no gráfico em 5.3.

Tabela 5.3: Tempo médio de execução da inserção de registos (10 execuções)

Users	Avg (ms)	Min (ms)	Max (ms)	Std. Dev (ms)
1	7104	6926	7832	258.6
5	8420	7487	9548	550.98
10	9736	7148	22800	2693.81
Total	9120	6926	22800	2297.13

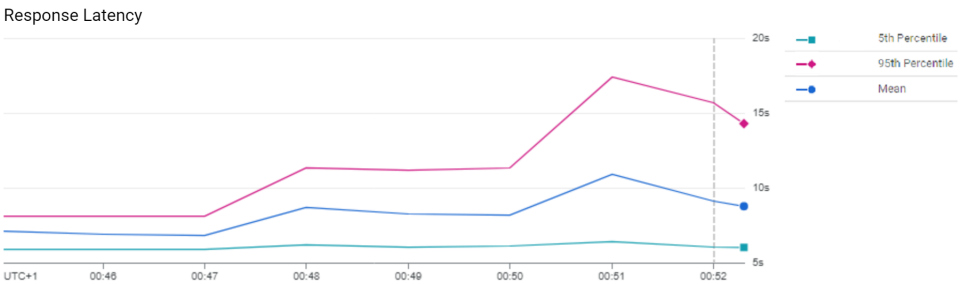


Figura 5.3: Latência do envio de 50 registos

Em relação aos pedidos realizados verificou-se que o tempo médio da sua execução piora à medida que o número de utilizadores em simultâneo aumenta, devido a uma

maior sobrecarga do sistema. No entanto, os valores de desvio padrão são relativamente pequenos, permitindo inferir que o comportamento do pedido se mantém constante. Embora o desvio do último caso seja ligeiramente superior aos outros dois, este acontece uma vez que uma instância apenas só pode realizar 5 ligações concorrentes à base de dados, pelo que no caso deste teste, um dos pedidos teve de esperar que uma instância nova fosse lançada.

Assim sendo, estes resultados permitem inferir que o sistema se adaptou com facilidade a uma situação que à partida será pouco vulgar, visto que não é esperado que o sistema receba de um utilizador cerca de 50 registos duma vez, sendo ainda mais raro para o caso de 5 ou 10 utilizadores em simultâneo. É também, preciso destacar que embora a execução média deste pedido seja de 7-9s, de forma a que o utilizador não fique bloqueado à espera desta, o pedido é executado através de um ciclo de atualizações por parte da aplicação, onde esta realiza em segundo plano a sincronização dos dados obtidos pelo utilizador com o servidor. No entanto, este é um caso que deve continuar a ser monitorizado à medida que apareçam mais utilizadores de teste.

5.7 Testes de Carga

A aplicação móvel já foi disponibilizada a alguns utilizadores, pelo que a sua ligação com o Sistema de Informação já se encontra desenvolvida e em produção. Dessa forma, visto que o número de utilizadores a testá-la é, ainda, reduzido, tem sido feita uma monitorização das ações realizadas por estes, de forma a garantir que não têm sido perdidos dados entre a aplicação e o Sistema, através da visualização e comparação dos *logs* produzidos por cada uma destas. Além disso, têm sido realizadas consultas à *Google Cloud Storage* e ao *Google Cloud SQL* de forma a verificar que os dados estão a ser armazenados da forma correta, pelo que até ao momento não existe nada de anormal a apontar.

No entanto, apesar destes resultados para utilizadores pontuais, é necessário ter uma noção da performance do *Google App Engine* quando este se encontra sobrecarregado por pedidos concorrentes de diversos utilizadores. Para tal, visto não existirem utilizadores de teste suficientes, foi utilizado o *Apache JMeter* [6] que é um *software* que permite realizar testes de carga num servidor, simulando vários ambientes de carga de forma a poder testar a performance do sistema.

Assim sendo, para a realização deste teste de carga foi escolhido o cenário hipotético de inicialização da aplicação por parte de utilizadores que já tinham realizado registos com esta, mas que limparam os seus dados previamente. Este cenário foi escolhido uma vez que representa a realização do maior número de pedidos num curto de espaço de tempo por parte da aplicação ao sistema, nomeadamente, cerca de 6 pedidos por cada utilizador. A aplicação divide estes 6 pedidos, em dois grupos paralelos que executam paralelamente, uma sequência de 2 e 4 pedidos respetivamente, tendo este caso também tido em conta na execução dos testes.

De forma a descobrir qual era o limite do servidor para as instâncias da *App Engine* escolhidas foram realizados testes com diferentes números de utilizador, nomeadamente 50, 100, 200, 240, 250, 260 e 270. É importante referir que é improvável que haja 270 utilizadores a utilizar a aplicação, simultaneamente, pelo que este teste foi realizado de forma a testar os limites do servidor.

Tabela 5.4: Métricas de desempenho para o *Load Testing* à *App Engine*

Users	Avg (ms)	Min (ms)	Max (ms)	Std. Dev (ms)	Error %	Throughput (Req/s)	Rec KB/s	Sent KB/s	Avg Bytes
50	1796	78	10323	1951.59	0.0	18.97	386.65	16.33	20870.7
100	2097	74	11547	2620.58	0.0	39.86	812.34	34.30	20870.7
200	4401	77	10670	1871	0.0	52.74	1074.97	45.39	20870.7
240	5481	77	12097	2392.91	0.0	54.49	1110.50	46.90	20870.7
250	5596	77	17410	3356.46	0.0	57.75	1176.94	49.70	20870.7
260	6080	73	17421	3105.49	0.0	50.38	1026.74	43.36	20870.7
270	6052	72	18200	3620.57	0.0025	56.28	1142.28	48.43	20785.1

Os resultados obtidos nos testes realizados com o JMeter demonstraram propriedades interessantes sobre o comportamento do Sistema de Informação. Primeiramente, é importante salientar que cada resultado apresentado foi realizado com nenhuma instância ativa de momento aquando do começo dos pedidos dos utilizadores de teste.

Passando aos resultados obtidos, é possível verificar que à medida que o número de utilizadores aumenta, o tempo de resposta também aumenta, traduzindo-se numa degradação da performance à medida que a *App Engine* fica mais sobrecarregada. A alteração do tempo de resposta de 50 utilizadores para 100 não ocorre de forma proporcional pelo que dá a entender que a configuração escolhida neste momento é adequada para este número de utilizadores apresentando médias de resposta consideravelmente baixas (cerca de 2s). No entanto é possível ver que quando o número de utilizadores em simultâneo é cerca de 200-270, este valor começa a ser proporcional o que pode indicar que está perto de atingir o limite do sistema.

Relativamente, aos valores mínimos e máximos de tempos de resposta é possível verificar que eles se mantêm relativamente próximos, com maior alteração para os valores de 260 e 270 utilizadores, sendo este um fator para considerar estes valores o limite da configuração escolhida. Esta discrepância de valores dá origem a valores do desvio padrão relativamente consideráveis, no entanto, é importante destacar que este desvio é influenciado pelo lançamento de novas instâncias que demoram cerca de 3-5s a serem lançadas, influenciando os tempos máximos de resposta. Além disso, quando foram simulados 270 utilizadores começaram a aparecer os primeiros erros de resposta por limite de tempo excedido na fila de uma instância, sendo este um sinal de que o limite se situa por volta destes valores.

Por fim, é importante realçar os valores obtidos no *throughput* onde é possível verificar que perante um aumento do número de utilizadores, o sistema é capaz de lidar com

mais pedidos por segundo, bem como nos valores de transferência e receção de dados, que também aumentam, demonstrando assim que o sistema é capaz de transferir maior quantidade de dados perante cargas mais altas de pedidos. Além dos resultados apresentados nas tabelas pelo *Apache JMeter*, são apresentadas, de seguida, as métricas obtidas diretamente a partir dos gráficos do *Metrics Explorer* da GCP relativamente ao *Google App Engine* de forma a suportar a informação apresentada.

Figura 5.4: Latência das respostas da App Engine - Teste de Carga

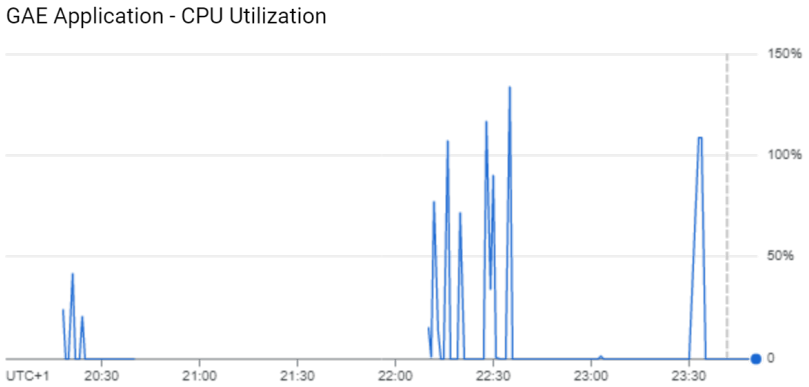
No gráfico em 5.4 é possível visualizar a latência das respostas do servidor, onde é dada informação extra sobre o percentil 95 e o percentil 5, onde podemos concluir da discrepância de valores apresentada, justificada pela demora dos pedidos que provocam o lançamento de novas instâncias.

Figura 5.5: Número de Instâncias do App Engine - Teste de Carga

Relativamente ao gráfico em 5.5, este demonstra a evolução do número de instâncias da *App Engine* ao longo do teste, que foi aumentando à medida que existia mais carga sobre esta. De referir, também, que neste gráfico é possível verificar 21 instâncias num dado momento, visto que apresenta nestes números uma instância que se encontra a ser desligada ou que ainda não foi processada como desligada por parte do *Metrics Explorer*.

