



MIGUEL RAMOS BAIÃO

Licenciado

OPERACIONALIZAÇÃO DE PLATAFORMA COLABORATIVA PARA AGRICULTURA

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Abril, 2025



OPERACIONALIZAÇÃO DE PLATAFORMA COLABORATIVA PARA AGRICULTURA

MIGUEL RAMOS BAIÃO

Licenciado

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

Coorientador: João Moura Pires
Professor Associado, Universidade Nova de Lisboa

Operacionalização de Plataforma Colaborativa para Agricultura

Copyright © Miguel Ramos Baião, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade NOVA de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

AGRADECIMENTOS

Gostaria de começar por agradecer ao meu orientador Carlos Viegas Damásio e ao meu coorientador João Moura Pires por todo o apoio e confiança que depositaram em mim, ao longo deste percurso. A sua orientação foi crucial para o meu crescimento pessoal como académico.

Em seguida, queria agradecer a toda a equipa do projeto FitoAgro, cuja colaboração, disponibilidade e *feedback* foram fundamentais, no desenvolvimento desta dissertação.

Um grande agradecimento aos amigos que fiz durante a faculdade, por todas as noites de estudo, por todos os conselhos que me deram nos momentos de dúvida, e pela vossa amizade, que me sustentou durante esta jornada.

Gostaria de agradecer aos meus amigos fora da faculdade, pela enorme paciência e compreensão com a minha frequente indisponibilidade e por ouvirem os meus desabafos, nos momentos mais exigentes, mesmo quando os temas escapavam ao seu interesse e compreensão. A vossa amizade foi um pilar fundamental ao longo deste período.

Por fim, um agradecimento muito especial aos meus pais, que foram o alicerce que me sustentou tanto em termos práticos, como emocionais, com o seu apoio incondicional, permitindo-me enfrentar os desafios encontrados, durante esta etapa da minha vida. Não há palavras que cheguem para descrever a minha gratidão por tudo o que fizeram e continuam a fazer por mim.

A todos, o meu sincero obrigado!

RESUMO

Atualmente, a produção hortofrutícola na região do Oeste, depara-se com problemas de índole fitossanitária, sendo as causas desses problemas: a limitação no uso de inúmeras substâncias ativas, que deixam muitas finalidades deficientemente cobertas, promovendo o aumento de populações que estavam em equilíbrio e que por isso passam a praga; às alterações climáticas que têm criado condições favoráveis ao aumento das populações das pragas e causado alterações nos ciclos de vida dos inimigos e a ocorrência de novas espécies.

Para a prevenção e controlo destes problemas, seria necessário criar um sistema de apoio para técnicos e agricultores no âmbito da fitossanidade, que os ajude a cumprir as obrigações legais e as metas impostas pelas estratégias europeias. Para satisfazer esta necessidade foi desenvolvida uma aplicação web de *back-office*, para a plataforma concebida no âmbito do projeto FitoAgro. Este projeto tem como objetivo principal, estudar as pragas emergentes na região do Oeste que afetam as pomóideas. Sendo que esta é uma aplicação, que permite a integração de informação sobre os problemas fitossanitários, a uma escala regional. Estes problemas são identificados por técnicos de campo durante as suas visitas ao terreno infestado, sendo registados através de uma aplicação móvel, cujo propósito é registar no sistema a informação fitossanitária recolhida durante as visitas. Esta aplicação móvel pode ser configurada por meio da aplicação web desenvolvida, de modo a adaptar-se às alterações nos ciclos de vida das pragas, ou ao aparecimento de novas espécies. Além da configuração da componente de recolha de informação fitossanitária, também é possível configurar a componente de recolha de informação meteorológica. Adicionalmente, é possível consultar e manipular a informação armazenada no sistema FitoAgro através de uma única interface. Por fim, é também possível gerir o acesso ao sistema por parte de todas as entidades interessadas no projeto, sejam elas organizações, produtores ou agentes de campo.

A fim de validar e avaliar a aplicação desenvolvida, foram conduzidos vários testes, sendo que alguns envolveram os futuros utilizadores da aplicação, que asseguraram que a recolha de informação e a sua interação numa plataforma inteligente permite ajudar a definir um conjunto de indicadores e serviços fundamentais para monitorizar a progressão dos problemas fitossanitários.

Palavras-chave: Fitossanidade, FitoAgro, aplicação web, *back-office*, regional

ABSTRACT

Currently, horticultural production in the Western region is facing phytosanitary problems, with the causes of these issues being: the limitation in the use of numerous active substances that leaves many purposes inadequately covered, promoting the increase of populations that were in balance and thus becoming pests; climate changes that have created favorable conditions for the increase of pest populations and caused alterations in the life cycles of these enemies and the occurrence of new species.

In order to prevent and control these problems, it would be necessary to create a support system for technicians and farmers in the field of phytosanitation, to help them comply with legal obligations and the targets imposed by European strategies. To meet this need, a back-office web application was developed for the platform designed as part of the FitoAgro project. This project's main goal is to study emerging pests in the West region affecting pome fruits. This is an application that allows information on phytosanitary problems to be integrated on a regional scale. These issues are identified by field technicians during their visits to the infected land, and are recorded in the system using a mobile application, whose purpose is to record the phytosanitary information collected during the visits. This mobile application can be configured via the web application developed to adapt to changes in pest life cycles or the appearance of new species of pests. In addition to configuring the phytosanitary information collection component, it is also possible to configure the weather information collection component. In addition, it is possible to consult and manipulate the information stored in the FitoAgro system via a single interface. Finally, it should be possible to manage access to the system by all the stakeholders interested in the project, be they organizations, producers or field agents.

In order to validate and evaluate the application developed, tests were carried out, some of which involved future users of the application, which ensured that the collection of information and its interaction on an intelligent platform helps to define a set of indicators and services that are fundamental to monitoring the progression of plant health problems.

Keywords: Phytosanitary, FitoAgro, web application, back-office, regional

ÍNDICE

Índice de Figuras	viii
Índice de Tabelas	x
1 Introdução	1
1.1 Motivação e Contexto	1
1.2 Projeto FitoAgro	2
1.3 Problema	2
1.4 Objetivos e Contribuições	3
1.5 Estrutura do documento	4
2 FitoAgro	5
2.1 Pragas emergentes	5
2.2 Sistema de Informação	8
2.3 Recolha de Informação	10
2.3.1 Informação fitossanitária	10
2.3.2 Informação Meteorológica	13
2.4 Visualização de Dados	14
2.5 Modelos Preditivos	14
2.6 <i>Stakeholders</i>	14
2.7 Limitações e Conclusões	15
3 Trabalho Relacionado	17
3.1 Projetos Relacionados	17
3.1.1 SafeBrócolo	17
3.1.2 Cropwise Operations	18
3.1.3 WiseCrop	19
3.1.4 Agroptima	20
3.1.5 SmartFarm	21
3.1.6 AGRIVI	22
3.1.7 Farmbrite	23
3.1.8 Conclusão	24
3.2 Tecnologias Relevantes	25
3.2.1 Tecnologias de Base de Dados	25
3.2.2 Bibliotecas para mapas	26

3.2.3	Bibliotecas de Visualização	27
3.2.4	Tecnologias para Desenvolvimento de Interfaces	28
4	Solução para gestão do sistema FitoAgro	32
4.1	Requisitos	32
4.2	Funcionalidades da Aplicação Web FitOps	33
4.2.1	Utilizadores	33
4.2.2	Estações meteorológicas	39
4.2.3	Parcelas	44
4.2.4	Períodos	53
4.2.5	Organizações	56
4.2.6	Visitas e Observações	57
4.2.7	Protocolos de recolha de informação fitossanitária	61
4.2.8	Controlo de Entidades do Sistema	63
4.3	Conclusão	64
5	Implementação	65
5.1	Tecnologias Usadas	65
5.1.1	<i>Frontend</i>	65
5.1.2	<i>Backend</i>	67
5.2	Arquitetura	69
5.2.1	Cliente	70
5.2.2	Servidor	75
5.2.3	Base de Dados	78
5.2.4	Scripts de Recolha de informação Meteorológica	80
5.3	Conclusão	80
6	Avaliação	81
6.1	Avaliação da interface	81
6.1.1	SUS	81
6.1.2	Usabilidade das funcionalidades	83
6.2	Avaliação do Servidor	88
6.2.1	Testes de Carga	89
6.2.2	Testes de Segurança	90
6.3	Conclusão	90
7	Conclusões e Trabalho Futuro	92
7.1	Conclusões	92
7.2	Trabalho Futuro	93
7.2.1	Integração de dados de satélite	93
7.2.2	Gestão de tratamentos e respetivos resíduos	93
7.2.3	Alertas	93
7.2.4	Sistema de mensagens	94
	Bibliografia	95
	Apêndices	
A	Listagem dos Requisitos	98

ÍNDICE DE FIGURAS

1.1	Modelo inicial do sistema de informação.	2
2.1	Colónia de filoxera na Primavera (fêmeas, ovos e larvas)	6
2.2	Ilustração de cecidómia e ataque a folhas	6
2.3	Cochonilha-Algodão	7
2.4	Larva e traça adulta da <i>Grapholita molesta</i> Busck	7
2.5	Modelo atual do sistema de informação	9
2.6	Vistas importantes da aplicação móvel ProtoCollector	11
2.7	Vistas importantes da aplicação web ProtoCollector	12
3.1	Algumas vistas <i>web</i> da plataforma SafeBrócolo	18
3.2	Exemplo da vista geral do sistema Cropwise Operations	19
3.3	Wisecrop	20
3.4	Agroptima	21
3.5	AGRIVI	23
3.6	Farmbrite	24
4.1	Registo de Utilizadores	34
4.2	Email de ativação de conta	34
4.3	Autenticação de Utilizadores	35
4.4	Recuperação de conta	35
4.5	Gestão da conta	36
4.6	Página de utilizadores	37
4.7	Detalhes de um utilizador	38
4.8	Editar utilizador	38
4.9	Gestão de membros da organização	39
4.10	Página de contas de fornecedores	40
4.11	Registo de conta de fornecedor	40
4.12	Estações de uma conta de fornecedor	41
4.13	Registo de estações meteorológicas	42
4.14	Página estações meteorológicas	42
4.15	Visualização de dados recolhidos	43
4.16	Editar estação meteorológica	43
4.17	Registo de parcelas	45
4.18	Página de parcelas	46

4.19	Detalhes de uma parcela	46
4.20	Editar parcela	46
4.21	Alterar e editar estações meteorológicas da parcela	47
4.22	Página de elementos de interesse	48
4.23	Editar elemento de interesse	48
4.24	Criar elemento de interesse	49
4.25	Página de períodos associados	49
4.26	Associar período a parcela	50
4.27	Inimigos estudados durante um período	50
4.28	Página de utilizadores associados	50
4.29	Associar utilizador a parcela	51
4.30	Gestão de utilizadores associados	51
4.31	Página de visitas	52
4.32	Registo de períodos	53
4.33	Duplicar período	54
4.34	Página de períodos	54
4.35	Detalhes de um período	55
4.36	Editar período	55
4.37	Registo de organizações	56
4.38	Página de organizações	56
4.39	Editar organização	57
4.40	Página de visitas	58
4.41	Detalhes de uma visita	59
4.42	Observações de uma visita	59
4.43	Fotos tiradas durante a visita	60
4.44	Trajetos percorridos durante a visita	60
4.45	Correção de valores	61
4.46	Página de protocolos	61
4.47	Registo de protocolos	62
4.48	Página de entidades do sistema	63
5.1	Arquitetura da aplicação FitOps.	69
5.2	Schema da tabela agentsregions	78
6.1	Gráfico SUS	82
6.2	Resultados SUS	83
6.3	Resultados de facilidade	87
6.4	Resultados de utilidade	88

ÍNDICE DE TABELAS

2.1	Sensores relevantes e a sua respetiva unidade.	13
3.1	Comparação entre Bootstrap, Vue.js, React, Angular e Flutter.	31
6.1	Métricas retiradas dos testes de carga efetuados com uma instância do servidor	89
6.2	Métricas retiradas dos testes de carga efetuados com duas instâncias do servidor	89

ÍNDICE DE LISTAGENS

4.1	Exemplo de ficheiro GeoJSON	44
5.1	accountStore.	70
5.2	generalStore.	71
5.3	Exemplo de rotas.	73
5.4	<i>Vue Router Middleware</i>	74
5.5	Exemplo de uso da função <i>hasRole</i>	75
5.6	Exemplo de uso do <i>middleware</i> numa rota.	77
5.7	Controlo de permissões no controlador.	77

INTRODUÇÃO

Neste capítulo é apresentada a motivação e o contexto (1.1) para a realização da dissertação, bem como o projeto em que esta se enquadra (1.2). O problema que pretende resolver (1.3) e os objetivos e contribuições atingidas (1.4). Por fim, é descrita a estrutura do documento onde são enumerados e descritos sumariamente os capítulos deste documento (1.5).

1.1 Motivação e Contexto

Ao longo da evolução da agricultura e da história da humanidade, as pragas têm causado danos económicos e sociais significativos, como por exemplo períodos de escassez alimentar e desperdício de recursos naturais, pois estas acabam por ser usados numa produção que irá ficar no campo, pois não pode ser colhida devido aos danos causados. Embora as consequências tenham diminuído em comparação com o passado, elas continuam a representar desafios substanciais para os agricultores, resultando em sérias perdas financeiras. Os produtores agrícolas adotam estratégias de controle e prevenção para lidar com essas pragas, recorrendo à aplicação de produtos fitofarmacêuticos quando as estratégias iniciais falham.

No entanto, a produção hortofrutícola contemporânea enfrenta novos desafios fitossanitários, decorrentes da saída de substâncias ativas que deixam finalidades deficientemente cobertas ou descobertas, contribuindo para o aumento de populações que estavam em equilíbrio e passam a se tornar pragas. As alterações climáticas também têm favorecido o aumento das populações de pragas e a ocorrência de doenças, juntamente com mudanças nos ciclos de vida e a emergência de novos inimigos. Além disso, a necessidade de informação e registo regional de pragas e doenças é crucial para a elaboração de planos de risco fitossanitário e a abertura de novos mercados de exportação.

