NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

BERNARDO FERNANDES OLIVEIRA

Licenciado em Ciências e Engenharia Informática

COMPARAÇÃO DE BASES DE DADOS RELACIONAIS COM BASES DE DADOS TEMPORAIS PARA ARMAZENAMENTO DE RASTERS

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Julho, 2023

DEPARTAMENTO
DE INFORMÁTICA

COMPARAÇÃO DE BASES DE DADOS RELACIONAIS COM BASES DE DADOS TEMPORAIS PARA ARMAZENAMENTO DE RASTERS

BERNARDO FERNANDES OLIVEIRA

Licenciado em Ciências e Engenharia Informática

Orientador: Carlos Viegas Damásio
Professor Associado, Universidade Nova de Lisboa

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Julho, 2023

Comparação de Bases de Dados Relacionais com Bases de Dados Temporais para armazenamento de rasters

Agradecimentos

Ao meu orientador Professor Carlos Damásio pelo acompanhamento que me deu ao longo de toda a elaboração da dissertação.

Aos meus pais, irmão e restante família por todos os ensinamentos e possibilidades que me deram desde sempre. Ao meu pai e mãe por suportarem todos estes anos de preparação para o mundo profissional.

À minha namorada Rita pela paciência que tem sempre para me aturar nos momentos menos positivos. E, por toda a motivação que me deu ao longo dos últimos tempos.

Aos meus colegas de curso mais próximos, que me ajudaram durante estes anos todos a terminar este curso, e que me ajudaram a pôr de lado a vida académica quando foi preciso.

A todos estes,
Muito Obrigado!

"No man ever threw away life while it was worth keeping."
(David Hume)

Resumo

Ao longos dos anos o setor da agricultura foi severamente afetado por diversas pragas, que arrasaram plantações inteiras e chegaram mesmo a provocar períodos de fome extrema. Desta forma, com o desenvolvimento desta dissertação vai ser possível auxiliar a formulação de modelos capazes de prever essas mesmas pragas, e assim impedir tanto prejuízos financeiros, como problemas sociais.

Para estes modelos serem eficientes, é necessário obter informação de elevada qualidade. Para tal, vão ser utilizados dados obtidos através de satélites. Por vezes, a informação obtida pelos satélites vem incompleta devido à existência de nuvens em certas regiões, logo é necessário recorrer-se também a modelos de previsão meteorológica para preencher a informação que falta e, desta forma, se obter os dados meteorológicos necessários.

Assim, o objetivo desta dissertação é criar um sistema de informação capaz de armazenar todos esses dados meteorológicos, de preferência, de elevada qualidade para posteriormente servirem de base para o cálculo de modelos de previsão de pragas.

De forma a desenvolver um sistema o mais eficiente possível, para o tipo de dados referido, nesta dissertação foram criadas duas bases de dados distintas, uma relacional e outra de séries temporais, com o intuito de se testar qual delas seria a mais útil ao sistema que se pretende criar. Para isso, foram realizadas inúmeras consultas, a ambas as bases de dados, tendo em conta diversos pormenores como, área da região analisada, intervalo temporal que se pretende analisar, agregação de dados diversos e ainda ordenação de resultados.

Palavras-chave: Sistema de informação, dados de satélite, modelos de previsão de pragas, modelos de previsão meteorológica, base de dados Relacional, base de dados de Séries Temporais, dados *raster*

Abstract

Over the years, the agriculture sector has been severely affected by pests, which have devastated entire plantations and even caused periods of extreme hunger. Thus, with the development of this dissertation, it will be possible to help formulating models capable of predicting these same pests, and thus prevent both financial losses and social problems.

For these models to be efficient, it is necessary to obtain high quality information. For this, data obtained from satellites will be used. Sometimes, the information obtained by the satellites is incomplete due to the existence of clouds in certain regions, therefore it is necessary to also use weather forecasting models to fill in the missing information and, in this way, to obtain the necessary meteorological data.

Thus, the objective of this dissertation is to create an information system capable of storing all these meteorological data, preferably of a high quality, to later serve as the basis for calculating pest prediction models.

In order to develop a system as efficient as possible, for the type of data mentioned, in this dissertation two distinct databases were created, one relational and the other a time series, to test which one would be the most useful for the system. For this, numerous queries were carried out to both databases, taking into account various details such as the area of the analyzed region, the time interval to be analyzed, aggregation of various data and even ordering of results.

Keywords: Information system, satellite data, pest prediction models, weather forecasting models, relational databases, timeseries databases, *raster* data

Índice

Índice de Figuras **xi**

Índice de Tabelas **xii**

1 Introdução **1**
1.1 Contexto . . . 1
1.2 Objetivos . . . 2
1.3 Aplicação . . . 2
1.4 Estrutura do documento . . . 3

2 Estado da arte **5**
2.1 Sistemas relacionados . . . 5
2.2 Bases de dados . . . 7
2.2.1 Tipos . . . 7
2.2.2 Base de dados de séries temporais . . . 8
2.2.3 Base de dados relacionais . . . 10
2.3 Dados meteorológicos . . . 11
2.3.1 Dados de estações meteorológicas . . . 11
2.3.2 Dados de satélite . . . 11
2.3.3 *Hierarchical Data Format 5* (HDF5) . . . 14
2.3.4 *Panoply* [22] . . . 15
2.3.5 Modelos de previsão meteorológica . . . 16
2.3.6 Interpolação espacial . . . 18
2.4 Representação de dados geográficos . . . 18
2.4.1 Modelos *raster* [20] . . . 18
2.4.2 Modelos vetoriais [20] . . . 19
2.4.3 QGIS . . . 20
2.4.4 GDAL . . . 21
2.5 Conclusão . . . 21

3 Modelação 22
3.1 Introdução . . . 22
3.2 Arquitetura . . . 23
3.3 Modelo de dados . . . 25
3.3.1 Características de Desenho . . . 28
3.3.2 Módulo de Fontes de Dados . . . 29
3.3.3 Módulo de Dados Genéricos . . . 30
3.3.4 Módulo de Dados Processados . . . 31
3.3.5 Módulo de Conteúdo de Dados . . . 31
3.3.6 Módulo de Séries Temporais . . . 32
3.4 Metodologia de desenvolvimento . . . 34
3.5 Conclusões . . . 34

4 Implementação **36**
4.1 Introdução . . . 36
4.2 Tecnologias escolhidas . . . 36
4.2.1 Bash . . . 36
4.2.2 Python . . . 37
4.2.3 QGIS . . . 37
4.2.4 PostgreSQL . . . 37
4.2.5 TimescaleDB . . . 37
4.2.6 pgAdmin . . . 38
4.3 Integração e obtenção de dados meteorológicos . . . 38
4.3.1 Descarregamento de dados . . . 38
4.3.2 Tratamento de dados . . . 40
4.4 Base de dados . . . 46
4.4.1 Povoamento das bases de dados . . . 46
4.4.2 Índices . . . 47
4.5 Conclusões . . . 48

5 Avaliação e testes **49**
5.1 Aplicabilidade . . . 49
5.2 Testes de desempenho . . . 50
5.2.1 Casos de uso . . . 50
5.2.2 Características do Sistema . . . 51
5.2.3 Integração dos Dados . . . 52
5.2.4 Consultas aos Dados . . . 52
5.2.5 Apresentação dos Resultados . . . 62
5.3 Conclusões . . . 67

6 Conclusões **69**
6.1 Conclusões . . . 69

6.2 Trabalho Futuro 70
6.2.1 Utilização de modelos de previsão meteorológica 70
6.2.2 Criação de uma aplicação web 70
6.2.3 Criação de Visualizações 70

Bibliografia **71**

Anexos

I Anexos **74**

Índice de Figuras

2.1 Lista de modelos meteorológicos calculados com dados em tempo real e lista de culturas com modelos de previsão, respetivamente, presentes no *CIPRA*. [5] 6
2.2 *Ranking* das bases de dados de séries temporais. [37] 9
2.3 Representação do produto LST, recolhido através de satélite. [17] 14
2.4 Representação gráfica da organização dos dados num ficheiro com formato HDF5. [13] 15
2.5 Representação de um *array* de um ficheiro HDF5 e representação gráfica de um ficheiro HDF5, respetivamente, com o programa *Panoply*. 16
2.6 Modelos raster vs Modelos vetoriais. [32] 19
2.7 Representação gráfica de um ficheiro *raster* com o programa QGIS. 20
2.8 Representação gráfica de um ficheiro *shapefile* com o programa QGIS. . . . 21

3.1 Arquitetura do sistema 24
3.2 Vista geral do modelo de dados relacional 26
3.3 Vista geral do modelo de dados *timeseries* 27
3.4 Módulo de Fontes de Dados 29
3.5 Módulo de Dados Genéricos 30
3.6 Módulo de Dados Processados 31
3.7 Módulo de Conteúdo de Dados 32
3.8 Módulo de Séries Temporais 33

Índice de Tabelas

5.1 Resultados obtidos - caso de uso 1 na base de dados de séries temporais . . 62
5.2 Resultados obtidos - caso de uso 1 na base de dados relacional 62
5.3 Resultados obtidos caso de uso 2 na base de dados de séries temporais . . . 63
5.4 Resultados obtidos caso de uso 2 na base de dados relacional 63
5.5 Resultados obtidos caso de uso 3 na base de dados de séries temporais . . . 63
5.6 Resultados obtidos caso de uso 3 na base de dados relacional 64
5.7 Resultados obtidos caso de uso 4 na base de dados de séries temporais . . . 64
5.8 Resultados obtidos caso de uso 4 na base de dados relacional 65
5.9 Resultados obtidos caso de uso 5 na base de dados de séries temporais . . . 65
5.10 Resultados obtidos caso de uso 5 na base de dados relacional 65
5.11 Resultados obtidos caso de uso 6 na base de dados de séries temporais . . . 66
5.12 Resultados obtidos caso de uso 6 na base de dados relacional 66
5.13 Resultados obtidos caso de uso 6 na base de dados relacional 66
5.14 Resultados obtidos caso de uso 6 na base de dados de séries temporais . . . 67

I.1 Resultados obtidos - caso de uso 1 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais 74
I.2 Resultados obtidos - caso de uso 1 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional 74
I.3 Resultados obtidos - caso de uso 4 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais 75
I.4 Resultados obtidos - caso de uso 4 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional 75
I.5 Resultados obtidos - caso de uso 1 utilizando a tabela com as coordenadas, na base de dados de séries temporais 75
I.6 Resultados obtidos - caso de uso 1 utilizando a tabela com as coordenadas, na base de dados relacional 76
I.7 Resultados obtidos - caso de uso 4 utilizando a tabela com as coordenadas, na base de dados de séries temporais 76

I.8 Resultados obtidos - caso de uso 4 utilizando a tabela com as coordenadas, na base de dados relacional 76
I.9 Resultados obtidos caso de uso 2 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais 76
I.10 Resultados obtidos caso de uso 2 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional 77
I.11 Resultados obtidos caso de uso 5 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais 77
I.12 Resultados obtidos caso de uso 5 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional 77
I.13 Resultados obtidos caso de uso 2 utilizando a tabela com as coordenadas, na base de dados de séries temporais 77
I.14 Resultados obtidos caso de uso 2 utilizando a tabela com as coordenadas, na base de dados relacional 77
I.15 Resultados obtidos caso de uso 5 utilizando a tabela com as coordenadas, na base de dados de séries temporais 77
I.16 Resultados obtidos caso de uso 5 utilizando a tabela com as coordenadas, na base de dados relacional 77
I.17 Resultados obtidos - caso de uso 3 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais 78
I.18 Resultados obtidos - caso de uso 3 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional 78
I.19 Resultados obtidos - caso de uso 6 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais 78
I.20 Resultados obtidos - caso de uso 6 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional 79
I.21 Resultados obtidos - caso de uso 3 utilizando a tabela com as coordenadas, na base de dados de séries temporais 79
I.22 Resultados obtidos - caso de uso 3 utilizando a tabela com as coordenadas, na base de dados relacional 79
I.23 Resultados obtidos - caso de uso 6 utilizando a tabela com as coordenadas, na base de dados de séries temporais 80
I.24 Resultados obtidos - caso de uso 6 utilizando a tabela com as coordenadas, na base de dados relacional 80

Índice de Listagens

4.1 Estruturação do nome do ficheiro a descarregar 38
4.2 Obtenção da data do ficheiro a descarregar 39
4.3 Como aceder ao dataset pretendido num ficheiro HDF5 42
4.4 Obtenção dos valores da Península Ibérica 42
4.5 Fórmulas para conversão de coordenadas geográficas 43
4.6 Criação dos layers 44
4.7 Criação do ficheiro .shp 45
4.8 Criação do raster 46
4.9 Inserção de um ficheiro .tiff na base de dados 47
4.10 Criação de indíces 47
4.11 Criação de indíces 48
5.1 Consultas relativas ao caso de uso 1 utilizando a função ST_*PixelAsCentroids* 53
5.2 Consultas relativas ao caso de uso 1 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela 53
5.3 Consultas relativas ao caso de uso 1 utilizando a tabela com as coordenadas 55
5.4 Consultas relativas ao caso de uso 2 utilizando a função ST_*PixelAsCentroids* 55
5.5 Consultas relativas ao caso de uso 2 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela 56
5.6 Consultas relativas ao caso de uso 2 utilizando a tabela com as coordenadas 56
5.7 Consultas relativas ao caso de uso 3 utilizando a função ST_*PixelAsCentroids* 57
5.8 Consultas relativas ao caso de uso 3 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela 57
5.9 Consultas relativas ao caso de uso 3 utilizando a tabela com as coordenadas 57
5.10 Consultas relativas ao caso de uso 4 utilizando a função ST_*PixelAsCentroids* 58
5.11 Consultas relativas ao caso de uso 4 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela 58
5.12 Consultas relativas ao caso de uso 4 utilizando a tabela com as coordenadas 58
5.13 Consultas relativas ao caso de uso 5 utilizando a função ST_*PixelAsCentroids* 59
5.14 Consultas relativas ao caso de uso 5 utilizando a tabela com as coordenadas 59

5.15 Consultas relativas ao caso de uso 5 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela 59
5.16 Consultas relativas ao caso de uso 6 utilizando a função ST_*PixelAsCentroids* 60
5.17 Consultas relativas ao caso de uso 6 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela 60
5.18 Consultas relativas ao caso de uso 6 utilizando a tabela com as coordenadas 61
5.19 Consultas relativas ao caso de uso 6 utilizando agregação por intervalo de tempo 61
5.20 Consultas relativas ao caso de uso 6 utilizando agregação por coordenadas x,y 61

1

Introdução

Ao longo deste capítulo vão ser descritos o contexto e o problema da dissertação realizada, bem como os objetivos que se pretendiam atingir. Vai ser ainda apresentada uma possível aplicação da mesma no nosso quotidiano. De seguida, teremos uma breve descrição dos objetivos relacionados com o trabalho a desenvolver nesta dissertação. Por fim, uma pequena apresentação dos conteúdos deste documento.

1.1 Contexto

Desde o ínicio da existência do ser humano, que a agricultura é uma atividade primária, e uma das mais importantes para o desenvolvimento e sobrevivência da nossa espécie. Porém, existem diversos fatores que ameaçam o bom funcionamento deste setor, entre eles o mais relevante ao longo deste documento, as pragas – surto de insetos nocivos ao desenvolvimento agrícola.

Estas pragas foram a causa de diversos períodos negros da história da humanidade. Capazes de contaminar ou mesmo destruir plantas e alimentos, são responsáveis por uma redução considerável do número de exemplares de algumas espécies e até por alguns períodos de fome, que posteriormente vieram a afetar o equilíbrio económico e/ou social de uma região. Atualmente, com o avanço da tecnologia e melhoramento das técnicas agrícolas já é possível prever e até mesmo exterminar algumas dessas pragas, mas se nada for feito nesse sentido, estas continuaram a ter um impacto económico negativo nas culturas dos agricultores.

Assim, de forma a reduzir ou mesmo evitar perdas monetárias, os produtores agrícolas tentam utilizar técnicas de prevenção de pragas, caso estas não surtam efeito têm de recorrer a pesticidas ou a um controlo biológico, ou seja, introdução de predadores de insetos de forma a impedir o crescimento da sua população. No entanto, o uso desmedido de pesticidas é bastante prejudicial para o meio ambiente e até para o ser humano, enquanto que um controlo biológico tem um preço elevado. Logo pretende-se que estas técnicas de prevenção tenham uma taxa de sucesso bastante elevada de forma a se evitar a utilização de soluções de recurso, como as referidas anteriormente.

Para aumentar a eficiência destas técnicas tem se recorrido cada vez mais a sistemas informáticos, quer para armazenamento e recolha de dados, quer para cálculos de probabilidades e/ou riscos. Desta forma, esta dissertação irá contribuir com um sistema capaz de, através de dados meteorológicos recolhidos por satélite, calcular os grau dias acumulados, isto é, a soma das temperaturas acima da condição mínima e abaixo da condição máxima para uma planta finalizar os diferentes subperíodos de desenvolvimento [2], ou outros modelos matemáticos geralmente usados para prever a evolução ou incidência das pragas, tanto de um ponto em Portugal como de uma qualquer região, contudo o foco da mesma será, numa fase inicial, apenas em pontos da Península Ibérica.

1.2 Objetivos

O objetivo desta dissertação foi o desenvolvimento de um sistema de informação capaz de armazenar a informação necessária à prevenção e identificação de pragas na agricultura de um dado ponto ou região da Península Ibérica, como referido anteriormente. Este vai fornecer informação relevante para os produtores agrícolas saberem como gerir as suas plantações da melhor forma possível, de modo a evitar a ocorrência de pragas.

Esta informação poderá ser depois utilizada por alguns modelos matemáticos, como o Graus Dia Acumulados. Porém, este tipo de modelos necessita de variada informação acerca das condições climatéricas, como a temperatura do ar, temperatura do solo, níveis de precipitação, entre outros. Desta forma, o sistema recorreu a satélites para ter acesso a todos estes dados e armazená-los. Uma vez que existiram condições que impediram a medição correta destas variáveis, como por exemplo a existência de nuvens numa dada região, existiram períodos em que não foi possível ter acesso a totalidade dos dados.

Tais informações serão muito importantes na gestão e proteção de uma plantação contra qualquer tipo de pragas.

Por fim, o objetivo principal foi a comparação do desempenho do sistema numa base de dados de séries temporais comparativamente a uma base de dados relacional, quando armazenando séries temporais de dados raster. Visto que, para o tipo de dados armazenados é aconselhável a utilização de uma base de dados de séries temporais, foi possível verificar que isto é de facto verdade.

1.3 Aplicação

A grande aplicação desta dissertação é a sua integração no sistema que foi desenvolvido para o Projeto FitoAgro.

O Projeto FitoAgro [20] é um projeto desenvolvido na região do Oeste de Portugal, que pretende monitorizar e estudar o ciclo de vida de novas pragas na agricultura de forma a encontrar formas alternativas para o combate das mesmas sem recurso a produtos químicos. Assim, este projeto irá tentar definir uma estimativa do risco e dos respetivos

níveis económicos de ataque (NEA) para uma melhor compreensão de como prevenir e/ou controlar um ataque.

Para cumprir os objetivos deste projeto é bastante importante que a informação obtida seja de elevada qualidade. De forma a garantir essa qualidade, foram instalados diversos tipos de dispositivos de monitorização em pontos de observação biológica (POB) de modo a estudar diversos aspetos da vida das espécies que são identificadas como pragas. Foram ainda instaladas diversas estações meteorológicas perto dos POB de modo a monitorizar os dados meteorológicos desses locais. E, para auxiliar o controlo desses mesmos dados, o projeto FitoAgro recorre ainda a satélites, previsões meteorológicas e fornecedores online. Sendo que estes são mesmo a base de grande parte dos modelos de previsão de pragas.

Assim, o sistema desenvolvido nesta dissertação, no projeto FitoAgro, vai ser inserido como um dos fornecedores de dados essenciais na criação dos modelos de previsão das pragas, visto que utiliza informação obtida através de satélites o que vai de encontro à necessidade de respeitar os padrões de qualidade necessários ao projeto FitoAgro. Desta forma, irá ser possível ter dados que possibilitam a estimativa de diversas características relativas às espécies que mostram ser uma ameaça às plantações dos produtores agrícolas. O sistema desenvolvido vai ainda possibilitar uma melhoria na eficiência do armazenamento/consulta dos dados, uma vez que através da comparação realizada às bases de dados permitiu identificar o melhor tipo para ir de encontro às necessidades do projeto.

1.4 Estrutura do documento

Este relatório encontra-se dividido em seis grandes capítulos e a estrutura de cada um deles será explicada de seguida:

- **Introdução**

  Ao longo deste capítulo é apresentado o tema que foi desenvolvido nesta dissertação. Dentro do qual são apresentados os objetivos, o contexto e possível aplicação da mesma.

- **Estado da arte**

  No segundo capítulo são apresentadas algumas tecnologias, técnicas e ferramentas que já existem e que foram úteis ao longo da elaboração desta dissertação.

- **Modelação**

  O capítulo 3 é responsável pela descrição de como foi idealizado o esqueleto do sistema. Começa por enunciar os requisitos para o bom funcionamento do mesmo, aborda toda a arquitetura e termina com a pormenorização de cada uma das suas componentes, como o modelo de dados utilizado.

- **Implementação**

Neste capítulo temos alguns exemplos que mostram como foram implementados alguns elementos do sistema.

- **Avaliação e testes**

  No capítulo 5, já com o sistema implementado, serão apresentados os testes realizados à solução desenvolvida, bem como os respetivos resultados.

- **Conclusões**

  Por fim, temos o sexto e último capítulo, dedicado às conclusões desta dissertação e a possíveis melhorias futuras.

2

Estado da arte

Neste capítulo são apresentadas as tecnologias e ferramentas que foram relevantes para o desenvolvimento desta dissertação, como referido anteriormente.

Primeiramente, são abordados conteúdos relacionados com sistemas já existentes que também têm como objetivo prever desenvolvimento de pragas na agricultura. De seguida, é apresentado um pequeno estudo acerca de dois tipos de base de dados existentes, que irão ser comparados posteriormente nesta dissertação, bem como algumas opções disponíveis para a implementação das mesmas, que irão ser o suporte desta dissertação.

Por fim, efetuámos um breve estudo acerca das diferentes formas de como irão ser obtidos alguns dados meteorológicos e também, de como representar dados geográficos e as aplicações utilizadas para tal.