Os gráficos de utilização quer do CPU quer da memória em relação ao GAE em 5.6 são de particular destaque, uma vez que comprovam que o servidor atinge valores acima dos limites, demonstrando que este está a receber mais pedidos do que aqueles que terá capacidade para lidar, provando-se, assim, uma necessidade de escalonamento.

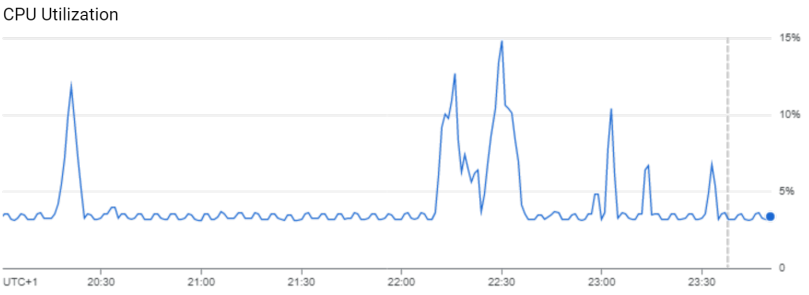
Por fim, em 5.7 são, também apresentadas as métricas referentes à utilização do CPU e da memória, mas desta vez no Cloud SQL. Ao contrário do que acontece com o *App Engine* é possível verificar que a utilização do CPU e da memória sofreram ligeiros aumentos



(a) Utilização do CPU da App Engine

(b) Utilização da memória da App Engine

Figura 5.6: Utilização do CPU e da memória da App Engine - Teste de Carga



(a) Utilização do CPU da Cloud SQL

(b) Utilização da memória da Cloud SQL

Figura 5.7: Utilização do CPU e Memória da Cloud SQL - Teste de carga

durante a realização destes testes, mas sem chegarem a valores perto dos seus limites, demonstrando que a instância da base de dados é eficaz face a situações de múltiplos acessos simultâneos.

5.7.1 Escalonamento Horizontal e Vertical

Para além dos resultados apresentados foi importante verificar como as técnicas de escalonamento horizontal e vertical afetavam a performance do Sistema de Informação. Para isso, numa primeira fase verificou-se o comportamento do Sistema já tendo 20 instâncias a correr, ou seja, já sob o efeito de elevado tráfego, não representando o número de pedidos concorrentes um elevado pico perante o que o Sistema já estava à espera. De notar que o número máximo de instâncias escolhidas é 20, uma vez que cada uma destas pode abrir uma *pool* de 5 conexões para a *Google Cloud SQL* que apresenta um valor máximo de 100 conexões simultâneas.

De seguida, numa outra fase procedeu-se à aplicação do escalonamento vertical, onde foram alteradas o tipo de instâncias utilizadas pelo GAE, passando de instâncias F1 para F4, que representam um aumento do CPU de 600 MHz para 2.4 GHz e do limite de memória de 256 MB para 1024 MB. Os resultados obtidos estão representados nas tabelas 5.5 e 5.6, sendo o primeiro exemplo o caso de escalonamento horizontal e o segundo caso o de escalonamento vertical. Nestes resultados por comparação com os apresentados anteriormente, é possível verificar uma clara melhoria nos valores obtidos, mostrando que com mais recursos é possível que o Sistema expanda a sua dimensão de utilização.

Tabela 5.5: Comparação de métricas após escalonamento horizontal

Num Inst.	Users	Avg (ms)	Min (ms)	Max (ms)	Std. Dev (ms)	Error %	Throughput (Req/s)	Rec KB/s	Sent KB/s	Avg Bytes
1	270	6052	72	18200	3620.57	0.0025	56.28	1142.28	48.43	20785.1
20	285	2334	72	8412	1583.87	0.0	103.10	2101.31	88.73	20785.1

Nota: Instâncias do tipo F1.

Tabela 5.6: Comparação de métricas após escalonamento vertical

Type Inst.	Users	Avg (ms)	Min (ms)	Max (ms)	Std. Dev (ms)	Error %	Throughput (Req/s)	Rec KB/s	Sent KB/s	Avg Bytes
F1	270	6052	72	18200	3620.57	0.0025	56.28	1142.28	48.43	20785.1
F4	500	3072	75	9540	1376.05	0.0	167.36	3383.95	144.05	20785.1

Nota: Uma instância a correr no momento em que os testes foram executados.

5.8 Conclusões

Em suma, as avaliações realizadas ao Sistema de Informação revelaram bons resultados para as arquiteturas desenvolvidas. Em relação aos recursos da Google é preciso ter em

conta que se procurou testar arquiteturas que fossem capazes de lidar com uma carga ligeira de utilização, de forma a tentar minimizar os custos que são aplicados nesta fase do projeto pelos recursos alocados.

Assim sendo, os *scripts* realizados em *Python* mostraram-se adequados para a quantidade de informação com que se pretende lidar, não criando limites de utilização do CPU nem de memória para os casos testados. Além disso, os tempos de execução demonstraram uma boa capacidade do sistema a lidar com cargas de vários tamanhos, especialmente graças à utilização de *threading* e *multiprocessing* que permite que os recursos sejam aproveitados de forma a maximizar a performance destes *scripts*. Ademais é importante destacar a interação destes processos com a componente da *Google Cloud Storage*, onde foi possível verificar a adaptabilidade desta para a receção de grandes quantidades de dados.

Quanto às consultas realizadas diretamente à base de dados e através da API REST disponibilizada no *App Engine* foi possível verificar uma diferença considerável entre estas. No entanto, esta discrepância é justificada pelo facto do servidor Node.js realizar transformações aos dados obtidos da base de dados de forma a serem enviados para a camada do cliente. Dessa forma, é importante verificar se existe possibilidade de otimizar a quantidade de dados que são devolvidos com vista a melhorar ainda mais a performance do servidor.

De seguida, após testar o comportamento das consultas foram testadas as inserções através da API REST, onde foi escolhido um caso que permitisse simular uma situação real que poderia provocar sobrecarga na quantidade de dados enviados num pedido ao Sistema. De facto, foi encontrado um limite, uma vez que para 50 registos de um utilizador são enviados cerca de 32 MB como resposta do servidor para a camada do cliente, que é o limite que pode ser enviado por este. No entanto, os resultados obtidos na latência são satisfatórios para um pedido que realiza diversas operações e que não bloqueia a experiência do utilizador, uma vez que é constantemente realizado pela aplicação móvel em certos intervalos de tempo de forma a sincronizar-se com o servidor.

Relativamente à interação da camada do cliente com a camada do servidor, nomeadamente através do envio de pedidos pela aplicação móvel para o *App Engine* também foi possível observar resultados de acordo com o previsto. De facto, tendo em conta que os casos de uso reais não apresentaram, até à data, nenhum erro identificado foi um primeiro passo importante para a garantia da estabilidade do servidor. No entanto, os testes de carga realizados ao Sistema de Informação permitiram reforçar a boa adequação do Sistema a diferentes tipos de carga, uma vez que mantiveram um tempo de resposta média razoável com uma baixa taxa de erros e bom *throughput*. Além disso, tendo em conta que, como já foi referido, foram utilizadas máquinas cujas especificações eram relativamente baixas, os testes realizados aos escalonamentos permitiram verificar que a performance do Sistema pode ser melhorada, caso se identifique a necessidade de expandir a sua utilização para elevados números de utilizadores.

Em conclusão, as avaliações realizadas neste capítulo permitiram comprovar o bom

desempenho das funcionalidades desenvolvidas, disponibilizando, assim, uma boa experiência de utilização aos seus clientes. No entanto, apesar dos resultados obtidos é preciso ter em consideração que nenhum destes casos teve em conta a elevada participação de utilizadores num cenário realmente real, pelo que é importante que numa próxima fase venha a ser analisado o comportamento do sistema sob estas condições.

Capítulo 6

Conclusão e Trabalho Futuro

Este último capítulo tem como objetivo fazer uma breve apreciação de todo o trabalho desenvolvido durante esta dissertação, comentando o estado do Sistema de Informação bem como dos resultados obtidos relacionados com este.

Por fim, apresenta também algumas noções de trabalho futuro a realizar de forma a melhorar o desempenho e estrutura do Sistema de Informação do projeto Floresta Limpa.

6.1 Conclusão

Esta dissertação incidiu-se sobre o tema das faixas de gestão de combustível para incêndios, foco principal do projeto Floresta Limpa, tendo como objetivo a modelação e conceção de um sistema de informação especializado para a monitorização destas.

De facto ao longo do desenvolvimento do projeto foi possível verificar a existência de grandes quantidades de dados sobre as faixas. No entanto, estes dados encontravam-se muitas vezes com erros e a não cumprir com as especificações definidas no documento em vigor para este fim (PMDFCI). Além disso, foram identificadas fontes de informação complementar de diversas entidades que quando agrupadas poderiam melhorar a monitorização das faixas, nomeadamente através da identificação de intervenções nas regiões por parte dos municípios, bem como da junção das mais diversas fontes de dados onde é possível obter informação sobre o território.

Assim sendo, o sistema de informação desenvolvido revelou-se uma ferramenta inovadora, não só para o ambiente do projeto como para o do território nacional, uma vez que permite o armazenamento e organização de todos estes dados, fornecendo, ainda, recursos às outras componentes do projeto que pretendem melhorar a monitorização das faixas, nomeadamente através da aplicação de algoritmos de aprendizagem automática sobre estes dados, de forma a identificar se as faixas se encontram limpas ou não, bem

como através da sensibilização de voluntários que possam fornecer ainda mais informação sobre as faixas, registando-a através de uma aplicação móvel desenvolvida para este fim.

Entrando na parte mais específica da dissertação, a modelação do sistema de informação revelou bons resultados ao longo do desenvolvimento do projeto, uma vez que as alterações que foram realizadas ao modelo inicial foram mínimas e pontuais, destacando a importância que esta fase teve na criação de um sistema que cumprisse com o que lhe foi proposto e que fosse extensível à medida da evolução do projeto.