Nesse contexto, surge a necessidade de desenvolver uma aplicação web que integra informações sobre problemas fitossanitários em escala regional, combinando dados de carácter fitossanitário como dados meteorológicos, de modo a serem consultados através de uma única interface, facilitando a sua análise. Esta abordagem permitiria monitorizar em tempo real as condições que influenciam os comportamentos das pragas e doenças, promovendo a implementação de medidas preventivas.

A coleta e interação de informações por meio de uma plataforma inteligente inovadora contribuiria para definir indicadores e serviços essenciais na monitorização da progressão dos problemas fitossanitários, com foco especial em pragas e doenças que impactam significativamente as culturas e os resíduos resultantes. Essa iniciativa visa não apenas atender às obrigações legais, mas também alinhar-se com as metas das estratégias europeias.

e infestantes que podem afetar as plantas em uma determinada área, como pomares, campos agrícolas ou estufas. Esses planos visam proteger a saúde das plantas, garantir a produção sustentável e minimizar os impactos ambientais e económicos.

Existia também a necessidade de um portal que permitisse a gestão do projeto FitoAgro. Este portal deveria possibilitar a administração do acesso ao sistema desenvolvido no âmbito do projeto, garantindo que todas as entidades interessadas pudessem interagir com o sistema de forma segura e controlada. Além disso, deveria permitir a consulta e manipulação dos dados recolhidos e armazenados pelo sistema, assegurando a sua integridade e acessibilidade para efeitos de análise. Finalmente, a ferramenta deveria permitir a configuração das componentes de recolha de informação, permitindo que estas se adaptem às alterações nos comportamentos das pragas, de modo a manter a eficácia da monitorização.

Consequentemente surgiu a necessidade de criar um sistema de informação e apoio para técnicos e agricultores no âmbito da fitossanidade, que o ajude a cumprir as obrigações legais e as metas impostas pelas estratégias europeias, nomeadamente a estratégia europeia farm-to-fork (do prado ao prato) [11]. Esta estratégia consiste em tornar a cadeia alimentar mais sustentável, saudável e transparente. Ela aborda questões como práticas agrícolas sustentáveis, bem-estar animal, transparência na cadeia de abastecimento, alimentos nutritivos, redução do desperdício, segurança alimentar e consumo sustentável. A estratégia visa promover a inovação, pesquisa e uma abordagem holística para melhorar todo o sistema alimentar da produção ao consumo.

1.4 Objetivos e Contribuições

Esta dissertação descreve e desenha a construção de uma plataforma aberta e colaborativa de monitorização regional para o oeste, especialmente para a fileira das pomóideas suportado em aplicações móveis com dados alojados e partilhados na nuvem (*Cloud*).

A recolha de informação fitossanitária é realizada por dispositivos móveis através de uma aplicação. Esta aplicação inicialmente continha um conjunto de protocolos predefinidos para a recolha de dados, mas através da plataforma web desenvolvida é possível criar novos protocolos, de modo a acomodar novas descobertas relacionadas às pragas e doenças. Após a recolha da informação também é possível fazer correções nos dados recolhidos garantindo quer a qualidade quer a uniformidade da informação carregada para a plataforma. Adicionalmente é possível gerir as parcelas em que cada técnico pode efetuar a recolha de informação. E em casos mais extremos é possível remover e conceder acesso à aplicação móvel

Inicialmente a recolha de informação meteorológica era efetuada através de um conjunto fixo de estações meteorológicas, sem possibilidade de alterar o tipo informação recolhida por estas. Com o desenvolvimento da aplicação proposta, tornou-se possível adicionar e remover estações, bem como alteração o tipo de informação que estas recolhem. Esta flexibilidade é fundamental para a adaptação do sistema às alterações no comportamento das pragas, uma vez que a dinâmica populacional e a distribuição geográfica destes organismos são diretamente influenciadas por fatores ambientais, como temperatura, humidade e precipitação. A possibilidade de adicionar, remover e editar estações meteorológicas permite adaptar a rede de monitorização em função da evolução das condições climáticas, assegurando que os dados recolhidos sejam sempre representativos da realidade no terreno.

Finalmente, toda a informação presente no sistema pode ser consultada e manipulada pelos intervenientes por intermédio de uma interface simples e interativa, ajudando à tomada de decisão informada permitindo a atuação coordenada a nível regional. Esta informação poderá ser fundamental no apoio à decisão, uma vez que possibilita a identificação precoce de potenciais surtos de pragas e doenças, a avaliação do impacto das condições meteorológicas na propagação dos mesmos e na definição de estratégias de intervenção mais eficazes. Adicionalmente, através da consulta destes dados, o sistema contribui para uma gestão mais eficiente e sustentável, otimizando o uso de recursos e reduzindo a necessidade de tratamentos fitossanitários preventivos excessivos.

1.5 Estrutura do documento

Este documento encontra-se dividido em sete capítulos organizados da seguinte forma:

- **Introdução**

Neste Capítulo 1 é apresentado o contexto e a motivação subjacentes ao tema da dissertação, o problema que se pretende resolver e os objetivos e contributos que se esperam vir a alcançar, com este trabalho.

- **FitoAgro**

Neste Capítulo 2 é descrito o projeto FitoAgro, as principais pragas que este estuda. São apresentadas as diversas componentes da plataforma, nomeadamente o sistema de informação, onde são armazenados os dados da plataforma, os tipos dos dados recolhidos, a componente de visualização dos dados armazenados no sistema de informação, os modelos preditivos e os *stakeholders* do projeto.

- **Trabalho Relacionado**

Neste Capítulo 3 é realizado um levantamento de aplicações existentes, que abordam uma temática igual ou semelhante ao da aplicação desta dissertação, e tecnologias que foram úteis, no desenvolvimento da aplicação para esta dissertação.

- **Solução para gestão do sistema FitoAgro**

Neste Capítulo 4 é apresentada a aplicação web desenvolvida para a gestão do sistema FitoAgro, onde são descritos os requisitos principais que a aplicação procura cumprir e as funcionalidades da aplicação desenvolvidas.

- **Implementação**

Neste capítulo 5 são abordados os aspetos mais técnicos da aplicação, como as tecnologias utilizadas, a arquitetura da aplicação e alterações efetuados ao sistema de informação e a outras componentes, para acomodar a aplicação web.

- **Avaliação**

Neste capítulo 6 são descritos os métodos de avaliação utilizados, para avaliar a aplicação desenvolvida e são apresentados e discutidos os resultados obtidos através destes métodos.

- **Conclusão e Trabalho Futuro**

Neste último capítulo 7 são apresentadas as conclusões retiradas do trabalho desenvolvido, durante esta dissertação e são indicadas sugestões para trabalho futuro.

FITOAGRO

Neste capítulo é especificado o projeto FitoAgro; as principais pragas que este estuda 2.1; as diversas componentes da plataforma; nomeadamente o sistema de informação 2.2; os dados recolhidos 2.3; a componente de visualização dos dados armazenados, no sistema de informação 2.4; os modelos preditivos 2.5 e os *stakeholders* do projeto 2.6.

Este projeto foi financiado através do "PDR2020 - Programa de Desenvolvimento Rural"[17] no âmbito do programa Portugal 2020, sendo constituído por membros de diversas instituições superiores e entidades agrícolas e hortofrutícolas. Existindo vontade por parte destes parceiros, de continuar com o projeto pelo sucesso alcançado. Os parceiros hortofrutícolas disponibilizaram, parcelas de pereiras e macieiras, para servirem de pontos de observação biológica (POB), nos quais foram instalados vários dispositivos de monitorização como armadilhas cromotrópicas, armadilhas sexuais, cintas para árvores, entre outros. Estes permitem o estudo e validação dos ciclos de vida, de maneira a definir-se a tão necessária estratégia, de controlo regional com principal foco na luta biotécnica e biológica.

2.1 Pragas emergentes

O projecto FitoAgro focou-se no estudo de espécies de insetos que têm vindo a afetar a produção de pomóideas, no Oeste. A presença destas espécies prejudica a produção dos frutos levando a uma diminuição da sua qualidade e consequentemente à diminuição do seu valor. As quatro espécies em estudo foram:

Filoxera (*Aphanostigma pyri* Chol.)

A filoxera da pereira é um pequeno homóptero da família *Phylloxeridae*, hiberna sob a forma de ovos durante o inverno e eclode em Março/Abril. As larvas resultantes transformam-se em fêmeas fundadoras, que realizam posturas agrupadas nos frutos. As larvas móveis causam necroses, enquanto a infestação favorece a ação de fungos. A dispersão ocorre através da ação humana, levando à desvalorização comercial dos frutos [13].

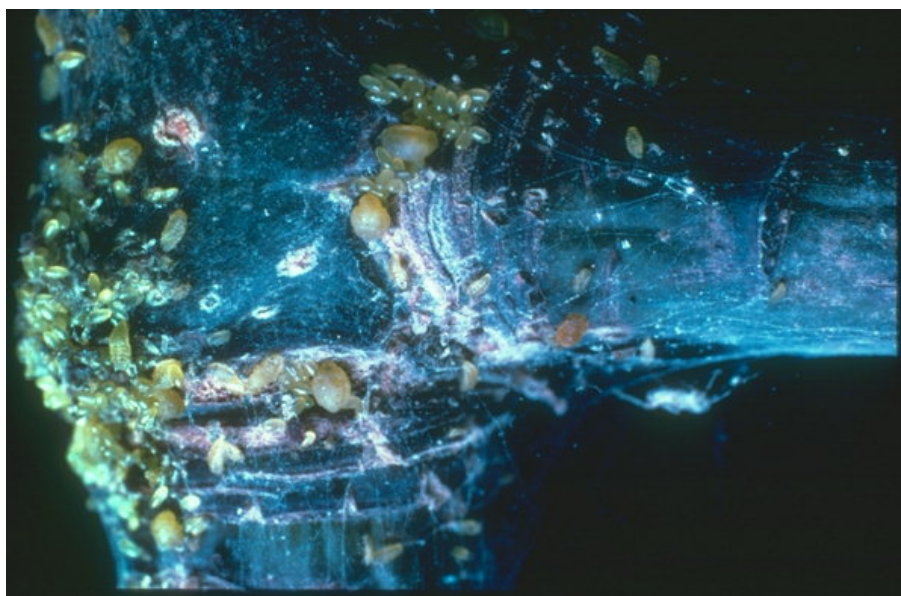
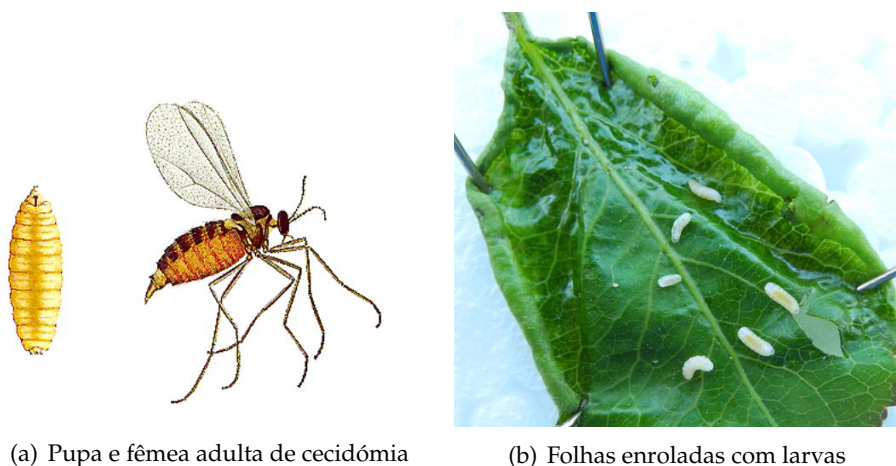


Figura 2.1: Colônia de filoxera na Primavera (fêmeas, ovos e larvas)

Cecidómia (*Dasineura pyri* Bouché)

O adulto é um pequeno díptero, com asas transparentes e patas longas, apresentando dimorfismo sexual. Os ovos, são depositados nas folhas das quais se irão alimentar ao eclodir, levando à desfoliação. Após o desenvolvimento, as larvas caem no solo e pupam em casulos. O ciclo repete-se 2 a 4 vezes por ano. Os danos incluem folhas enroladas, dessecação e impacto em pomares jovens. Nas árvores adultas, a praga tem menos influência na frutificação, mas pode servir de abrigo para outras pragas [7].



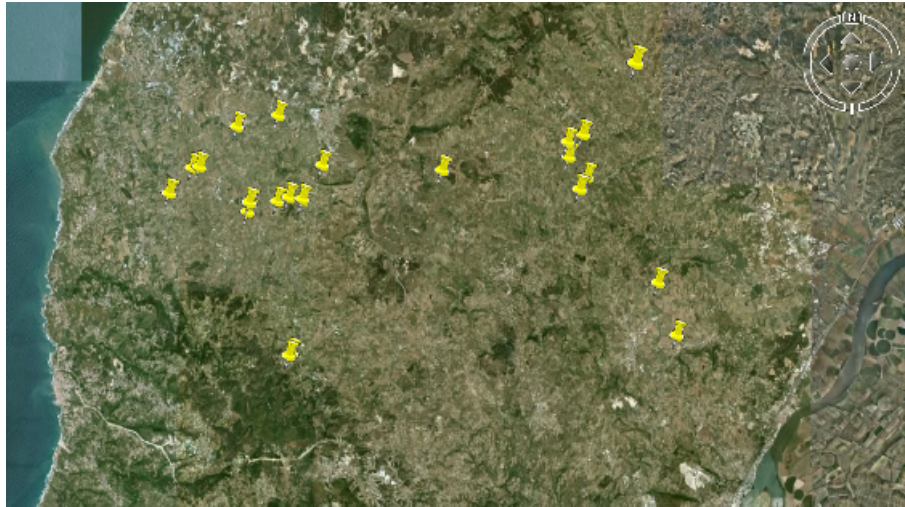
(a) Pupa e fêmea adulta de cecidómia

(b) Folhas enroladas com larvas

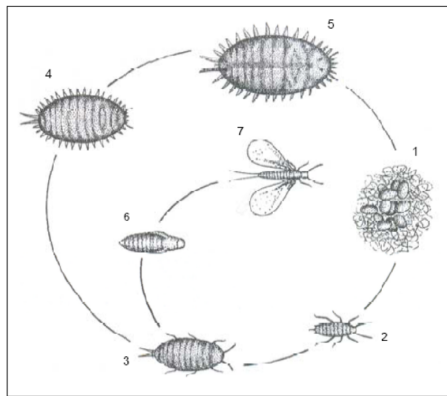
Figura 2.2: Ilustração de cecidómia e ataque a folhas

Cochonilha-Algodão (*Pseudococcus viburni*)

A cochonilha-algodão, um homóptero da família *Pseudococcidae*, ataca vinhas e pomóideas, preferindo regiões costeiras e mediterrânicas. A fecundidade máxima ocorre entre 18 e 22 °C. Seu ciclo de vida inclui ovos, ninfas de primeiro, segundo e terceiro instar, e adultos. Os danos resultam da alimentação nas folhas e da excreção de melada nos frutos, o que atrai organismos nocivos causando prejuízos aos produtores [26].