2.1 Sistemas relacionados

Após uma breve pesquisa sobre modelos de previsão de pragas na agricultura, foram descobertos diversos sistemas capazes efetuar previsões acerca do desenvolvimento de pragas, entre insetos e doenças, nas culturas. Chegando à conclusão que na maioria dos casos, os modelos de previsão para os insetos são baseados em modelos graus-dia acumulados [26]. É nesta secção que esses sistemas vão ser apresentados e os seus objetivos descritos.

Computer Centre for Agricultural Pest Forecasting [5]

O CIPRA é uma ferramenta que é capaz de prever o desenvolvimentos de pragas, colheitas e alguns problemas pós-colheita baseados em dados meteorológicos. Estes dados são obtidos de duas formas diferentes, através de estações meteorológicas automáticas no Quebec, Canadá, e de modelos de previsão meteorológica. Após a recolha de toda a informação necessária, o sistema recorre a modelos bioclimáticos científicos para calcular a probabilidade do crescimento das pragas.

Esta ferramenta, conta ainda com cerca de 130 modelos de previsão de pragas para cerca de 25 diferentes tipos de culturas. Mesmo assim, todos os anos são adicionados

novos dados ao sistema, o que faz com que este esteja em constante evolução.

O CIPRA é, atualmente, o maior repositório de modelos de previsão bioclimáticos existente no Canadá.

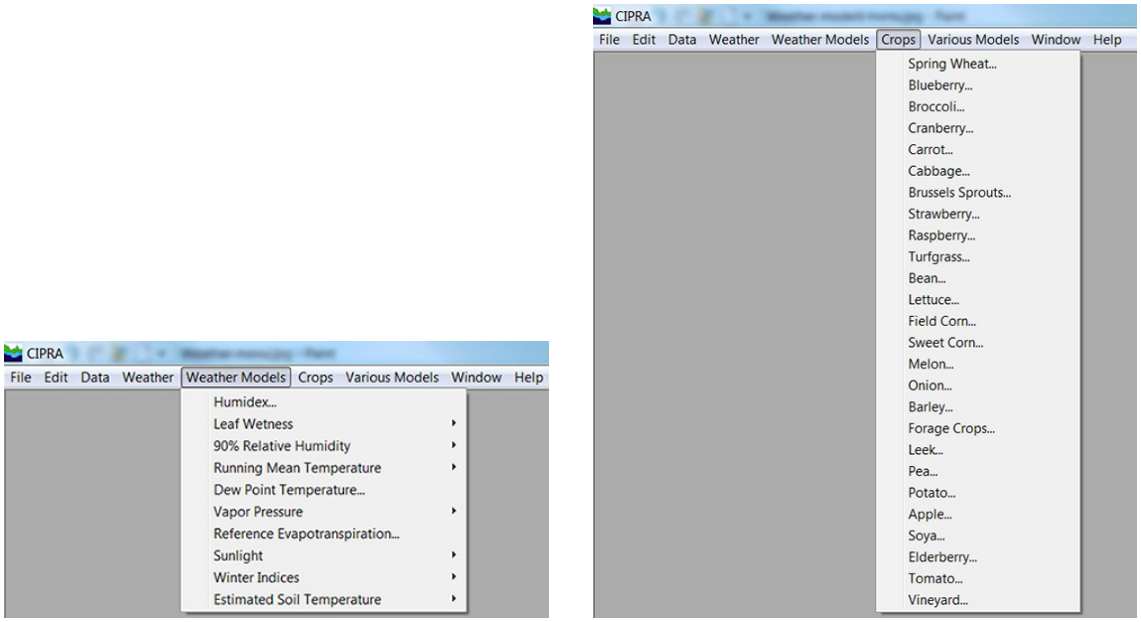

Figura 2.1: Lista de modelos meteorológicos calculados com dados em tempo real e lista de culturas com modelos de previsão, respetivamente, presentes no *CIPRA*. [5]

Previsão de pragas de insetos no arroz [34]

Num estudo sobre as pragas nas plantações de arroz na Tailândia, foram comparados três diferentes formas de prever a população de uma espécie de inseto, muito comum neste tipo de plantações. As formas utilizadas para a comparação foram redes neurais artificiais - modelos em que se tem uma determinada quantidade de entradas, e através de uma função de ativação serão transformadas numa saída [33] - e árvores de decisão, ambos modelos de aprendizagem automática, e regressão linear múltipla. Para o cálculo destes modelos utilizaram como input a fenomenologia do clima e da planta hospedeira, durante a estação seca da colheita, tendo estes dados sido obtidos através de satélites e obervações no solo.

No final de todos os testes efetuados, as redes neurais artificiais foram o modelo que obteve a melhor precisão numa previsão feita a curto prazo.

Mapas de risco de pragas considerando temperatura à superfície do solo [21]

Neste estudo, os autores utilizaram a temperatura à superfície do solo, obtida através de satélites, para gerar um dos parâmetros mais utilizados na previsão de pragas, e aquele que esta dissertação também vai usar maioritariamente, graus dia acumulados. Os resultados do estudo foram bastante positivos e permitiram a criação de mapas de risco para

pragas na agricultura com maior detalhe que os realizados através de dados de estações meteorológicas.

Trap System [39]

Com o aumento da rede de monitorização de insetos em diversos estados do Brasil, foi necessário criar um sistema capaz de armazenar toda a informação obtida através dessa mesma rede. Assim, foi criada uma aplicação, disponível não só para computadores como para telemóveis, capaz de armazenar toda a informação e que permite aos seus utilizadores a partilha de informação entre eles.

Essa aplicação é a *Trap System* e através das suas funcionalidades e informação armazenada, vai permitir aos utilizadores monitorizar os padrões de miragão das populações de insetos e estimar o impacto de diversas espécies dos mesmos no nosso quotidiano.

Sistema de informação para previsão de doenças nas plantas, baseado em dados meteorológicos [1]

De forma a auxiliar os produtores agrícolas da região de Gyeonggi-do, na Coreia do Sul, foi desenvolvido um sistema de informação capaz de prever e depois avisar, de hora a hora ou diariamente baseado em dados meteorológicos, os produtores de quando devem começar a pulverizar os seus fungicidas, de modo a controlar as doenças nas suas plantações.

Este sistema é composto por 4 componentes, um sistema capaz de adquirir a informação meteorológica da região, outro para gerir os processos que depois lançam os avisos aos utilizadores, um sistema para armazenar os dados recolhidos e, por fim, o sistema de *web services*, que é a componente que disponibiliza a informação aos utilizadores através de navegadores *web*.

2.2 Bases de dados

A base de dados é uma componente essencial no desenvolvimento de um sistema de informação, por isso foi necessária uma pesquisa detalhada sobre o seu atual estado da arte, de forma a escolher algo apropriado para o tema e, que não esteja desatualizado.

2.2.1 Tipos

Como já foi referido anteriormente, nesta dissertação são abordados dois tipos de base de dados diferentes, as base de dados relacionais e as base de dados de séries temporais, uma vez que se pretende verificar se a utilização das bases de dados temporais irá beneficiar o projeto FitoAgro - atualmente utiliza base de dados relacional - no armazenamento dos seus tipos de dados. As bases de dados de séries temporais começaram por ser usadas no processamentos de dados financeiros muito voláteis e também em aplicações

industriais que guardavam os valores medidos por sensores. Porém, desde 2015 que a sua utilização tem crescido a um ritmo elevadíssimo.

Estas base de dados, tal como o nome indica, destinam-se a guardar séries temporais como pares de tempo e valor, tornando assim mais fácil a análise das alterações de um conjunto de dados ao longo do tempo. Esta forma de armazenar os dados, permite também que um utilizador adicione, processe e analise elevadas quantidades de dados com elevada precisão e num curto espaço de tempo [38].

Por outro lado, as base de dados relacionais têm como objetivo guardar e fornecer acesso a dados que estão relacionados entre si. Estas baseiam-se no modelo relacional, em que os dados são representados por tuplos e agrupam-se em relações. Desta forma, numa base de dados deste tipo, cada linha de uma tabela é um registo com um identificador único chamado chave primária (podendo ser constituída por um ou mais atributos) e cada coluna representa um atributo [20].

Com isto, para esta dissertação o principal foco foi nas base de dados de séries temporais que, para além de serem a base do tema, também são as mais indicadas, teoricamente, para o tipo de dados utilizado ao longo da dissertação – dados meteorológicos recolhidos através de satélites. Porém, também foram utilizadas base de dados relacionais espaciais de modo a tornar possível a comparação entre ambas.

2.2.2 Base de dados de séries temporais

Atualmente, nas bases de dados de séries temporais temos quatro que se destacam em termos de popularidade aquando da pesquisa sobre as mesmas:

- **InfluxDB** [15];
- **Prometheus** [30];
- **TimescaleDB** [38];
- **M3** [18].

Tendo em conta que o objetivo da dissertação incide sobre dados geográficos, mais concretamente na região da Península Ibérica, a base de dados escolhida teve de ter uma componente geográfica forte, que permitisse um armazenamento capaz de a manipular de forma fácil e eficiente.

InfluxDB [15]

Desenvolvida pela *InfluxData*, esta base de dados é *open-source* e está escrita em *Go*. Apresenta uma grande disponibilidade de armazenamento e recuperação de dados de séries temporais em áreas como a monitorização de operações, métricas de aplicações e dados sensoriais. InfluxDB não tem depêndencias externas e fornece uma linguagem *SQL*.

Prometheus [30]

Um *software* gratuito que regista métricas em tempo real numa base de dados de séries temporais. Tal como o anterior, é escrito em *Go* e apresenta uma licensa *Apache 2 License*. Licensa esta que garante aos utilizadores a receção de uma licença para qualquer patente relacionada com o *software* em questão. Prometheus fornece a sua própria linguagem, *PromQL*, que permite aos seus utilizadores selecionar e gerir os dados das séries temporais em tempo real.

TimescaleDB [38]

É um *software* de base de dados de séries temporais *open-source*, optimizado para lidar com esses mesmos tipos de dados. Tem suporte *SQL* e é considerado uma extensão do PostgreSQL, o que torna possível utilizá-lo como se fosse o próprio PostgreSQL.

Entre as diversas funcionalidades oferecidas pelo PostgreSQL, estão as relacionadas com a consulta de séries temporais num ponto geográfico específico. Neste sistema de base de dados é possível obter os valores de um dado conjunto de dados durante um tempo escolhido pelo utilizador, num ponto pretendido ou até numa região. Para tal, existe a função *ST Distance*, que permite definir uma região à volta de um certo ponto para o qual pretendemos obter os valores das séries temporais e a função *time bucket()*, que permite definir os intervalos de tempo referidos anteriormente.

M3 [18]

É uma base de dados open-source, e foi inicialmente desenvolvida para localizar todos os condutores da Uber, em tempo real. Inicialmente, era uma base de dados de séries temporais para a Prometheus, já falada anteriormente, mas agora é o agregado de três componentes, o que a torna mais uma plataforma que uma base de dados. Esses três componentes são a base de dados em si, um serviço de agregação e um motor de busca, todos juntos numa só plataforma que é a M3.

Rank			DBMS	Database Model	Score		
Sep 2022	Aug 2022	Sep 2021			Sep 2022	Aug 2022	Sep 2021
1.	1.	1.	InfluxDB	Time Series, Multi-model	29.15	-0.63	-0.35
2.	2.	2.	Kdb+	Time Series, Multi-model	8.14	-1.20	+0.01
3.	↑ 4.	↑ 4.	Graphite	Time Series	6.56	+0.52	+1.46
4.	↓ 3.	↓ 3.	Prometheus	Time Series	6.53	-0.09	+0.09
5.	5.	5.	TimescaleDB	Time Series, Multi-model	4.87	+0.08	+1.15
6.	6.	6.	Apache Druid	Multi-model	2.63	-0.10	-0.64
7.	7.	7.	RRDtool	Time Series	2.51	-0.01	+0.07
8.	8.	8.	OpenTSDB	Time Series	2.23	+0.19	+0.36
9.	9.	↑ 11.	DolphinDB	Time Series, Multi-model	1.62	+0.03	+0.54
10.	10.	↓ 9.	Fauna	Multi-model	1.56	+0.18	-0.17

Figura 2.2: *Ranking* das bases de dados de séries temporais. [37]

2.2.3 Base de dados relacionais

Após uma pesquisa acerca das base de dados relacionais, foram identificadas algumas com capacidade para manipular dados geográficos com alguma eficiência, entre elas temos:

- **PostgreSQL** com a extensão **PostGIS**;
- **SQLite** com a extensão **SpatiaLite**;
- **Oracle** com a componente **Oracle Spatial and Graph**.

PostgreSQL [28]

Também conhecida por Postgres, é uma base de dados relacional gratuita. Esta usa e estende a linguagem *SQL* em conjunto com diversas funcionalidades que guardam e escalam os mais diferentes conjuntos de dados. PostgreSQL é compatível com qualquer um dos sistemas operativos mais conhecidos e tem diversas extensões, como a PostGIS, que irá ser abordada abaixo.

PostGIS [27]

É uma base de dados espacial que extende o sistema PostgreSQL. Esta adiciona diversas funcionalidades relativas a dados geográficos, como queries de localização em *SQL* e indexação espacial.

SQLite [36]

O SQLite é uma biblioteca em C que implementa um mecanismo de base de dados *SQL*. Esta é a mais usada em todo o mundo e é conhecida por ser utilizada em grande parte dos telemóveis e computadores. É também de utilização gratuita.

SpatiaLite [35]

É uma biblioteca gratuita com o objetivo de estender o sistema SQLite de forma a suportar funcionalidades espaciais. Esta fornece diversas funcionalidades relacionadas com base de dados geográficas de dados vetoriais.

Oracle [24]

É uma base de dados multi-modal desenvolvida e comercializada pela *Oracle Corporation*. Esta é regularmente utilizada para suportar aplicações relacionadas com transações, como por exemplo aplicações de bancos, data warehousing ou então uma mistura de ambas.

Oracle Spatial and graph [25]

É uma opção gratuita disponibilizada pela base de dados da Oracle, que ajuda os utilizadores a gerir os dados geográficos em tipos nativos dentro de uma base de dados Oracle. Assim, a base de dados passa a suportar uma grande variedade de aplicações que sem esta componente não seria possível. Entre elas estão as aplicações relacionadas com sistemas de informação geográficos.

2.3 Dados meteorológicos

Esta terceira secção, tem como objetivo indicar quais e como são obtidos os dados meteorológicos que, como já foi referido anteriormente, são essenciais nos cálculos dos modelos matemáticos para aplicações agronómicas. Para tal, este tipo de dados tem de apresentar uma elevada qualidade e baixa taxa de erros, ou seja, os dados têm de ser o mais próximos da realidade possível e pormenorizados, não apenas disponibilizar valores para grandes regiões, mas sim para pontos geográficos mais específicos. Modelos estes que facilitam a gestão das culturas aos produtores agrícolas, oferecendo diversas previsões relacionadas com a mesma.

2.3.1 Dados de estações meteorológicas

Os dados das estações meteorológicas automáticas são cada vez mais usados em situações relacionadas com a agricultura, porém sem qualquer tipo de controlo de qualidade estes possuem diversos tipos de erros [3]. Assim, este tipo de dados irá ser evitado ao máximo nesta dissertação, como já foi referido anteriormente.

2.3.2 Dados de satélite

Os dados obtidos através de satélite são a prioridade e um dos grandes objetivos desta dissertação, apenas as observações que o satélite não conseguir oferecer poderão ser recolhidas através de outro método.

Os satélites oferecem diversos tipos de produtos, que irão ser explicados posteriormente, mas o que interessa para esta dissertação é a temperatura da superfície da terra. Todos os produtos, abordados ao longo deste documento, são disponibilizados por satélites da *European Organization for Exploitation of Meteorological Satellites* (EUMETSAT).

2.3.2.1 EUMETSAT

A EUMETSAT é uma organização intergovernamental situada em Darmstadt, Alemanha, que opera com diversos satélites de forma a monitorizar o clima, as condições meteorológicas e o ambiente, a partir do espaço. Esta apresenta oito *Satellite Application Facilities* (SAFs), cada uma destinada ao processamento de dados de uma variável meteorológica diferente:

- ***Atmospheric Composition Monitoring*** **(AC SAF)** **[9]**

  Área que analisa dados de satélites acerca da camada de ozono, aerossóis, ultra-violeta e outro gases.

- ***Climate Monitoring*** **(CM SAF)** **[9]**

  Departamento que produz *datasets* climáticos de elevada qualidade.

- ***Ocean and Sea Ice*** **(OSI SAF)** **[9]**

  Fornece informação acerca da interface oceano-atmosfera.

- ***Numerical Weather Prediction*** **(NWP SAF)** **[9]**

  Nesta área é suportada a interface entre os dados de satélite e as atividades europeias de previsões meteorológicas.

- ***Radio Occultation Meteorology*** **(ROM SAF)** **[9]**

  De seguida, o departamento que gera dados *GNSS Radio Occultation (RO)*, uma técnica sensorial remota de satélite que usa medidas GNSS (Global Navigation Satellite System) [11] para representar a atmosfera e ionosfera do planeta Terra, de elevada qualidade, para o NWP .

- ***Nowcasting and Very Short Range Forecasting*** **(NWC SAF)** **[9]**

  Uma previsão meteorológica para as próximas horas, baseada em informação atual.

- ***Operational Hydrology and Water Management*** **(H SAF)** **[9]**

  É nesta área que são produzidos os *datasets* e produtos para aplicações hidrológicas operacionais.

- ***Land Surface Analysis*** **(LSA SAF)** **[9]**

  Por último, o departamento que a área de investigação inside sobre dados sensoriais remotos em terra, interações terra-atmosfera e aplicações da biosfera. Os produtos analisados nesta dissertação, serão na sua maioria, provenientes deste departamento, da responsabilidade do IPMA em Portugal.

É a partir deste último que são obtidos todos os dados satélites utilizados na realização desta dissertação. Dados estes que estão divididos entre cinco produtos diferentes:

- ***Daily Evapotranspiration*** **(DET)** **[17]**

  Este produto fornece informação sobre a evapotranspiração - fluxo de água evaporado na relação terra-atmosfera e transpirado pela vegetação nas suas folhas como consequência de processos fotossintéticos - do dia, em milímetros por dia (mm/dia).

- ***Evapotranspiration* (ET) [17]**

  Tal como o acima referido, dá os valores de evapotranspiração, porém ao invés de ser diário, as medições são feitas de 30 em 30 minutos em milímetros por hora (mm/hora).

- ***Reference Evapotranspiration* (ETREF) [17]**

  Parecido ao primeiro produto enunciado, DET, este também está relacionado com a evapotranspiração diária, contudo não fornece os seus valores ao longo do dia mas sim uma estimativa da radiação global diária derivada a partir do *Spinning Enhanced Visible and InfraRed Imager* (SEVIRI/MSG) - instrumento pertencente à EUMETSAT capaz de observar o planeta Terra em doze canais/bandas espectrais diferentes - em milímetros por dia (mm/dia).

- ***Land Surface Temperature* (LST) [17]**

  Produto que mede a temperatura do solo ao longo do dia em graus celsius (ºC). Este fornece as medições de 15 em 15 minutos, um total de 96 vezes por dia. Contudo, as 96 medições diárias são apenas um número teórico, visto que caso existam nuvens, o satélite pode não conseguir efetuar todas essas medições.

- ***Land Surface Temperature - All Sky* (LST-AS) [17]**

  Igualmente aos primeiros três produtos descritos, os últimos dois também estão relacionados, ambos medem a temperatura do solo ao longo do dia em ºC. Porém, este produto em específico, LST-AS, fornece a temperatura de todos os pontos da superfície terrestre, de 30 em 30 minutos, mesmo quando existem nuvens.

Uma vez que todos os produtos listados anteriormente provêm da mesma fonte de dados, era expectável que a sua resolução, área analisada e formato dos produtos gerados sejam igualmente semelhantes. Assim, podemos concluir que todos eles analisam a área referente à Europa, África e maioria da América do Sul com uma resolução de cerca de 3 quilómetros (km) *at sub satellite point* - ponto à superfície terrestre diretamente por baixo do satélite em linha reta - num formato *Hierarchical Data Format 5* (HDF5).

Na figura 2.2 podemos observar um exemplo de representação do produto LST, recolhido através dos satélites, neste caso no dia 1 de Fevereiro de 2021 pelas 8 horas da tarde (UTC). A escala das cores representa da temperatura mínima, abaixo dos -45ºC a cinzento, até à temperatura máxima, cerca de 75ºC a vermelho escuro. Como referido anteriormente, existem algumas informações que não são possíveis obter devido à existência de nuvens na região, essa é explicação para a existência da cor branca na imagem e não na escala.

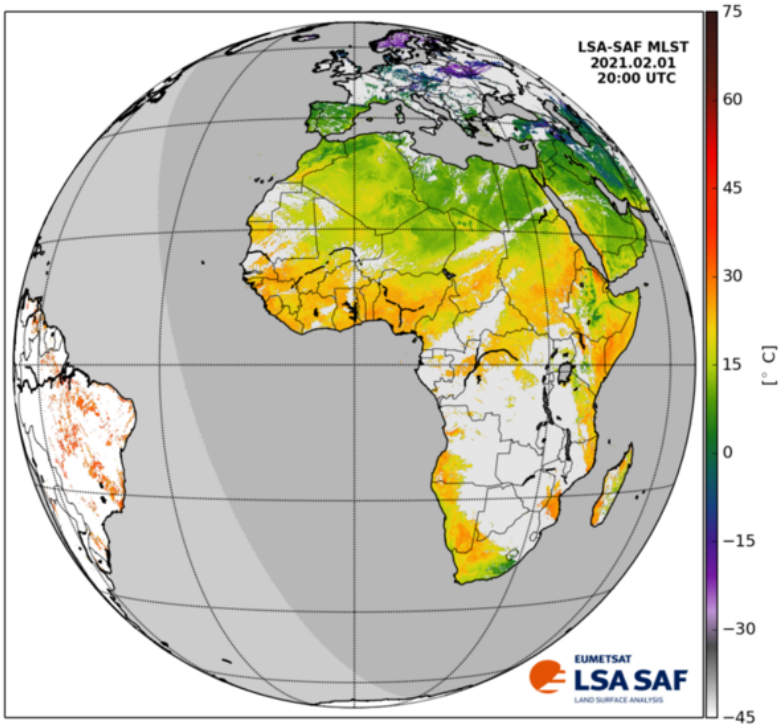

Figura 2.3: Representação do produto LST, recolhido através de satélite. [17]

2.3.3 *Hierarchical Data Format 5* (HDF5)

O formato HDF foi desenhado pela *National Center for Supercomputing Applications* (NCSA), dos Estados Unidos da América (EUA), de forma a manipular conjuntos de dados científicos muito grandes e complexos e já é utilizado por diversas fontes de dados sensoriais remotas, como a referida acima neste capítulo. [19]

Este formato de dados é conhecido pela sua forma característica de organizar os dados de um ficheiro, que tal como o nome indica, segue uma hierarquia parecida às diretorias de ficheiros. Assim, cada ficheiro armazenado com este formato é composto por:

- **Grupos:** estrutura composta por instâncias de zero ou mais grupos ou *datasets*, juntos com os seus metadados. Estes grupos são compostos por duas partes, um *header* com o nome do grupo e os seus atributos, e, ainda uma tabela de símbolos que contém uma lista de todos os seus objetos HDF5.
- ***Dataset*:** um *array* multidimensional de dados junto com os seus metadados. Cada *dataset* é guardado num ficheiro com duas partes, um *header* com a informação necessária para interpretar o *array* do *dataset* e os seus metadados e um *array* com os dados. Cada *header* apresenta quatro blocos de informação relevantes:
  - ***name*:** sequência de carateres *ASCII alfanuméricos*.