No que concerne ao desenvolvimento da API REST, este foi um processo trabalhoso que teve por base o seguimento de regras previamente estabelecidas fundamentais na sua criação, encontrando-se neste momento exposta e documentada de forma a permitir a sua fácil compreensão e utilização por parte dos utilizadores. É, também, necessário destacar que o sistema procurou não ser apenas um repositório de dados, fornecendo, ainda, funcionalidades adicionais para as várias componentes do projeto, nomeadamente através da criação e do envio de alertas de eventos que serão propagados para os utilizadores através da aplicação móvel, de forma a que estes estejam sempre informados sobre o que se passa nas faixas em seu redor e do seu interesse, bem como através da integração da ferramenta do *Google Earth Engine* que apresenta um caráter fundamental no desenvolvimento dos algoritmos de aprendizagem automática.

Por fim, foi possível verificar o bom desempenho do sistema nas mais diversas funcionalidades a que se propôs, especialmente nas técnicas de integração de dados desenvolvidas que foram capazes de lidar com diversos tipos de dados, mostrando que pode ser incluída no sistema qualquer tipo de informação que possa vir a ser identificada como útil no âmbito deste tema. Ademais, foi possível verificar boas respostas do servidor Node.js alocado na GAE, com recursos reduzidos, o que demonstra que este se encontra pronto para lidar cada vez com mais dados e cargas à medida que a dimensão do projeto avance, desde que seja acompanhado pela evolução dos equipamentos que o suportam.

6.2 Trabalho Futuro

Embora o Sistema, desenvolvido no âmbito desta dissertação, cumpra, de forma eficaz, os objetivos propostos no projeto, existem características que podem ser adicionadas futuramente para que este apresente ainda melhores resultados, quer em nível de performance quer em nível de experiência do utilizador. Dessa forma nesta secção serão apresentadas algumas funcionalidades que poderão a vir ser incluídas em versões futuras do Sistema.

6.2.1 *Hypermedia Controls* na API REST

Embora não fosse algo prioritário no desenvolvimento da API REST para o projeto, a API REST do Sistema de Informação poderá sofrer alterações de forma a desenvolver-se para o nível 3 no Modelo de Maturidade de Richardson [47]. Este nível diferencia-se do

nível 2 pela introdução de HATEOAS (*Hypermedia as the Engine of Application State*) que consiste na introdução de *Hypermedia controls* nas respostas a um pedido do utilizador.

Estes controladores têm a função de indicar ao utilizador os pedidos que podem ser feitos a partir do recurso que o utilizador obteve no pedido que realizou, facilitando assim o processo de utilização da API REST. No caso do Sistema de Informação, uma das aplicações possíveis seria devolver, após um pedido POST ao */records*, um *hyperlink* para disponibilizar a inserção de respostas associadas a este recurso, nomeadamente /records/:id/answers.

Assim sendo, esta extensão da API para o nível 3 poderia facilitar a utilização do Sistema de Informação, uma vez que tornava mais intuitiva a linha de aprendizagem que neste momento é necessária e providenciada através da documentação existente.

6.2.2 Cache com Redis

No desenvolvimento deste Sistema em conjunto com a aplicação móvel existiu o cuidado de tentar diminuir o número de pedidos que a aplicação tem de fazer ao Sistema, bem como a quantidade de dados que tem de receber. Para isso foram criadas opções de *query* do lado do servidor, onde a aplicação irá enviar o último identificador do recurso que pretende obter e onde o servidor apenas irá devolver os recursos cujo identificador é superior a esse. Dessa forma diminui-se o envio de informação duplicada, ou seja informação que a aplicação já possui.

No entanto, para recursos, frequentemente acedidos, a experiência do utilizador pode ser melhorada com recurso ao Redis, que consiste numa base de dados em memória que pode ser utilizada como camada intermediária entre o Servidor do GAE e a base dados do *Cloud SQL*. O Redis guarda em memória dados frequentemente acedidos, permitindo que a aplicação aceda a estes dados sem ter de realizar acessos ao *Cloud SQL*, permitindo assim uma melhor performance na resposta a pedidos dos utilizadores.

Dessa forma, a performance do Sistema de Informação pode ainda ser melhorada com recurso à aplicação do Redis, sendo uma funcionalidade importante a desenvolver num futuro próximo para que o Sistema esteja pronto para disponibilizar a melhor experiência de utilização aos seus clientes.

6.2.3 Visualizações de dados

Como foi referido previamente em 4.1.4, o Sistema de Informação começou a ser preparado através da criação de vistas materializadas para a disponibilização de dados analíticos que necessitarão de ser partilhados com as autoridades e outras entidades especializadas na área.

Uma vez que a listagem final dos requisitos analíticos está, ainda, pendente, este é um processo que deverá ser concluído até ao final do projeto de forma a garantir que este cumpre com todas as fases propostas. Para isso, após a decisão final dos recursos a apresentar, poderá ser necessário proceder à criação de um *data warehousing* de forma a

dar suporte a esta funcionalidade, pelo que o sistema deverá ser estendido de forma a suportar essa integração.

Por fim, delineada a abordagem em relação aos dados a serem disponibilizados, será do interesse do projeto originar ferramentas de visualizações de dados, de forma a facilitar quer a receção quer a interpretação dos dados por parte das entidades competentes.

6.2.4 Envio de Dados para o Mapbox

O Sistema de Informação cumpriu com o objetivo principal de armazenar informação sobre as mais diversas FGCI presentes no território nacional, no entanto, devido à restrição do tamanho das respostas do GAE ser de 32MB, pode vir a tornar-se pesado que a aplicação móvel requisite estas porções por vários pedidos ao Sistema de Informação.

Para isso, atualmente o processo de envio de faixas para a aplicação móvel, baseia-se no acesso das faixas presentes no sistema por parte do QGIS, sendo este responsável pela criação do *shapefile* que é, posteriormente, enviado e colocado no MapBox da aplicação móvel.

No entanto, o MapBox disponibiliza uma API que facilita este processo, permitindo a ligação direta entre o *App Engine* e o MapBox associado à aplicação móvel. Desta forma, o processo de envio de dados poderá consistir na obtenção dum GeoJSON que será diretamente enviado à camada do MapBox que a aplicação móvel suporta.

6.2.5 Integração de código do Google Earth Engine

No âmbito do projeto Floresta Limpa continuam a ser desenvolvidas técnicas de aprendizagem automática que recorrem às bibliotecas do GEE. Embora já tenham sido introduzidas ferramentas que integram funcionalidades do GEE no GAE, nomeadamente o acesso ao vasto catálogo de dados deste, o Sistema pode ainda ser desenvolvido de forma a integrar os *scripts* elaborados nesta área do projeto, onde são definidas técnicas de pré-processamento dos dados obtidos, dando origem a séries temporais e informação adicional sobre o estado das faixas.

No entanto, é preciso ter atenção ao facto de computações pesadas terem um custo adicional do lado do GAE. Para isso, a integração do código do GAE presente no Sistema deve apenas chamar a criação de tarefas que serão realizadas na parte do GEE para computações mais pesadas. Dessa forma, a infraestrutura do *Earth Engine* será responsável pelo processamento dessas tarefas, evitando a sobrecarga do *App Engine* com estes processos.

Bibliografia

[1] *About high availability*. URL: https://cloud.google.com/sql/docs/mysql/high-availability (acedido em 2022-01-31) (ver p. 27).

[2] R. Afonso et al. «Assessment of Interventions in Fuel Management Zones Using Remote Sensing». Em: *ISPRS International Journal of Geo-Information* 9.9 (2020). ISSN: 2220-9964. DOI: 10.3390/ijgi9090533. URL: https://www.mdpi.com/2220-9964/9/9/533 (ver p. 2).

[3] R. F. S. Afonso. *Avaliação por Deteção Remota do Efeito das Operações de Limpeza nas Faixas de Gestão de Combustível de Incêndios*. 2019. URL: http://hdl.handle.net/10362/93767 (acedido em 2022-01-27) (ver p. 32).

[4] M. da Agricultura do Desenvolvimento Rural e das Pescas. *Decreto-Lei n.º 124/2006*. 2006-06-28. URL: https://dre.pt/dre/detalhe/decreto-lei/124-2006-358491 (acedido em 2022-01-17) (ver p. 1).

[5] *An Overview of App Engine*. URL: https://cloud.google.com/appengine/docs/standard/java-gen2/an-overview-of-app-engine (acedido em 2022-01-31) (ver p. 24).

[6] *Apache JMeter*. URL: https://jmeter.apache.org/ (acedido em 2023-03-25) (ver p. 101).

[7] *Carta de Uso e Ocupação do Solo 2018*. URL: https://snig.dgterritorio.gov.pt/rndg/srv/por/catalog.search#/metadata/b498e89c-1093-4793-ad22-63516062891b (acedido em 2023-03-01) (ver p. 14).

[8] *Choosing between Native mode and Datastore mode*. URL: https://cloud.google.com/datastore/docs/firestore-or-datastore (acedido em 2022-02-01) (ver p. 25).

[9] P. do Conselho de Ministros. *Decreto-Lei n.º 82/2021*. 2021-10-13. URL: https://dre.pt/dre/legislacao-consolidada/decreto-lei/2021-172745166 (acedido em 2022-01-26) (ver pp. 1, 2, 6).

[10] I. da Conservação da Natureza e das Florestas. *Plano Municipal de Defesa da Floresta Contra Incêndios*. 2012. URL: http://www2.icnf.pt/portal/florestas/dfci/Resource/doc/Guia-Tecnico-PMDFCI-AFN-Abril2012-v1.pdf (acedido em 2022-01-28) (ver pp. 7, 13–15, 142–144).

[11] I. da Conservação da Natureza e das Florestas. *Manual de Rede Primária*. 2014. URL: http://www2.icnf.pt/portal/florestas/dfci/Resource/doc/cartografia-dfci/manual-RPFGC-20mai2014.pdf (acedido em 2022-01-27) (ver pp. 6, 7).