(a) Zonas atacadas



(b) Ciclo de vida

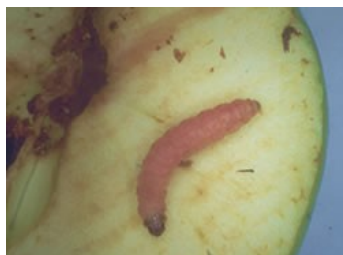


(c) Danos causados

Figura 2.3: Cochonilha-Algodão

Traça Oriental (*Grapholita molesta* Busck)

Adulto com asas, castanho escuro, com mancha esbranquiçada, ovos elípticos depositados nas folhas. Larvas de corpo inicialmente branco, torna-se posteriormente cor-de-rosa, atacam rebentos jovens e frutos, formando galerias descendentes. Ciclo de vida com 4 gerações, borboletas de março a abril e setembro a outubro. Danos significativos em frutos, especialmente em pomares próximos a pessegueiros. Na macieira, os ramos raramente são afetados, as larvas atacam frutos pela cavidade peduncular [43].



(a) Larva



(b) Traça Adulta

Figura 2.4: Larva e traça adulta da *Grapholita molesta* Busck

2.2 Sistema de Informação

Como referido anteriormente [1.2](#) o sistema de informação foi desenvolvido inicialmente para manter informação relevante ao estudo das pragas emergentes, na região Oeste. O sistema que foi criado tinha um carácter genérico e flexível para ser facilmente adaptado, a requisitos emergentes.

O modelo da base de dados do sistema de informação, após ter sido estendido encontra-se dividido em seis módulos ([Figura 2.5](#)):

Módulo de Agentes e Contas

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

Módulo Espaço-Temporal

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

Módulo de Dados de Campo

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

Módulo de Séries Temporais

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

Módulo de Configurações

Este módulo disponibiliza a informação sobre as configurações base do modelo de dados, do Sistema de Informação, da aplicação móvel e web e da componente de visualização.

O modelo representado na [Figura 2.5](#) é resultante do trabalho da dissertação de D. Lopes [[22](#)], cujo o objetivo era criar um sistema de informação geográfica, no âmbito do ordenamento do território e da conservação da natureza, para ajudar na prevenção de incêndios florestais.

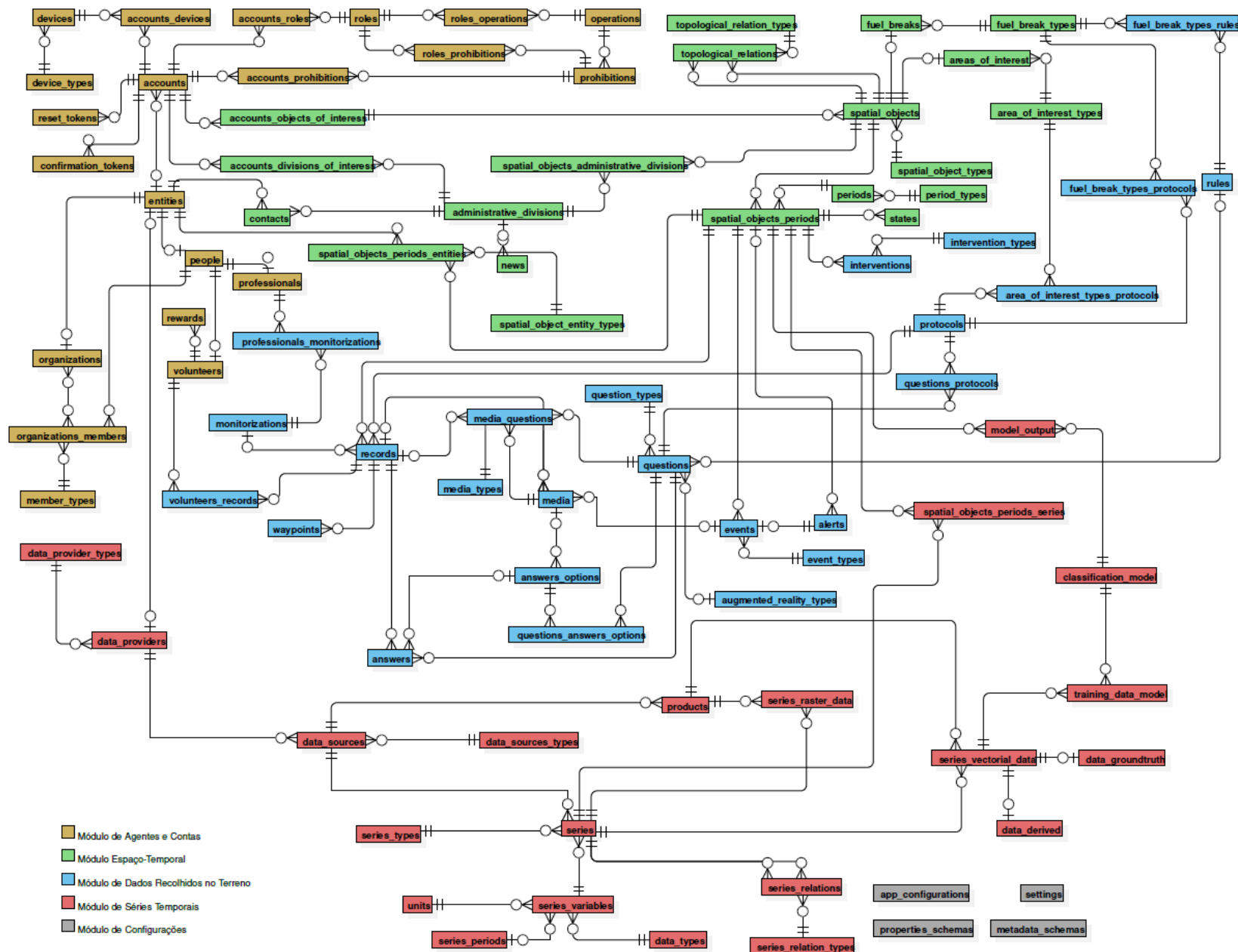


Figura 2.5: Modelo atual do sistema de informação

2.3 Recolha de Informação

Através do projeto FitoAgro foram recolhidos diversos tipos de dados de diversas fontes que foram registados no sistema de informação previamente apresentado.

A informação recolhida pertence a duas categorias:

Fitossanitária: baseia-se em dados que são recolhidos no campo;

Meteorológica: consiste em dados provenientes de estações meteorológicas internas e externas ao projeto.

2.3.1 Informação fitossanitária

Este tipo de informação é recolhida semanalmente pelos técnicos que se dirigem ao local onde se encontram as POBs com o propósito de registarem suas observações relativamente aos *elements of interest* (EOIs) que podem ser árvores ou equipamentos de monitorização.

Para além da cultura, cada POB tem um inimigo principal associado, que por sua vez tem um protocolo de recolha de informação específico que deve ser executado pelos técnicos durante as suas visitas. Estes protocolos foram desenvolvidos pelos parceiros do projeto FitoAgro, com o objetivo de estudar as pragas principais que afetam a região Oeste referidos em (2.1).

Para além dos protocolos específicos para inimigo em estudo, existe ainda um protocolo generalista que deve ser executado em todos os POBs quando são visitados. Este protocolo descreve as observações genéricas a serem realizadas para todas as pragas em estudo, sejam estas principais ou secundárias.

Para além dos dados recolhidos durante a execução dos protocolos, é também da competência do técnico registar informação adicional relativa:

- **Estado fenológico** do pomar;
- **Fenómenos meteorológicos** que ocorreram durante a visita;
- **Tratamentos** feitos por parte dos produtores;
- **Anotações adicionais** que os técnicos entendam relevantes;

A recolha dos dados é realizada ao longo de determinados períodos de tempo denominados campanhas. Campanhas variam dependendo da cultura. No projeto FitoAgro as campanhas são anuais, iniciando-se em outubro.

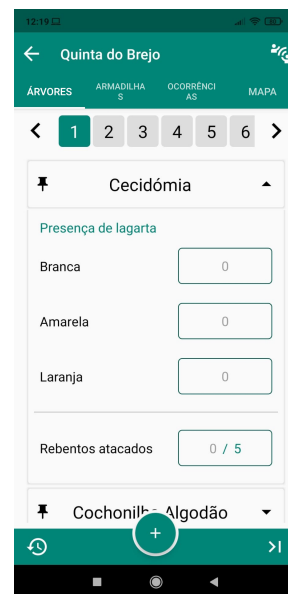
Inicialmente esta recolha de dados era registada em papel e transcrita manualmente para uma folha *Excel*, que estaria armazenada na *Google Drive*; tudo isto era realizado pelo técnico, o que era exaustivo e redundante, podendo vir ainda a introduzir erros de transcrição. Por estas razões foi desenvolvida uma aplicação móvel denominada de ProtoCollector (figura 2.6), implementada no contexto da dissertação de M. Correia [10], que torna o processo mais intuitivo e após cada visita, a informação é validada e integrada automaticamente no sistema de informação.

Posteriormente foi desenvolvida uma versão do ProtoCollector para web, onde é possível ter acesso a todas as visitas e os respetivos dados (acima referidos) registados pelo técnico durante a visita.

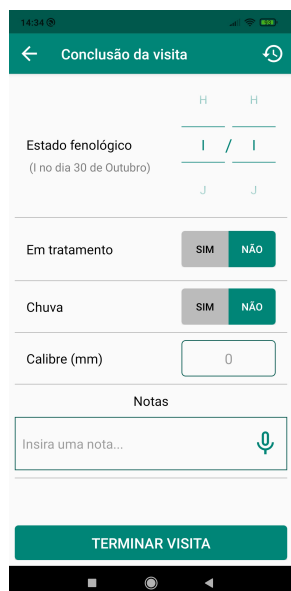
Uns dos grandes objetivos desta dissertação foi desenvolver um portal para configurar a aplicação móvel de cada técnico, alterando as parcelas atribuídas ao técnico ou mudar o protocolo que o técnico executa em determinada parcela.