- ***datatype*:** existem dois tipos de *datatypes*, os *atomic datatypes* que representam os que não se conseguem decompor e os *compound datatypes* que são constituídos por vários *atomic datatypes*.
- ***dataspace*:** indica a dimensionalidade do *dataset*, esta pode ser fixa ou ilimitada. O *dataspace* pode ainda indicar partes do *dataset* em vez dele todo, o que torna possível fazer operações apenas com a porção selecionada.
- ***storage layout*:** é possível guardar os dados de diversas formas diferentes, de uma forma linear semelhante à armazenada em memória, *contiguous storage* que é a forma *default*, caso os dados sejam de pequenas dimensões é possível guardar tudo apenas no *header*, *compact storage*, e, por fim, dividindo o *dataset* em *chunks* ou porções todas do mesmo tamanho, guardadas de forma separada, *chunked storage*.

Para além dos grupos e *datasets* podem ainda existir atributos. Estes são contituídos por um nome e o seu respetivo valor, podendo ser utilizados para representar o uso de um determinado grupo ou *dataset* [12].

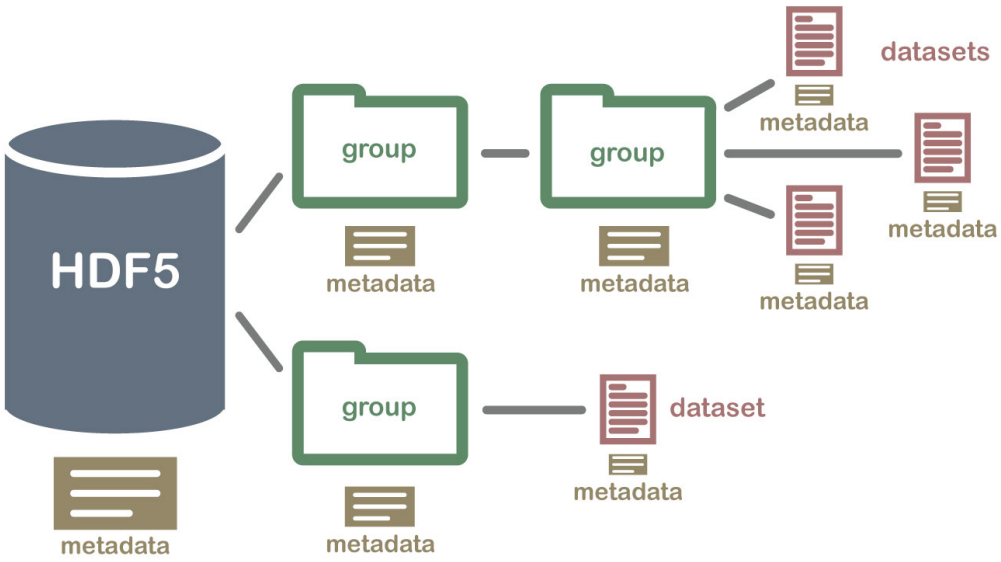

Figura 2.4: Representação gráfica da organização dos dados num ficheiro com formato HDF5. [13]

2.3.4 *Panoply* [22]

A aplicação *Panoply* representa *arrays* georreferenciados e outros a partir de dados em ficheiros *Network Common Data Form* (NetCDF), HDF, ou *GRIdded Binary* (GRIB). Esta aplicação fornece ainda algumas funcionalidades como truncar ou representar *arrays*, combinar *arrays* e exportar essas mesmas representações. Panoply está disponível em

diversos sistemas operativos, e foi desenvolvido pelo *Goddard Institute for Space Studies* (GISS) da *National Aeronautics and Space Administration* (NASA).

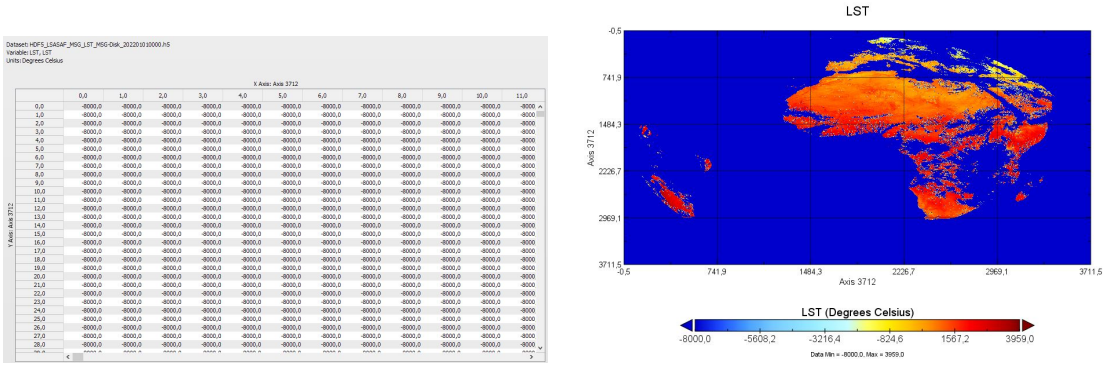

Figura 2.5: Representação de um *array* de um ficheiro HDF5 e representação gráfica de um ficheiro HDF5, respetivamente, com o programa *Panoply*.

2.3.5 Modelos de previsão meteorológica

Como referido anteriormente, nem sempre é possível obter toda a informação necessária através dos satélites. Por isso, essa informação poderá ser complementada com o cálculo da temperatura através de modelos de previsão meteorológica. Estes modelos são capazes de produzir uma aproximação do estado futuro da atmosfera e são calculados através de equações que utilizam as leis da física para prever o comportamento da mesma. Assim, é possível obter previsões de diferentes varáveis atmosféricas, como é o caso da temperatura [16].

Após uma breve pesquisa, estes foram os seis modelos encontrados:

- Modelo do ECMWF;
- AROME;
- ALADIN;
- ERA5;
- *Global Forecast System* (GFS).
- ICON

Modelo do ECMWF [16]

O modelo desenvolvido pelo ECMWF é produzido através de um sistema chamado *Integrated Forecasting System* (IFS), e é um modelo global, o que significa que efetua previsões para todo o planeta. Este modelo é também conhecido como o "Modelo Europeu" pelos Norte Americanos. Este modelo é executado com um alcance máximo de 10 dias,

uma resolução vertical de 137 níveis e uma resolução horizontal de 16 km. Porém, tal como nos dados de satélite, a existência de nuvens faz com que exista uma margem de erro nas previsões.

ALADIN [16]

É um modelo de previsão de área limitada com 46 níveis verticais e uma resolução horizontal de 9 km. Este foi desenvolvido por diversos grupos de previsão europeus e norte africanos, sobre a coordenação da *Météo-France*. Este modelo é capaz de prever diversos parâmetros como por exemplo, a temperatura e humidade relativa a 2 metros e a precipitação de neve, entre outros.

AROME [16]

É também um modelo de previsão de área limitada mas de alta resolução, desenvolvido pela *Météo-France*. Este modelo utiliza as previsões do modelo ALADIN como condições iniciais, e é executado com uma resolução vertical de 46 níveis e resolução horizontal de 2.5 km.

ERA5 [7]

O ERA5 é a última reanálise climática produzida pelo ECMWF que fornece informação durante todas as horas, sobre diversos parâmetros atmosféricos.

Este conjunto de dados está disponível com uma resolução de cerca de 28 km, e vai cobrir toda a informação desde 1950 até aos dias de hoje. Porém, atualmente ainda só está disponível informação a partir do ano de 1979. É esperado que as previsões de parâmetros atmosféricos futuros seja disponibilizada cinco dias antes da data.

GFS [23]

O GFS é um modelo de previsão meteorológica produzido pelo *National Centers for Environmental Prediction* (NCEP), que disponibiliza dados sobre dezenas de parâmetros atmosféricos. Este modelo pode ter 28 km de resolução base horizontal ou descer para os 44 km, dependendo de quantos dias no futuro se pretende prever o tempo.

ICON [14]

O modelo global de previsões meteorológicas ICON, é utilizado pelo Deutscher Wetterdienst (DWD), sistema meteorológico alemão, desde o início de 2015.

A estrutura deste modelo é baseada num icosaedro, representando a forma do planeta Terra, tal como os dados que este modelo disponibiliza. Posteriormente, a estrutura dos dados pode ser interpolada numa estrutura mais comum, latitude/longitude, utilizando *Climate Data Operators* [4], uma ferramenta com uma interface de linha de comandos, utilizada para analisar e manipular dados de modelos de previsão climática e numérica.

2.3.6 Interpolação espacial

Para além dos modelos de previsão, outra opção para solucionar o problema dos satélites quando não é possível obter os valores de uma região devido à existência de nuvens é a interpolação espacial. Esta consiste em utilizar pontos com valores já conhecidos para obter valores aproximados em pontos desconhecidos. Porém, tal como os dados de estações meteorológicas, estas técnicas são pouco fiáveis.

As duas técnicas encontradas após uma breve pesquisa foram IDW (*Inverse Distance Weighting*) e *Kriging*, a primeira é uma técnica determinista enquanto que a segunda é geo-estatística. As técnicas deterministas criam valores através de pontos já conhecidos, baseando-se na semelhança entre os pontos ou no grau de variação. Enquanto que as técnicas geo-estatísticas utilizam as propriedades estatísticas dos dados já conhecidos.

IDW (*Inverse Distance Wighting*) [40]

A técnica IDW consiste na atribuição de um certo nível de influência a cada ponto de valor conhecido, baseado na sua proximidade ao ponto de valor desconhecido. Ou seja, quanto mais próximo um ponto está daquele que pretendemos calcular, mais impacto vai ter o seu valor. Este grau de influência é representado por um parâmetro chamado potência. Se essa potência for zero, o valor vai ser a média de todos os valores próximos. Caso tenha um valor elevado, apenas os pontos mais próximos irão influenciar o valor final.

Kriging [40]

Esta técnica é semelhante à anterior, na medida em que utiliza uma combinação linear dos pesos de pontos com valores conhecidos para estimar pontos de valores desconhecidos. Sendo que estes pesos variam consoante a posição espacial dos pontos.

2.4 Representação de dados geográficos

Sendo o sistema a desenvolver relacionado com dados geográficos, existiu a necessidade de realizar uma pequena pesquisa sobre os mesmos.

Assim, esta secção é uma introdução às duas principais formas de representação destes tipo de dados – modelo *raster* e modelo vetorial.

2.4.1 Modelos *raster* [20]

Nos modelos raster, a informação é representada através de matrizes de células, também chamadas de pixeis, sendo estas a únidade mais pequena de representação. Cada um destes pixeis tem um valor atribuído, normalmente relacionado com uma cor, quando juntos formam uma imagem.

A qualidade da imagem está diretamente relacionada com o número de pixeis que a formam, ou seja, quanto maior o número de pixeis, maior qualidade terá a imagem. Porém, uma das desvantagens deste modelo é quando se pretende ampliar uma imagem, quanto maior a ampliação mais se irão notar os pixeis, tornando assim a imagem noutra cada vez mais desfocada.

Os modelos *raster* são úteis para guardar informação que está sempre a ser alterada, uma vez que ao apresentarem estruturas relativamente simples conseguem processar os dados de uma forma rápida. Com o modelo *raster*, os dados espaciais não são contínuos mas sim divididos em unidades discretas, isto torna este tipo de dados bastante útil para alguns tipos de operações espaciais, como sobreposições e cálculos de áreas.

2.4.2 Modelos vetoriais [20]

Nos modelos vetoriais a informação é representada através de formas geométricas. Cada uma dessas formas é constituída por um ou mais vértices, ou pontos, e cada um deles descreve uma posição no espaço através de coordenadas x e y, e algumas vezes com um eixo z.

Quando a forma geométrica tem apenas dois pontos, sendo que o inicial é diferente do final, temos uma linha, enquanto que, quando temos três ou mais pontos e o último ponto coincide com o primeiro, estamos perante um polígono. Desta forma, a qualidade de uma imagem, quando representada por modelos vetoriais, não se altera consoante a ampliação da mesma. Tornando assim estes modelos bastante úteis para fazer mapas e imagens de grandes resoluções.

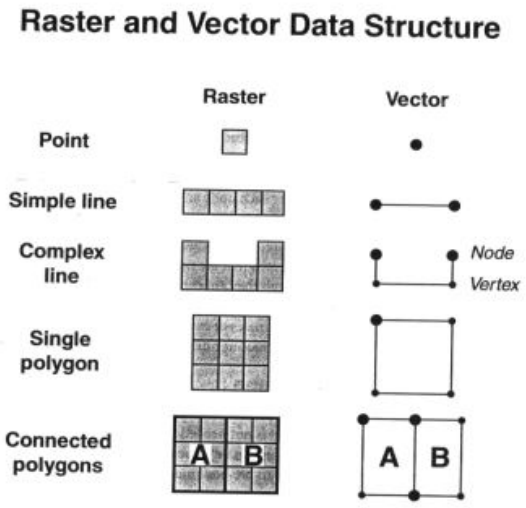

Figura 2.6: Modelos raster vs Modelos vetoriais. [32]

Um exemplo da aplicação destes modelos é o formato de ficheiros *Shapefile* (.shp). Este é um formato de armazenamento de dados vetoriais que guarda a localização, a forma e alguns atributos desses mesmos dados. Foi desenvolvido pelo *Environmental Systems*

Research Institute (Esri) e apesar de ser chamado *Shapefile*, este formato é uma coleção de ficheiros guardados na mesma diretoria, só o .shp não é suficiente para obter toda a informação, este apenas contém os dados geométricos. Desta forma, é necessário ter também os ficheiros:

- ***Shapefile shape index format*(.shx)**: contém o índice da funcionalidade geométrica, o que torna possível navegar pelos *records* do ficheiro de maneira a encontrar a posição pretendida do ficheiro .shp.

- ***Shapefile attribute format* (.dbf)**: armazena os atributos de cada forma presente no *shapefile* utilizando uma base de dados.

Existem ainda outros ficheiros que se podem encontrar numa diretoria de um *Shapefile*, porém não são essenciais [8].

2.4.3 QGIS

QGIS é uma aplicação *open-source cross-platform* gratuita. Esta permite visualizar, editar e analisar dados geoespaciais, o que a torna um *geographic information system* (GIS), tal como o nome indica.

Esta aplicação suporta diversos formatos de vetores, raster e base de dados, o que a torna ideal para representar os modelos de dados referidos anteriormente neste capítulo. Esta permite ainda a integração com outras aplicações GIS *open-source*, como *PostGIS*, *GRASS GIS* e *MapServer* [31].

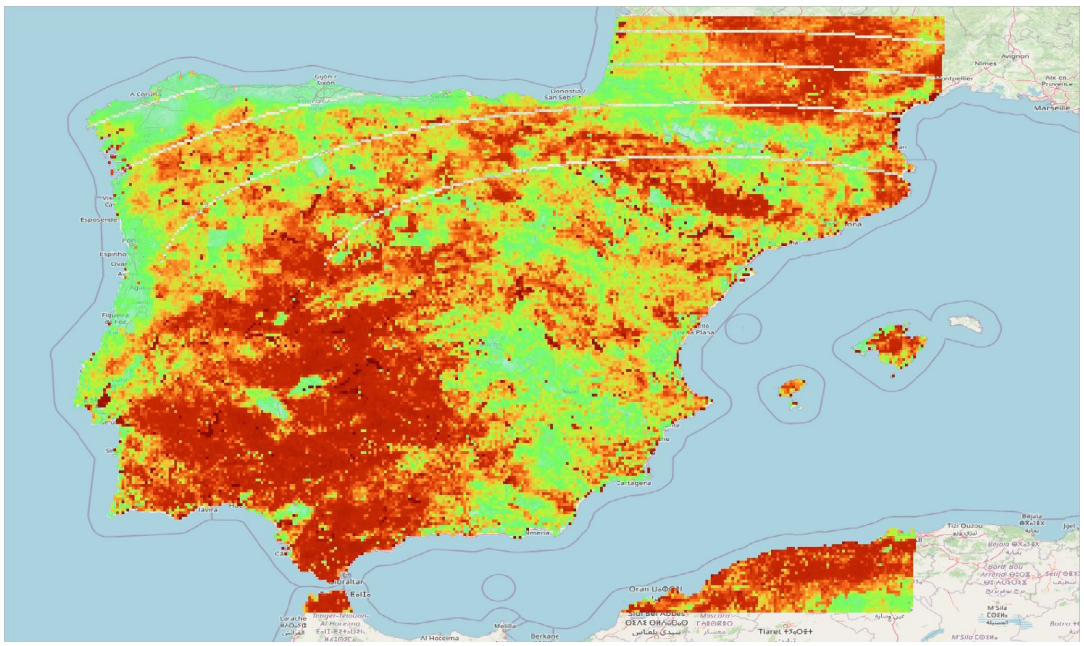

Figura 2.7: Representação gráfica de um ficheiro *raster* com o programa QGIS.

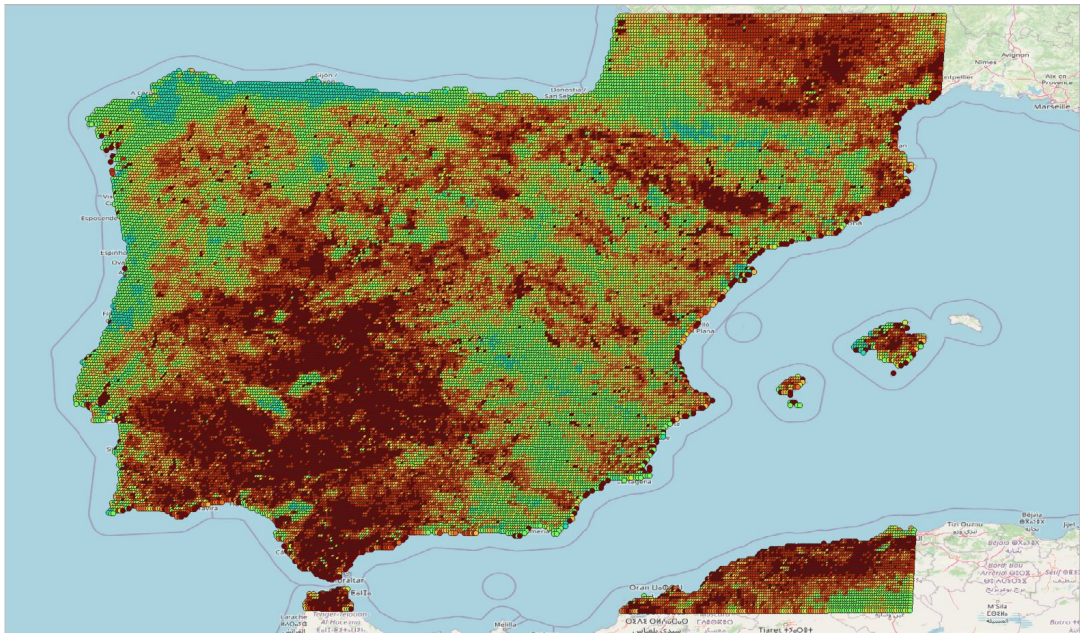

Figura 2.8: Representação gráfica de um ficheiro *shapefile* com o programa QGIS.

2.4.4 GDAL

GDAL é uma biblioteca *open-source* para dados do tipo *raster* e vetores geoespaciais, licensiada pela *The open source Geospatial Foundation*, OSGeo. Esta biblioteca oferece uma linha de comandos diversificada para tradução e processamento de dados. A GDAL está também integrada na aplicação referida anteriormente, QGIS [10].

2.5 Conclusão

Ao longo deste capítulo foram apresentados os estudos e pesquisas acerca das temáticas e tecnologias essenciais para a realização desta dissertação.

A pesquisa de sistemas relacionados com pragas na agricultura ajudou a compreender que dados são relevantes para o desenvolvimento de eficientes modelos de previsões. As base de dados são essenciais na construção de um sistema de informação, pelo que o seu estudo é imperativo para se obter o melhor desempenho possível, neste caso será recorrendo à utilização do *Postgresql + PostGIS* e do *TimescaleDB* para séries temporais. Já sabendo como guardar a informação, era necessário perceber como obter a mesma, então efetuou-se uma pesquisa relacionada com os dados meteorológicos e onde os obter. Desta forma decidiu-se solicitar a ajuda do IPMA, uma vez que eles detêm um servidor com toda a informação que será necessária, e descarregar diversos produtos disponibilizados por eles diariamente.

Por fim, existindo a necessidade de os manipular de forma a apenas contemplar as regiões pretendidas, realizou-se uma pesquisa mais pormenorizada sobre os formatos em que os mesmos eram descarregados e como os alterar para a forma pretendida.

3

Modelação

Ao longo deste capítulo irá ser apresentada e explicada a modelação da solução que foi implementada, tanto de uma perspectiva geral, como cada componente ao pormenor.

Inicialmente é apresentada uma breve descrição do sistema de informação, enunciando os requisitos prioritários funcionais. Em seguida, a arquitetura e seus componentes, e ainda, o modelo de dados.

Por fim, é discutida a metodologia de desenvolvimento e uma breve conclusão.