[12] I. da Conservação da Natureza e das Florestas. *8.º Relatório Provisório de Incêndios Rurais*. URL: https://www.icnf.pt/api/file/doc/504914cdd1a211bb (acedido em 2022-01-17) (ver p. 1).

[13] I. da Conservação da Natureza e das Florestas. *Inventário Florestal Nacional*. URL: http://www2.icnf.pt/portal/florestas/ifn (acedido em 2022-01-29) (ver p. 17).

[14] I. da Conservação da Natureza e das Florestas. *Perigosidade de Incêndio Rural*. URL: http://www2.icnf.pt/portal/florestas/dfci/inc/cartografia/perigosidade (acedido em 2022-01-29) (ver p. 16).

[15] I. da Conservação da Natureza e das Florestas. *Planos PMDFCI - Público*. URL: https://fogos.icnf.pt/infoPMDFCI/PMDFCI_PUBLICOlist.asp (acedido em 2022-01-30) (ver p. 3).

[16] I. da Conservação da Natureza e das Florestas. *Territórios Ardidos*. URL: https://sig.icnf.pt/portal/home/item.html?id=983c4e6c4d5b4666b258a3ad5f3ea5af (acedido em 2022-01-29) (ver p. 17).

[17] K. Eldrandaly. «GIS AND SPATIAL DECISION MAKING». Em: 2011-01. ISBN: 978-1-61209-925-5 (ver pp. 9, 10).

[18] V. Gandhi. «Vector Data». Em: *Encyclopedia of GIS*. Ed. por S. Shekhar e H. Xiong. Boston, MA: Springer US, 2008, pp. 1217–1221. ISBN: 978-0-387-35973-1. DOI: 10.1007/978-0-387-35973-1_1438. URL: https://doi.org/10.1007/978-0-387-35973-1_1438 (ver p. 10).

[19] GDAL/OGR contributors. *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. 2022. DOI: 10.5281/zenodo.5884351. URL: https://gdal.org (ver p. 22).

[20] *Geodata*. URL: https://www.ibm.com/topics/geospatial-data (ver p. 8).

[21] *Google Cloud Logging*. URL: https://cloud.google.com/logging/docs/overview (acedido em 2023-03-10) (ver p. 31).

[22] N. Gorelick et al. «Google Earth Engine: Planetary-scale geospatial analysis for everyone». Em: *Remote Sensing of Environment* 202 (2017). Big Remotely Sensed Data: tools, applications and experiences, pp. 18–27. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2017.06.031. URL: https://www.sciencedirect.com/science/article/pii/S0034425717302900 (ver p. 29).

[23] A. Gupta et al. «Deploying an Application using Google Cloud Platform». Em: *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. 2020, pp. 236–239. DOI: 10.1109/ICIMIA48430.2020.9074911 (ver pp. 22, 34).

[24] P. Guth et al. «Digital Elevation Models: Terminology and Definitions». Em: *Remote Sensing* 13 (2021-09). DOI: 10.3390/rs13183581 (ver pp. 17, 18).

[25] E. Hoel. «Data Models in Commercial GIS Systems». Em: *Encyclopedia of GIS*. Ed. por S. Shekhar e H. Xiong. Boston, MA: Springer US, 2008, pp. 215–219. ISBN: 978-0-387-35973-1. DOI: 10.1007/978-0-387-35973-1_247. URL: https://doi.org/10.1007/978-0-387-35973-1_247 (ver p. 9).

[26] O. G. C. Inc. *OpenGIS® Web Map Server Implementation Specification*. 2006. URL: https://portal.ogc.org/files/?artifact_id=14416 (acedido em 2022-01-27) (ver p. 10).

[27] O. G. C. Inc. *OpenGIS Web Feature Service 2.0 Interface Standard – With Corrigendum*. 2014. URL: https://portal.opengeospatial.org/files/09-025r2 (acedido em 2022-01-27) (ver p. 11).

[28] O. G. C. Inc. *About OGC*. URL: https://www.ogc.org/about (acedido em 2022-01-27) (ver p. 10).

[29] A. Interna. *Decreto-Lei n.º 10/2018*. 2018-02-14. URL: https://dre.pt/dre/detalhe/decreto-lei/10-2018-114685734 (acedido em 2022-01-27) (ver pp. 6, 8).

[30] *Landsat 8*. URL: https://landsat.gsfc.nasa.gov/satellites/landsat-8/ (acedido em 2022-02-03) (ver p. 21).

[31] H. Lim. «Raster Data». Em: *Encyclopedia of GIS*. Ed. por S. Shekhar e H. Xiong. Boston, MA: Springer US, 2008, pp. 949–955. ISBN: 978-0-387-35973-1. DOI: 10.1007/978-0-387-35973-1_1080. URL: https://doi.org/10.1007/978-0-387-35973-1_1080 (ver p. 9).

[32] P. A. Longley et al. *Geographic Information Science and Systems*. 4ª ed. Wiley, 2015. ISBN: 978-1-119-03130-7 (ver p. 9).

[33] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (ver p. ii).

[34] A. T. M. Malafaia. *Sistema de Informação para Projetos de Monitoração Fitossanitária*. 2019. URL: http://hdl.handle.net/10362/92289 (acedido em 2022-01-27) (ver p. 33).

[35] *Mapa de Perigosidade Conjuntural*. URL: https://sig.icnf.pt/portal/home/item.html?id=f91cfac165904f999c2f1c65d3ed7914 (acedido em 2023-03-01) (ver p. 15).

[36] *Mapa de Perigosidade Estrutural*. URL: https://sig.icnf.pt/portal/home/item.html?id=65e7a435415e467b82f84b0640205409 (acedido em 2023-03-01) (ver p. 15).

[37] *Mapa de Risco de Incêndio Rural*. URL: https://www.ipma.pt/pt/riscoincendio/rcm.pt/ (acedido em 2023-03-01) (ver p. 16).

[38] I. P. do Mar e da Atmosfera. *Cálculo do Índice de Risco de Incêndio Rural Risco Conjuntural e Meteorológico – RCM*. 2020. URL: https://www.ipma.pt/export/sites/ipma/bin/docs/relatorios/meteorologia/nota-metodologica-calculo-RCM2020-v20200713.pdf (acedido em 2022-01-29) (ver p. 16).

[39] G. Mateo-García, J. Muñoz-Marí e L. Gómez-Chova. «Cloud detection on the Google Earth engine platform». Em: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2017, pp. 1942–1945. DOI: 10.1109/IGARSS.2017.8127359 (ver p. 29).

[40] *MODIS*. URL: https://modis.gsfc.nasa.gov/ (acedido em 2022-02-03) (ver p. 21).

[41] *PCIF/MOG/0161/2019*. URL: https://www.fct.pt/apoios/projectos/consulta/vglobal_projecto?idProjecto=159692&idElemConcurso=14228 (acedido em 2022-02-04) (ver p. 2).

[42] *PostGIS*. URL: https://postgis.net/ (acedido em 2022-01-31) (ver p. 28).

[43] *PostgreSQL*. URL: https://www.postgresql.org/ (acedido em 2022-01-30) (ver p. 27).

[44] *Postman*. URL: https://www.postman.com/product/what-is-postman/ (acedido em 2023-03-04) (ver p. 32).

[45] *Put your archive data on ice with new storage offering*. URL: https://cloud.google.com/blog/products/storage-data-transfer/archive-storage-class-for-coldest-data-now-available (acedido em 2022-01-31) (ver p. 26).

[46] *QGIS*. URL: https://www.qgis.org/ (acedido em 2022-01-30) (ver p. 22).

[47] *Richardson Maturity Model*. URL: https://martinfowler.com/articles/richardsonMaturityModel.html (acedido em 2022-09-11) (ver pp. 49, 109).

[48] *Sequelize*. URL: https://sequelize.org/ (acedido em 2023-03-01) (ver p. 30).

[49] *Sequelize Auto*. URL: https://github.com/sequelize/sequelize-auto (acedido em 2023-03-02) (ver p. 62).

[50] A. Shelestov et al. «Large scale crop classification using Google earth engine platform». Em: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2017, pp. 3696–3699. DOI: 10.1109/IGARSS.2017.8127801 (ver p. 34).

[51] *StarUML*. URL: https://staruml.io (acedido em 2023-03-01) (ver p. 61).

[52] *Structure of a Google App Engine application*. URL: https://www.oreilly.com/library/view/google-cloud-platform/9781788837675/a10eee42-5613-463e-8bd0-d76dd8ef6216.xhtml (acedido em 2022-01-31) (ver p. 24).

[53] D.-G. do Território. *Especificações Técnicas da Carta de Uso e Ocupação do Solo (COS) de Portugal Continental para 2018*. 2019. URL: http://mapas.dgterritorio.pt/atom-dgt/pdf-cous/COS2018/ET-COS-2018_v1.pdf (acedido em 2022-01-28) (ver p. 14).

[54] D.-G. do Território. *Ortofotos*. URL: https://www.dgterritorio.gov.pt/cartografia/cartografia-topografica/ortofotos/ortofotos-digitais (acedido em 2022-01-29) (ver p. 18).

[55] *What is Cloud SQL?* URL: https://cloud.google.com/sql/docs/introduction (acedido em 2022-02-01) (ver p. 26).

[56] *What is Cloud Storage?* URL: https://cloud.google.com/storage/docs/introduction (acedido em 2022-02-01) (ver p. 25).

[57] *What is Spatial Data and Non-Spatial Data?* URL: https://www.safe.com/blog/2021/10/non-spatial-data-difference-fme/ (acedido em 2023-03-17) (ver p. 10).

[58] L. Zhang e J. Yi. «Management methods of spatial data based on PostGIS». Em: *2010 Second Pacific-Asia Conference on Circuits, Communications and System*. Vol. 1. 2010, pp. 410–413. DOI: 10.1109/PACCS.2010.5626962 (ver p. 29).