(a) Protocolos suportados



(b) Protocolo da Cecidómia

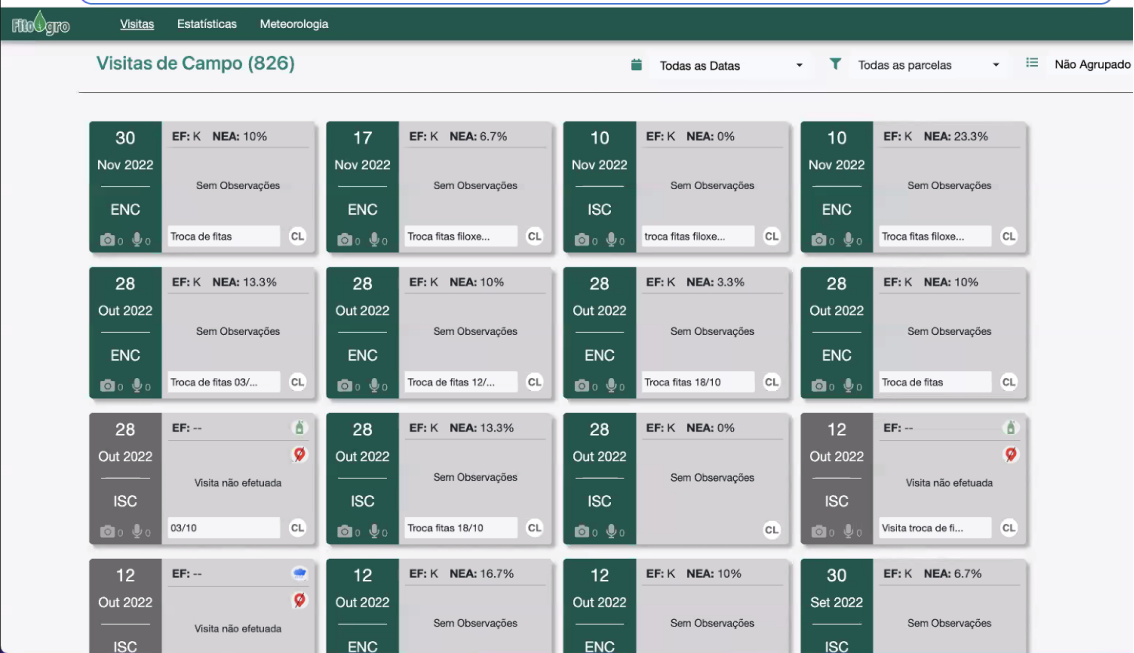


(c) Informação adicional sobre a visita

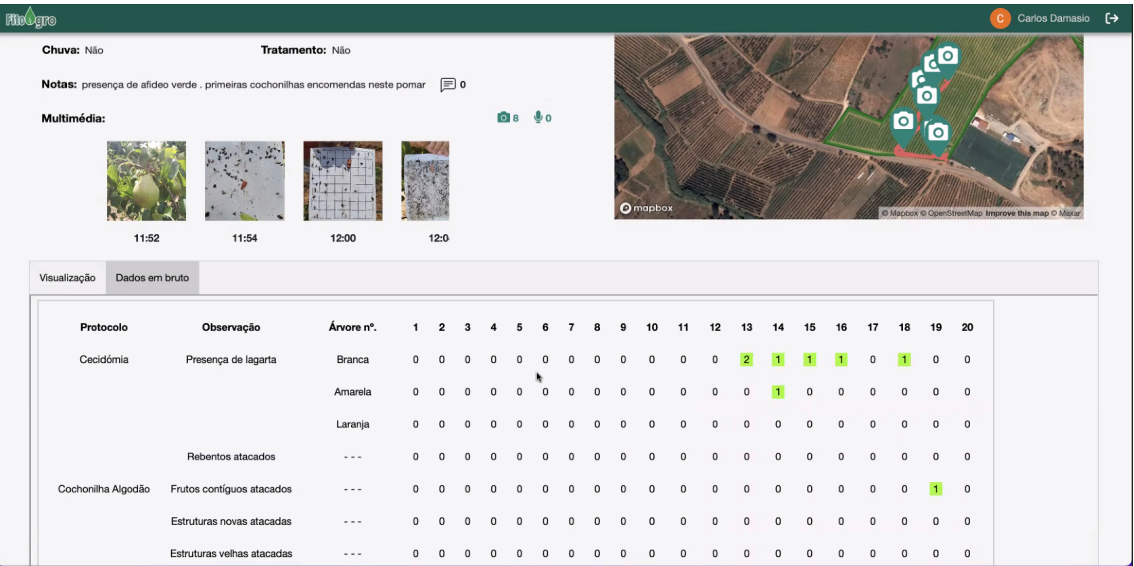


(d) Resumo da visita

Figura 2.6: Vistas importantes da aplicação móvel ProtoCollector



(a) Página inicial com todas as visitas



(b) Página de uma visita

Figura 2.7: Vistas importantes da aplicação web ProtoCollector

2.3.2 Informação Meteorológica

Esta informação é recolhida de diversas estações meteorológicas fornecidas por:

- **Parceiros do projeto**
 - Associação dos Produtores Agrícolas da Sobrena (APAS) [3]
 - Pessl Instruments [29]
 - TerraPro [41]
 - WiseCrop [45]
- **Entidades externas**
 - Instituto Português do Mar e da Atmosfera (IPMA) [18]

Todos estes fornecedores e as suas respetivas estações já foram previamente configurados no sistema de informação [24], com o propósito de que dos dados gerados por estas estações sejam automaticamente integrados no sistema.

Devido ao facto dos dados serem provenientes de fontes diferentes, o seu formato também irá ser diferente, o que poderia vir a gerar complicações. Neste intuito, foi desenvolvido, um processo [9] que tem como objetivo uniformizar todas as estações para o fuso horário UTC (Coordinated Universal Time) e para uma frequência de amostragem horária.

Apesar de cada estação fornecer um conjunto variado de sensores meteorológicos, apenas foram selecionados e utilizados os sensores cujos dados são relevantes para o estudo do desenvolvimento de pragas.

Sensor	Unidade de medida
Temperatura do ar	Grau Celsius (° C)
Humidade relativa	Percentagem (%)
Precipitação acumulada	Milímetro (mm)
Ponto de orvalho	Grau Celsius (° C)
Indicador de folha molhada	Minuto (min)
Velocidade do vento	Metro por segundo (m/s)
Radiação solar	Watt por metro quadrado (W/m ²)

Tabela 2.1: Sensores relevantes e a sua respetiva unidade.

Através da plataforma a desenvolver, é possível alterar a estação meteorológica mais próxima, atribuída a cada parcela, para outra em que os dados que recolhe sejam mais significativos, para a análise e estudo do inimigo. No futuro pretende-se estender a aplicação, para incorporar dados de previsão meteorológica.

2.4 Visualização de Dados

Durante a evolução projeto FitoAgro ocorreu a necessidade de criar uma componente de visualização que fosse capaz de atender os requisitos analíticos elicitados pelos seus diferentes elementos e que servisse como ferramenta de apoio à decisão no delinear de estratégias de controlo biológico, para os diferentes tipos de pragas na região do Oeste. Para que tal fosse possível, deveria conter mecanismos para a análise, de dados meteorológicos e de dados obtidos durante as visitas de campo, bem como promover a descoberta de correlações entre estes dois tipos de dados. Neste propósito, foi estendida no âmbito da dissertação de M. Ferreira [12] a base de dados analítica desenvolvida por A. Malafaia [24] e foi criada uma componente de visualização usando a ferramenta Tableau. A base de dados analítica tem como função de um modo geral, guardar informação que possa ser útil para análise de dados do sistema de informação, que por sua vez irá ser exibida graficamente, através componente de visualização.

2.5 Modelos Preditivos

O grupo operacional FitoAgro tem como um dos objetivos desenvolver modelos de previsão para todas as pragas que afetam a região Oeste, tanto as principais referidas em (2.1), como para novas pragas, que possam vir a causar preocupação na região.

Os modelos preditivos servem para ajustar modelos de desenvolvimento, que são modelos desenvolvidos em laboratório num ambiente controlado, ou seja, as condições de estudos das pragas no laboratório são bastante diferentes das que ocorrem no campo. Devido a esta discrepância, utilizam-se modelos preditivos para ajustar os modelos de desenvolvimento obtidos em laboratório, às condições climáticas registadas em cada local e a outras variáveis que possam ser consideradas relevantes. Estas informações climáticas são obtidas de estações meteorológicas próximas, ao local de interesse.

A principal vantagem dos modelos preditivos consiste, no facto de estes poderem estimar possíveis datas, em que é expectável que ocorram as diferentes fases de desenvolvimento da praga. Por sua vez, esta informação torna-se valiosa para os produtores pois auxilia-os na tomada de decisão relativamente à aplicação de medidas de combate, tornando-as mais eficazes e eficientes.

2.6 Stakeholders

O projeto FitoAgro tem parceiros provenientes de diversas instituições e entidades, com diferentes contributos e objetivos. Identificamos assim sete tipos de *stakeholders* tendo em conta os seus objetivos:

- **Produtores** - Donos ou responsáveis pelos pomares.
- **Técnicos de Recolha de Dados** - Monitorização da recolha dos dados de campo.
- **Técnicos da Produção** - Gestão das culturas, POBs e campanhas.
- **Organizações de produtores ou associações** - Constituídas por produtores e técnicos.
- **Desenvolvedores** - Desenvolvimento e manutenção da aplicação.

- **Back Office** - Principais utilizadores da aplicação responsáveis pela manutenção do sistema FitoAgro.
- **Gestão do Projeto** - Gestão do projeto de monitorização fitossanitária, tanto da parte de monitorização e controlo como da analítica dos dados.

A plataforma a desenvolver no âmbito desta dissertação, tenta responder a todas necessidades e requisitos dos *stakeholders*, de forma a que estes consigam através da aplicação, cumprir seus objetivos.

2.7 Limitações e Conclusões

Após a análise dos diferentes componentes da plataforma original do projeto FitoAgro, encontrou-se um problema principal comum ao sistema todo: falta de flexibilidade na manipulação de informação inserida no sistema de informação. através das interfaces disponíveis. Passa-se agora a detalhar para cada componente da plataforma, as limitações encontradas.

Sistema de informação

O processo para inserção de novos utilizadores, novas parcelas e novas estações meteorológicas não é possível, pois esta informação tem de ser inserida manualmente na base de dados, algo que acaba por ser bastante complicado e moroso.

Recolha de informação fitossanitária

Apesar do processo de inserção dos dados, no sistema de informação, durante a recolha ter sido extremamente melhorado, após a informação ser inserida não é possível fazer qualquer alteração à mesma, algo que seria bastante útil, no caso de ocorrência de um erro, no processo de recolha e consequente registo de informação errada.

Outro problema encontrado, foi o de não ser possível alterar o técnico responsável por certa parcela e o respetivo protocolo que este tem de efetuar. Estes protocolos que são imutáveis, ou seja, não é possível nem adicionar protocolos novos, nem fazer alterações aos já existentes.

Recolha de informação meteorológica

Neste caso o problema não ocorre no processo de recolha dos dados, mas sim na sua fonte, ou seja, não é possível alterar a estação meteorológica responsável pela parcela em questão para outra em que os dados recolhidos sejam mais significativos, para a análise e estudo do inimigo da parcela. Outro problema é o facto do sistema não englobar a componente de previsão meteorológica.

Visualização dos dados

Como referido pelo seu nome, esta componente apenas permite a visualização dos dados, não a sua inserção ou alteração, impossibilitando a resolução dos problemas anteriormente referidos.

Modelos Preditivos

O problema referente aos modelos preditivos é semelhante ao dos protocolos de recolha de informação fitossanitária, que é o facto de serem imutáveis, ou seja, não é possível fazer a sua afinação.

Estes problemas poderiam ser endereçados, caso houvesse uma aplicação de *backoffice*, onde fosse possível gerir e manter o sistema, fazendo inserções, remoções ou alterações, nos dados guardados no sistema caso necessário.

TRABALHO RELACIONADO

Neste capítulo é elencado um levantamento de algumas aplicações existentes que abordam uma temática igual, ou semelhante ao da aplicação desenvolvida para esta dissertação (3.1). Por fim encontra-se presente o trabalho de pesquisa relativo ao levantamento de tecnologias consideradas, para o desenvolvimento da aplicação (3.2), abrangendo tecnologias tanto de servidor como da componente cliente.

3.1 Projetos Relacionados

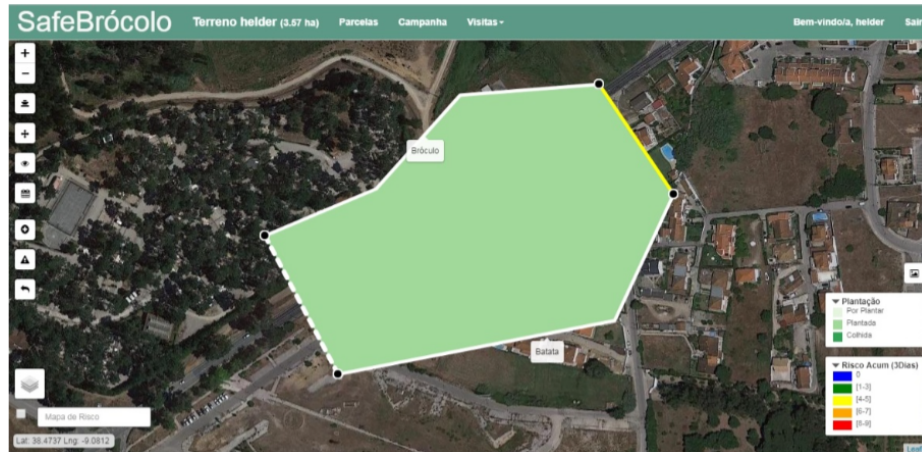
Esta primeira secção tem o objetivo de fazer o levantamento de aplicações cujo domínio de ação é semelhante ao da aplicação desenvolvida, durante esta dissertação.

3.1.1 SafeBrócolo

O Safebrócolo [31] é um projeto que resultou na criação de uma plataforma inovadora que oferece uma componente móvel e web, cujo o objetivo foi é o desenvolvimento de modelos de previsão, de ocorrência de *alternariose* e ataques de mosca da couve.

A aplicação móvel permite aos utilizadores registarem as suas parcelas e as observações efetuadas no campo. Além disso, disponibiliza mapas de risco de Portugal Continental e indicadores de risco para cada uma das parcelas do utilizador, com alertas personalizáveis para o mesmo.

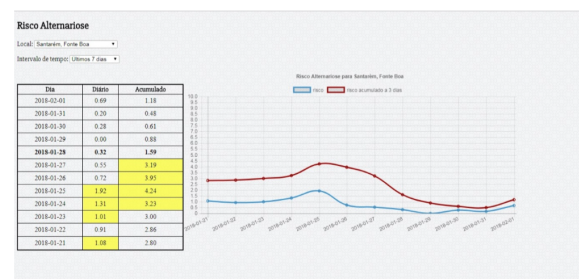
Através da aplicação web é possível observar os dados georreferenciados recolhidos pela aplicação móvel bem como permitir a adição e complementação de informação registada. Permite também o registo de parcelas e observações, mas, ao contrário da aplicação móvel, é possível definir cabeceiras e bordaduras e fazer divisões de áreas de plantação, dentro de uma parcela. Esta plataforma apresenta não só visualizações para os dados georreferenciados, mas também para estimativas de risco por parcela ou estação e mapas de risco.



(a) Página da parcela



(b) Página de risco da parcela



(c) Visualização do risco de *alternariose* por estação.



(d) Mapa de risco de *alternariose* para Portugal Continental.

Figura 3.1: Algumas vistas *web* da plataforma SafeBrócolo

3.1.2 Cropwise Operations

A plataforma Cropwise Operations [40] representa um sistema avançado de gestão agrícola via satélite, proporcionando a monitorização remota e eficaz de parcelas. Esse sistema permite aos utilizadores terem a capacidade de planear e executar suas operações agrícolas de maneira eficiente, com o objetivo de maximizar a produtividade. Utilizando imagens

de satélite, previsões meteorológicas precisas e outras informações cruciais, a plataforma fornece aos seus utilizadores, uma visão abrangente do estado das suas culturas, níveis de vegetação e potenciais áreas propensas a problemas.

Adicionalmente, o sistema gera semanalmente um resumo de dados pertinentes sobre as condições das parcelas monitorizadas. Tal possibilita, um acompanhamento contínuo do crescimento da cultura, desde da plantação até à colheita. A aplicação está disponível em várias línguas, e o acesso ao sistema, é mediante o pagamento.

Em resumo, a plataforma destaca-se como uma ferramenta abrangente e eficaz para a gestão agrícola, oferecendo informações precisas e oportunas que auxiliam os agricultores na tomada de decisões fundamentadas, para otimizar seus resultados.