3.1 Introdução

A solução desenvolvida, tal como referido em capítulos anteriores (1.2), tem como objetivo integrar dados meteorológicos recolhidos através de satélites num sistema de informação, capaz de os analisar de forma a poderem ser utilizados na elaboração de modelos de previsão e controlo de pragas de forma eficiente. Assim, o sistema será capaz de cumprir o seu principal caso de uso, obter a informação de uma dada região, num determinado instante.

De forma a cumprir todos os referidos objetivos, o sistema de informação necessita de cumprir os seguintes requisitos:

- **Automatização da integração dos dados**: Ser capaz de, através da informação fornecida pelos satélites, gerar e integrar os ficheiros *raster* e/ou *shapefile* da região pretendida, diariamente.

- **Capacidade de armazenamento**: Ser capaz de armazenar elevadíssimas quantidades de dados e diversos tipos de ficheiros diferentes, como *raster*, *shapefile*, etc.

- **Componente espácio-temporal**: Ser capaz de armazenar dados geográficos e temporais, e relacioná-los.

- **Eficiência nas procuras**: Ser capaz de fornecer a informação que o utilizador pretende de forma eficiente e o mais rápido possível.

- **Extensível**: Ter a capacidade de ser estendido aquando da integração de novas fontes de dados ou requisitos.

3.2 Arquitetura

Ao longo desta secção, é explicada toda a arquitetura do sistema de informação que recolhe, trata, armazena e analisa os dados meteorológicos recolhidos na região da Península Ibérica. Esta divide-se em quatro grandes componentes a fonte de dados - Satélites, o integrador de dados - componente que consiste num conjunto de *scripts* com a função de descarregar, tratar e inserir os dados na componente seguinte, a base de dados - componente que armazena toda a informação relevante para o objetivo final do sistema, e, por fim, a camada de visualização - onde se encontram as visualizações obtidas através da análise dos dados armazenados na componente referida anteriormente.

- **Fonte de dados:** É nesta componente que são disponibilizados, pelo IPMA, num servidor próprio para o efeito, todos os produtos descritos na dissertação, na subsecção 2.3.2.1, entre outros.

- **Integrador de dados:** Este integrador vai comunicar com a fonte de dados de forma a descarregar todos os produtos necessários ao sistema. No final de obter toda a informação necessária, vai, através de diversos *scripts*, formatar a informação. Primeiramente, vai obter a informação de cada produto apenas para a região pretendida, a Península Ibérica, em seguida converte a informação para coordenadas geográficas, e, finalmente, através das mesmas gera o ficheiro *raster* que vai ser armazenado e, posteriormente analisado.

- **Camada de Armazenamento:** É nesta camada que é armazenada toda a informação relevante ao sistema. Como esta dissertação tem por objetivo comparar a eficiência de diferentes tipos de base de dados, nesta componente iremos ter duas a guardar exatamente a mesma informação, uma relacional e outra *timeseries* que irão ser detalhadas já na secção seguinte.

- **Camada de visualização:** E, por fim, a camada onde são representadas graficamente as análises feitas à informação presente no sistema e relevante para a criação de modelos agronómicos.

Na figura 3.1 pode-se ter uma ideia da representação gráfica final da arquitetura geral do sistema descrita no parágrafo anterior. Analisando a imagem, e tendo em conta que o fluxo de informação se desloca na direção das setas, podemos concluir que a informação é descarregada da fonte de dados através do *curl* [6] - um *software open source* e gratuito utilizado na linha de comandos ou em *scripts* para tranferir dados - pelo integrador, que, de seguida, a trabalha com recurso a diversos *scripts* e a coloca na base de dados. Por fim,

através da realização de *queries* para a análise da informação armazenada, são criadas as visualizações que compõem a última camada da arquitetura, camada de visualização.

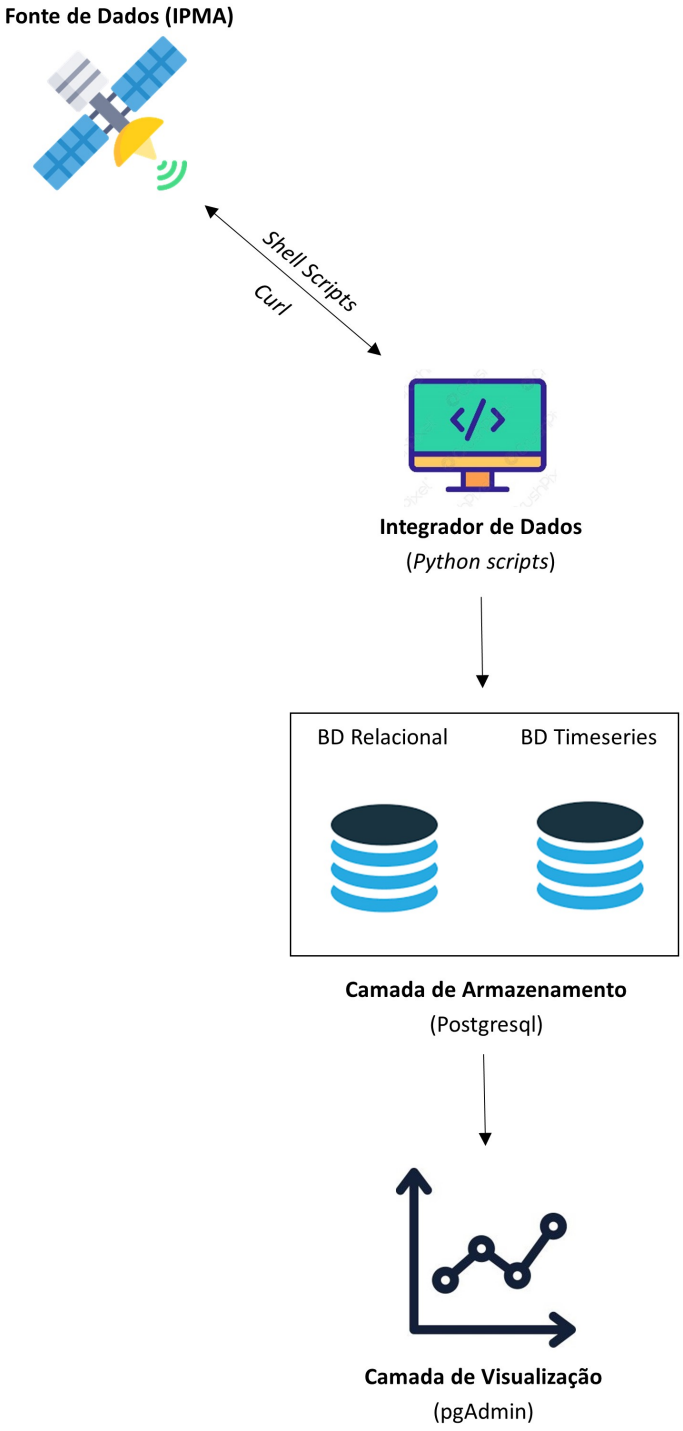

Figura 3.1: Arquitetura do sistema

3.3 Modelo de dados

O desenvolvimento do modelo de dados foi algo demorado e com diversas fases até obter o representado na figura 3.2. Uma das razões para o tempo dispendido, foi o facto de se pretender que o mesmo possa ser utilizado no futuro e não só ser adequado para os objetivos desta dissertação. A outra razão foi o facto de se utilizar uma base de dados *timeseries*, o que levou a pequenas alterações de um modelo para o outro.

Antes de começar a explicação ao pormenor, será apresentada uma pequena justificação sobre a nomenculatura atribuída a cada entidade/atributo, que se manteve ao longo de ambos os modelos de Dados. De forma a facilitar a explicação dos mesmos, estes serão divididos por módulos, cada um com uma cor, e apenas serão discutidas as entidades mais relevantes.

Com a cor azul temos o módulo de Fontes de Dados (3.3.2), a vermelho o módulo de Dados genéricos (3.3.3), a verde o módulo de Dados processados (3.3.4) e a roxo o módulo de Conteúdo de dados (3.3.5).

No caso do modelo para a base de dados *timeseries* (figura 3.3), teremos ainda, a amarelo, o módulo de Séries temporais (3.3.6).

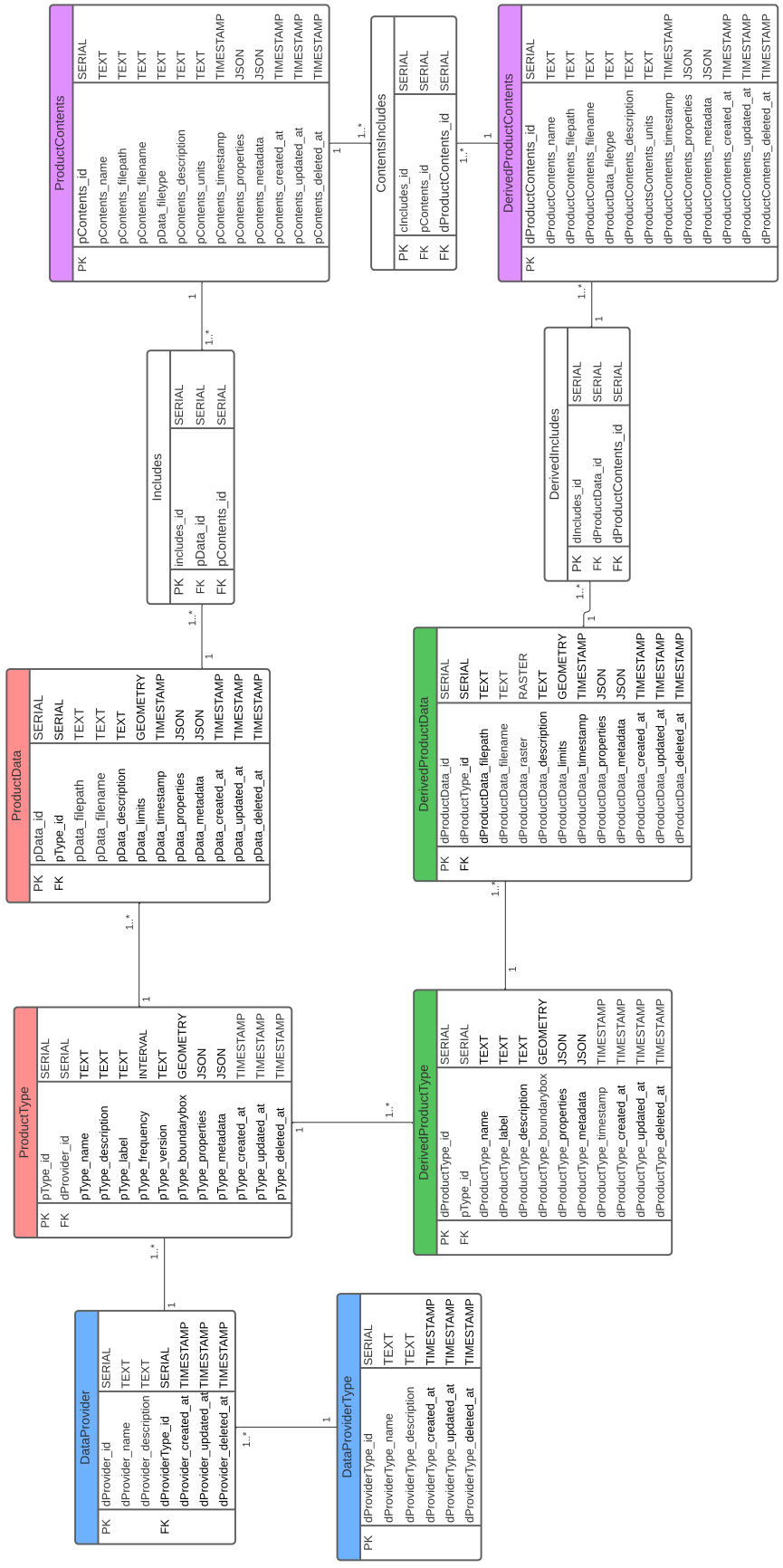

Figura 3.2: Vista geral do modelo de dados relacional

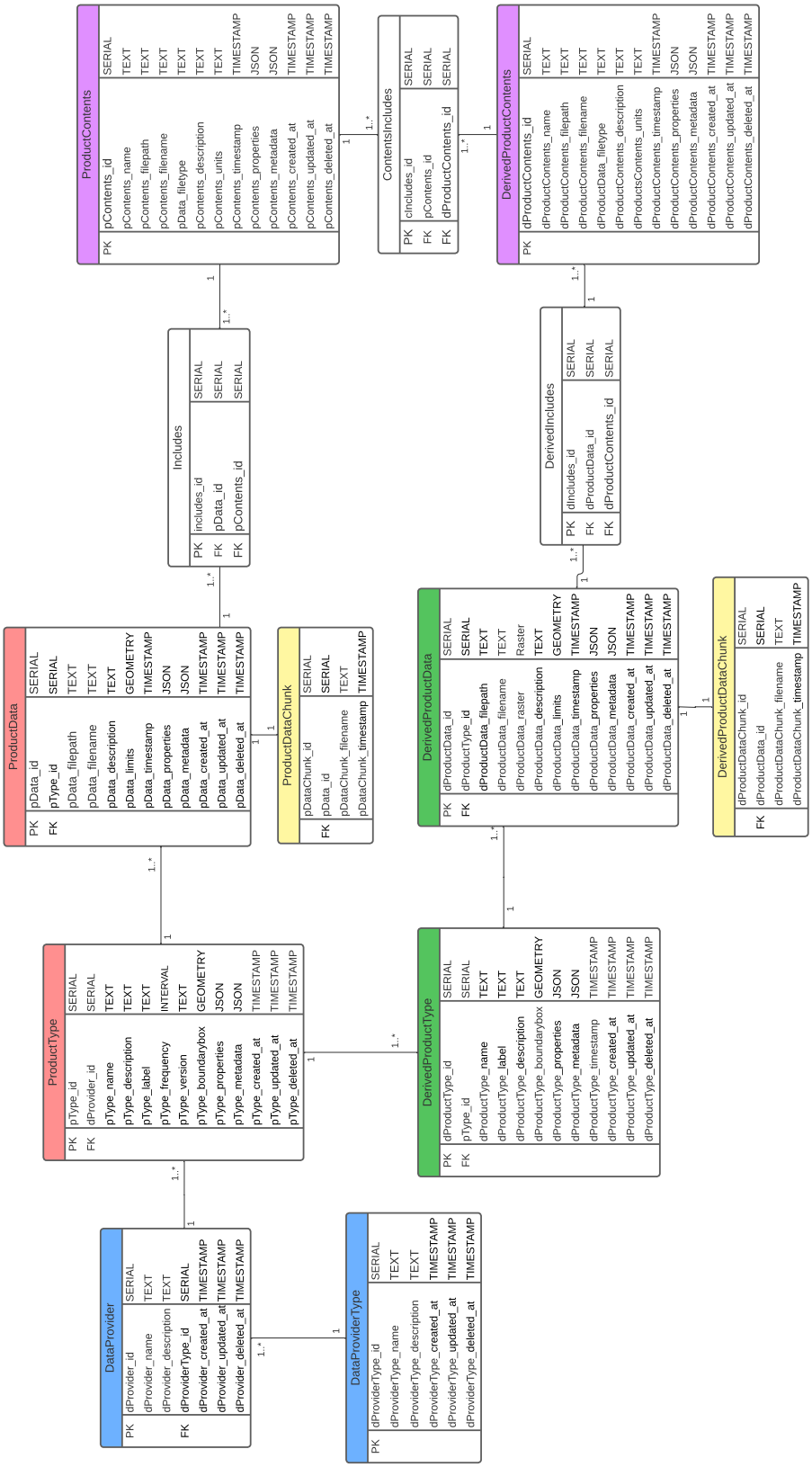

Figura 3.3: Vista geral do modelo de dados *timeseries*

3.3.1 Características de Desenho

Ao longo do processo de desenvolvimento dos modelos de dados, seguiram-se diversas características que se acredita que facilitem, não só toda a criação, como depois a leitura e compreensão dos mesmos. Assim, todas essas características serão enunciadas e explicadas de seguida.

Nomenculatura de Tabelas/Atributos

Os nomes escolhidos para as tabelas estão em inglês, uma vez que se pretende que o modelo seja acessível a qualquer um. E foram ainda escolhidos de forma a que mesmo sem qualquer explicação se tenha ideia de que dados estão guardados em cada tabela.

Já os atributos, seguem a característica <nome_tabela_abreviado>_<nome_atributo>, onde a palavra inicial do nome da tabela correspondente é abreviada apenas para a sua inicial. Desta forma, caso existam dois atributos com o mesmo nome mas de tabelas diferentes, são facilmente distinguíveis através do nome.

Atributos *properties* e *metadata*

Ao longo de ambos os modelos de dados, é de notar a presença dos atributos *properties* e *metadata*. Estes são ambos em formato JSON, de forma a facilitar a sua integração nas tabelas, no caso do atributo *properties*, que pode acrescentar novos atributos à tabela sem ter de a atualizar.

Enquanto que, o atributo *metadata* tem como objetivo, tal como o nome indica, armazenar os metadados dos dados armazenados na tabela correspondente.

Estampilhas temporais

Também presentes em todas as tabelas de ambos os modelos de dados, existem três estampilhas temporais: ***created_at***, ***updated_at***, ***deleted_at***.

Tal como referido no início deste capítulo, todos os nomes de atributos pretendem ser intuitivos, assim, facilmente se entende que uma estampilha é para guardar o momento da inserção de um tuplo na tabela, outra para quando se atualiza e outra quando é "eliminado"um tuplo, respetivamente.

Mas, ao contrário dos atributos ***created_at*** e ***updated_at***, o ***deleted_at*** não tem só como objetivo guardar a data de uma operação na base de dados, neste caso a de remoção. Este serve também para evitar que se perca informação relevante devido a algum engano ou ignorância do utilizador, desta forma quando se pretende eliminar algum registo altera-se o atributo mas nunca se elimina o mesmo.

Tabelas de Tipos

Sempre que possível, e faça sentido, as tabelas dos modelos têm associadas uma outra

tabela mais genérica, tabela de tipo. Assim, é possível ter uma classificação sobre cada tuplo guardado.

3.3.2 Módulo de Fontes de Dados

O primeiro módulo é o módulo que estrutura os fornecedores dos dados armazenado. É nele que são armazenados os tipos existentes e também especificado cada fornecedor. Este módulo é representado pela cor azul, tal como representado na figura 3.4.

Quanto aos atributos para além dos referentes às estampilhas temporais, já anteriormente explicados 3.3.1, existem ainda outros três atributos que, tal como o nome indica, representam o identificador único da tabela (id), nome e descrição.

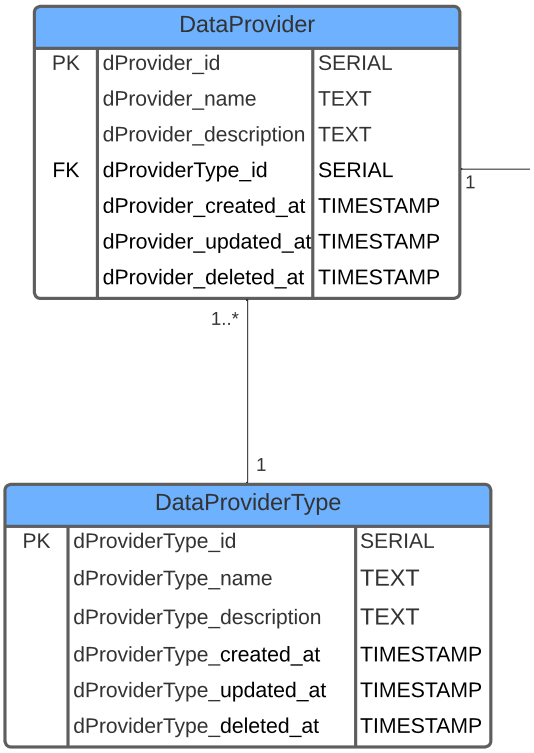

Figura 3.4: Módulo de Fontes de Dados

Fornecedor de Dados

O fornecedor de dados representa a entidade que fornece acesso aos dados que são guardados no sistema. Neste caso, apenas o IPMA se encaixa nesta tabela, pois será o único fornecedor a utilizar.

Tipo de Fornecedor de Dados

Enquanto que o tipo de fornecedor de dados é relativo à forma como são recolhidos os dados. Ao longo da dissertação, todos os dados são provenientes de satélites, porém existem outros tipos, como as estações meteorológicas.

3.3.3 Módulo de Dados Genéricos

Representado pela cor vermelha, o módulo de Dados Genéricos é o responsável por descrever os tipos de produto obtidos através do módulo referido anteriormente, e por armazenar todos os produtos de cada tipo que foram fornecidos. Na figura 3.4, podem-se observar as tabelas que formam este módulo.

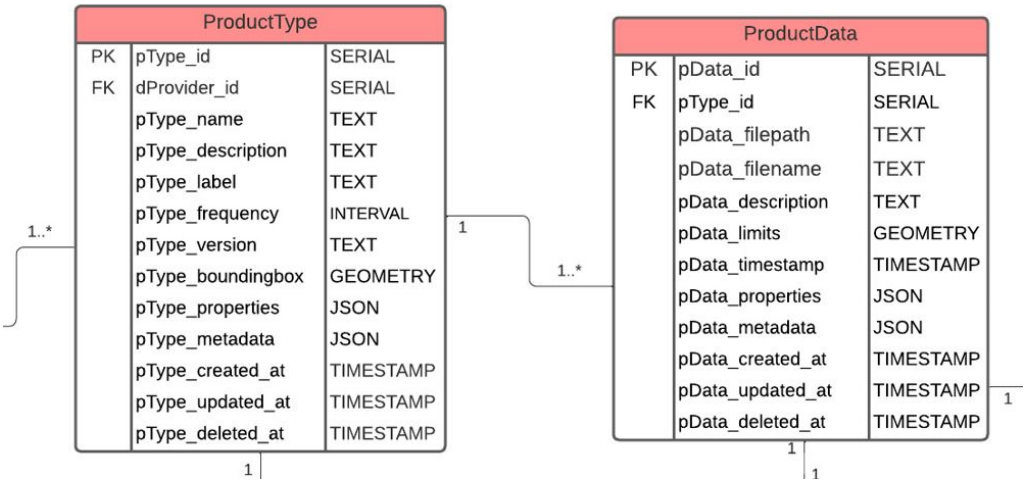

Figura 3.5: Módulo de Dados Genéricos

Tipo de Produto

Nesta tabela são armazenados os tipos de produto, estes têm associados a si um fornecedor, um nome (sempre completo, nunca abreviado), uma breve descrição, uma sigla - uma vez que o nome completo é extenso -, a frequência com que são produzidos (desde diários a de 15 em 15 minutos) e os seus limites geográficos. Desta forma, o utilizador pode ter uma ideia geral do que é que está armazenado no sistema.