Apêndice A

Listagem dos Requisitos Funcionais

- **Utilizadores Genéricos:**
  - O sistema de informação deverá permitir a criação de contas para os utilizadores.
  - O sistema de informação deverá permitir a edição dos dados de cada conta.
  - O sistema de informação deverá permitir a remoção de contas.
  - O sistema de informação deverá apresentar a faixa de gestão de combustível onde um utilizador se encontra, bem como carregar os seus dados informativos.
  - O sistema de informação deverá apresentar as faixas de gestão de combustível da zona onde um utilizador se encontra, bem como carregar os seus dados informativos, assim que possível (quando houver rede).
  - O sistema de informação deverá apresentar uma vista cartográfica das faixas para os utilizadores nos vários dispositivos.
  - O sistema de informação deverá apresentar uma vista e modelos 3D das estruturas para os utilizadores da aplicação móvel.
  - O sistema de informação deverá apresentar uma lista das faixas de que necessitam de dados no terreno para serem consultadas pelos utilizadores.
- **Organizações e CEO's:**
  - O sistema de informação deverá permitir a introdução de cargos para outros utilizadores dentro duma organização.
- **Organizações e/ou membros:**

  - O sistema de informação deverá permitir a inserção de dados open-source, quer por parte das entidades responsáveis pela emissão destes dados quer por parte dos membros do projeto.

- **Pessoas / Voluntários:**

  - O sistema de informação deverá permitir a visualização bem como o armazenamento de fotos tiradas no local pelos utilizadores.
  - O sistema de informação deverá suportar a adição de áreas de interesse por parte dos utilizadores de modo a que estas passem a ser monitorizadas.
  - O sistema de informação deverá permitir que os utilizadores sejam avisados quando uma das suas áreas de interesse necessita de ser limpa.
  - O sistema de informação deverá permitir que os utilizadores saibam qual é o período de obrigatoriedade de limpeza de faixas.
  - O sistema de informação deverá permitir que caso um proprietário do setor duma faixa se encontre no sistema que este seja notificado caso o seu segmento de faixa necessite de intervenção.

- **Voluntários:**

  - O sistema de informação deverá apresentar as contribuições que cada utilizador fez na inserção de novos dados no sistema.
  - O sistema de informação deverá fornecer *feedback* aos utilizadores sobre a qualidade da sua contribuição.

- **Membros do projeto:**

  - O sistema de informação deverá apresentar os dados obtidos através de satélites para serem aplicados nos algoritmos por parte dos membros do projeto.
  - O sistema de informação deverá apresentar imagens cartográficas de todas as faixas, de forma a poderem ser utilizadas nos algoritmos.
  - O sistema de informação deverá apresentar dados históricos e de previsão meteorológica para serem incluídos nos algoritmos de aprendizagem automática por parte dos membros do projeto.
  - O sistema de informação deverá apresentar dados obtidos através de visualização no terreno, de forma a funcionarem como *groundtruth* para os algoritmos de aprendizagem automática.
  - O sistema de informação deverá apresentar estatísticas sobre a cartografia base, o estado e a condição dos recursos florestais para serem aplicados nos algoritmos.

- O sistema de informação deverá permitir a edição do estado de limpeza das faixas de combustível.

- **Faixas e áreas de interesse:**

  - O sistema de informação deverá apresentar características das faixas, nomeadamente altitude, área, tipo de cobertura, índice de vegetação e o responsável pela limpeza da faixa.
  - O sistema de informação deverá permitir a filtragem dos atributos enumerados no ponto anterior.
  - O sistema de informação deverá possuir informação sobre as datas de intervenção e as formas de intervenção para permitir um estudo temporal da evolução das faixas.
  - O sistema de informação deverá permitir a filtragem das faixas por tipo, de forma a facilitar a aplicação dos requisitos estipulados na lei para cada tipo de faixa.
  - O sistema de informação deverá permitir a filtragem das faixas por estado de limpeza para identificar as faixas que carecem de intervenção.
  - O sistema de informação deverá permitir a filtragem do tipo de intervenção, de forma a permitir identificar as áreas ardidas.
  - O sistema de informação deverá apresentar dados sobre a vegetação ao longo do tempo nas faixas, armazenando séries temporais.
  - O sistema de informação deverá permitir o acesso aos dados de risco de incêndio de cada área.

- **Dados estatísticos:**

  - O sistema de informação deverá providenciar dados de caráter analítico sobre os diversos tipos de dados que tem no sistema (número de faixas intervencionadas, número de faixas por intervencionar, número de faixas classificadas de forma correta, entre outros).

Apêndice B

Listagem dos Requisitos Informacionais

- Faixas de Gestão de Combustível de Incêndios georreferenciadas. (Shapefile Format)
- Áreas de Interesse a monitorizar georreferenciadas. (Shapefile Format)
- Áreas ardidas georreferenciadas. (Shapefile Format)
- Carta de ocupação de solo (Shapefile Format)
- Mapa de Perigosidade Estrutural 2020-2030 (GeoTIFF / KML Format)
- Mapa de Perigosidade Conjuntural (GeoTIFF / KML Format)
- Ortofotos (WMS Service)
- Fotografias / Vídeos dos utilizadores (JPG, PNG, GIF, mp4, avi, wmv e outros formatos associados)
- Inventário Florestal Nacional (Shapefile Format)
- Cartografia estática, como por exemplo Google Maps, OpenStreetMaps (KML, OSM Format)
- Modelos 3D de paisagens, edifícios e estradas (FBX, DEM Format)
- Produtos de satélites (Rasters, Shapefiles, HDF5, GeoTIFF, JPEG2000 Format)
- Ficheiros gerados a partir dos algoritmos de aprendizagem automática (CSV Format)
- Informação Meteorológica (API IPMA, API OpenWeatherMap – JSON Format)

Apêndice C

Listagem dos casos de uso da API REST

- **Autenticação**

  C1. O agente autentica-se pelo provedor da Google.

  C2. O agente autentica-se pela aplicação móvel.

  C3. O agente autentica-se com a inserção do email e password registada no Sistema.

- **Contas**

  C4. O agente consulta as contas.

  C5. O agente insere uma ou várias contas.

  C6. O agente edita uma conta.

  C7. O agente elimina uma conta.

- **Divisões Administrativas**

  C8. O agente consulta as divisões administrativas.

  C9. O agente insere uma ou várias divisões administrativas.

  C10. O agente edita uma divisão administrativa.

  C11. O agente elimina uma divisão administrativa.

- **Alertas**

  C12. O agente consulta os alertas.

  C13. O agente insere um ou vários alertas.

  C14. O agente edita um alerta.

C15. O agente elimina um alerta.

- **Respostas**

C16. O agente consulta as respostas dum registo.

C17. O agente insere uma ou várias respostas num registo.

C18. O agente edita uma resposta dum registo.

C19. O agente elimina uma resposta dum registo.

- **Opções de resposta**

C20. O agente consulta as opções de resposta.

C21. O agente insere uma ou várias opções de resposta.

C22. O agente edita uma opção de resposta.

C23. O agente elimina uma opção de resposta.

- **Configurações da Aplicação**

C24. O agente consulta as configurações da aplicação.

C25. O agente insere uma configuração da aplicação.

C26. O agente elimina uma configuração da aplicação.

- **Áreas de Interesse**

C27. O agente consulta as áreas de interesse.

C28. O agente cria uma ou várias áreas de interesse e os seus respetivos objetos espaciais.

C29. O agente edita uma área de interesse.

C30. O agente elimina uma área de interesse.

- **Tipos de Áreas de Interesse**

C31. O agente consulta os tipos de áreas de interesse.

C32. O agente insere um ou vários tipos de áreas de interesse.

C33. O agente edita um tipo de área de interesse.

C34. O agente elimina um tipo de área de interesse.

- **Tipos de Realidade Aumentada**

C35. O agente consulta os tipos de realidade aumentada.

C36. O agente insere um ou vários tipos de realidade aumentada.

C37. O agente edita um tipo de realidade aumentada.

C38. O agente elimina um tipo de realidade aumentada.

- **Fornecedores de Dados**

C39. O agente consulta os fornecedores de dados.

C40. O agente insere um ou vários fornecedores de dados.

C41. O agente edita um fornecedor de dados.

C42. O agente elimina um fornecedor de dados.

- **Tipos de Fornecedores de Dados**

C43. O agente consulta os tipos de fornecedores de dados.

C44. O agente insere um ou vários tipos de fornecedores de dados.

C45. O agente edita um tipo de fornecedor de dados.

C46. O agente elimina um tipo de fornecedor de dados.

- **Fontes de Dados**

C47. O agente consulta as fontes de dados.

C48. O agente insere uma ou várias fontes de dados.

C49. O agente edita uma fonte de dados.

C50. O agente elimina uma fonte de dados.

- **Tipos de Fontes de Dados**

C51. O agente consulta os tipos de fontes de dados.

C52. O agente insere um ou vários tipos de fontes de dados.

C53. O agente edita um tipo de fonte de dados.

C54. O agente elimina um tipo de fonte de dados.

- **Séries**

C55. O agente consulta as séries de uma fonte de dados.

C56. O agente insere uma ou várias séries de uma fonte de dados.

C57. O agente edita uma série.

C58. O agente elimina uma série.

C59. O agente consulta os dados raster ou vetoriais de uma série temporal.

- **Tipos de Séries**

  C60. O agente consulta os tipos de séries.

  C61. O agente insere um ou vários tipos séries.

  C62. O agente edita um tipo de séries.

  C63. O agente elimina um tipo de séries.

- **Relações entre Séries**

  C64. O agente consulta as relações duma série.

  C65. O agente insere uma associação de relação entre duas séries.

  C66. O agente edita uma associação de relação entre duas séries.

  C67. O agente elimina uma associação de relação entre duas séries.