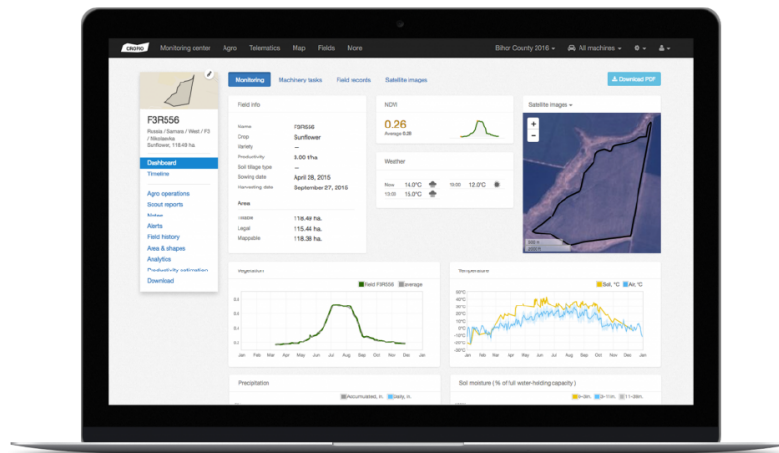


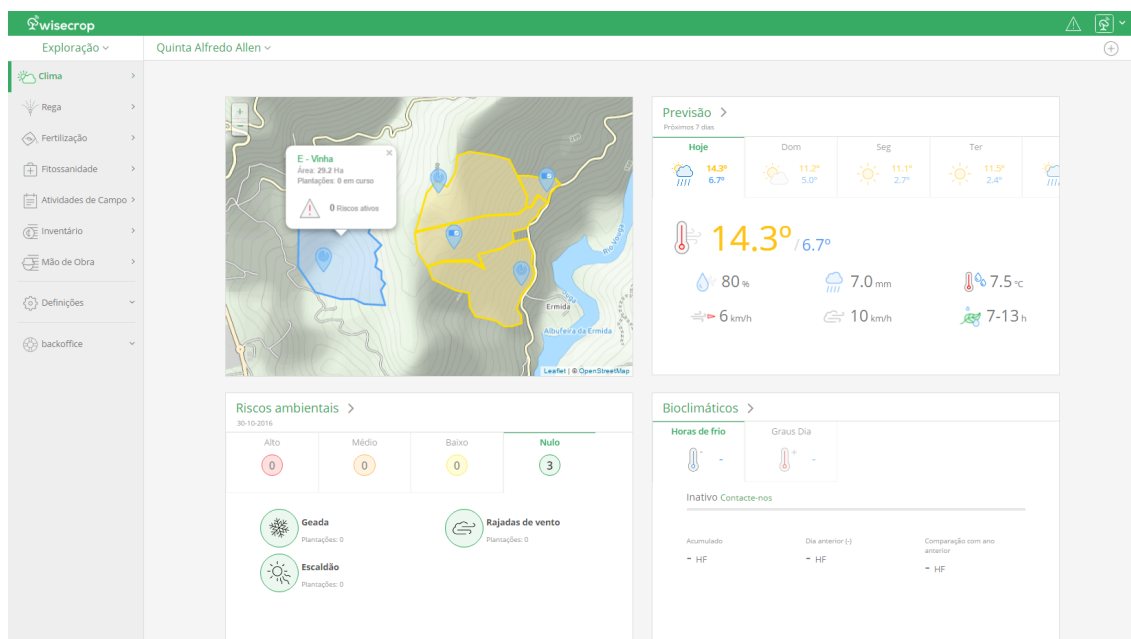
Figura 3.2: Exemplo da vista geral do sistema Cropwise Operations

3.1.3 WiseCrop

A WiseCrop [45] é uma *startup* portuguesa que disponibiliza um sistema de apoio à gestão agrícola, auxiliando o produtor na economia de recursos e no aumento da produção. Proporciona ao empresário agrícola, dados quase em tempo real (atualizados a cada 15 minutos) sobre o estado das suas culturas, através da medição de diversos parâmetros meteorológicos como temperatura; humidade; velocidade e direção do vento; radiação; e humidade do solo; entre outros. Esta medição é realizada por vários sensores colocados nas parcelas que os utilizadores desejam monitorizar.

Além das medições, a plataforma inclui um sistema de alertas que utiliza modelos de previsão, para informar o produtor sobre a ocorrência de pragas, doenças ou variações climáticas, que possam prejudicar a cultura. Oferece também uma personalização significativa, permitindo ao utilizador definir os seus próprios alertas.

Este sistema é pago anualmente e é adquirido por módulos, de acordo com as necessidades específicas de cada utilizador.



(a) Módulo do clima



(b) Módulo atividades de campo



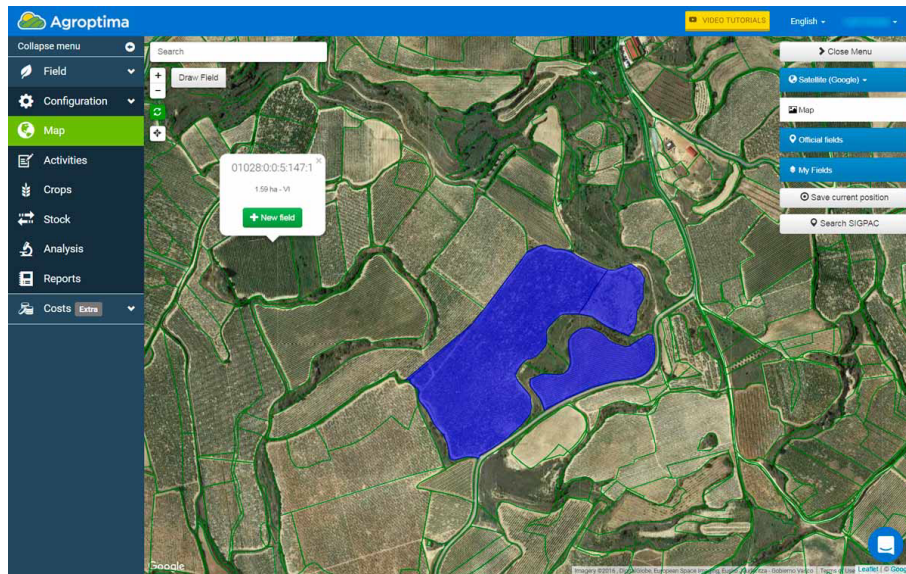
(c) Módulo fitossanidade

Figura 3.3: Wisecrop

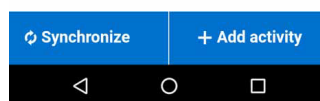
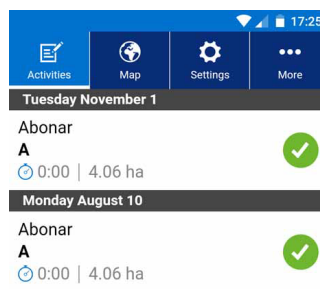
3.1.4 Agroptima

A Agroptima [1] é uma aplicação móvel paga com uma versão para *desktop*, que oferece aos utilizadores um período de testes. Na versão móvel é permitido a definição e localização de campos e colheitas, agendamento de tarefas e registo de atividades no campo. A informação é inicialmente armazenada localmente e posteriormente sincronizada com a nuvem. Além disso, disponibiliza um extenso catálogo de fertilizantes, para consulta pelos utilizadores.

A aplicação *desktop* tem como objetivo proporcionar aos utilizadores, uma ferramenta para consultar todos os dados registados pela aplicação, oferecendo uma visão abrangente e facilitando a análise das informações recolhidas.



(a) Aplicação desktop



(b) Aplicação móvel

Figura 3.4: Agroptima

3.1.5 SmartFarm

O SmartFarm da CropIn [37] é uma plataforma abrangente e altamente configurável de gestão agrícola projetada para eficiência, previsibilidade e sustentabilidade. Funciona como uma suíte completa de *Enterprise Resource Planning* (ERP) e *Customer Relationship Management* (CRM) para agricultura, fornecendo soluções digitais para o setor. A plataforma oferece recursos para gestão de plantações, utilizando tecnologias inteligentes como análise de *Big Data*, inteligência artificial e *machine learning*.

O SmartFarm da CropIn visa preencher a necessidade crítica de informações agrícolas credíveis, proporcionando uma plataforma digital para agricultores e *stakeholders* do setor. Suporta várias funcionalidades, incluindo monitorização de culturas, otimização de recursos e perspectivas gerais, baseados em dados recolhidos.

3.1.6 AGRIVI

A plataforma AGRIVI 360 Farm Enterprise é uma solução centralizada baseada em nuvem que reúne todos os dados agrícolas, tais como, dados de IoT e informações em tempo real do campo, fornecendo uma plataforma poderosa de apoio à decisão que oferece *insights* agrónomos e económicos acionáveis.

A plataforma está disponível como uma aplicação web e uma aplicação móvel para dispositivos iOS e Android, com a função principal de modo *offline* para a aplicação móvel AGRIVI.

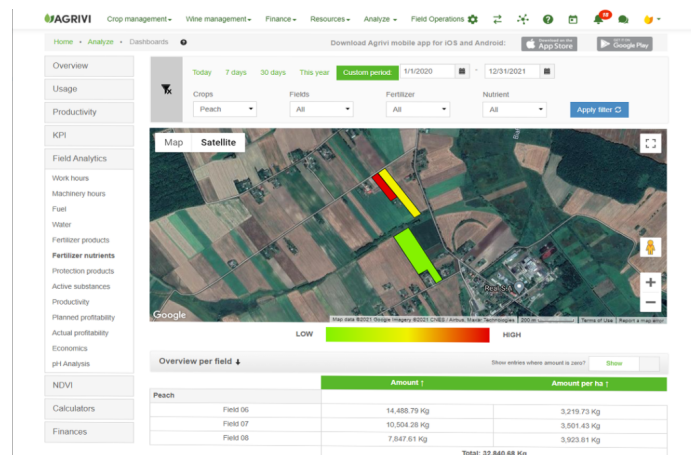
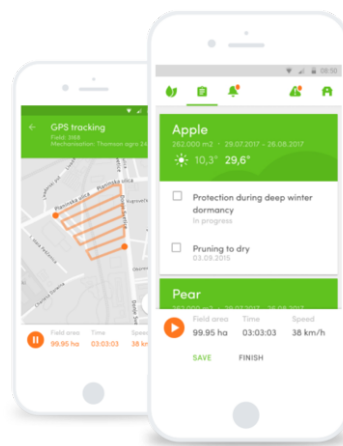
A AGRIVI 360 Farm Enterprise é perfeitamente adequada para integração com *Enterprise Resource Planning* (ERP) dos clientes ou outras soluções de software existentes.

Com suporte para 15 línguas, serviço internacional de apoio ao cliente, a AGRIVI oferece a capacidade de cobrir operações internacionais com uma única plataforma.

A AGRIVI 360 Farm Enterprise oferece suporte a todos os tipos de culturas, como frutas, legumes, grãos e outras plantas.

Principais funcionalidades da AGRIVI 360 Farm Enterprise:

- Gestão de recursos da propriedade (Culturas, campos, maquinaria, pessoas, inventário)
- Operações da propriedade (planeamento, rastreamento e análise completa da produção de culturas)
- Vigilância da propriedade (Imagens de Satélite NDVI, base de conhecimento de pragas e doenças, evidências fotográficas georreferenciadas)
- Inteligência climática e de prevenção de riscos (dados climáticos, alertas climáticos, alertas de riscos de pragas e doenças)
- Gestão Financeira da propriedade (faturas, contas a pagar e a receber, orçamento, gestão de empréstimos)
- *Dashboards* e análises da propriedade (análise de campos, análise de rendimento, rentabilidade das culturas, ROI, KPIs financeiros da fazenda, relatórios)
- Relatório de Rastreabilidade

(a) Aplicação *desktop*

(b) Aplicação móvel

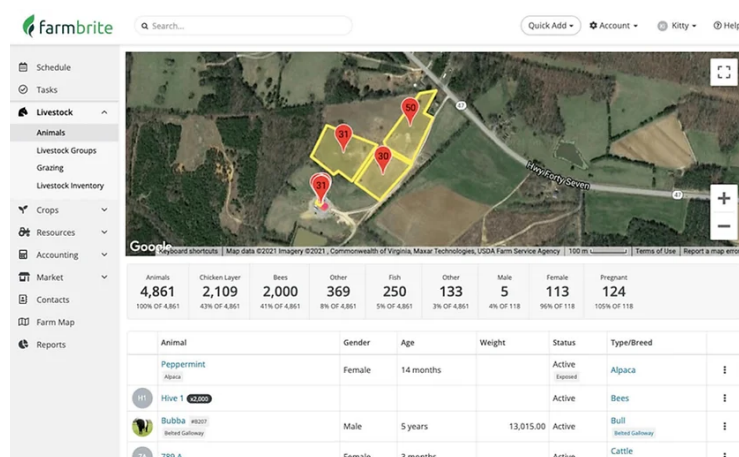
Figura 3.5: AGRIVI

3.1.7 Farmbrite

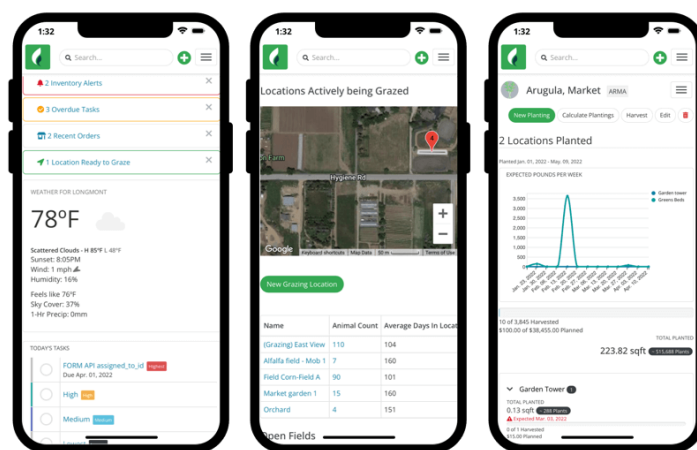
O Farmbrite é uma plataforma de gestão agrícola baseada na nuvem, que fornece ferramentas para gerir operações agrícolas, planejar e otimizar produção, agendar tarefas e atividades, rastrear finanças, gerir clientes e muito mais. A plataforma oferece tanto uma componente web como móvel.