Dados de Produto

Os dados de produto são todos os ficheiros armazenados no sistema de informação que não sofreram qualquer alteração desde que foram descarregados do fornecedor. As únicas informações extra que não vêm associadas ao seu tipo é o seu *timestamp* e os limites, ou seja, a data do exato momento em que os satélites obtiveram a informação que está contida no ficheiro e os seus limites geográficos.

3.3.4 Módulo de Dados Processados

O módulo acima referido (3.3.3) é a origem deste módulo. É aqui onde são descritos os tipos de dados criados a partir do módulo referido, e onde se armazenam todos esses ficheiros. Apesar de os dados não sofrerem qualquer alteração, a maneira como são representados e a área que abrangem vão facilitar a análise dos mesmos. É através desta análise dos dados aqui armazenados que vão ser obtidos os resultados desta dissertação.

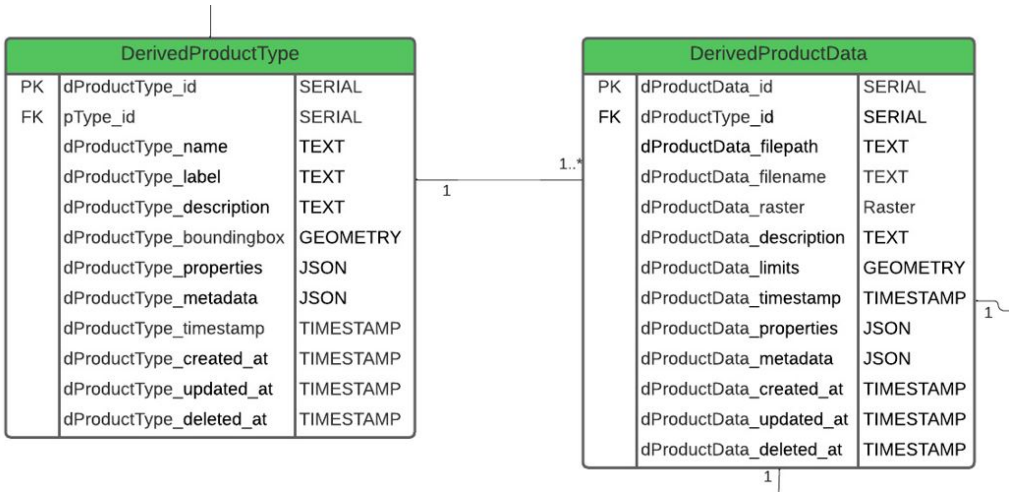

Figura 3.6: Módulo de Dados Processados

Tipo de produto derivado

O tipo de produto derivado descreve o tipo de ficheiro obtido através da transformação dos ficheiros originais. Nesta dissertação serão utilizados, maioritariamente, os ficheiro *raster*, porém também foram criados ficheiros *shapefile*.

Dados de produto derivado

Tal como no módulo anterior, nesta tabela serão armazenados todos os ficheiros gerados. Porém esta tabela armazena grande parte dos dados relevantes para toda a dissertação no atributo *raster* associado. Desta forma, os ficheiros *raster* são armazenados internamente na base de dados e não apenas numa diretoria exterior.

3.3.5 Módulo de Conteúdo de Dados

O módulo de conteúdo de dados é o responsável por armazenar toda a informação dos produtos analisados, não só a utilizada para obter os valores de cada ponto, mas qualquer informação necessária para saber do que se trata um determinado produto.

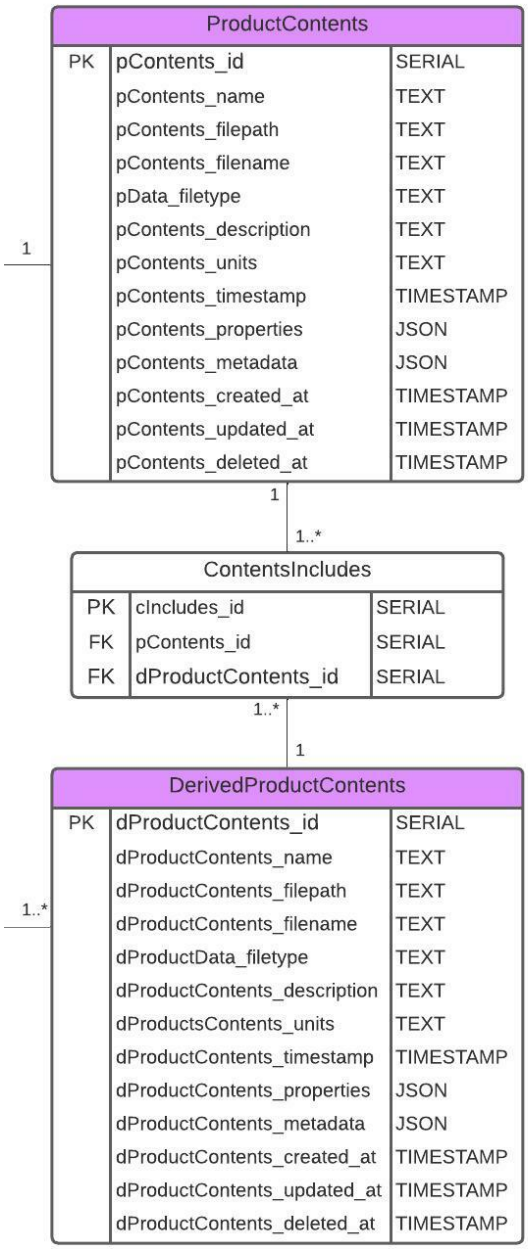

Figura 3.7: Módulo de Conteúdo de Dados

3.3.6 Módulo de Séries Temporais

Por fim, temos o módulo de séries temporais que apenas está presente num dos modelos de dados apresentados. Este é representado pela cor amarela, e aparece na imagem interligado com tabelas de outros módulos uma vez que este módulo ordena os dados tendo em conta um atributo relativo ao tempo e não pode conter índices únicos, logo não permite a existência de uma chave primária. Condição esta, existente na base de dados *Postgresql + TimescaleDB*, aquando da criação de uma *hypertable*, tabelas que particionam

os dados automaticamente por uma variável de tempo. Assim, para ter acesso ao resto da informação dos ficheiros, foi necessário a criação da chave estrangeira das outras tabelas.

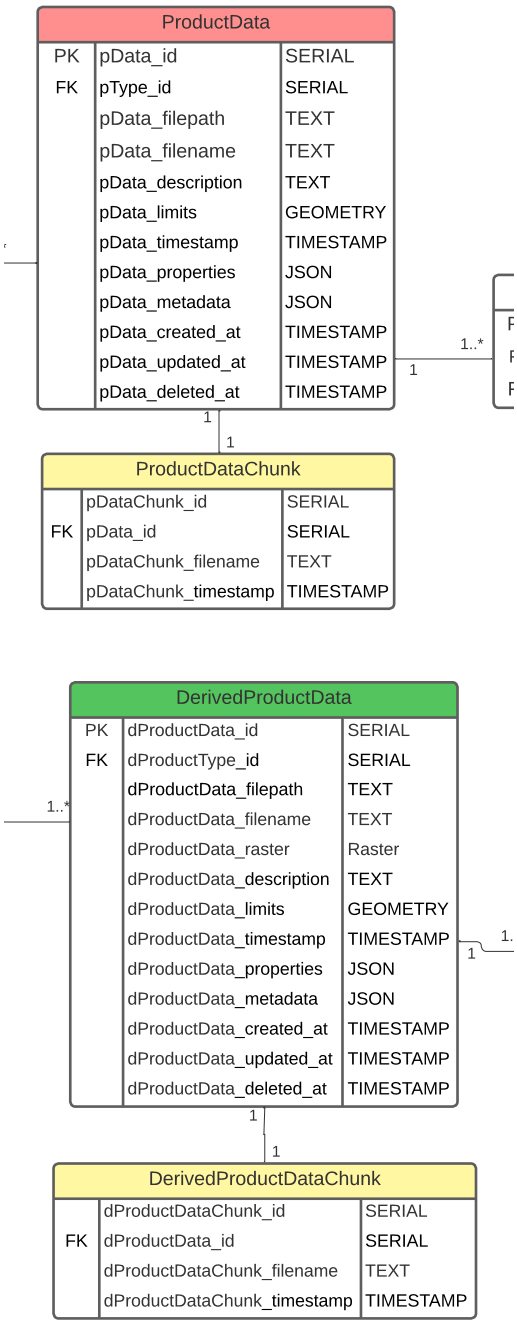

Figura 3.8: Módulo de Séries Temporais

3.4 Metodologia de desenvolvimento

Nesta pequena secção será abordada a metodologia de desenvolvimento adoptada ao longo da implementação desta dissertação. Esta metodologia baseou-se na construção da arquitetura por etapas de forma a organizar melhor o trabalho, e não na implementação de várias componentes em simultâneo. Desta forma, foi possível já ter garantias acerca do que seria necessário para a continuação da implementação para a próxima componente.

Assim, primeiramente, através da arquitetura do sistema dividiu-se a implementação em 3 etapas:

- **Descarregamento e processamento de dados**

  Em primeiro lugar, definiram-se que produtos descarregar do IPMA que tivessem os dados pretendidos para armazenar. Após decidida que informação ir obter, criaram-se os *scripts* necessários para o fazer. Após ter a certeza que estava tudo a funcionar como previsto, criaram-se os *scripts* para processar os dados escolhidos, do produto que fornece a temperatura do solo 2.3.2.1, da forma pretendida.

- **Modelação e criação de bases de dados:**

  Assim que terminada a etapa anterior, já com ideia do volume e tipo de dados a tratar, foi modelada a base de dados relacional de forma a ser o máximo eficiente, quando lidando com os dados a armazenar. Em seguida, criou-se e povoou-se a base de dados de forma a correr pequenos testes para verificar se estava tudo a funcionar. Só no fim de ter a certeza que o modelo não precisava de mais ajustes, é que se repetiu o processo para a base de dados de séries temporais.

- **Análise dos dados e Resultados (Camada de visualização):**

  Por fim, já tendo praticamente todo o sistema a funcionar, efetuaram-se diversas *queries* de modo a registar o quão eficientes são as bases de dados, para posteriormente compará-las e tirar as conclusões necessárias, baseadas nos objetivos da dissertação.

3.5 Conclusões

Concluindo, foi pensado um sistema de informação o mais abrangente e simples de compreender possível. Essas características foram obtidas com a ajuda dos dois modelos de dados apresentados, que armazenam a informação de uma forma genérica, e não só direcionada para o tipo de dados utilizado ao longo desta implementação, bem como a nomenculatura, referida na secção 3.3.1, que ajuda qualquer utilizador a compreender a que informação se refere cada entidade ou atributo presentes nas tabelas. Estes modelos cumprem também todos os requisitos que foram levantados como essenciais, aquando do desenvolvimento dos mesmos.

Por fim, a metodologia de desenvolvimento utilizada, permitiu uma implementação mais estruturada e consolidada, o que tornou a construção do sistema mais fluida. Assim, quando se avançavam etapas, raramente foi necessário efetuar correções nas anteriores tendo a implementação seguido uma ordem contínua, e não uma ordem desconectada.

4

Implementação

Ao longo deste capítulo será detalhado todo o processo de implementação do sistema de informação modelado no capítulo anterior.

4.1 Introdução

Após estar completa toda a modelação do sistema, neste capítulo explicaremos ao pormenor toda a implementação do mesmo.

Neste quarto capítulo, inicialmente, vão ser discutidas as tecnologias escolhidas para a conclusão do sistema, em seguida são explicados todos os processos que envolvem a obtenção e tratamento dos dados meteorológicos armazenado pelo sistema e, por fim, o servidor utilizado para testes e as respetivas base de dados.

Em suma, serão explicadas ao pormenor cada uma das componentes presentes na arquitetura do sistema, apresentada em 3.2.

4.2 Tecnologias escolhidas

Nesta secção são apresentadas todas as tecnologias escolhidas para implementar as quatro principais componentes deste sistema, de forma a cumprir todos os requisitos e objetivos descritos ao longo do documento.

4.2.1 Bash

A primeira tecnologia escolhida, coincide também com a primeira fase do sistema, obter os dados meteorológicos. Desta forma, para descarregar a informação necessária do servidor do IPMA, onde estão disponíveis todos os produtos gerados a partir dos satélites, foi escolhida a linguagem de *scripting* Bash.

Uma vez que era necessária uma gestão de diretorias e ligação a um servidor por protocolos SSH/SFTP, ligação essa feita através da linha de comandos, a utilização do Bash foi bastante óbvia, tendo em conta que esta linguagem permite a utilização da ferramenta/biblioteca curl, ideal para a tranferência de dados com a utilização de URLs.

4.2.2 Python

Após já terem sido descarregados todos os dados, é necessário, novamente, uma linguagem de *scripting* de forma a manipular os dados obtidos para posteriormente serem armazenados no sistema de informação.

A escolha pelo Python, neste caso foi fácil, visto que a informação vem toda em ficheiros do tipo HDF5 (2.3.3) e esta linguagem tem uma biblioteca que facilita a manipulação deste tipo de ficheiros. Apesar de não ser a única com esta biblioteca, é também uma das mais aconselhadas para tratar grandes volumes de dados, para além de que já existe familiarização com a linguagem.

4.2.3 QGIS

Apesar de numa fase final do sistema não existir nada que indique a utilização desta ferramenta, era necessário utilizar algo que tornasse possível a visualização gráfica dos ficheiros do tipo *raster* e *shp*, de forma a verificar que o tratamento dos ficheiros HDF5 foi realizado como pretendido.

Assim, sendo *open-source*, executável em praticamente todos os sistemas de operação, apresenta uma documentação bastante útil, e, principalmente, fácil de adaptar as suas ações para Python, QGIS foi a ferramenta utilizada para as visualizações gráficas.

4.2.4 PostgreSQL

A base de dados adotada foi o PostgreSQL 12. Tendo em conta que os dados a armazenar no sistema têm uma forte componente geográfica, a extensão PostGIS desta base de dados vem responder a esses pré-requisitos.

A outra razão que também influenciou bastante a escolha, prende-se no facto de que o Projeto FitoAgro (1.3) também está implementado com esta mesma base de dados, funciona corretamente e de forma eficiente.

4.2.5 TimescaleDB

Após a escolha da base de dados relacional, é preciso também escolher uma de time series. Não existindo qualquer familiarização com nenhuma existente, a escolha foi baseada no facto de o PostgreSQL ter uma extensão para timeseries, o que aproxima bastante as duas base de dados. E, tendo em conta que um dos grandes objetivos é a comparação das mesmas, manter o mesmo sistema e apenas usar extensões diferentes torna a comparação ainda mais relevante.

Assim, TimescaleDB 1.7.5 foi a extensão adotada para a implementação da segunda base de dados. Esta tem ainda uma documentação muito rica, o que facilita a compreensão da mesma.

4.2.6 pgAdmin

Uma vez já escolhido o sistema a utilizar para implementar a base de dados, era necessário escolher uma ferramenta capaz de administrar e gerir a mesma.

Sendo um dos objetivos representar graficamente as informações obtidas através de *queries* às bases de dados, a escolha para representá-las foi a utilização do *pgAdmin*, uma das ferramentas mais populares e ricas em funcionalidades para o sistema PostgreSQL, que permite diversas formas de visualização e análise das *queries*.

Tal como já referido anteriormente, esta foi mais uma das ferramentas já utilizadas, pelo que já existe algum conhecimento acerca da mesma, o que também influenciou na escolha.

4.3 Integração e obtenção de dados meteorológicos

Ao longo desta secção vai ser explicada com algum pormenor a implementação das componentes integração de dados e obtenção de dados.

4.3.1 Descarregamento de dados

A obtenção dos dados é feita através da execução de um *shell script*, que descarrega os produtos pretendidos de um servidor disponibilizado pelo IPMA.

Uma vez que os produtos presentes no servidor do LANDSAF têm todos o nome de ficheiro formados da mesma forma - <Tipo de ficheiro>_<Tipo de satélite>_<Nome do produto>_<Data do produto>- é pela formação do nome de ficheiro que o *script* começa. Inicialmente, guarda o prefixo (que inclui o tipo de ficheiro e de satélite) numa variável, uma vez que é constante em todos os produtos, depois com o auxílio de um ficheiro .txt onde estão guardados os nomes dos produtos a descarregar e a frequência, em minutos, com que são produzidos, ou seja, de quantos em quantos minutos os satélites disponibilizam novos ficheiros, obtém-se o nome do produto só ficando a faltar preencher a parte da data do produto.

```
# prefixo dos ficheiros a descarregar
fileprefixname = "HDF5_LSASAF_"

# le de um ficheiro com que frequencia existem novos ficheiros
# para descarregar
freq = $(echo "$line" | cut -d ">" -f2)

# le do ficheiro o nome do produto a descarregar
variableshortname = $(echo "$line" | cut -d ">" -f1)
```

```
# a partir do nome do produto retira apenas a abreviatura do dado
# meteorologico que o produto analisa
product = $(echo "$variableshortname" | cut -d "_" -f2)

if [ "$product" == "EDLST-DAY" ] ||
    [ "$product" == "EDLST-NIGHT" ] ;
then
    product = "EDLST"
elif [ "$product" == "ET" ] && [ "$product" == "LST" ]
then
    product="M$product"
fi
```

Listagem 4.1: Estruturação do nome do ficheiro a descarregar

Para adicionar a data ao nome do ficheiro é necessário um processo mais complexo, uma vez que é necessário saber qual a data do último ficheiro transferido. Assim, coloca-se o *script* a executar na diretoria do ficheiro que pretendemos saber a data e faz-se o tratamento do nome do ficheiro até ficar só a data do mesmo. Em seguida, uma vez que através do Bash não foi possível manipular a data, converte-se a mesma para segundos e depois adiciona-se a frequência com que o produto é produzido à data obtida, também em segundos, volta-se a converter para o formato de data pretendido, (AAMMDDhhmm), e tem-se o nome completo do primeiro ficheiro a descarregar do servidor. O seguinte obtem-se através da soma da frequência guardada no ficheiro .txt, referido anteriormente, à data do ficheiro que foi descarregado por último, e assim sucessivamente.

```
# vai buscar a data do ultimo ficheiro descarregado na diretoria
file = `gdalinfo $file_name | grep "SENSING_START_TIME"
        | cut -d "=" -f2`

# obtem o ano
year = $(echo ${file::4})

# obtem o mes
month = $(echo ${file:4:2})

# obtem o dia
day = $(echo ${file:6:2})
```

```
# obtem a hora
hour = $(echo ${file:8:2})

# obtem o minuto
minute = $(echo ${file: -2})

# conversao da hora e minutos em segundos
sum = $((hour*60*60 + minute*60))

# data sem horas
thedate="${year}${month}${day}"

# data sem horas em segundos epoch
file_date = `date "+%s" -d $thedate`

# data com horas em segundos epoch
file_date = $((file_date+sum))

# juncao do nome do ficheiro inteiro como esta no servidor, para
# depois descarregar
bandFilename="${fileprefixname}${variableshortname}_${file_date}"
```

Listagem 4.2: Obtenção da data do ficheiro a descarregar

Uma vez obtido o nome do ficheiro a descarregar, através de um ciclo descarregam-se todos os ficheiros até a data ser igual à do dia em que se corre o *script*, com a ajuda da ferramenta *curl*, mencionada na secção 4.2.1. Onde, apenas é necessário indicar o caminho do ficheiro que se pretende descarregar, daí a necessidade de se saber o nome do ficheiro.

4.3.2 Tratamento de dados

Após ter os ficheiros dos produtos, foi necessário manipular esses mesmos ficheiros de forma a limitar a área a analisar. Visto que os ficheiros descarregados apresentam valores relativos a cerca de metade da superfície do globo, pretende-se que essa área seja reduzida apenas à Península Ibérica.

Para conseguir reduzir a área, à qual o ficheiro fornece valores, foi criado um *script*, em Python, que transforma uma matriz bidimensional - ou *array* - com os valores de todos os continentes noutra mais pequena, onde apenas estão os valores correspondentes à Península Ibérica.

Porém, como explicado na subsecção 2.3.3, os ficheiros HDF5 podem ter diversos *datasets*, logo em primeiro lugar foi necessário aceder ao dataset correto para depois

se conseguir manipular a informação pretendida. Na listagem seguinte, 4.3, é possível observar o código responsável por todo esse processo.

```
# iterar sobre os ficheiros na diretoria
for filename in os.listdir(DIRECTORY):
    f = os.path.join(DIRECTORY, filename)

    # abrir o ficheiro e aceder ao dataset
    file = h5py.h5f.open(f.encode())
    dset = h5py.h5d.open(file, DATASET.encode())
```

Listagem 4.3: Como aceder ao dataset pretendido num ficheiro HDF5

Uma vez tendo acesso ao dataset pretendido, define-se o *hyperslab* - porção de dados da matriz bidimensional definido através dos parâmetros referidos na listagem seguinte - onde estão os dados referentes à Península Ibérica e lê-se essa informação para uma outra matriz bidimensional. O *offset* escolhido para representar a Península Ibérica foi determinado manualmente, a partir de um exemplo de pontos, visto que não foi encontrada qualquer informação acerca do mesmo.

```
# parametros para execucao da funcao que escolhe apenas a parte
# pretendida do dataset
space = dset.get_space()
start = (450, 1590)
stride = (1,1)
count = (221, 361)
block = (1, 1)
space.select_hyperslab(start, count, stride, block)

# ler os dados utilizando o hyperslab definido anteriormente
rdata = np.zeros(dims, dtype=np.int16)
dset.read(h5py.h5s.ALL, space, rdata)

raster = np.array(rdata)
```

Listagem 4.4: Obtenção dos valores da Península Ibérica

Por fim, é feita a conversão dos índices da matriz para as respetivas coordenadas geográficas através de diversas fórmulas fornecidas pelo IPMA, através do manual dos produtos disponibilizados [29].