- **Tipos de Relações de Séries**

  C68. O agente consulta os tipos de relações de séries.

  C69. O agente insere um ou vários tipos de relações de séries.

  C70. O agente edita um tipo de relação de séries.

  C71. O agente elimina um tipo de relação de séries.

- **Variáveis de Séries**

  C72. O agente consulta as variáveis de séries.

  C73. O agente insere uma ou várias variáveis de séries.

  C74. O agente edita uma variável de série.

  C75. O agente elimina uma variável de série.

- **Períodos de Séries**

  C76. O agente consulta os períodos de série.

  C77. O agente insere um ou vários períodos de série.

  C78. O agente edita um período de série.

  C79. O agente elimina um período de série.

- **Tipos de Dados**

  C80. O agente consulta os tipos de dados.

  C81. O agente insere um ou vários tipos de dados.

C82. O agente edita um tipo de dados.

C83. O agente elimina um tipo de dados.

- **Dispositivos**

C84. O agente consulta os dispositivos.

C85. O agente insere um ou vários dispositivos.

C86. O agente edita um dispositivo.

C87. O agente elimina um dispositivo.

- **Tipos de Dispositivo**

C88. O agente consulta os tipos de dispositivo.

C89. O agente insere um ou vários tipos de dispositivo.

C90. O agente edita um tipo de dispositivo.

C91. O agente elimina um tipo de dispositivo.

- **Entidades**

C92. O agente consulta as entidades.

C93. O agente insere uma ou várias entidades.

C94. O agente edita uma entidade.

C95. O agente elimina uma entidade.

- **Eventos**

C96. O agente consulta os eventos.

C97. O agente insere um ou vários eventos.

C98. O agente edita um evento.

C99. O agente elimina um evento.

- **Tipos de Evento**

C100. O agente consulta os tipos de evento.

C101. O agente insere um ou vários tipos de evento.

C102. O agente edita um tipo de evento.

C103. O agente elimina um tipo de evento.

- **Faixas de Gestão de Combustível**

C104. O agente consulta as faixas de gestão de combustível.

C105. O agente insere uma ou várias faixas de gestão de combustível e o seu respetivo objeto espacial.

C106. O agente edita uma faixa de gestão de combustível.

C107. O agente elimina uma faixa de gestão de combustível.

- **Tipos de Faixa de Gestão de Combustível**

C108. O agente consulta os tipos de faixa de gestão de combustível.

C109. O agente insere um ou vários tipos de faixa de gestão de combustível.

C110. O agente edita um tipo de faixa de gestão de combustível.

C111. O agente elimina um tipo de faixa de gestão de combustível.

- **Intervenções**

C112. O agente consulta as intervenções de um objeto espacial num certo período.

C113. O agente insere uma ou várias intervenções num objeto espacial num certo período

C114. O agente edita uma intervenção.

C115. O agente elimina uma intervenção.

- **Tipos de Intervenções**

C116. O agente consulta os tipos intervenções.

C117. O agente insere um ou vários tipos de intervenções

C118. O agente edita um tipo de intervenção.

C119. O agente elimina um tipo de intervenção.

- **Multimédia**

C120. O agente consulta os conteúdos multimédia.

C121. O agente insere um ou vários conteúdos multimédia.

C122. O agente edita um conteúdo multimédia.

C123. O agente elimina um conteúdo multimédia.

- **Tipos de Multimédia**

C124. O agente consulta os tipos de conteúdo multimédia.

C125. O agente insere um ou vários tipos de conteúdo multimédia.

C126. O agente edita um tipo de conteúdo multimédia.

C127. O agente elimina um tipo de conteúdo multimédia.

- **Tipos de Membro**

C128. O agente consulta os tipos de membro.

C129. O agente insere um ou vários tipos de membro.

C130. O agente edita um tipo de membro.

C131. O agente elimina um tipo de membro.

- **Esquemas de Metadados**

C132. O agente consulta os esquemas de metadados.

C133. O agente insere um ou vários esquemas de metadados.

C134. O agente edita um esquema de metadados.

C135. O agente elimina um esquema de metadados.

- **Esquemas de Propriedades**

C136. O agente consulta os esquemas de propriedades.

C137. O agente insere um ou vários esquemas de propriedades.

C138. O agente edita um esquema de propriedades.

C139. O agente elimina um esquema de propriedades.

- **Monitorizações**

C140. O agente consulta as monitorizações.

C141. O agente insere uma ou várias monitorizações.

C142. O agente edita uma monitorização.

C143. O agente elimina uma monitorização.

- **Notícias**

C144. O agente consulta as notícias.

C145. O agente insere uma ou várias notícias.

C146. O agente edita uma notícia.

C147. O agente elimina uma notícia.

- **Organizações**

C148. O agente consulta as organizações.

C149. O agente insere uma ou várias organizações.

C150. O agente edita uma organização.

C151. O agente elimina uma organização.

- **Membros de Organizações**

C152. O agente consulta os membros de uma dada organização.

C153. O agente consulta as organizações de que um membro faz parte.

C154. O agente cria uma associação entre uma pessoa e uma organização.

C155. O agente edita uma associação entre uma pessoa e uma organização.

C156. O agente elimina uma associação entre uma pessoa e uma organização.

- **Pessoas**

C157. O agente consulta as pessoas.

C158. O agente insere uma ou várias pessoas.

C159. O agente edita uma pessoa.

C160. O agente elimina uma pessoa.

- **Períodos**

C161. O agente consulta os períodos.

C162. O agente insere um ou vários períodos.

C163. O agente edita um período.

C164. O agente elimina um período.

- **Tipos de Período**

C165. O agente consulta os tipos de período.

C166. O agente insere um ou vários tipos de período.

C167. O agente edita um tipo de período.

C168. O agente elimina um tipo de período.

- **Produtos**

C169. O agente consulta os produtos de uma fonte de dados.

C170. O agente insere um ou vários produtos de uma fonte de dados.

C171. O agente edita um produto.

C172. O agente elimina um produto.

- **Profissionais**

C173. O agente consulta os profissionais.

C174. O agente marca uma pessoa como profissional e insere a sua informação.

C175. O agente edita um profissional.

C176. O agente elimina um profissional.

- **Voluntários**

C177. O agente consulta os profissionais.

C178. O agente marca uma pessoa como voluntário e insere a sua informação.

C179. O agente edita um voluntário.

C180. O agente elimina um voluntário.

- **Protocolos**

C181. O agente consulta os protocolos.

C182. O agente insere um ou vários protocolos.

C183. O agente edita um protocolo.

C184. O agente elimina um protocolo.

- **Questões**

C185. O agente consulta as questões.

C186. O agente insere uma ou várias questões.

C187. O agente edita uma questão.

C188. O agente elimina uma questão.

- **Tipos de Questão**

C189. O agente consulta os tipos de questão.

C190. O agente insere um ou vários tipos de questão

C191. O agente edita um tipo de questão.

C192. O agente elimina um tipo de questão.

- **Questões de Protocolos**

C193. O agente consulta as questões de um protocolo.

C194. O agente consulta os protocolos de que uma questão faz parte.

C195. O agente insere uma associação entre uma questão e um protocolo.

C196. O agente edita uma associação entre uma questão e um protocolo.

C197. O agente elimina uma associação entre uma questão e um protocolo.

- **Opções de resposta de Questões**

C198. O agente consulta as opções de resposta de uma questão.

C199. O agente insere uma associação entre uma opção de resposta e uma questão.

C200. O agente edita uma associação entre uma opção de resposta e uma questão.

C201. O agente elimina uma associação entre uma opção de resposta e uma questão.

- **Multimédia de Questões**

C202. O agente consulta o conteúdo multimédia associado à pergunta.

C203. O agente consulta o conteúdo multimédia associado à pergunta numa resposta de um dado registo.

C204. O agente insere uma associação entre um conteúdo multimédia e uma questão.

C205. O agente insere uma associação entre um conteúdo multimédia e uma questão num dado registo.

C206. O agente edita uma associação entre uma opção de resposta e uma questão.

C207. O agente elimina uma associação entre uma opção de resposta e uma questão.

- **Registos**

C208. O agente consulta os registos.

C209. O agente insere um ou vários registos.

C210. O agente edita um registo.

C211. O agente elimina um registo.

- **Regras**

C212. O agente consulta as regras.

C213. O agente insere uma ou várias regras.

C214. O agente edita uma regra.

C215. O agente elimina uma regra.

- **Recompensas**

C216. O agente consulta as recompensas de um voluntário.

C217. O agente insere uma ou várias recompensas num voluntário.

C218. O agente edita uma recompensa.

C219. O agente elimina uma recompensa.

- **Objetos Espaciais**

C220. O agente consulta os objetos espaciais.

C221. O agente insere um ou vários objetos espaciais.

C222. O agente edita um objeto espacial.

C223. O agente elimina um objeto espacial.

- **Tipos de Objetos Espaciais**

C224. O agente consulta os tipos de objetos espaciais.

C225. O agente insere um ou vários tipos de objetos espaciais.

C226. O agente edita um tipo de objeto espacial.

C227. O agente elimina um tipo de objeto espacial.

- **Relações Topológicas**

C228. O agente consulta as relações topológicas de um objeto espacial.

C229. O agente insere uma ou várias relações topológicas entre dois objetos espaciais.

C230. O agente edita uma relação topológica entre dois objetos espaciais.

C231. O agente elimina uma relação topológica entre dois objetos espaciais.

- **Tipos de Relações Topológicas**

C232. O agente consulta os tipos de relações topológicas.

C233. O agente insere um ou vários tipos de relações topológicas.

C234. O agente edita um tipo de relação topológica.

C235. O agente elimina um tipo de relação topológica.

- **Unidades**

C236. O agente consulta as unidades.

C237. O agente insere uma ou várias unidades.

C238. O agente edita uma unidade.

C239. O agente elimina uma unidade.