Com o Farmbrite, os produtores podem organizar, otimizar e gerir seus negócios agrícolas usando ferramentas operacionais gerais, como um calendário integrado, listas de tarefas, estimativas, relatórios, rastreamento de atividades. O Farmbrite também oferece uma variedade de ferramentas específicas da indústria para gestão de gado, cultivo de culturas e operações agrícolas. O gado pode ser gerido usando ferramentas para rastreamento e planeamento de reprodução, genealogia, saúde do rebanho, pastoreio e muito mais. Os agricultores podem usar o Farmbrite para relatar colheitas, contagem de nutrientes, monitorizar a saúde do solo, estimar encomendas de sementes.

Projetado para ajudar as empresas agrícolas a simplificar os seus processos digitais, o Farmbrite oferece acesso a dados, em qualquer lugar e pode ser integrado aos fluxos de trabalho digitais existentes. Relatórios podem ser gerados para identificar tendências, avaliar o desempenho, monitorizar receitas e despesas.



(a) Aplicação desktop



(b) Aplicação móvel

Figura 3.6: Farmbrite

3.1.8 Conclusão

Após a análise detalhada dos sistemas anteriormente apresentados, é possível identificar várias características e limitações que influenciam a eficácia e a flexibilidade das soluções de gestão agrícola, especialmente no que diz respeito à prevenção e controlo de pragas e doenças.

Primeiramente, constatou-se que os sistemas analisados anteriormente oferecem funcionalidades robustas para a gestão e monitorização de parcelas agrícolas. Sendo que estas plataformas recorrem a tecnologias modernas e avançadas, tais como imagens de satélite, sensores IoT e modelos preditivos, com o propósito de fornecer em tempo real informação relativa ao estado das culturas e fornecer previsões precisas sobre a ocorrência de pragas e doenças.

No entanto, uma limitação comum entre esses sistemas reside na reduzida flexibilidade na adaptação dos modelos preditivos e dos protocolos para estudo dos inimigos. A maioria das plataformas utiliza modelos predefinidos que não permitem ajustes ou personalizações pelos utilizadores. Esta rigidez pode ser um obstáculo significativo, uma vez que a capacidade de ajustar as estratégias de prevenção e controlo, com base em

alterações climáticas ou alterações no comportamento do inimigo em análise, poderia aumentar significativamente a eficácia e eficiência das mesmas.

Em relação à usabilidade das plataformas. Embora as suas interfaces sejam intuitivas, a complexidade de algumas funcionalidades pode ser um desafio para utilizadores menos experientes. Esta circunstância pode originar a necessidade de formação adicional ou suporte técnico contínuo, o que pode constituir um obstáculo à adoção destas plataformas.

Em conclusão, enquanto as plataformas analisadas oferecem uma gama abrangente de funcionalidades para a gestão agrícola, existe uma necessidade de maior flexibilidade e personalização nos modelos de previsão e protocolos para estudo do inimigos. A capacidade de adaptar estas ferramentas às necessidades específicas dos utilizadores e às condições ambientais é crucial para maximizar a eficácia das estratégias de prevenção e controlo de pragas e doenças.

3.2 Tecnologias Relevantes

Nesta secção são exploradas todas as tecnologias consideradas para o desenvolvimento da aplicação. São abrangidas tecnologias de base de dados, onde a sua principal função é a do armazenamento de informação. São também exploradas tecnologias para o cliente, cuja a sua principal função é construir uma interface iterativa onde é possível representar informação, para o utilizador. Para complementar as tecnologias para o cliente são exploradas bibliotecas que permitem o uso de mapas, na aplicação e bibliotecas que permitem visualizar grandes quantidades de informação, de forma intuitiva, recorrendo a representações gráficas.

3.2.1 Tecnologias de Base de Dados

Nesta subsecção são identificadas tecnologias de base de dados onde a sua principal função será o armazenamento de informação.

3.2.1.1 PostgreSQL

O PostgreSQL [33] é um sistema de gestão de base de dados *open source* conhecido pela sua arquitetura confiável, priorizando a integridade, a consistência e a replicação dos dados. Este sistema destaca-se como um dos mais avançados em bases de dados relacionais devido aos seus recursos sofisticados, como o método de controlo multiversão, que gere a consistência de dados em acessos concorrentes à mesma tabela, e o método de replicação lógica, proporcionando controlo refinado sobre a replicação e a segurança dos dados. Devido à sua robustez, tem a capacidade de lidar com mecanismos complexos e bases de dados de grande escala.

Além disso, o PostgreSQL possui a extensão PostGIS [32], conferindo-lhe capacidades significativas na representação de informação geográfica. Essa extensão adiciona tipos de pontos geométricos, multipontos, retas, polígonos e multipolígonos, tornando o PostgreSQL uma escolha valiosa, como base de dados para sistemas de informação geográfica como o que pretendemos desenvolver.

Este é o sistema de gestão de base dados utilizado no sistema de informação FitoAgro.

3.2.1.2 MySQL

O MySQL [27] é um sistema de gestão de bases de dados relacional *open source*, amplamente utilizado para construir bases de dados escaláveis e eficientes. Desenvolvido pela *Oracle Corporation*, o MySQL utiliza a *Structured Query Language*(SQL) para gerir e manipular dados. Suporta vários sistemas operativos e é reconhecido pela sua rapidez, fiabilidade e facilidade de utilização.

As características do MySQL incluem a conformidade com *ACID*, garantindo a fiabilidade das transações, e o suporte a múltiplos motores de armazenamento, permitindo aos utilizadores escolher o motor mais adequado às suas necessidades específicas. É comum ser utilizado em aplicações que vão desde *websites* de pequena escala até a sistemas empresariais de grande envergadura, tornando-se uma solução de base de dados versátil e amplamente adotada.

3.2.1.3 SQLite

SQLite [39] é uma biblioteca em C que implementa uma base de dados transacional *standalone* embebida. Ao contrário da maior parte das bases de dados SQL, SQLite não tem um processo servidor em separado. Os programas que usam esta biblioteca podem ler e escrever diretamente nos ficheiros de base de dados, no disco. É uma base de dados com reputação de ser bastante confiável, lidando bem com falhas de alocação de memória e erros de *input/output* no disco. Garante ainda transações *ACID* mesmo no caso do sistema falhar ou de haver alguma falha de energia. No entanto, não lida bem com grandes quantidades de dados. Este é o sistema de gestão de base de dados usado na aplicação móvel ProtoCollector.

SpatiaLite [38] é uma biblioteca *open source* desenvolvida para ser integrada com SQLite, oferecendo suporte a capacidades de representação de informação geográfica (de forma semelhante ao que o PostGIS oferece ao PostgreSQL).

3.2.2 Bibliotecas para mapas

Nesta subsecção são exploradas tecnologias que através da sua integração é possível a utilização de mapas interativos, onde se pode visualizar e gerar informação georreferenciada.

3.2.2.1 Mapbox

O Mapbox [25] é uma plataforma de localização que fornece APIs (Interfaces de Programação de Aplicações) e SDKs (Kits de Desenvolvimento de Software) para a criação de mapas personalizados em aplicações web, móveis, automóveis e de realidade aumentada (AR). A plataforma oferece serviços como mapas interativos, navegação, busca de localização e dados em tempo real.

Os programadores podem utilizar o Mapbox para criar visualizações de mapas altamente personalizadas, adaptando estilos, camadas e funcionalidades de acordo com as necessidades específicas de suas aplicações. Ele é frequentemente utilizado em diversos setores, desde aplicações móveis e jogos até ferramentas de análise de dados geoespaciais.

O Mapbox é reconhecido pela sua flexibilidade e por permitir uma ampla gama de personalizações visuais e funcionais nos mapas, criados por meio de sua plataforma.

Foi a opção usada para a componente de mapas, da aplicação móvel e web ProtoCollector.

3.2.2.2 OpenLayers

OpenLayers [28] é uma biblioteca JavaScript *open source* utilizada para exibir dados de mapas em navegadores web, conhecidos como mapas interativos. Ela fornece uma API para a criação de mapas dinâmicos, permitindo a exibição de dados cartográficos em páginas da web de forma interativa. A OpenLayers é especialmente útil para criar mapas “slippy maps”, nos quais os utilizadores podem interagir, fazer zoom e *pan* nos mapas.

As principais características da biblioteca *OpenLayers* são:

- Possibilita a exibição de mapas com diferentes tipos de dados, incluindo camadas de mapa, dados vetoriais e marcadores.
- Pode ser integrada com outros *frameworks* e bibliotecas para aprimorar suas funcionalidades.

A biblioteca OpenLayers é utilizada habitualmente para criar aplicações web que exigem a visualização de dados geoespaciais de maneira interativa.

3.2.2.3 Leaflet

Leaflet [21] é uma biblioteca JavaScript *open source* que se destaca na criação de mapas interativos e amigáveis, para dispositivos móveis e *desktop*. Com um foco significativo em desempenho, usabilidade e simplicidade. A biblioteca Leaflet é uma escolha popular para programadores que necessitam incorporar mapas interativos, nas suas aplicações web.

Com um tamanho leve de aproximadamente 42 KB de JavaScript e 4 KB de CSS compactados, Leaflet oferece uma variedade de recursos, incluindo suporte para camadas raster e vetoriais, marcadores, círculos e outras funcionalidades de mapas interativos, tudo isto explicado detalhadamente, na documentação.

Além disso, Leaflet é reconhecida por sua facilidade de uso, tornando-a acessível para programadores com diferentes níveis de experiência. A sua presença ativa na comunidade de desenvolvimento e atualizações regulares, garantem que os programadores possam contar com uma biblioteca robusta, para suas necessidades de mapeamento interativo.

3.2.3 Bibliotecas de Visualização

Nesta subsecção serão exploradas bibliotecas que permitem a visualização e análise de grandes quantidades de dados de forma intuitiva recorrendo a representações gráficas.

3.2.3.1 Chart.js

Chart.js [8] é uma biblioteca JavaScript de *open source* usada para criar gráficos interativos e visualizações de dados, em páginas da web. Ela é projetada para ser simples e flexível, permitindo aos programadores criar facilmente gráficos dinâmicos e responsivos.

Chart.js é amplamente utilizada devido à sua facilidade de uso e documentação abrangente. Ela oferece uma variedade de tipos de gráficos, incluindo barras, linhas, radar, pizza e muito mais. Além disso, permite personalizar cores, rótulos e outros elementos gráficos para atender às necessidades específicas do programador.

3.2.3.2 JSCharting

JSCharting [19] é uma biblioteca de gráficos em JavaScript que oferece uma ampla variedade de tipos de gráficos, desde gráficos básicos até gráficos avançados, incluindo mapas, ações, medidores de painel e gráficos de calendário. Com mais de 150 tipos de gráficos, a biblioteca é conhecida por sua versatilidade e capacidade de atender as diversas necessidades de visualização de dados.

A JSCharting suporta a criação de visualizações interativas e responsivas, incluindo gráficos dinâmicos e tabelas. Além disso, oferece integrações oficiais para React.js e Vue.js, tornando-a compatível, com várias estruturas de desenvolvimento.

A biblioteca é reconhecida por sua abordagem simples e flexível, permitindo aos programadores criar visualizações de dados personalizadas, sem complicações. A sua documentação abrangente fornece *API descriptions* e exemplos de cada tipo de gráfico, facilitando o desenvolvimento dos mesmos.

3.2.4 Tecnologias para Desenvolvimento de Interfaces

Nesta subsecção são exploradas tecnologias para o desenvolvimento da interface cliente, cuja a sua principal função é construir uma interface iterativa onde é possível demonstrar informação para o utilizador. Nestas tecnologias destaca-se ainda uma tecnologia *cross-platform*, que é uma *framework* que, a partir de um único código fonte possibilita construir aplicações para várias arquiteturas e dispositivos.

3.2.4.1 BootStrap

O Bootstrap [5] é um conhecido *framework* de *front-end open source* desenvolvido pelo Twitter. Consiste numa coleção de ferramentas *HTML*, *CSS* e *JavaScript* projetadas para otimizar o processo de desenvolvimento web. O Bootstrap oferece um conjunto de componentes, como barras de navegação, botões, formulários, entre outros, que os programadores podem incorporar facilmente nos seus projetos. Inclui também um sistema de grelha responsivo e várias classes utilitárias, para garantir interfaces consistentes e visualmente apelativos em diferentes dispositivos e tamanhos de ecrã.

3.2.4.2 Vue.js

Vue.js [0] é um *framework* de JavaScript utilizado para construir interfaces de utilizador (UI). Vue.js é um *framework* progressiva, ou seja, é concebido para ser adotado de forma incremental, o que significa que os programadores podem começar a utiliza-lo em pequenas partes dos seus projetos e aumentar gradualmente a sua utilização. O Vue.js concentra-se na camada de visualização da aplicação e fornece uma maneira flexível e eficiente de gerir o estado da interface do utilizador. É conhecido pela sua simplicidade, facilidade de integração e por um sistema de ligação de dados reativo, permitindo aos programadores criar aplicações web dinâmicas e interativas.

Vue.js adota uma arquitetura baseada em componentes, ou seja, encapsula unidades independentes da interface, promovendo a reutilização e a manutenção do código.

Vue.js utiliza um DOM (*Document Object Model*) virtual (VDOM) no seu mecanismo de renderização. O DOM virtual é um conceito onde a representação ideal ou "virtual" da interface é guardada em memória e é sincronizada com o DOM real. O principal propósito é otimizar e melhorar a eficiência em que a interface é atualizada.

Vue.js adota uma abordagem declarativa, onde o programador apenas tem de descrever onde é a localização do componente na interface e o *framework* cuida da manipulação do DOM.