Constantes:

COFF = 1857
LOFF = 1857
LFAC = 13642337
CFAC = 13642337
P3 = 1737121856
P2 = 1.006803
P1 = 42164

COFF e LOFF dependem da localizacao de cada area geografica LANDSAF no disco Meteostat

LFAC e CFAC fatores de escala de colunas e linhas que dependem do input dos dados SEVIRI

$$\text{Longitude (em graus)} \rightarrow lon = \arctan\left(\frac{S_2}{S_1}\right)$$

$$\text{Latitude (em graus)} \rightarrow lat = \arctan\left(p_2 \cdot \frac{S_3}{S_{xy}}\right)$$

onde:

$$s_1 = p_1 - s_n \cdot \cos x \cdot \cos y$$
$$s_2 = s_n \cdot \sin x \cdot \cos y$$
$$s_3 = s_n \cdot \sin y$$
$$s_{xy} = \sqrt{s_1^2 + s_2^2}$$
$$s_d = \sqrt{(p_1 \cdot \cos x \cdot \cos y)^2 - (\cos^2 y + p_2 \cdot \sin^2 y) \cdot p_3}$$
$$s_n = \frac{p_1 \cdot \cos x \cdot \cos y - s_d}{\cos^2 y + p_2 \cdot \sin^2 y}$$

onde:

$$x = \frac{ncol - COFF}{2^{-16} \cdot CFAC}$$
$$y = \frac{nlin - LOFF}{2^{-16} \cdot LFAC}$$

Listagem 4.5: Fórmulas para conversão de coordenadas geográficas

E, criado um ficheiro .txt com toda essa informação disponibilizada em três colunas, latitude, longitude e valor de temperatura, em graus *celsius*, mas apenas as coordenadas que têm valores de temperatura válidos. Por fim, com as mesmas três colunas, é guardada num ficheiro csv toda a informação, até as coordenadas com valores de temperatura que não foram possíveis captar através dos satélites.

Com os dois ficheiros referidos anteriormente, que contêm os valores e coordenadas geográficas de cada píxel presente na região escolhida, são criados os ficheiros *raster* e shp que são armazenados na base de dados. Estes ficheiros são gerados através da execução de mais um *script*, este simulando a utilização da ferramenta QGIS, referida na secção 2.4.3.

Inicialmente, adicionam-se os ficheiros .txt e .csv como *layer* (camada).

```
# caminho onde armazenar os dados
datapath = "file:///F:/Tese/LSASAF_Products/Data/"

# link para o ficheiro .txt
uri = datapath + producttypem + "/" + str(filename) +
"?type=csv&useHeader=No&maxFields=10000&detectTypes=yes&xField=
field_1&yField=field_2&crs=EPSG:4326&spatialIndex
=no&subsetIndex=no&watchFile=no"

# adicionar camada a partir de ficheiro .txt
txtlayer = QgsVectorLayer(uri, splitfilename[0] + '_txt',
"delimitedtext")

QgsProject.instance().addMapLayer(txtlayer)

# link para o ficheiro .csv
uri = datapath + producttypem + "/" + str(filename) + "?type=csv
&maxFields=10000&detectTypes=yes&xField=Longitude%0(degrees
%20east)&yField=Latitude%20(degrees%20north)&crs=EPSG:4326&
spatialIndex=no&subsetIndex=no&watchFile=no"

# adicionar camada a partir do ficheiro .csv
csvlayer = QgsVectorLayer(uri, splitfilename[0] + '_csv',
"delimitedtext")

QgsProject.instance().addMapLayer(csvlayer)
```

Listagem 4.6: Criação dos layers

Em seguida, a partir dos *layers* adicionados cria-se o ficheiro .shp correspondente.

```
# retorna a primeira camada presente no QGIS
layers = QgsProject.instance().mapLayersByName(splitfilename[0]
+ '_txt')
layer = layers[0]

# caminhos para os ficheiros
shppath = "F:/Tese/LSASAF_Products/Shapefiles/"
shpfoldername = "/Portugal_Evapormap_"

# criar um ficheiro shapefile a partir de um ficheiro .txt
# e adiciona-lo como camada
if(producttypem == "DMET" or producttypem == "MET" or
producttypem == "METREF"):

    os.mkdir(shppath + producttypem + shpfoldername + producttype
    + "_" + splitfilename[0].split("_")[1])

    shapefn = shppath + producttypem + shpfoldername +
    producttype + "_" + splitfilename[0].split("_")[1] +
    shpfoldername + producttype + "_" +
    splitfilename[0].split("_")[1] + ".shp"

    writer = QgsVectorFileWriter.writeAsVectorFormat(layer,
    shapefn, 'utf-8', \driverName='ESRI Shapefile')

    shapelayer = iface.addVectorLayer(shapefn, '', 'ogr')

    del(writer)
```

Listagem 4.7: Criação do ficheiro .shp

E, por fim, a partir dos *layers* adicionados e o .shp criado é gerado o ficheiro *raster* através de um comando, fornecido pela ferramenta QGIS, executado na linha de comandos.

```
# caminhos para os ficheiros
rasterFolderName = "Portugal_Heatmap_"
rasterpath = "F:/Tese/LSASAF_Products/Rasters/"
coords = "-10.680006226 35.559069713 3.738128082 44.738354486"

# comando que cria um ficheiro raster atraves de rasterize
# (shapefile to raster)
cmd = "gdal_rasterize -l" + rasterfoldername + producttype
+ "_" + splitfilename[0].split("_")[1] + " -a field_3 -ts
361.0 221.0 -a_nodata 0.0 -te " + coords + " -ot Float32 -of
GTiff " + shppath + producttypem + "/" + rasterfoldername +
producttype + "_" + splitfilename[0].split("_")[1] + "/" +
rasterfoldername + producttype + "_" +
splitfilename[0].split("_")[1] + ".shp " + rasterpath +
producttypem + "/Raster_" + producttype + "_" +
splitfilename[0].split("_")[1] + ".tif"

os.system(cmd)
```

Listagem 4.8: Criação do raster

4.4 Base de dados

A presente secção aborda os temas essenciais à implementação da base de dados. Como referido anteriormente, nas secções 4.2.4, 4.2.5 e 4.2.6, os sistemas utilizados para a implementação de ambas as base de dados foram PostgreSQL e o TimeScaleDB, enquanto que a gestão de grande parte da implementação foi realizada através do pgAdmin.

4.4.1 Povoamento das bases de dados

Após a criação de ambas as bases de dados com a ajuda do pgAdmin, foi necessário inserir todos os dados descarregados nas base de dados, de modo a estarem prontas para ser testadas. Para tal, foi criado um *script* capaz de gerar um ficheiro .sql com todos os dados necessários para inserir um registo na base de dados.

Este *script* através dos raster gerados, utiliza o comando **raster2pgsql**, para criar um ficheiro com a informação necessária para inserir um raster numa base de dados a partir do seu ficheiro .tiff. Porém, como a tabela onde se colocam os raster não tem apenas essa coluna, foi necessário adicionar as restantes colunas de forma a inserir um registo válido.

Por fim, através do comando **psql** insere-se o documento .sql preenchido na base de dados pretendida, como se pode ver na listagem seguinte.

```
# comando que cria o raster a adicionar na base de dados a
# partir do ficheiro . tiff
raster2pgsql -a -f dproductdatachunk_raster $filename
derivedproductdatachunk > $concat.sql

polygonshape = 'POLYGON((-10.680 44.738,-9.157 35.611,3.249
35.533,3.778 44.608,-10.680 44.738))'
connectionstring = -p 5432 -h 127.0.0.1 -d lsasaf_timeseries

# insercao na base dados atraves de um ficheiro .sql
"INSERT INTO derivedproductdata (dproducttype_id,
dproductdata_filename, dproductdata_filepath,
dproductdata_raster, dproductdata_limits,
dproductdata_timestamp, dproductdata_created_at)
VALUES("$id", '"$filename"', '"$filepath"',
"$rast_filename", ST_GeometryFromText(" + polygonshape + ",4326),
'"$timestamp"', current_timestamp);" > $concat.sql

psql connectionstring -U postgres -f
$concat.sql
```

Listagem 4.9: Inserção de um ficheiro .tiff na base de dados

4.4.2 Índices

Uma vez que ambas as bases de dados armazenam ficheiros do tipo *raster*, cada um com bastantes píxeis, foi necessário tornar mais eficientes os acessos aos valores desses mesmos pixéis. Desta forma, após uma breve pesquisa, criaram-se dois índices espaciais em cada uma das bases de dados, relacional e de séries temporais.

Para a criação desses mesmos índices foi também necessário adicionar duas novas colunas às tabelas, não representadas no modelo de dados anterior, que armazenam os ficheiros *raster*, uma que representa a extensão da grelha do ficheiro e outra que representa apenas a extensão dos píxeis com valores válidos do ficheiro.

Assim, foram criados os seguintes indíces:

```
CREATE INDEX derivedproductdata_tile_extent_idx ON
derivedproductdata USING GIST(ST_Envelope(tile_extent));

CREATE INDEX derivedproductdata_raster_valued_extent_idx ON
derivedproductdata USING GIST(ST_Envelope(raster_valued_extent));
```

Listagem 4.10: Criação de indíces

Após diversos testes à performance das bases de dados, conclui-se que nenhum dos índices criados foi utilizado, tendo em conta que as *queries*, que devolvem os resultados pretendidos, não envolvem nenhuma das colunas indexadas.

Desta forma, criaram-se duas novas tabelas com o intuito de inserir pesquisas às bases de dados, em colunas indexadas. Assim, criaram-se os mesmos três indíces GIST em ambas as tabelas.

```
CREATE INDEX idx_geom ON
derivedproductdata USING GIST(geom);

CREATE INDEX idx_xy ON
derivedproductdata USING GIST(x,y);

CREATE INDEX idx_yx ON
derivedproductdata USING GIST(y, x);
```

Listagem 4.11: Criação de indíces

Sendo que desta vez, todos os índices foram utilizados e testados com sucesso.

4.5 Conclusões

A implementação do sistema foi bem sucedida apesar de, inicialmente terem existido algumas dificuldades na tranformação dos ficheiros descarregados do servidor do IPMA, uma vez que se tentou utilizar uma solução desenvolvida em java, já antiga, que não se conseguiu compilar, mesmo após a atualização das bibliotecas necessárias. A implementação das bases de dados foi apenas desenvolvida localmente, o que não possibilitou povoá-las com elevadas quantidades de informação, apenas dados relativos a um mês.

Relativamente às tecnologias utilizadas, as escolhas facilitaram bastante o desenvolvimento da solução. Como foi o caso da escolha das bases de dados, uma vez que o TimescaleDB é uma extensão do PostgreSQL permitiu que fosse apenas necessário dominar os comandos e construção de *queries* de uma base de dados.

No que diz respeito à integração e tratamento dos dados, a utilização da ferramenta QGIS, revelou-se fundamental. Não só permite ao utilizador observar graficamente os dados que se estão a tratar, podendo fazer uma análise detalhada se estes correspondem ao pretendido ou não, como mostra ao utilizador os comandos que este executa para obter os gráficos e imagens produzidas, facilitando assim a criação de *scripts*.

Por fim, não foram apenas as bases de dados escolhidas que facilitaram implementação da camada de armazenamento do sistema, a ferramenta pgAdmin tornou tudo ainda mais eficiente, tendo em conta que apresenta uma interface capaz de criar todas as tabelas, índices e atributos, presentes em ambas as bases de dados. Isto, faz com que o utilizador não sinta necessidaade de recorrer a uma interface de linha de comandos e pesquisar sobre os comandos necessários para efetuar essas mesmas operações.

5

Avaliação e testes

O presente capítulo serve para avaliar o resultado final do sistema de informação implementado, e também para apresentar os resultados obtidos através da mesma.

Primeiramente, será apresentada uma breve avaliação quantitativa quanto à aplicabilidade do sistema desenvolvido, noutros projetos que não o Fitoagro. Em seguida, uma apresentação pormenorizada dos resultados obtidos. E, por fim, uma breve conclusão.

5.1 Aplicabilidade

Numa altura em que a informática está cada vez mais presente na agricultura, será que um sistema de informação capaz de integrar dados obtidos através de satélites, e outros fornecedores de dados idênticos, é aplicável a outros sistemas, utilizados para controlar diversas condições relacionadas com a produção agrícola?

Para responder à pergunta anterior, pode-se assumir dois tipos diferentes de aplicabilidade, aplicabilidade indireta e direta. A indireta, no sentido em que, mesmo não integrando as componentes deste sistema nalgum projeto do mesmo âmbito do descrito nesta dissertação, Fitoagro, é possível analisar os resultados obtidos, que irão ser apresentados e explicados na próxima secção, e fazer diversas decisões. Entre elas, a escolha das bases de dados ou mesmo diferentes formas de como tratar os dados para povoar um sistema. Assim, mesmo não utilizando qualquer parte da implementação, o conteúdo desta dissertação será aplicável.

Por outro lado, temos a aplicabilidade direta. É desta forma, que o sistema, desenvolvido no âmbito desta dissertação, oferece o maior número de vantagens. Porém, nem todos os sistemas beneficiam com esta integração. Para tal, é necessário que possuam alguns pré-requisitos funcionais, como:

- **Descarregamento e processamento de dados**

  Caso o projeto nessecite de obter dados de um fornecedor de dados, como um satélite ou um qualquer modelo de previsões meteorológicas, desde que o formato inicial sejam ficheiros HDF5 e o final pretendido sejam ficheiros *raster* ou .shp, facilmente poderia aplicar os *scripts* desenvolvidos de forma a automatizar todo o

processo de descarregamento e processamento desses mesmos dados. Necessitando apenas de efetuar pequenas alterações em variáveis relativamente a algumas diretorias e/ou servidor.

- **Modelação e criação de bases de dados espaciais:**

  Quanto à modelação e criação da camada de armazenamento, uma vez que o modelo desenvolvido tinha como um dos principais objetivos, ser o mais genérico possível, qualquer sistema que necessecite de implementar uma base de dados espacial, capaz de armazenar dados relativos a coordenadas geográficas, beneficiará da utilização do mesmo.

- **Análise dos dados e Resultados:**

  Por fim, para análise dos dados ou mesmo para tirar resultados, as *queries*, que irão ser demonstradas na próxima secção, permitem tirar conclusões quanto à eficiência do sistema e também à utilidade dos dados armazenados, caso se pretendam analisar as mesmas condições que na dissertação desenvolvida.

5.2 Testes de desempenho

Ao longo desta secção, serão apresentados todos os testes de desempenho efetuados a ambas as bases de dados presentes no sistema, relacional e de séries temporais.

Primeiramente, serão enunciados os casos de uso que originaram as *queries* executadas para os testes, de seguida expostos os resultados das mesmas, e, por último, uma breve explicação de como as bases de dados percorrem a sua informação para devolver os resultados das *queries*.

5.2.1 Casos de uso

De forma a testar a eficiência das bases de dados da forma mais completa e obtendo-se os resultados mais próximos da realidade possíveis, tiveram-se quatro aspetos em consideração que eram necessários incluir em cada análise:

- Regiões geográficas;
- Intervalos de Tempo;
- Valores dos píxeis;
- Agregação de valores.

Desta forma, foram escolhidos seis casos de uso para efetuar os testes de desempenho ao sistema. Destes seis casos, podem-se separar dois grupos, o primeiro grupo onde se obtêm os valores dos píxeis por intervalos de tempo e o segundo em que se obtem a média dos valores dos píxeis por ficheiros *raster* também por intervalos de tempo. Ambos os

grupos sempre incluindo procuras relacionadas com os primeiros três aspetos referidos anteriormente, porém apenas o segundo inclui o quarto e último, a agregação de valores.

Assim, os seis casos de uso são:

- Obter os valores de um píxel específico ao longo de um determinado intervalo de tempo;
- Obter os valores de todos os píxeis presentes numa região escolhida, num determinado instante;
- Obter os valores de todos os píxeis presentes numa região, ao longo de um determinado intervalo de tempo;
- A média dos valores de um determinado píxel por cada ficheiro ao longo de um determinado intervalo de tempo;
- A média dos valores de um determinado conjunto de píxeis num determinado instante do tempo;
- A média dos valores de um determinado conjunto de píxeis num determinado intervalo de tempo.

5.2.2 Características do Sistema

A máquina utilizada para a realização dos testes de desempenho foi a mesma utilizada para o desenvolvimento de todo o sistema de informação, um portátil LENOVO ideapad 320S, cujas suas características são:

- **Processador:** Intel® Core™ i5-8250U CPU @ 1.60GHz
- **Memória:** 8 GB
- **GPU:** NVIDIA GeForce MX130 e Intel® UHD Graphics 620
- **Disco:** SSD 212 GB, com cerca de 56 GB livres
- **Sistema Operativo:** Windows 10 Home Versão 22H2

Para além destas características, é de salientar que o servidor onde as base de dados foram integradas foi o *Windows Subsystem for Linux*, WSL. Este executa o sistema Ubuntu 20.04.4 LTS, a partir de um sistema operativo *Windows*.

5.2.3 Integração dos Dados

A integração dos dados nas respetivas bases de dados, foi, como anteriormente explicado, realizada através da execução de alguns *scripts* desenvolvidos para o efeito.

Uma vez que, a máquina utilizada para realizar os testes de desempenho tem pouca capacidade, do ano inteiro de dados processados, apenas se inseriram os ficheiros relativos a um mês, neste caso, o mês de abril de 2022, o que gerou 5655 entradas na tabela que armazena os ficheiros *raster*, cerca de 13,5 GB.

5.2.4 Consultas aos Dados

Para efetuar as consultas foram escolhidas trinta e seis *queries* por base de dados, apenas com pequenas diferenças entre elas, de maneira a ir de encontro aos casos de uso referidos anteriormente, 5.2.1, e também a avaliar ambas as bases de dados em diferentes aspetos, podendo assim tornar a sua comparação mais rica e pormenorizada. Para além das diferenças necessárias efetuar para cada caso de uso, existem ainda diferentes intervalos de tempo a ser analisados e ordenação crescente dos resultados obtidos. Desta forma é possível testar diferentes volumes de resultados, em cada caso de uso, e entender o comportamento das bases de dados para diversas situações e não apenas uma em específico.

Para além destas trinta e seis consultas, foram ainda realizadas outras setenta e duas semelhantes, outros dois conjuntos de trinta e seis. A única diferença entre cada um desses três conjuntos é a forma como se obtêm os valores de todos os pontos dos ficheiros raster, ou seja, os valores da temperatura em cada ponto da Península Ibérica, existente no ficheiro.

No primeiro conjunto de consultas recorre-se à função ST_PixelAsCentroids(raster), função esta que recebe um ficheiro do tipo raster e retorna o centróide para cada píxel existente no ficheiro junto com o seu valor, as coordenadas x e y, e a sua geometria.

No segundo conjunto, ao invés de se utilizar a função explicada anteriormente, guardaram-se todos os resultados da mesma numa tabela nomeada idxRasterData e utilizou-se essa mesma tabela para obter os valores necessários à realização das consultas.

E, por fim, pesquisaram-se todas as coordenadas geográficas (x e y), não repetidas, existentes em todos os ficheiros armazenados nas bases de dados e criou-se uma tabela, idxPixels, que guarda numa coluna todas as coordenadas x, noutra as y e na última a geometria, ou seja, o ponto que delas resulta. Após ter a tabela criada e preenchida com a informação necessária, utiliza-se a função ST_Value(raster, x, y), esta recebe um ficheiro raster, um x e um y de um ponto e retorna o valor de um ficheiro na coluna y e linha x. Com os valores obtidos efetuam-se as consultas necessárias, porém as consultas deste último conjunto apresentam maior número de registos , uma vez que apresentam os valores para todos os pontos da península ibérica, mesmo os que não se conseguem obter devido à existência de nuvens na altura em que o satélite obteve a informação.

É de referir que, sendo um dos objetivos desta dissertação essa mesma comparação, cada consulta foi testada diversas vezes, de maneira a aproximar-se de um valor real do seu tempo de execução nas condições anteriormente enunciadas. Para além dos diversos ensaios, foi feita uma média com os cinco intervalos de tempo registados, sendo esse o valor final que é utilizado, posteriormente, nas comparações.

Por último antes de serem apresentados os resultados, todas as consultas apenas contemplam os produtos do tipo LST, 2.3.2.1, visto serem os mais regulares, apesar de as tabelas terem os registos de todos eles.

Caso de Uso 1: Obter os valores de um píxel específico ao longo de um determinado intervalo de tempo

Para testar o primeiro caso de uso, foram executadas oito consultas que apenas diferem em alguns pormenores. Todas elas devolvem os valores de um píxel ao longo de um determinado tempo.

A primeira, 5.1, devolve o nome do ficheiro, e os valores de temperatura dos píxeis que se encontram a menos de 10 quilómetros (km) de um píxel específico, ao longo de um dia inteiro. As outras três consultas devolvem exatamente o mesmo, porém em vez de ser apenas ao longo de um dia é ao longo de dez, vinte e trinta dias.

Enquanto que, as outras quatro consultas analisam exatamente o mesmo que as anteriores, mas no final, os valores obtidos são ordenados por ordem crescente. O número de valores obtidos em cada consulta, respetivamente, é 712, 4074, 8790 e 13463.

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2022-04-01 23:59:59'
AND ST_Distance(geom::geography,
ST_Point(lon, lat, srid)::geography) < 10000
```

Listagem 5.1: Consultas relativas ao caso de uso 1 utilizando a função ST_*PixelAsCentroids*

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id
INNER JOIN "idxRasterData" idx ON idx.filename = chunks.filename
AND idx.timestamp = chunks.timestamp WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2022-04-01 23:59:59'
```

```
AND ST_Distance(idx.geom::geography,
ST_Point(lon, lat, srid)::geography) < 10000
```

Listagem 5.2: Consultas relativas ao caso de uso 1 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL (SELECT * FROM "idxPixels" WHERE
ST_Distance(geom::geography, ST_Point(lon, lat, srid)::geography)
< 10000) idx WHERE dtype.id = 4 AND timestamp BETWEEN
'2022-04-01 00:00:00' AND '2022-04-01 23:59:59'
```

Listagem 5.3: Consultas relativas ao caso de uso 1 utilizando a tabela com as coordenadas

Caso de Uso 2: Obter os valores de todos os píxeis presentes numa região escolhida, num determinado instante

No segundo caso de uso, foram executadas apenas duas consultas. Ambas devolvem os valores de todos os píxeis numa região num determinado instante.

A primeira, 5.4, devolve o nome do ficheiro, e os valores de temperatura dos píxeis que se encontram a menos de 10 km de uma região específica, numa hora exata do dia. A segunda consulta devolve exatamente o mesmo, porém ordena os resultados por ordem crescente dos valores.

Nas consultas realizadas obtêm-se dez valores diferentes, respetivamente.

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp= '2022-04-01 00:00:00' AND ST_Distance(geom::geography,
ST_MakeEnvelope(min lon, min lat, max lon, max lat, srid)
::geography) < 10000
```

Listagem 5.4: Consultas relativas ao caso de uso 2 utilizando a função ST_*PixelAsCentroids*

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id
INNER JOIN "idxRasterData" idx ON idx.filename = chunks.filename
AND idx.timestamp = chunks.timestamp
WHERE dtype.id = 4 AND timestamp= '2022-04-01 00:00:00'
AND ST_Distance(idx.geom::geography,
ST_MakeEnvelope(min lon, min lat, max lon, max lat, srid)
::geography) < 10000
```

Listagem 5.5: Consultas relativas ao caso de uso 2 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL (SELECT * FROM "idxPixels" WHERE
ST_Distance(geom::geography, ST_MakeEnvelope(min lon, min lat,
max lon, max lat, srid)::geography) < 10000) idx WHERE
dtype.id = 4 AND timestamp= '2022-04-01 00:00:00'
```

Listagem 5.6: Consultas relativas ao caso de uso 2 utilizando a tabela com as coordenadas

Caso de Uso 3: Obter os valores de todos os píxeis presentes numa região escolhida, num determinado intervalo de tempo

No terceiro caso de uso, foram executadas, de novo, oito consultas. Todas devolvem os valores de todos os píxeis numa região num determinado intervalo de tempo.