- **Pontos de Recolha**

C240. O agente consulta os pontos de recolha de um registo.

C241. O agente insere um ou vários pontos de recolha de um registo.

C242. O agente edita um ponto de recolha.

C243. O agente elimina um ponto de recolha.

- **Estados**

C244. O agente consulta os estados associados a um objeto espacial num certo período de tempo.

C245. O agente insere um ou vários estados associados a um objeto espacial num certo período de tempo

C246. O agente edita um estado.

C247. O agente elimina um estado.

- **Registos de Voluntários**

C248. O agente consulta os registos de um voluntário.

C249. O agente consulta os voluntários que realizaram um registo.

C250. O agente cria uma associação entre um registo e um voluntário.

C251. O agente edita uma associação entre um registo e um voluntário.

C252. O agente elimina uma associação entre um registo e um voluntário.

- **Monitorizações de Profissionais**

C253. O agente consulta as monitorizações de um profissional.

C254. O agente consulta os profissionais que realizaram uma monitorização.

C255. O agente cria uma associação entre uma monitorização e um profissional.

C256. O agente edita uma associação entre uma monitorização e um profissional.

C257. O agente elimina uma associação entre um registo e um voluntário.

- **Contactos**

C258. O agente consulta os contactos das entidades numa divisão administrativa.

C259. O agente cria uma associação entre uma entidade e uma divisão administrativa e insere informação sobre o seu contacto.

C260. O agente edita uma associação entre uma entidade e uma divisão administrativa.

C261. O agente elimina uma associação de contacto entre uma entidade e uma divisão administrativa.

- **Dispositivos de Contas**

C262. O agente consulta os dispositivos de uma conta.

C263. O agente consulta as contas associadas a um dispositivo.

C264. O agente cria uma associação entre uma conta e um dispositivo.

C265. O agente edita uma associação entre uma conta e um dispositivo.

C266. O agente elimina uma associação entre uma conta e um dispositivo.

- **Objetos de interesse de uma conta**

C267. O agente consulta os objetos espaciais de interesse associados a uma conta.

C268. O agente cria uma associação entre uma conta e um objeto espacial.

C269. O agente edita uma associação entre uma conta e um objeto espacial.

C270. O agente elimina uma associação entre uma conta e um objeto espacial.

- **Divisões de interesse de uma conta**

C271. O agente consulta as divisões administrativas de interesse associados a uma conta.

C272. O agente cria uma associação entre uma conta e uma divisão administrativa.

C273. O agente edita uma associação entre uma conta e uma divisão administrativa.

C274. O agente elimina uma associação entre uma conta e uma divisão administrativa.

- **Objetos espaciais em Divisões de Interesse**

C275. O agente consulta os objetos espaciais de uma divisão de interesse.

C276. O agente consulta as divisões de interesse em que um objeto espacial se situa.

C277. O agente cria uma associação entre um objeto espacial e uma divisão administrativa.

C278. O agente edita uma associação entre um objeto espacial e uma divisão administrativa.

C279. O agente elimina uma associação entre um objeto espacial e uma divisão administrativa.

- **Objetos espaciais em Períodos**

C280. O agente consulta os períodos associados a um objeto espacial.

C281. O agente consulta os objetos espaciais associados a um período.

C282. O agente cria uma associação entre um objeto espacial e um período.

C283. O agente edita uma associação entre um objeto espacial e um período.

C284. O agente elimina uma associação entre um objeto espacial e um período.

- **Entidades de Objetos espaciais em Períodos**

C285. O agente consulta as entidades de um objeto espacial num certo período.

C286. O agente consulta os objetos espaciais num certo período associados a uma entidade.

C287. O agente cria uma associação entre um objeto espacial num certo período e uma entidade.

C288. O agente edita uma associação entre um objeto espacial num certo período e uma entidade.

C289. O agente elimina uma associação entre um objeto espacial num certo período e uma entidade.

- **Tipos de relação entre Objetos espaciais em Períodos e Entidades**

C290. O agente consulta os tipos de relação entre objetos espaciais em certos períodos e entidades.

C291. O agente cria um ou vários tipos de relação entre objetos espaciais em certos períodos e entidades.

C292. O agente edita um tipo de relação entre objetos espaciais em certos períodos e entidades.

C293. O agente elimina um tipo de relação entre objetos espaciais em certos períodos e entidades.

- **Definições**

C294. O agente consulta as definições.

C295. O agente cria uma ou várias definições.

C296. O agente edita uma definição.

C297. O agente elimina uma definição.

- **Funções**

C298. O agente consulta as funções.

C299. O agente cria uma ou várias funções.

C300. O agente edita uma função.

C301. O agente elimina uma função.

- **Operações**

C302. O agente consulta as operações.

C303. O agente cria uma ou várias operações.

C304. O agente edita uma operação.

C305. O agente elimina uma operação.

- **Proibições**

C306. O agente consulta as proibições.

C307. O agente cria uma ou várias proibições.

C308. O agente edita uma proibição.

C309. O agente elimina uma proibição.

- **Operações de Funções**

C310. O agente consulta operações que uma determinada função pode realizar.

C311. O agente consulta que funções podem realizar uma determinada operação.

C312. O agente cria uma associação entre uma função e uma operação.

C313. O agente edita uma associação entre uma função e uma operação.

C314. O agente elimina uma associação entre uma função e uma operação.

- **Proibições de Funções**

C315. O agente consulta as proibições que uma determinada função tem.

C316. O agente consulta que funções têm uma determinada proibição.

C317. O agente cria uma associação entre uma função e uma proibição.

C318. O agente edita uma associação entre uma função e uma proibição.

C319. O agente elimina uma associação entre uma função e uma proibição.

- **Proibições de Contas**

C320. O agente consulta as proibições que uma determinada conta tem.

C321. O agente consulta que contas têm uma determinada proibição.

C322. O agente cria uma associação entre uma conta e uma proibição.

C323. O agente edita uma associação entre uma conta e uma proibição.

C324. O agente elimina uma associação entre uma conta e uma proibição.

- **Funções de Contas**

C325. O agente consulta as funções que uma determinada conta tem.

C326. O agente consulta que contas têm uma determinada função.

C327. O agente cria uma associação entre uma conta e uma função.

C328. O agente edita uma associação entre uma conta e uma função.

C329. O agente elimina uma associação entre uma conta e uma função.

- **Protocolos de Tipos de Áreas de Interesse**

C330. O agente consulta os protocolos de um tipo de área de interesse.

C331. O agente consulta os tipos de área de interesse que podem realizar um determinado protocolo.

C332. O agente cria uma associação entre um protocolo e um tipo de áreas de interesse.

C333. O agente edita uma associação entre um protocolo e um tipo de áreas de interesse.

C334. O agente elimina uma associação entre um protocolo e um tipo de áreas de interesse.

- **Protocolos de Tipos de Faixas**

C335. O agente consulta os protocolos de um tipo de faixa.

C336. O agente consulta os tipos de faixa que podem realizar um determinado protocolo.

C337. O agente cria uma associação entre um protocolo e um tipo de faixa.

C338. O agente edita uma associação entre um protocolo e um tipo de faixa.

C339. O agente elimina uma associação entre um protocolo e um tipo de faixa.

- **Regras de Tipos de Faixas**

C340. O agente consulta as regras de um tipo de faixa.

C341. O agente consulta os tipos de faixa que têm determinada regra.

C342. O agente cria uma associação entre uma regra e um tipo de faixa.

C343. O agente edita uma associação entre uma regra e um tipo de faixa.

C344. O agente elimina uma associação entre uma regra e um tipo de faixa.

- **Séries de Objetos Espaciais**

C345. O agente consulta as séries temporais relevantes para um objeto espacial.

C346. O agente consulta os objetos espaciais para qual uma determinada série é relevante.

C347. O agente cria uma associação entre um objeto espacial e uma série.

C348. O agente edita uma associação entre um objeto espacial e uma série.

C349. O agente elimina uma associação entre um objeto espacial e uma série.

- **Séries de Objetos Espaciais**

C350. O agente consulta as séries temporais relevantes para um objeto espacial.

C351. O agente consulta os objetos espaciais para qual uma determinada série é relevante.

C352. O agente cria uma associação entre um objeto espacial e uma série.

C353. O agente edita uma associação entre um objeto espacial e uma série.

C354. O agente elimina uma associação entre um objeto espacial e uma série.

- **Modelos de Classificação**

C355. O agente consulta os modelos de classificação.

C356. O agente insere um ou vários modelos de classificação.

C357. O agente edita um modelo de classificação.

C358. O agente elimina um modelo de classificação.

- **Dados de treino**

C359. O agente consulta os dados de treino de um modelo.

C360. O agente insere um ou vários dados como dados de treino.

C361. O agente edita um dado de treino.

C362. O agente elimina um dado de treino.

- **Dados Derivados**

C363. O agente consulta os dados derivados de dados de uma série temporal.

C364. O agente insere dados derivados de dados de uma série temporal.

C365. O agente edita os dados derivados.

C366. O agente elimina os dados derivados.

- **Dados *Groundtruth***

C367. O agente consulta quais são os dados *groundtruth* de um modelo.

C368. O agente insere um ou vários dados como dados *groundtruth* de um modelo.

C369. O agente edita um dado *groundtruth*.

C370. O agente elimina um dado dado *groundtruth*.

- **Resultados do Modelo**

C371. O agente consulta quais são os resultados do modelo.

C372. O agente insere um ou vários resultados do modelo.

C373. O agente edita um resultado do modelo.

C374. O agente elimina um resultado do modelo.

- **Casos de Uso Específicos da Aplicação Móvel**

C375. A aplicação móvel consulta quais são os protocolos ativos, obtendo informação sobre os restantes recursos associados.

C376. A aplicação móvel acede a um protocolo antigo, obtendo informação sobre os restantes recursos associados.