A sintaxe do Vue.js foca-se na utilização de *templates*, que se assemelham a HTML algo que pode ser vantajoso para os programadores que estejam habituados a trabalhar com HTML nativo.

Em relação à gestão do estado global da aplicação, Vue.js contem integrado um sistema de gestão de estado chamado Vuex, que é opcional, ou seja, na criação do projeto pode ou não optar por integrar o Vuex.

3.2.4.3 React

React [35] é um *framework* de JavaScript desenvolvido pelo Facebook (atual Meta) para a construção de interfaces de utilizador (UI) em aplicações web. Diferentemente de outros *frameworks*, o React é considerado uma biblioteca devido à sua abordagem modular e flexível, não impondo regras estritas de organização de código.

O React apresenta várias semelhanças do Vue.js, pois a sua arquitetura é também baseada em componentes, faz uso de um DOM virtual e adota uma abordagem declarativa.

A sintaxe do React é baseada em JSX (JavaScript XML), uma extensão de sintaxe JavaScript que permite a escrita de elementos de UI de forma mais declarativa, algo que pode ser mais desafiador de entender para um programador iniciante.

Em relação à gestão do estado global da aplicação, React não contem nenhum mecanismo, sendo apenas possível através da integração da biblioteca Redux.

3.2.4.4 Angular

Angular [2] é uma plataforma e framework para a construção de aplicações cliente de página única (SPA) usando HTML e TypeScript. Ele segue uma arquitetura modular, onde os componentes são os blocos de construção principais. Os módulos organizam componentes relacionados e fornecem coesão e acoplamento controlado.

A manipulação do DOM no Angular é simplificada por meio de um sistema de templates declarativos. A sintaxe Angular inclui recursos como data binding, pipes e diretivas, facilitando a interação entre o código TypeScript e o DOM.

Angular segue a arquitetura MVC, o que significa que ele separa a lógica de negócios (Model), a apresentação (View) e o controlo do fluxo (Controller). Isso facilita a manutenção e o desenvolvimento modular da aplicação.

A gestão do estado global em Angular pode ser abordada de várias maneiras. Algumas práticas recomendadas incluem o uso do serviço de gestão de estado NgRx, que utiliza padrões Redux para oferecer uma solução robusta e escalável.

3.2.4.5 Flutter

O Flutter [14] é um conjunto de ferramentas de interface do utilizador (UI toolkit) e um framework desenvolvido pela Google que revoluciona todo o processo de desenvolvimento de aplicações. Permite aos programadores criar aplicações visualmente atraentes e compiladas nativamente para dispositivos móveis, web, *desktop* e incorporados, utilizando um único código-fonte. O Flutter foi projetado para fornecer uma interface do utilizador consistente e expressiva, em diferentes plataformas.

O framework vem com um conjunto rico de widgets pré-projetados, permitindo que os programadores criem interfaces personalizadas. Uma das principais características do Flutter é o "hot reload," que permite que os programadores vejam instantaneamente as alterações feitas no código sem reiniciar toda a aplicação.

A flexibilidade e eficiência do Flutter tornam-no numa excelente escolha para o desenvolvimento *cross-platform*, reduzindo a necessidade de códigos separados para iOS e Android. O Flutter ganhou popularidade pela facilidade de uso, ciclos de desenvolvimento rápidos e capacidade de criar aplicações de alta performance.

3.2.4.6 Comparação

Como é possível observar na tabela (3.1) a tecnologia mais adequada para a aplicação a desenvolver seria um *framework* focado em desenvolvimento web, com arquitetura baseada em componentes como o Vue.js, React ou Angular, pois promovem a reutilização de código, algo que irá acelerar o processo de desenvolvimento e são as únicas tecnologias que suportam todas as bibliotecas identificadas anteriormente. Das três *frameworks* o Vue.js é o que se destaca pois é tem uma curva de aprendizagem mais baixa e contém um sistema de gestão do estado global da aplicação nativo ao *framework*, que é mais intuitivo de usar pois assemelha-se ao usado em Javascript e HTML nativo.

Bootstrap	Vue.js	React	Angular	Flutter
JavaScript	JavaScript/TypeScript	JavaScript/TypeScript	TypeScript	Dart
HTML,CSS	HTML Templates,JSX e CSS	JSX e CSS	HTML Templates e CSS	Flutter Widgets
Aprendizagem fácil	Aprendizagem moderada	Aprendizagem difícil		Aprendizagem difícil
Ocupa muito pouco espaço	Ocupa algum espaço	Ocupa algum espaço	Ocupa muito espaço	Ocupa muito espaço
Desenvolvimento web	Desenvolvimento web	Desenvolvimento web	Desenvolvimento web	Desenvolvimento web e móvel
DOM real	DOM virtual	DOM virtual	DOM real	DOM virtual
Utiliza variáveis globais e armazenamento local (<i>local storage/session storage</i>) para gerir o estado global da aplicação.	Implementa o Vuex para gestão de estado, centralizando dados compartilhados em uma <i>store</i> global.	Adota o Redux para uma solução mais robusta de gestão do estado global da aplicação.	Adota o NgRX (baseado em Redux) para uma solução mais robusta de gestão do estado global da aplicação.	Utiliza o Provider para gerir a árvore de widgets e implementa o BLoC Pattern para um fluxo de dados reativo e organizado.
É compatível com JSCharting	Suporta JSCharting	Suporta JSCharting	Suporta JSCharting	Não possui uma integração nativa com JSCharting
É compatível com Chart.js	Suporta Chart.js	Suporta Chart.js	Suporta Chart.js	Não possui uma integração nativa com Chart.js
Pode ser integrado com Mapbox, mas pode requerer pacotes adicionais	Suporta Mapbox	Suporta Mapbox	Suporta Mapbox	Suporta Mapbox
Pode ser integrado com OpenLayers, mas pode requerer pacotes adicionais	Suporta OpenLayers	Suporta OpenLayers	Suporta OpenLayers	Não há suporte nativo para OpenLayers.
Suporta Leaflet	Suporta Leaflet	Suporta Leaflet	Suporta Leaflet	Não há suporte nativo para Leaflet.

Tabela 3.1: Comparação entre Bootstrap, Vue.js, React, Angular e Flutter.

SOLUÇÃO PARA GESTÃO DO SISTEMA FITOAGRO

Neste capítulo é apresentada uma visão geral do estado da solução, serão referidos inicialmente os principais requisitos que levaram à sua concepção [4.1](#), de seguida são listadas e descritas todas as funcionalidades implementadas da aplicação web [4.2](#).

4.1 Requisitos

No início da realização desta dissertação, foi necessário efetuar um estudo completo do sistema existente, pois a aplicação a desenvolver deveria ser capaz de comunicar com as diferentes componentes do sistema, de modo a que fosse possível fazer a gestão das mesmas, para tal foram identificados os requisitos no (apêndice [A](#)). Sendo identificados como principais os seguintes requisitos:

- **Registo, gestão e autenticação de utilizadores**

A aplicação deveria permitir o registo de novos utilizadores, os quais podem ser de diferentes tipos, conforme o seu papel no sistema. Os utilizadores deveriam ser capazes de autenticar-se perante o sistema para que possam ter acesso ao mesmo e aos seus serviços. Deveria ser possível também a gestão dos utilizadores, ou seja, a possibilidade de alterar os seus dados e ativar ou desativar, as contas dos mesmos.

- **Registo e gestão de estações meteorológicas**

Deveria ser possível registar novas estações meteorológicas ou criar estação meteorológicas virtuais, ou seja, combinar informação de várias estações e definir como uma única estação. Deveria ser possível visualizar a informação das estações meteorológicas, como por exemplo, a localização, a informação meteorológica que recolhem, fornecedor, entre outros. Deveria ser possível ainda a gestão das estações meteorológicas, ou seja, a possibilidade de alterar os seus dados ou removê-la do sistema.

- **Registo e gestão de parcelas**

Deveria ser possível registar novas parcelas facilmente através da interface, e remover existentes. Deveria ser possível visualizar, editar e remover toda a informação associada a uma parcela.

- **Registo e gestão de períodos**

Deveria ser possível registar novos períodos facilmente através da interface e remover existentes. Adicionalmente deveria ser possível visualizar e editar os detalhes de um período.

- **Registo e gestão de organizações**

Deveria ser possível registar novas organizações facilmente através da interface e remover existentes. Adicionalmente deveria ser possível visualizar e editar os detalhes de uma organização e ter a capacidade de gerir dos membros da mesma.

- **Verificação de visitas**

Deveria ser possível verificar a informação recolhida durante as visitas e fazer alterações caso seja detetado um erro.

- **Gestão de protocolos**

Deveria ser possível alterar protocolos de recolha de informação relativo a pragas já existentes e criar novos mais simples.

4.2 Funcionalidades da Aplicação Web FitOps

Nesta secção serão detalhadas as várias funcionalidades da aplicação web FitOps, e como estas ajudam a cumprir os requisitos identificados na secção 4.1. Esta secção está dividida em várias subsecções, cada uma dedicada a uma componente principal do sistema FitoAgro.

4.2.1 Utilizadores

Na aplicação web existem vários tipos de contas de utilizadores, cada tipo com diferentes permissões e funcionalidades. Sendo estes:

- **Administrador:** É como o nome indica, responsável pela administração e gestão da aplicação.
- **Organização:** É responsável pela gestão da organização a que pertence.
- **Produtor:** É responsável pela gestão das parcelas do qual é proprietário.
- **Agente de Campo:** É responsável pela recolha de informação no campo.

4.2.1.1 Registo

É possível registar todos os tipos de utilizadores através da interface da aplicação, exceto o administrador, pois devido à sua natureza, este deve ser registado diretamente na base de dados.

O processo de registo está dividido em três etapas (figura 4.1).

1. **Tipo de Conta e Organização:** É selecionado o tipo de conta a criar e a organização a que está afiliada.
2. **Dados Pessoais:** São introduzidos os dados pessoais do utilizador.

3. Dados da Conta: É introduzido o nome de utilizador e a senha de acesso.

Após a criação da conta é necessário ativar a mesma para que o utilizador possa aceder ao sistema, este processo é diferente conforme o tipo de utilizador, no caso de uma organização a ativação tem de ser efetuada por um administrador, enquanto para os restantes tipos de utilizadores é lhes enviado um email de confirmação semelhante ao da figura 4.2.

O diagrama ilustra o processo de registo de utilizadores em três etapas principais:

- Tipo de Conta e Organização:** O utilizador seleciona o tipo de conta e a organização. Há um link "Já tem conta" e um botão "PRÓXIMO".
- Dados Pessoais:** O utilizador preenche campos obrigatórios: Nome Pessoal, Telemóvel, Gmail, Cidade e Morada. Há um link "Já tem conta" e botões "ANTERIOR" e "PRÓXIMO".
- Dados de Conta:** O utilizador cria a conta definindo o Nome de Utilizador, a Senha e a Confirmação da Senha. Há um link "Já tem conta", um botão "ANTERIOR" e um botão "ADICIONAR".

Figura 4.1: Registo de Utilizadores



Obrigado por se registar.

Por favor, clique [aqui](#) para ativar a sua conta.

Se você não se registou, por favor ignore este email.

Figura 4.2: Email de ativação de conta

4.2.1.2 Autenticação

Todos os utilizadores tem de se autenticar perante o sistema para poderem aceder ao mesmo. Este processo é feito através de um formulário de autenticação (figura 4.3), onde é introduzido o *nome de utilizador* e a *senha*. Os utilizadores só se podem autenticar, se a sua conta estiver aceite no sistema, não bloqueada e ativa.

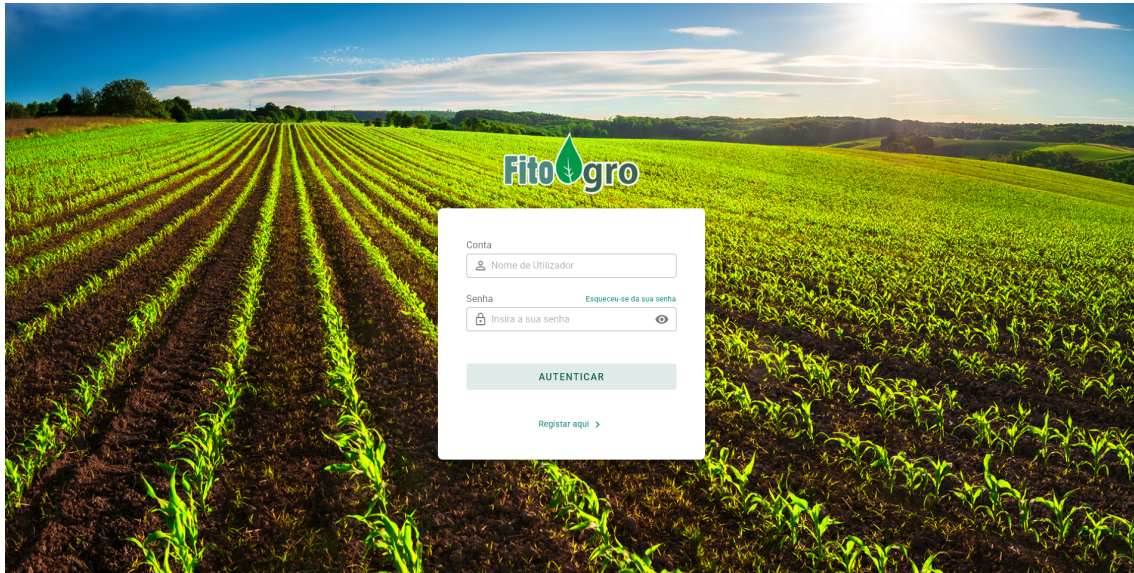


Figura 4.3: Autenticação de Utilizadores

4.2.1.3 Recuperação de senha

Caso um utilizador se tenha esquecido da sua senha de acesso, a aplicação permite que este recupere a conta desde que tenha acesso ao email associado à mesma. Como é possível ver na figura 4.4. Primeiramente é necessário inserir o email num formulário, após a submissão do mesmo, o utilizador recebe um email com um link para redefinir a senha, que o irá redirecionar para uma página onde será possível efetuar a alteração da senha de acesso. Após este processo ser concluído, o utilizador poderá aceder à aplicação com a nova senha de acesso.