A consulta, 5.7, devolve o nome do ficheiro, e os valores de temperatura dos píxeis que se encontram a menos de 10 km de uma região específica, durante um dia inteiro. As seguintes, apenas muda o facto de que são os valores durante dez, vinte e trinta dias. E, tal como no primeiro caso de uso, as últimas quatro são idênticas às primeiras, apenas ordenam os valores obtidos por ordem crescente.

O números de valores retornados, é, respetivamente, 34706, 203358, 421855 e 608570.

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2020-05-01 00:00:00'
AND ST_Distance(geom::geography,ST_MakeEnvelope(min lon, min lat,
max lon, max lat, srid)::geography) < 10000
```

Listagem 5.7: Consultas relativas ao caso de uso 3 utilizando a função ST_*PixelAsCentroids*

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id
INNER JOIN "idxRasterData" idx ON idx.filename = chunks.filename
AND idx.timestamp = chunks.timestamp WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2020-05-01 00:00:00'
AND ST_Distance(idx.geom::geography, ST_MakeEnvelope(min lon,
min lat, max lon, max lat, srid)::geography) < 10000
```

Listagem 5.8: Consultas relativas ao caso de uso 3 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela

```
SELECT filename, value FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL (SELECT * FROM "idxPixels" WHERE ST_Distance(geom::
geography, ST_MakeEnvelope(min lon, min lat, max lon, max lat,
srid)::geography) < 10000) idx WHERE dtype.id = 4 AND timestamp
BETWEEN '2022-04-01 00:00:00' AND '2020-05-01 00:00:00'
```

Listagem 5.9: Consultas relativas ao caso de uso 3 utilizando a tabela com as coordenadas

Caso de uso 4: Média dos valores de um determinado píxel por cada ficheiro ao longo de um determinado intervalo de tempo

No primeiro caso de uso do segundo grupo, executaram-se oito consultas, mas tal como todos os outros casos de uso, as alterações de uma consulta para outra são pormenores.

A primeira consulta, 5.10, dá a informação acerca da média dos valores de um determinado píxel por cada ficheiro *raster*, ao longo de um dia inteiro. As restantes, apenas diferem no intervalo de tempo analisado, dez, vinte ou trinta dias.

Já as últimas quatro, são idênticas às referidas anteriormente, porém os valores vêm ordenados por ordem crescente.

O número de registos retornados por consulta é 96, 611, 1315 e 2004, respetivamente

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2022-04-01 23:59:59'
AND ST_Distance(geom::geography,
ST_Point(lon, lat, srid)::geography) < 10000 GROUP BY filename
```

Listagem 5.10: Consultas relativas ao caso de uso 4 utilizando a função ST_*PixelAsCentroids*

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id
INNER JOIN "idxRasterData" idx ON idx.filename = chunks.filename
AND idx.timestamp = chunks.timestamp WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2022-04-01 23:59:59'
AND ST_Distance(geom::geography, ST_Point(lon, lat, srid)
::geography) < 10000 GROUP BY filename
```

Listagem 5.11: Consultas relativas ao caso de uso 4 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL (SELECT * FROM "idxPixels" WHERE
ST_Distance(geom::geography, ST_Point(lon, lat, srid)::geography)
< 10000) idx WHERE dtype.id = 4 AND timestamp BETWEEN
'2022-04-01 00:00:00' AND '2022-04-01 23:59:59' GROUP BY filename
```

Listagem 5.12: Consultas relativas ao caso de uso 4 utilizando a tabela com as coordenadas

Caso de uso 5: Média dos valores de um determinado conjunto de píxeis num determinado instante do tempo

Baseado no caso de uso número cinco, apenas se realizaram duas consultas. Uma obtendo a média dos valores de um determinado conjunto de píxeis num determinado instante, 5.13, e a outra, devolvendo os mesmos valores mas ordenados por ordem crescente.

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp = '2022-04-01 00:00:00' AND ST_Distance(
geom::geography,
ST_MakeEnvelope(min lon, min lat, max lon, max lat, srid)
::geography) < 10000 GROUP BY filename
```

Listagem 5.13: Consultas relativas ao caso de uso 5 utilizando a função ST_*PixelAsCentroids*

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL (SELECT * FROM "idxPixels" WHERE ST_Distance(geom
::geography, ST_MakeEnvelope(min lon, min lat, max lon, max lat,
srid)::geography) < 10000) idx WHERE dtype.id = 4 AND
timestamp = '2022-04-01 00:00:00' GROUP BY filename
```

Listagem 5.14: Consultas relativas ao caso de uso 5 utilizando a tabela com as coordenadas

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id
INNER JOIN "idxRasterData" idx ON idx.filename = chunks.filename
AND idx.timestamp = chunks.timestamp WHERE dtype.id = 4 AND
timestamp = '2022-04-01 00:00:00' AND ST_Distance(geom::geography
, ST_MakeEnvelope(min lon, min lat, max lon, max lat, srid)
::geography) < 10000 GROUP BY filename
```

Listagem 5.15: Consultas relativas ao caso de uso 5 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela

Caso de uso 6: Média dos valores de um determinado conjunto de píxeis ao longo de um determinado intervalo de tempo

Por fim, o sexto caso de uso, onde é retornada a média dos valores de um determinado conjunto de píxeis ao longo de um determinado intervalo de tempo.

Das oito consultas realizadas, as primeiras quatro apenas se altera o intervalo de tempo em que é feita a análise, um (5.16), dez, vinte ou trinta dias.

As outras quatro são iguais porém devolvem os valores ordenados do menor valor para o maior.

Nas consultas, o número de valores retornados é, respetivamente, 96, 805, 1748 e 2573.

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2020-05-01 00:00:00'
AND ST_Distance(geom::geography,ST_MakeEnvelope(min lon, min lat,
max lon, max lat, srid)::geography) < 10000 GROUP BY filename
```

Listagem 5.16: Consultas relativas ao caso de uso 6 utilizando a função ST_*PixelAsCentroids*

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
INNER JOIN "idxRasterData" idx ON idx.filename = chunks.filename
AND idx.timestamp = chunks.timestamp WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2020-05-01 00:00:00'
AND ST_Distance(geom::geography,ST_MakeEnvelope(min lon, min lat,
max lon, max lat, srid)::geography) < 10000 GROUP BY filename
```

Listagem 5.17: Consultas relativas ao caso de uso 6 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela

```
SELECT filename, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL (SELECT * FROM "idxPixels" WHERE
ST_Distance(geom::geography,ST_MakeEnvelope(min lon, min lat,
max lon, max lat, srid)::geography) < 10000) idx WHERE
dtype.id = 4 AND timestamp BETWEEN '2022-04-01 00:00:00' AND
'2020-05-01 00:00:00' GROUP BY filename
```

Listagem 5.18: Consultas relativas ao caso de uso 6 utilizando a tabela com as coordenadas

Posteriormente, de forma a testar diferentes agregações de valores, realizaram-se mais seis consultas por base de dados. Todas elas idênticas às deste caso de uso, duas por cada conjunto de testes, porém ao invés de agrupar pelo nome do ficheiro agruparam-se por coordenadas e por intervalo de tempo. Estas apenas foram testadas com um intervalo de tempo de trinta dias.

```
SELECT timestamp, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2020-05-01 00:00:00'
AND ST_Distance(geom::geography,ST_MakeEnvelope(min lon, min lat,
max lon, max lat, srid)::geography) < 10000 GROUP BY timestamp
```

Listagem 5.19: Consultas relativas ao caso de uso 6 utilizando agregação por intervalo de tempo

```
SELECT x, y, AVG(value) FROM derivedproductdatachunk chunks
INNER JOIN derivedproductdata datas ON datas.id = chunks.id
INNER JOIN derivedproducttype dtype ON dtype.id = datas.id,
LATERAL ST_PixelAsCentroids(raster) WHERE dtype.id = 4 AND
timestamp BETWEEN '2022-04-01 00:00:00' AND '2020-05-01 00:00:00'
AND ST_Distance(geom::geography,ST_MakeEnvelope(min lon, min lat,
max lon, max lat, srid)::geography) < 10000 GROUP BY x, y
```

Listagem 5.20: Consultas relativas ao caso de uso 6 utilizando agregação por coordenadas x,y

5.2.5 Apresentação dos Resultados

Na secção anterior, foram explicadas ao pormenor as consultas utilizadas, tendo em atenção a ordem pela qual as mesmas foram referidas, e os nomes nas listagens enunciados, 5.1, 5.4, 5.7, 5.10, 5.13, 5.16, serão apresentados nesta secção os resultados apenas das consultas cujos valores obtidos utilizam a função ST_PixelAsCentroids.

Uma vez que são demasiadas tabelas, as restantes serão apresentadas no anexo, no final do documento (I).

1º Caso de Uso

Através da observação das tabelas em baixo, 5.1 & 5.2, verifica-se que a base de dados de séries temporais é bastante mais eficaz que a relacional. Quando se retornam poucos registos, 712, elas apresentam praticamente o mesmo desempenho, porém à medida que o número de registos retornados aumenta, a diferença entre ambas as bases de dados fica cada vez mais óbvia, chegando quase a *timeseries* a ser duas vezes mais rápida a executar a mesma consulta que a relacional.

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	4,579	4,680	4,648	5,079	7,138	5,225
Query 2	38,298	38,864	39,583	40,452	41,123	39,664
Query 3	85,002	80,114	85,843	83,599	82,957	83,503
Query 4	115,454	112,345	107,887	111,294	115,324	112,461
Query 5	9,544	4,546	4,719	4,731	6,256	5,959
Query 6	40,401	37,550	42,887	39,330	43,661	40,766
Query 7	85,979	77,485	79,164	85,782	84,052	82,492
Query 8	117,801	111,374	117,123	119,350	116,621	116,454

Tabela 5.1: Resultados obtidos - caso de uso 1 na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	7,534	6,024	6,093	6,165	6,021	6,367
Query 2	65,485	69,351	72,364	69,466	73,087	69,951
Query 3	152,394	143,336	146,803	145,191	149,710	147,487
Query 4	219,682	218,466	211,458	203,353	205,214	211,635
Query 5	6,591	6,126	6,036	6,126	6,123	6,200
Query 6	69,958	63,411	69,007	71,643	66,784	68,160
Query 7	133,942	123,618	134,174	119,598	208,087	143,884
Query 8	286,251	193,050	199,010	183,952	190,673	210,587

Tabela 5.2: Resultados obtidos - caso de uso 1 na base de dados relacional

2º Caso de Uso

No segundo caso de uso, a base de dados de séries temporais apresenta uma velocidade bastante superior à relacional, cerca de três vezes mais rápida, porém o facto de as consultas apenas retornarem 10 registos, faz com que a velocidade das mesmas seja inferior a um segundo. Desta forma, e tendo em conta a máquina utilizada para alocar as bases de dados e fazer os testes, pode existir alguma incerteza quanto à discrepância dos tempos de execução apresentados.

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 17	0,162	0,292	0,282	0,154	0,200	0,218
Query 18	0,160	0,158	0,160	0,156	0,181	0,163

Tabela 5.3: Resultados obtidos caso de uso 2 na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 17	0,663	0,598	0,804	0,782	0,836	0,737
Query 18	0,631	0,565	0,572	0,484	0,779	0,606

Tabela 5.4: Resultados obtidos caso de uso 2 na base de dados relacional

3º Caso de Uso

No terceiro caso de uso, é onde fica mais evidente a diferença entre a utilização de uma base de dados *timeseries* e uma relacional. Em todas as consultas a base de dados *timeseries* apresenta uma velocidade aproximadamente três vezes superior, exceptuando na primeira (*query* 21) que é apenas cerca de duas vezes superior, uma vez que o número de registos retornado é apenas de 34706 contra os 608570 da *query* 28. Os valores que estão representados por um -"significa que foram apenas testadas três vezes essas *queries* em vez das cinco.

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 21	14,796	14,751	14,781	14,752	14,707	14,758
Query 22	118,051	118,998	120,317	101,461	115,076	114,781
Query 23	232,069	234,917	237,582	235,302	236.214	234,968
Query 24	337,268	337,289	321,977	317,501	331,836	329,174
Query 25	15,848	14,814	14,885	15,250	14,894	15,138
Query 26	115,567	123,014	129,252	120,920	125,191	122,789
Query 27	247,443	263,368	256,091	247,759	264,853	255,903
Query 28	361,897	345,943	329,698	346,579	359,356	348,695

Tabela 5.5: Resultados obtidos caso de uso 3 na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 21	41,966	20,676	24,833	32,817	32,974	30,653
Query 22	312,096	317,991	330,145	351,465	342,408	330,821
Query 23	691,616	709,755	748,232	745,015	747,980	728,519
Query 24	998,737	1071,298	1072,497	-	-	1047,511
Query 25	44,100	38,244	35,563	40,888	38,583	39,476
Query 26	344,512	311,299	321,918	346,937	354,840	335,901
Query 27	745,469	740,447	725,101	756,413	746,200	742,726
Query 28	1076,825	1069,086	1024,009	-	-	1056,640

Tabela 5.6: Resultados obtidos caso de uso 3 na base de dados relacional

4º Caso de Uso

É neste caso de uso, que começam as consultas com agregações de valores, ou seja, consultas em que se agregam os valores de temperatura dos píxeis calculando uma média ou um valor máximo, por exemplo. Isto faz com que os registos retornados sejam menores, porém as consultas têm o mesmo fluxo de execução, mas no final agrupam os valores por nome de ficheiro, neste caso.

Ao contrário dos outros três casos de uso, já foi possível observar duas consultas (*query* 9 e 13) em que a base de dados relacional é mais eficiente que a de séries temporais, porém nas restantes seis a diferença é bastante significativa, sendo a de séries temporais claramente melhor para este tipo de consultas. Esta superiorização da base de dados relacional nas duas consultas mencionadas, pode dever-se ao facto de serem devolvidos poucos registos e também de a base de dados de séries temporais realizar mais um *inner join* que a relacional.

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 9	9,778	9,722	5,171	6,893	4,607	7,234
Query 10	43,531	44,578	43,088	39,042	41,372	42,322
Query 11	86,850	84,124	83,699	78,674	78,643	82,398
Query 12	113,798	110,823	119,403	119,318	113,362	115,341
Query 13	9,938	10,025	9,572	7,504	7,720	8,952
Query 14	45,199	43,171	43,305	36,002	43,473	42,230
Query 15	82,511	77,530	83,518	83,746	84,135	82,288
Query 16	118,767	116,492	117,370	117,048	116,514	117,238

Tabela 5.7: Resultados obtidos caso de uso 4 na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 9	6,777	6,390	6,212	6,209	6,227	6,363
Query 10	60,047	53,900	61,782	63,586	58,085	59,480
Query 11	125,975	120,397	126,921	116,953	125,993	123,248
Query 12	189,183	176,635	169,944	179,300	183,456	179,703
Query 13	6,388	6,148	6,131	6,379	6,168	6,243
Query 14	59,013	57,859	67,202	61,126	61,894	61,419
Query 15	122,227	130,237	129,456	126,560	128,273	127,351
Query 16	185,056	175,469	181,133	184,301	181,440	181,480

Tabela 5.8: Resultados obtidos caso de uso 4 na base de dados relacional

5º Caso de Uso

No quinto e penúltimo caso de uso, o facto de conter agregações de valores não pareceu fazer qualquer diferença, uma vez que os tempos de execução são bastante idênticos aos do segundo caso de uso. Assim, a base de dados *timeseries* demora cerca de um terço do tempo a executar as consultas, comparativamente à base de dados relacional. É de notar que a agregação de valores não alterou os tempos de execução significativamente, isto deve-se ao facto de apenas ser retornado um registo.

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 19	0,181	0,159	0,201	0,164	0,158	0,173
Query 20	0,281	0,168	0,169	0,159	0,165	0,188

Tabela 5.9: Resultados obtidos caso de uso 5 na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 19	0,631	0,882	0,482	0,487	0,544	0,605
Query 20	0,480	0,793	0,481	0,552	0,602	0,582

Tabela 5.10: Resultados obtidos caso de uso 5 na base de dados relacional

6º Caso de Uso

Por fim, o sexto caso de uso apresenta resultados muito parecidos com o terceiro caso de uso, pela análise das tabelas (5.14 & 5.11) percebemos que a base de dados de séries temporais consegue executar as consultas cerca de três vezes mais rápido que a relacional. É salientar que a agregação de valores faz com que as consultas sejam um pouco mais demoradas do que as realizadas no caso de uso número três, sendo que a diferença do tempo de execução aumenta juntamente com o número de registos retornados, mas não de forma proporcional.

	Tentativa 1	Tentativa 2	Tentativa 3	Tentativa 4	Tentativa 5	Average (s)
Query 29	21,572	15,963	16,790	20,231	16,874	18,286
Query 30	119,672	122,234	125,064	121,358	131,144	123,894
Query 31	254,039	252,996	250,823	254,193	249,316	252,273
Query 32	367,728	386,943	352,860	352,605	344,542	360,936
Query 33	29,915	15,988	16,121	17,708	15,986	19,143
Query 34	123,263	124,605	130,225	129,415	131,145	127,731
Query 35	260,012	267,728	271,984	253,547	267,926	264,239
Query 36	388,641	387,609	375,131	397,578	391,991	388,190

Tabela 5.11: Resultados obtidos caso de uso 6 na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 29	45,803	37,004	41,618	44,787	31,825	40,207
Query 30	383,449	379,481	377,411	385,909	378,356	380,921
Query 31	767,273	763,243	768,472	764,546	765,603	765,827
Query 32	1101,373	1098,544	1100,600	-	-	1100,172
Query 33	45,296	40,891	43,357	42,426	44,818	43,358
Query 34	383,020	382,098	385,442	377,069	382,487	382,023
Query 35	767,227	770,358	765,242	766,213	768,272	767,462
Query 36	1098,856	1084,565	1097,830	-	-	1093,750

Tabela 5.12: Resultados obtidos caso de uso 6 na base de dados relacional

Nas consultas de agregação por intervalo de tempo e coordenadas os resultados também apontam para a maior eficiência da base de dados de séries temporais. Contudo na *query* 5, feita à base de dados de séries temporais, notou-se uma irregularidade anormal, o que fez com que na relacional o tempo de execução tenha sido consideravelmente menor. Uma vez que aquando destes testes, o espaço livre na máquina utilizada era escasso, ao contrário dos primeiros testes, que foram efetuados sem a existência das tabelas utilizadas para os índices. Tabelas estas que apresentam milhões de tuplos, cerca de 53,5.

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	1144,172	1137,824	1111,639	1144,34	1135,58	1134,711
Query 2	423,553	425,729	428,459	431,021	423,763	426,505
Query 3	1793,118	1787,562	1240,392	1422,562	1698,739	1588,475
Query 4	1130,14	1138,589	1092,238	1090,954	1111,836	1112,751
Query 5	419,098	431,863	430,615	416,824	402,85	420,250
Query 6	1792,236	1790,393	1771,996	1770,179	1771,512	1779,263

Tabela 5.13: Resultados obtidos caso de uso 6 na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	438,734	437,82	443,696	442, 773	465,319	446,392
Query 2	454,968	451,094	461,799	456,128	449,40	454,678
Query 3	1585,246	1577,112	1557,576	1575,913	1573,886	1573,947
Query 4	431,716	437,508	457,162	410,555	417,665	430,921
Query 5	743,109	802,837	728,495	685,878	420,765	676,217
Query 6	1764,708	1799,093	1798,366	1747,806	1799,767	1781,948

Tabela 5.14: Resultados obtidos caso de uso 6 na base de dados de séries temporais

5.3 Conclusões

A aplicação deste sistema em projetos, do mesmo âmbito que o Fitoagro, 1.3, seria realizada com alguma facilidade e alguns benefícios. Contudo, existiria espaço para melhorias.

Quanto aos testes de desempenho, esperavam-se resultados mais satisfatórios relativamente ao tempo de execução das consultas. Porém, também é preciso ter em conta a máquina utilizada para os mesmos.

Não obstante, uma vez que todas as consultas foram efetuadas no mesmo panorama, as conclusões retiradas abordam três importantes condições presentes nas consultas feitas, o tipo de base de dados, se devolvem valores agrupados e se os valores são ordenados ou não.

Desta forma, utilizando a função ST_PixelAsCentroids() para obter o valor dos pontos num ficheiro *raster*, pode-se concluir com os resultados obtidos nos testes, que quanto mais registos forem retornados numa consulta, maior a diferença nos tempos de execução de uma base de dados para a outra, o que torna a base de dados de séries temporais bastante mais útil em contextos mais abrangentes.

Comparando os resultados obtidos, utilizando os resultados da função ST_PixelAs Centroids() armazenados numa tabela para se obter os valores dos pontos dos *rasters*, conclui-se que guardar os dados em bases de dados de séries temporais é mais vantajoso em grande parte das situações testadas. Porém, quando os valores retornados por uma consulta são bastante elevados, a base de dados relacional apresenta um desempenho ligeiramente superior . É ainda de salientar, que nos casos de uso 1 e 4 ou 3 e 6, quando armazenados em base de dados relacional, o desempenho das consultas efetuadas foi bastante idêntico independentemente do número de valores devolvido.