C377. A aplicação móvel consulta os registos de um utilizador, obtendo informação sobre os restantes recursos associados.

C378. A aplicação móvel requere novas hiperligações de *resumable uploads* para aceder a fotos.

C379. A aplicação móvel atualiza a última localização de um utilizador.

C380. A aplicação móvel consulta quais são as notícias da localização de um utilizador.

C381. A aplicação móvel consulta quais são as notícias globais.

Apêndice D

Listagem de *Endpoints* específicos da Aplicação Móvel

- **POST /mobileapp/records** – *Endpoint* criado com o objetivo de satisfazer uma das principais funções da aplicação móvel, permitindo criar um registo completo após a resposta a várias questões de um protocolo. Dessa forma, é possível a criação em simultâneo de múltiplos tuplos nas tabelas *records*, *answers*, *media* e *waypoints*, sendo assim, possível que a aplicação guarde vários registos em modo *offline* e que os envie assim que exista ligação à Internet. Este *endpoint* permite que sejam enviadas as várias respostas dadas às múltiplas questões do protocolo daquela faixa, bem como os múltiplos pontos onde o utilizador se encontrava a recolher informação, sem ser necessário o envio de novos pedidos por parte da aplicação móvel para cada uma das tabelas onde pretende inserir dados. Por fim, permite, ainda a criação de conteúdo multimédia que será armazenado na *Google Cloud Storage*, registando na base de dados um apontador para esse ficheiro.

- **GET /mobileapp/spatial_objects/waypoints** – *Endpoint* utilizado para apresentar os pontos de recolha associados a um objeto espacial, de forma a que seja possível que a aplicação móvel gere um *heatmap* com base nesses pontos, para que seja possível ao utilizador perceber que zonas dentro duma faixa carecem de informação. Uma vez que os pontos de recolha estão associados a cada registo realizado numa faixa num certo período de tempo, para reduzir o número de pedidos que a aplicação móvel teria de fazer foi identificada a necessidade de criar esta operação que trata diretamente do processamento do conjunto de pontos por faixa, permitindo a filtragem pelo identificador da faixa bem como por intervalos temporais.

- **GET /mobileapp/protocols** - *Endpoint* criado com o intuito de disponibilizar à aplicação móvel todos os recursos associados aos protocolos, nomeadamente, questões,

fotografias e opções de resposta associadas à questões, bem como o tipo de faixas associado, apenas na realização dum pedido. De notar que este pedido é utilizado pela aplicação acompanhado pelos parâmetro protocol_active de forma a retornar apenas os protocolos ativos.

- **GET /mobileapp/protocols/:id** - *Endpoint* que tal como o anterior devolve todos os recursos associados a um protocol em específico. É geralmente utilizado pela aplicação para retornar um protocolo antigo em específico quando um utilizador deseja consultar um registo realizado com este.

- **GET /mobileapp/currentuser/records** - *Endpoint* criado com o objetivo de devolver os registos antigos de um utilizador que eliminou/reinstalou a aplicação ou eliminou os seus dados desta do seu telemóvel. Devolve também, toda a informação associada aos registos, tal como explicado no *endpoint* da sua inserção, permitindo assim restaurar toda a informação sobre estes.

- **POST /mobileapp/media_urls** -*Endpoint* que disponibiliza a criação de *links* de *Resumable Uploads* para serem disponibilizados novamente acesso às fotografias presentes no *bucket*, caso ocorra algum problema de ligação na obtenção das fotografias de um utilizador aquando da recuperação de registos. De notar que o método é uma operação POST uma vez que é necessário enviar no *body* os identificadores da multimédia que a aplicação não conseguiu obter.

- **POST /mobileapp/app_configurations** -*Endpoint* utilizado para a criação de novos ficheiros de configuração dedicados à aplicação móvel. Este pedido diferencia-se do genérico, uma vez que realiza o envio de uma notificação de alerta para os utilizadores a indicar esta atualização.

- **POST /mobileapp/app_logs** -*Endpoint* que permite o envio e armazenamento dos *logs* criados pela aplicação móvel no Sistema de Informação.

- **PATCH /mobileapp/currentuser/location** -*Endpoint* utilizado para realizar atualizações à localização do utilizador corrente da aplicação móvel.

- **POST /mobileapp/currentuser/zones_of_interest** -*Endpoint* criado com o intuito de gerar objetos espaciais por parte de um utilizador que representa uma zona de interesse do mesmo, realizando por isso a inserção da associação entre o objeto espacial criado com a conta do utilizador atual, nomeadamente na tabela de zonas de interesse do utilizador.

- **PATCH /mobileapp/zones_of_interest** -*Endpoint* que pretende a remoção de zonas de interesse de utilizadores bem como do objeto espacial associado. Tal como no pedido relacionado com os *media_urls* este método é um POST uma vez que recebe no *body* uma lista de identificadores a remover.

- **POST /mobileapp/events** -*Endpoint* cujo motivo de criação é semelhante ao do *app_configurations*, uma vez que insere um evento no sistema e cria, também, um alerta que vai ser enviado para todos os utilizadores que podem estar interessados na zona onde este está a ocorrer, nomeadamente, caso se situe próxima da sua localização ou de uma das suas zonas de interesse.

- **POST /mobileapp/local_news** -*Endpoint* cujo objetivo é obter as notícias (dos últimos 7 dias) apenas relativas ao local onde o utilizador se encontra. De notar que este método é um POST uma vez que é requisitado pela aplicação sempre que um utilizador a inicia e dessa forma, vai enviar uma lista dos identificadores de notícias que já tem no sistema de forma a diminuir a carga do pedido.

- **POST /mobileapp/all_news** -*Endpoint* cujo objetivo é obter as notícias (dos últimos 7 dias) mas desta vez de todas as localizações, permitindo ao utilizador obter informação sobre as outras regiões.

- **POST /mobileapp/accounts_prohibitions** -*Endpoint* que pretende adicionar uma lista de proibições a um utilizador que excedeu os limites ou usou as ferramentas com má intenção. Este pedido envia uma notificação de alerta para o respetivo utilizador de forma a avisá-la da alteração de permissões na sua conta.

ANEXO I

TABELAS DE CÓDIGOS ASSOCIADAS AOS ATRIBUTOS DAS FGCI

Tabela I.1: Descrição das faixas[10]

Código	Designação
1	Edificações integradas em espaços rurais
2	Aglomerados populacionais
3	Parques de campismo, infraestruturas e equipamentos florestais de recreio, parques e polígonos industriais, plataformas de logística e aterros sanitários
4	Rede viária florestal
5	Rede ferroviária
6	Rede de transporte de gás
7	Linhas de transporte e distribuição de energia elétrica em muito alta tensão
8	Redes primárias de faixas de gestão de combustível
9	Rede terciária de faixas de gestão de combustível
10	Linhas de transporte e distribuição de energia elétrica em média tensão
11	Mosaico de parcelas de gestão de combustível
12	Pontos de água
13	Linhas de transporte e distribuição de energia elétrica em alta tensão
14	Silvicultura no âmbito da DFCI

Tabela I.2: Objetivos da Faixa [10]

Código	Designação
1	Diminuir a superfície percorrida por grandes incêndios. Facilitar combate/intervenção (in)direta na frente de fogo ou nos seus flancos.
2	Reduzir os efeitos da passagem de incêndios. Proteger de forma passiva, zonas edificadas, vias de comunicação, infraestruturas, povoamentos florestais.
3	Isolar focos potenciais de incêndios. Reduzir a probabilidade de propagação de incêndios a áreas adjacentes a linhas elétricas, à e ferroviária, a parques de recreio, entre outros.
4	Outro

Tabela I.3: Tipo de intervenção a realizar nas faixas [10]

Código	Designação
SSS	Sem intervenção
SAR	Sem intervenção. Área ardida no ano anterior ou no ano actual
DDD	Correcção de densidades excessivas
DRO	Correcção de densidades excessivas e desramações
AAA	Criação faixas ou manchas por alteração do coberto vegetal
RRR	Desramações
QQQ	Gestão com fogo controlado
GFI	Gestão de combustíveis com aplicação de fitocidas
GAG	Gestão de combustíveis com culturas agrícolas
GSI	Gestão de combustíveis com silvopastorícia
MAO	Gestão mecânica de combustível e alteração do coberto vegetal
MDO	Gestão mecânica de combustível e correcção de densidades excessivas
MDR	Gestão mecânica de combustível, correcção de densidades excessivas e desramação
MDE	Gestão mecânica de combustível e desramação
MDQ	Gestão mecânica de combustível e gestão com fogo controlado
CAO	Gestão moto-manual de combustível e alteração do coberto vegetal
CDO	Gestão moto-manual de combustível e correcção de densidades excessivas
CDR	Gestão moto-manual de combustível, correcção de densidades excessivas e desramação
CDE	Gestão moto-manual de combustível e desramação
CDQ	Gestão moto-manual de combustível e gestão com fogo controlado

Tabela I.4: Meios de execução das faixas [10]

Código	Designação
1	Equipa de Sapadores Florestais da Autarquia
2	Equipa de Sapadores Florestais da Organização de Produtores Florestais/Baldios
3	Equipa de Defesa da Floresta contra Incêndios
4	Empresa de Prestação de Serviços/Prestadores de Serviços
5	Meios próprios da Autarquia
6	Programas ocupacionais – Instituto de Emprego e Formação Profissional
7	Outro
8	Equipa do Corpo Nacional de Agentes Florestais - CNAF

Tabela I.5: Meios de financiamento para execução de faixas [10]

Código	Designação
1	AGRIS 3.4
2	Autarquia
3	Fundo Florestal Permanente
4	Outro
5	Serviço Público – Programa de Sapadores Florestais
6	ProDeR

Tabela I.6: Fase do projeto faixas [10]

Código	Designação
1	Marcação no gabinete
2	Validação no terreno
3	Elaboração do plano de intervenção
4	Execução no terreno
5	Conclusão