Por fim, as *queries*, que recorreram a uma tabela com todas as coordenadas existentes num ficheiro *raster* para obter os valores dos pontos do mesmo, não foram diferentes dos anteriores dois conjuntos de resultados, porém os seus resultados foram um pouco irregulares, o que os tornam um pouco discutiveís. Na origem destas irregularidades acredita-se que está a fraca disponibilidade de memória e qualidade da máquina utilizada.

Em suma, o sistema desenvolvido será mais eficiente utilizando uma base de dados de séries temporais ao invés da relacional e obtendo os valores de temperatura de cada píxel através da função ST_PixelAsCentroids, caso a quantidade de dados a devolver seja na casa dos milhares. Caso contrário, a diferença não será significativa. Pode-se também afirmar, que se consegue integrar o sistema desenvolvido noutros projetos relacionados, sem grandes complicações.

6

Conclusões

Ao longo deste último capítulo, será feito um resumo acerca do trabalho efetuado e também algumas conclusões retiradas a partir do mesmo.

No final, serão apresentadas algumas ideias sobre temas a trabalhar a partir do sistema implementado.

6.1 Conclusões

Os objetivos definidos, anteriormente, para esta dissertação foram a implementação de um sistema capaz de auxiliar na prevenção de pragas, na agricultura, através da análise de dados meteorológicos, e ainda, verificar que tipo de base de dados é mais eficaz a armazenar e, consequentemente, devolver dados *raster*.

Desta forma, os objetivos foram cumpridos completamente, visto que o sistema implementado é capaz de analisar os dados armazenados e efetuar comparações na execução das *queries* a ambas as bases de dados.

O integrador de dados, constituído por diversos scripts, tem a particularidade de permitir descarregar qualquer tipo de dados de um servidor, e gerar os ficheiros *raster* e .shp dos mesmos, porém, caso os ficheiros descarregados não sejam HDF5, terá de ser criado outro *script* para obter a informação dos mesmos.

O modelo de dados desenvolvido permite que, projetos do mesmo âmbito, consigam utilizá-lo e adaptá-lo às suas necessidades, facilmente.

Quanto às bases de dados desenvolvidas, apesar de serem bastante adequadas ao tipo de dados que armazenam, a inconstância dos resultados obtidos, com a máquina utilizada, pode ter afetado o desempenho das consultas efetuadas às mesmas.

Por fim, a camada de visualização serviu para obter os resultados pretendidos, contudo existem soluções mais completas que a utilizada e que possam abranger outros objetivos que não o cálculo de modelos numéricos.

6.2 Trabalho Futuro

Mesmo com os objetivos atingidos, esta secção visa abordar diversos aspetos que não chegaram a ser implementados neste sistema.

6.2.1 Utilização de modelos de previsão meteorológica

Apesar dos dados provenientes de satélites serem bastante precisos, por vezes, como já foi referido ao longo do documento, a existência de nuvens ou nevoeiro prejudica a extensão da superfície terrestre que se consegue analisar. Desta forma, no futuro, incluir também dados de modelos de previsão meteorológica, seria uma mais valia para o sistema, ajudando a culmatar a ausência de dados em algumas zonas.

6.2.2 Criação de uma aplicação web

A criação de uma aplicação web capaz de oferecer uma melhor experiência interativa ao utilizador, seria outra das possíveis implementações futuras. Apesar da ferramenta pgAdmin oferecer todos os elementos necessários para analisar os dados armazenados, uma interface gráfica poderia não só ter as mesmas funcionalidades, como também permitiria ao utilizador utilizar esses mesmos dados para calcular modelos de previsão de pragas, como os graus dia acumulados.

6.2.3 Criação de Visualizações

Por fim, a criação de diferentes formas de visualização dos dados, para além de em formato tabela, poderá ser útil para a análise de outros parâmetros que sejam do interesse fitossanitário.

Bibliografia

[1] "A Web-based Information System for Plant Disease Forecast Based on Weather Data at High Spatial Resolution". Em: (). URL: http://koreascience.or.kr/article/JAK0201028552428036.pdf (ver p. 7).

[2] A. M. d. A. R. Angélica Prela. "Determinação de graus-dia acumulados e sua aplicação no planejamento do cultivo de feijão-vagem (Phaseolus vulgaris L.) para Londrina-PR". Em: *Revista Brasileira de Agrometeorologia* (mai. de 2002) (ver p. 2).

[3] P. B. Cerlini, L. Silvestri e M. Saraceni. "Quality control and gap-filling methods applied to hourly temperature observations over central Italy". Em: *Meteorological Applications* 27.3 (2020), e1913. DOI: https://doi.org/10.1002/met.1913. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/met.1913. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/met.1913 (ver p. 11).

[4] *Climate Data Operators*. URL: https://www.dwd.de/DE/leistungen/opendata/hilfe.html?nn=495490 (ver p. 17).

[5] *Computer Centre for Agricultural Pest Forecasting (CIPRA)*. 2017. URL: https://agriculture.canada.ca/en/agricultural-science-and-innovation/agricultural-research-results/computer-centre-agricultural-pest-forecasting-cipra (ver pp. 5, 6).

[6] *cURL*. URL: https://curl.se/ (ver p. 23).

[7] *ERA5*. URL: https://climate.copernicus.eu/climate-reanalysis (ver p. 17).

[8] Esri. "ESRI Shapefile Technical Description". Em: (1998). URL: https://www.esri.com/content/dam/esrisites/sitecore-archive/Files/Pdfs/library/whitepapers/pdfs/shapefile.pdf (ver p. 20).

[9] *EUMETSAT*. URL: https://www.eumetsat.int/about-us/satellite-application-facilities-safs (ver p. 12).

[10] *GDAL*. URL: https://gdal.org/ (ver p. 21).

[11] *GNSS RO*. URL: https://www.cosmic.ucar.edu/what-we-do/gnss-radio-occultation (ver p. 12).

[12] *HDF5*. URL: http://web.mit.edu/fwtools_v3.1.0/www/H5.intro.html (ver p. 15).

[13] *HDF5 Hierarchy*. URL: https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5 (ver p. 15).

[14] *ICON weather forecast model*. URL: https://www.dwd.de/EN/ourservices/nwp_forecast_data/nwp_forecast_data.html (ver p. 17).

[15] *InfluxDB*. URL: https://www.influxdata.com/products/influxdb/ (ver p. 8).

[16] *IPMA*. URL: http://www.ipma.pt/pt/index.html (ver pp. 16, 17).

[17] *LSA SAF*. URL: https://landsaf.ipma.pt/en/ (ver pp. 12–14).

[18] *M3DB*. URL: https://m3db.io/about/ (ver pp. 8, 9).

[19] W. Macho. "Interactive Time series analysis on raster data using QGIS, an open source GIS". Em: (set. de 2016). URL: http://unigis.sbg.ac.at/files/Mastertheses/Full/102723.pdf (ver p. 14).

[20] A. T. M. Malafaia. "Sistema de Informação para Projetos de Monitoração Fitossanitária". Em: (2019) (ver pp. 2, 8, 18, 19).

[21] J. Marques da Silva et al. "Agriculture pest and disease risk maps considering MSG satellite data and land surface temperature". Em: *International Journal of Applied Earth Observation and Geoinformation* 38 (2015), pp. 40–50. ISSN: 0303-2434. DOI: https://doi.org/10.1016/j.jag.2014.12.016. URL: https://www.sciencedirect.com/science/article/pii/S0303243414002918 (ver p. 6).

[22] *NASA Panoply*. URL: https://www.earthdata.nasa.gov/technology/panoply (ver p. 15).

[23] *NOAA*. URL: https://www.noaa.gov/ (ver p. 17).

[24] *Oracle DB*. URL: https://www.oracle.com/pt/database/technologies/ (ver p. 10).

[25] *Oracle Spatial and Graph*. URL: https://www.oracle.com/database/technologies/spatialandgraph.html (ver p. 11).

[26] *Pest Models*. URL: http://ipm.ucanr.edu/WEATHER/index.html#PESTPLANTMODELS (ver p. 5).

[27] *PostGis*. URL: https://postgis.net/ (ver p. 10).

[28] *PostgreSQL*. URL: https://www.postgresql.org/about/ (ver p. 10).

[29] *Product User Manual*. URL: https://nextcloud.lsasvcs.ipma.pt/s/SwRtreJqPPbrTc9 (ver p. 42).

[30] *Prometheus Software*. 2021. URL: https://prometheus.io/docs/introduction/overview/ (ver pp. 8, 9).

[31] *QGIS*. URL: https://pt.wikipedia.org/wiki/QGIS (ver p. 20).

[32] *Raster vs Vetorial*. URL: https://docs.ufpr.br/~firk/pessoal/Carto_Digital/BED3.pdf (ver p. 19).

[33] T. W. Rauber. "Redes Neurais Artificiais". Em: (). URL: https://www.researchgate.net/profile/Thomas-Rauber-2/publication/228686464_Redes_neurais_artificiais/links/02e7e521381602f2bd000000/Redes-neurais-artificiais.pdf (ver p. 6).

[34] S. Skawsang et al. "Predicting Rice Pest Population Occurrence with Satellite-Derived Crop Phenology, Ground Meteorological Observation, and Machine Learning: A Case Study for the Central Plain of Thailand". Em: *Applied Sciences* 9 (nov. de 2019), p. 4846. DOI: 10.3390/app9224846 (ver p. 6).

[35] *SpatiaLite*. URL: https://www.gaia-gis.it/fossil/libspatialite/index (ver p. 10).

[36] *SQLite*. URL: https://www.sqlite.org/index.html (ver p. 10).

[37] *Time series DB ranking*. URL: https://db-engines.com/en/ranking/time+series+dbms (ver p. 9).

[38] *TimescaleDB*. URL: https://docs.timescale.com/latest/main (ver pp. 8, 9).

[39] " TRAPSYSTEM - UMA APLICAÇÃO PARA GERENCIAMENTO DE DADOS COLETADOS A PARTIR DE ARMADILHAS DE INSETOS". Em: (). URL: https://ainfo.cnptia.embrapa.br/digital/bitstream/item/158724/1/ID43959-2016RCBPTT10DOUGLAS36.pdf (ver p. 7).

[40] J. Yang, Y. Wang e P. August. "Estimation of Land Surface Temperature Using Spatial Interpolation and Satellite-Derived Surface Emissivity". Em: *Journal of Environmental Informatics* 4 (set. de 2004), pp. 37–44. DOI: 10.3808/jei.200400035 (ver p. 18).

I

Anexos

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	7,904	10,339	9,84	8,048	8,06	8,838
Query 2	38,679	25,617	28,784	25,045	24,85	28,595
Query 3	79,385	94,345	90,859	98,528	105,877	93,799
Query 4	140,302	141,801	149,861	148,572	140,153	144,138
Query 5	15,802	17,045	17,014	17,847	19,82	17,506
Query 6	52,852	57,61	43,999	56,457	58,876	53,959
Query 7	108,206	107,866	101,201	99,572	99,468	103,263
Query 8	137,016	156,696	149,01	142,394	134,775	143,978

Tabela I.1: Resultados obtidos - caso de uso 1 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	122,483	119,871	112,464	118,04	127,477	120,067
Query 2	127,337	125,268	130,916	126,692	131,451	128,333
Query 3	128,66	128,955	129,115	111,252	128,587	125,314
Query 4	131,161	128,845	119,212	125,346	134,514	127,816
Query 5	128,894	151,189	128,163	110,685	125,481	128,882
Query 6	112,387	128,12	128,57	112,881	132,835	122,959
Query 7	133,569	122,088	124,376	122,133	133,529	127,139
Query 8	137,88	125,843	133,69	124,301	114,187	127,180

Tabela I.2: Resultados obtidos - caso de uso 1 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 9	8,187	12,95	14,805	15,858	14,286	13,217
Query 10	43,661	43,234	39,863	33,169	46,18	41,221
Query 11	102,055	90,34	87,77	88,348	91,213	91,945
Query 12	136,563	139,01	141,022	135,021	140,397	138,403
Query 13	7,978	11,344	13,818	12,192	9,682	11,003
Query 14	38,411	34,627	33,885	33,333	34,446	34,940
Query 15	88,513	85,906	101,997	98,386	102,439	95,448
Query 16	145,046	144,671	126,471	129,171	141,187	137,309

Tabela I.3: Resultados obtidos - caso de uso 4 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 9	122,606	129,798	125,6	127,387	130,897	127,258
Query 10	117,449	125,133	120,305	127,315	133,204	124,681
Query 11	137,458	127,258	116,173	109,695	128,828	123,882
Query 12	132,049	127,481	123,706	127,391	123,672	126,860
Query 13	136,318	138,11	123,86	132,031	126,912	131,446
Query 14	130,222	124,457	126,289	126,966	126,432	126,873
Query 15	110,472	128,102	131,56	125,334	112,245	121,543
Query 16	141,671	132,049	130,92	127,845	120,17	130,531

Tabela I.4: Resultados obtidos - caso de uso 4 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	3,65	3,289	3,298	3,312	3,3	3,370
Query 2	23,881	23,505	23,709	23,86	23,528	23,697
Query 3	48,602	47, 063	47,777	47,697	47,371	47,862
Query 4	77,782	78,946	78,92	79,056	79,027	78,746
Query 5	3,284	3,244	3,281	3,354	3,23	3,279
Query 6	24,726	24,155	23,71	23,609	24,074	24,055
Query 7	48,957	48,447	48,023	48,327	48,554	48,462
Query 8	81,074	80,534	80,314	80,111	79,318	80,270

Tabela I.5: Resultados obtidos - caso de uso 1 utilizando a tabela com as coordenadas, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 1	1,786	1,458	1,465	1,498	1,562	1,554
Query 2	17,326	23,292	24,234	26,569	25,362	23,357
Query 3	56,834	51,063	49,341	49,123	50,399	51,352
Query 4	84,176	83,574	82,681	83,621	84,603	83,731
Query 5	3,6	3,473	3,47	3,349	3,32	3,442
Query 6	25,141	25,095	24,869	24,474	24,942	24,904
Query 7	50,751	47,479	47,758	48,12	48,253	48,472
Query 8	79,298	78,307	78,554	79,388	80,222	79,154

Tabela I.6: Resultados obtidos - caso de uso 1 utilizando a tabela com as coordenadas, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 9	3,634	3,483	3,479	3,517	3,513	3,525
Query 10	24,747	24,258	24,11	24,07	24,932	24,423
Query 11	52,408	51,391	51,972	51,607	52,425	51,961
Query 12	79,954	79,83	80,377	80,151	79,614	79,985
Query 13	3,487	3,437	3,42	3,468	3,497	3,462
Query 14	25,013	24,846	24,496	24,327	24,409	24,618
Query 15	52,198	52,519	53,031	53,082	53,106	52,787
Query 16	81,164	81,877	81,824	81,912	82,057	81,767

Tabela I.7: Resultados obtidos - caso de uso 4 utilizando a tabela com as coordenadas, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 9	3,262	3,309	3,251	3,21	3,187	3,244
Query 10	24,396	23,99	23,821	24,092	24,097	24,079
Query 11	51,765	52,057	51,799	52,133	52,504	52,052
Query 12	81,129	81,784	83,018	85,428	86,149	83,502
Query 13	3,361	3,431	3,486	3,57	3,416	3,453
Query 14	24,918	25,198	24,782	24,889	25,359	25,029
Query 15	54,345	56,063	54,862	55,647	57,203	55,624
Query 16	85,568	82,971	84,477	84,467	87,307	84,958

Tabela I.8: Resultados obtidos - caso de uso 4 utilizando a tabela com as coordenadas, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 17	0,14	0,215	0,161	0,178	0,129	0,165
Query 18	0,147	0,138	0,129	0,16	0,134	0,142

Tabela I.9: Resultados obtidos caso de uso 2 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 17	65,654	65,667	66,34	66,544	66,153	66,072
Query 18	65,753	65,843	66,177	65,68	65,413	65,773

Tabela I.10: Resultados obtidos caso de uso 2 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 19	0,171	0,141	0,157	0,101	0,103	0,135
Query 20	0,16	0,137	0,13	0,131	0,107	0,133

Tabela I.11: Resultados obtidos caso de uso 5 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 19	71,912	65,514	66,309	65,992	66,343	67,214
Query 20	65,65	66,412	66,381	66,02	65,83	66,059

Tabela I.12: Resultados obtidos caso de uso 5 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 17	0,71	0,713	0,778	1,46	1,492	1,031
Query 18	1,67	1,398	1,417	0,808	0,783	1,215

Tabela I.13: Resultados obtidos caso de uso 2 utilizando a tabela com as coordenadas, na base de dados de séries temporais

	Tentativa 1	Tentativa 2	Tentativa 3	Tentativa 4	Tentativa 5	Average (s)
Query 17	1,49	1,553	1,613	1,491	2,009	1,631
Query 18	1,387	1,391	1,376	1,484	1,571	1,442

Tabela I.14: Resultados obtidos caso de uso 2 utilizando a tabela com as coordenadas, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 19	1,478	1,651	1,536	1,577	0,811	1,411
Query 20	1,897	1,852	1,818	2,21	2,034	1,962

Tabela I.15: Resultados obtidos caso de uso 5 utilizando a tabela com as coordenadas, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 19	2,009	1,751	1,728	1,789	1,9	1,835
Query 20	1,791	1,749	1,862	1,687	1,68	1,754

Tabela I.16: Resultados obtidos caso de uso 5 utilizando a tabela com as coordenadas, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 21	35,297	26,816	24,544	33,758	29,82	30,047
Query 22	150,693	152,913	158,972	164,358	159,122	157,212
Query 23	326,355	321,359	287,273	287,569	292,298	302,971
Query 24	470,878	444,199	459,682	391,904	385,161	430,365
Query 25	26,414	36,127	33,664	21,069	25,63	28,581
Query 26	156,868	163,999	162,907	147,218	153,03	156,804
Query 27	331,031	311,951	284,084	283,127	327,49	307,537
Query 28	435,835	478,22	476,501	429,213	428,228	449,599

Tabela I.17: Resultados obtidos - caso de uso 3 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 21	404,486	411,272	404,271	420,259	407,669	409,591
Query 22	409,873	410,181	397,804	413,834	410,95	408,528
Query 23	412,233	414,76	410,037	416,796	423,261	415,417
Query 24	420,089	410,037	413,625	421,047	419,649	416,889
Query 25	376,263	411,499	409,99	402,466	411,407	402,325
Query 26	409,426	408,671	412,841	408,448	412,034	410,284
Query 27	409,809	410,583	400,576	412,292	417,478	410,148
Query 28	422,702	419,69	430,049	408,064	416,898	419,481

Tabela I.18: Resultados obtidos - caso de uso 3 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 29	52,109	47,576	45,29	45,311	48,152	47,688
Query 30	173,918	166,29	180,832	175,158	182,939	175,827
Query 31	346,972	320,46	303,324	330,328	334,463	327,109
Query 32	456,873	481,688	492,854	462,253	442,033	467,140
Query 33	43,793	47,229	47,801	44,85	47,498	46,234
Query 34	168,369	174,026	169,7	166,898	148,6	165,519
Query 35	293,927	293,582	299,005	348,403	342,5	315,483
Query 36	498,438	423,294	433,25	488,814	495,195	467,798

Tabela I.19: Resultados obtidos - caso de uso 6 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 29	423,524	415,449	417,98	416,11	404,105	415,434
Query 30	420,67	419,64	428,34	381,908	412,484	412,608
Query 31	412,173	402,668	374,921	393,893	404	397,531
Query 32	414,551	414,644	409,351	405,821	416,448	412,163
Query 33	377,632	399,857	414,884	378,96	374,642	389,195
Query 34	427,7	396,199	410,061	419,471	418,707	414,428
Query 35	422,93	425,946	408,769	403,378	422,914	416,787
Query 36	416,899	410,45	407,963	407,72	412,815	411,169

Tabela I.20: Resultados obtidos - caso de uso 6 utilizando o resultado da função ST_*PixelAsCentroids* numa tabela, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 21	57,596	59,865	61,096	60,958	57,265	59,356
Query 22	348,105	293,353	294,464	318,245	323,128	315,459
Query 23	716,756	743,461	704,627	712,44	649,863	705,429
Query 24	1357,441	1213,276	1323,132	-	-	1297,950
Query 25	57,872	56,489	56,947	56,374	56,418	56,820
Query 26	364,457	363,587	384,374	397,844	362,851	374,623
Query 27	712,068	719,721	678,315	-	-	703,368
Query 28	1247,386	1317,148	1301,981	-	-	1288,838

Tabela I.21: Resultados obtidos - caso de uso 3 utilizando a tabela com as coordenadas, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 21	39,469	43,722	39,359	37,419	36,768	39,347
Query 22	504,554	503,649	367,865	348,863	499,85	444,956
Query 23	824,174	806,988	761,464	-	-	797,542
Query 24	1230,87	1215,772	1256,895	-	-	1234,512
Query 25	57,53	57,668	58,114	56,685	58,032	57,606
Query 26	502,987	503,034	496,317	502,312	502,467	501,423
Query 27	638,957	716,509	616,717	-	-	657,394
Query 28	1412,786	1379, 12	1411,612	-	-	1412,199

Tabela I.22: Resultados obtidos - caso de uso 3 utilizando a tabela com as coordenadas, na base de dados relacional

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 29	56,876	52	56,975	56,959	57,154	55,993
Query 30	357,757	338,574	382,721	362,83	361,786	360,734
Query 31	778,696	832,099	829,532	-	-	813,442
Query 32	1096,584	1270,915	1264,665	-	-	1210,721
Query 33	58,061	57,852	57,588	57,784	57,517	57,760
Query 34	375,254	343,235	381,74	356,313	355,199	362,348
Query 35	742,997	797,389	810,358	-	-	783,581
Query 36	1233,02	1224,206	1252,804	-	-	1236,677

Tabela I.23: Resultados obtidos - caso de uso 6 utilizando a tabela com as coordenadas, na base de dados de séries temporais

	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média (s)
Query 29	58,391	58,39	57,41	58,472	58,372	58,207
Query 30	506,189	511,266	511,949	506,658	512,208	509,654
Query 31	988, 153	996,209	967,894	-	-	982,052
Query 32	1257,499	1125,111	1435,099	-	-	1272,570
Query 33	41,954	45,413	27,742	27,564	27,575	34,050
Query 34	343,985	349,358	376,739	358,719	368,305	359,421
Query 35	838,312	905,444	853,808	-	-	865,855
Query 36	1363,474	1277, 99	1237,705	-	-	1300,590

Tabela I.24: Resultados obtidos - caso de uso 6 utilizando a tabela com as coordenadas, na base de dados relacional