Figura 4.4: Recuperação de conta

4.2.1.4 Edição da própria conta

Cada utilizador pode editar a sua própria conta através da aba apresentada na figura 4.5 que está disponível no canto superior direito da tela. Sendo possível alterar os seus dados

peçoais como nome, morada e cidade. Pode alterar também a senha de acesso. E por fim cada utilizador tem a possibilidade de sair da aplicação.

(a) Menu de gestão da conta

(b) Formulário para alteração de senha

(c) Formulário para editar detalhes da conta

Figura 4.5: Gestão da conta

4.2.1.5 Consulta

Apenas os administradores e organizações podem consultar os dados de outros utilizadores. Os administradores podem consultar todos os utilizadores, enquanto as organizações apenas podem consultar os utilizadores associados à sua organização.

A consulta destes dados é feita através de uma tabela (figura 4.6), onde o tipo de informação apresentada varia conforme o tipo do utilizador autenticado, ou seja, para os administradores é apresentada informação pertinente para administração do sistema, como por exemplo, o estado da conta e a organização a que pertence, enquanto para as organizações é apresentada informação pertinente para a gestão da organização, como por exemplo, o cargo e se este está aceite na organização.

Sendo também possível filtrar a informação apresentada na tabela, através de um campo de pesquisa, onde é possível pesquisar por nome de utilizador, nome, email, telefone, organização (**administrador**) e tipo de utilizador (**administrador**) ou tipo de membro (**organização**).

4.2. FUNCIONALIDADES DA APLICAÇÃO WEB FITOPS

Caso se pretenda visualizar mais detalhes de um utilizador, é possível clicar no botão azul localizado no lado esquerdo da tabela, que irá abrir um painel com os detalhes do utilizador, semelhante ao da figura 4.7.

Filtro						
Organização	Tipo de Conta	Conta	Nome	Telefone	Email	

Organização	Tipo	Conta	Nome	Estado				
		clucas	Carolina Lucas					
		Isilva	Luísa Silva					
		deptecnicoapas	Dep. Técnico APAS					
		Org João	João Silva					
		Prod Pedro	Pedro Santos					
		Agente Miguel	Miguel Baião					

(a) Página de Administrador

Filtro						
Tipo de Membro	Conta	Nome	Telefone	Email		

Tipo	Conta	Nome	Estado			
		Prod Pedro	Pedro Santos			
		Agente Miguel	Miguel Baião			

(b) Página de Organização

Figura 4.6: Página de utilizadores

Detalhes da Conta Prod Pedro

Nome: Pedro Santos

Morada: Rua Luís Gonzaga do Nascimento 19-1

Cidade: Beja

Email: ps@email.com

Telefone: +351 912331412

Aceite: ☒ Bloqueado: ☐ Ativo: ☒

Figura 4.7: Detalhes de um utilizador

4.2.1.6 Gestão de outros utilizadores

A funcionalidade de gestão de outros utilizadores é apenas disponível para o **administradores** e **organizações**.

Os **administradores** podem editar e remover do sistema as contas de todos os utilizadores, com exceção de outros administradores. Ao editar um utilizador, é possível alterar o nome, morada e cidade, adicionalmente é possível alterar o estado da conta, ou seja, se esta, está aceite no sistema, bloqueada ou ativa (figura 4.8).

Quando uma conta de um **agente de campo** é desativada este ficará bloqueado de aceder a aplicação móvel desenvolvida no âmbito da Dissertação do M. Correia [10], só voltando a ter acesso se a conta for reativada.

Editar Conta Prod Pedro

Nome: Pedro Santos

Morada: Rua Luís Gonzaga do Nascimento 19-1

Cidade: Beja

Aceite: ☒ Bloqueado: ☐ Ativo: ☒

⌕ ⌕

Figura 4.8: Editar utilizador

As **organizações** podem adicionar e remover membros da sua organização, através da interface apresentada na figura 4.9, membros estes que são **produtores** ou **agentes de campo**, sendo necessário, tanto no adicionar como na remoção, a introdução das datas oficiais de início e fim da associação do utilizador à organização, datas estas que podem ser importantes para efeitos de rastreabilidade. Para além destas funcionalidades, também é possível mudar o estado dos membros dentro da organização, para aceite ou rejeitado.

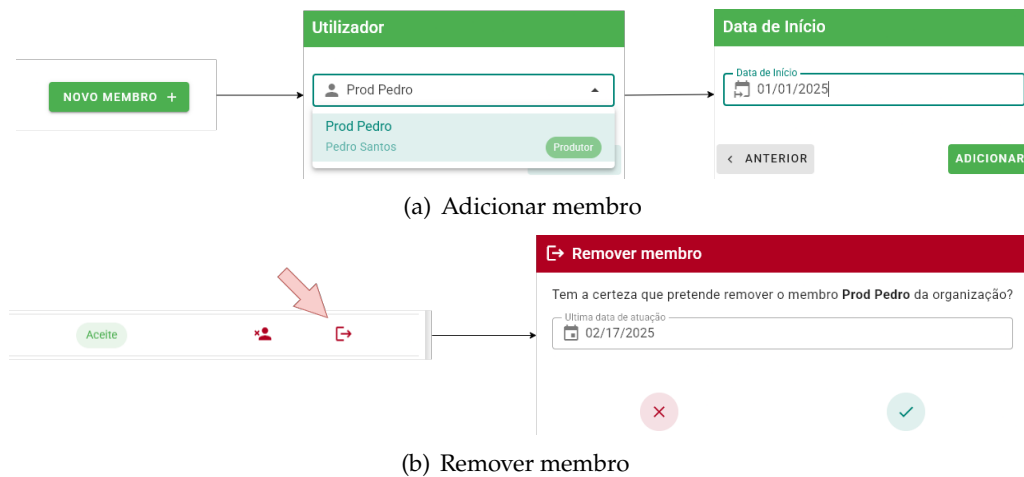


Figura 4.9: Gestão de membros da organização

4.2.2 Estações meteorológicas

A aplicação suporta a gestão de estações meteorológicas, sendo possível registar, consultar, editar e remover estações meteorológicas dos fornecedores APAS, PESSL, IPMA, WiseCrop e TerraPro.

4.2.2.1 Contas de Fornecedores

As contas de fornecedores são entidades que não são utilizadas para autenticação no nosso sistema, mas sim no sistema do fornecedor. Estas contas contém as credenciais de acesso de uma conta do sistema do fornecedor, tornando assim possível aceder às estações meteorológicas dessa conta.

Estas contas podem ser geridas através de uma página própria (figura 4.10) por **administradores**, sendo que estes podem visualizar detalhes, filtrar, criar, editar e remover contas. Atualmente o sistema suporta contas dos fornecedores PESSL, WiseCrop e TerraPro, sendo que para os fornecedores APAS e IPMA não é necessário uma conta de fornecedor para aceder às estações.

O processo de registo pode ser efetuado de maneira diferente que dependendo do fornecedor como é possível ver na figura 4.11. Sendo que este divide em três etapas:

1. **Fornecedor:** Nesta etapa é selecionado o fornecedor das estações meteorológicas
2. **Credenciais de Acesso:** São introduzidas as credenciais de acesso ao sistema do fornecedor.
3. **Detalhes:** São introduzidos os detalhes da conta de fornecedor.

Adicionalmente é possível também visualizar todas as estações da conta e se estas já estão registadas no nosso sistema, como é possível ver na figura 4.12.

CAPÍTULO 4. SOLUÇÃO PARA GESTÃO DO SISTEMA FITOAGRO

Filtro

Fornecedor

Conta

Nome

NOVA CONTA +

	Conta	Nome	Telefone	Email	Endereço	Cidade	Fornecedor	Ações
	cothn1	Goncalo Lopes	960000000	onnoschaap@aquagri.com	Estrada Leiria sn, 2460 Alcobaca	Alcobaca	PESSL	 
	Carlos	Carlos Damásio	961111111	cd@fct.unl.pt			WiseCrop	 
	Cothn	Cothn TerraPro	982222222	cothn@terrapro.com			TerraPro	 
	AM	André Malafala	921311992	a.malafala@campus.fct.unl.pt			APAS	 
	frubaca	Nuno Franco	960000099	nunohefranco@gmail.com	Acipreste, 2460 Alcobaca	Alcobaca	PESSL	 

Figura 4.10: Página de contas de fornecedores

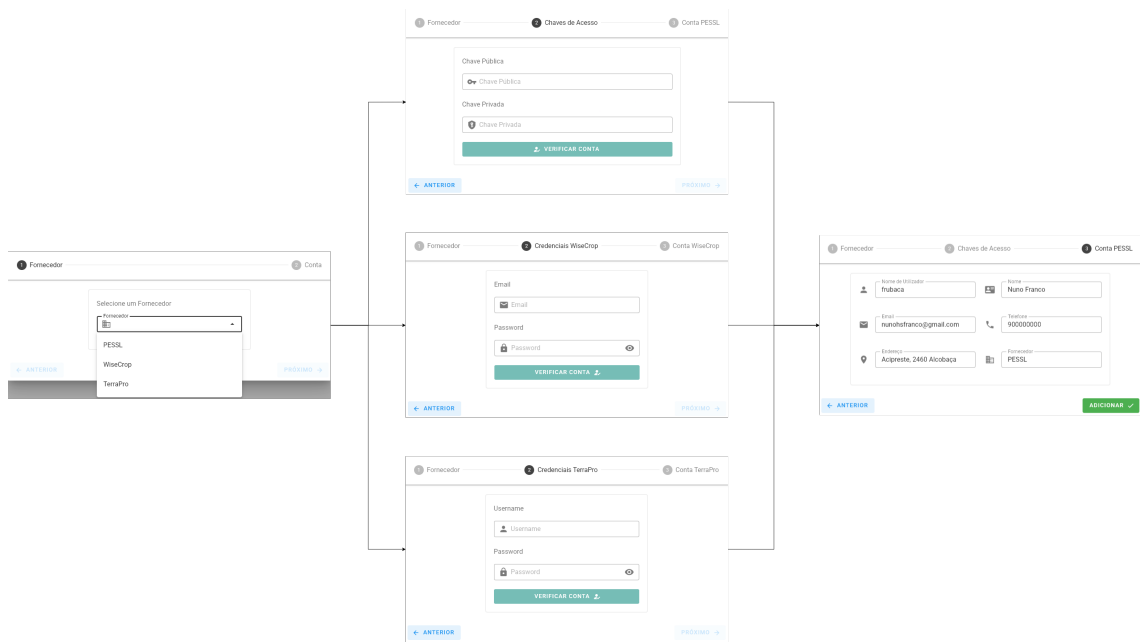


Figura 4.11: Registo de conta de fornecedor

frubaca

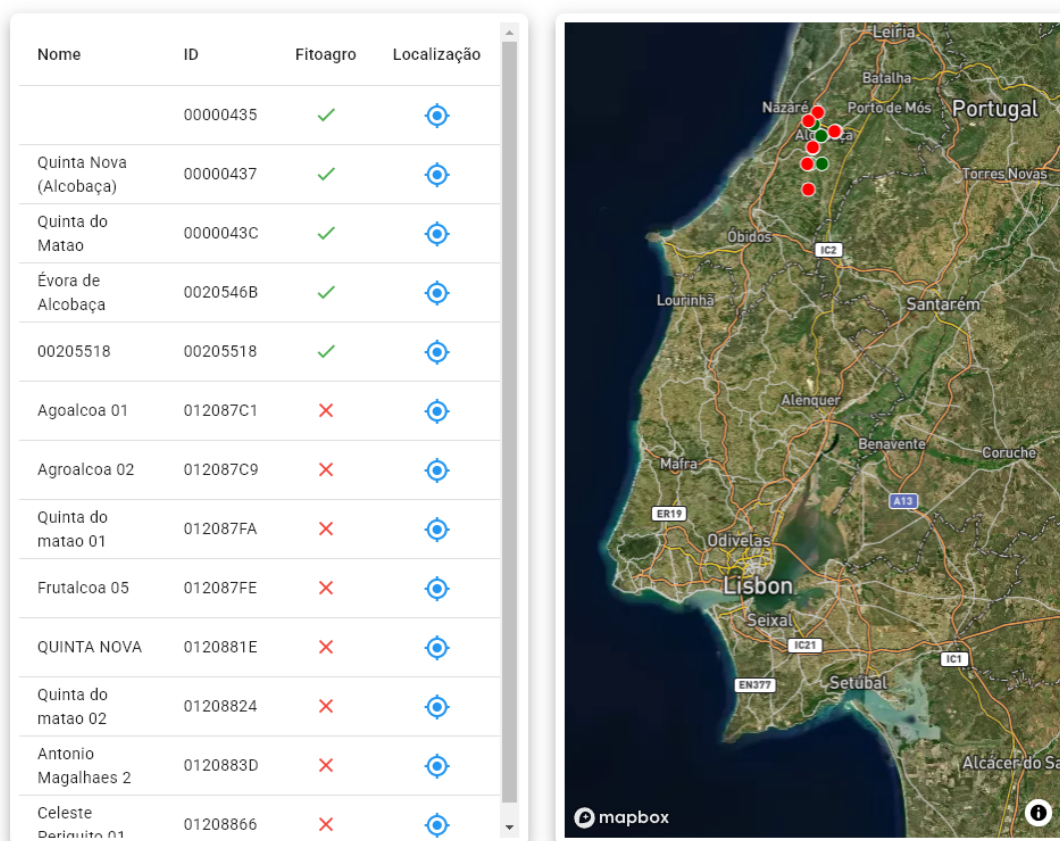


Figura 4.12: Estações de uma conta de fornecedor

4.2.2.2 Registo de estações meteorológicas

O processo de registo de uma estação meteorológica só pode ser efetuado por **administradores** e está dividido em três etapas (figura 4.13).

1. Fornecedor

Nesta etapa é selecionado o fornecedor das estações meteorológicas e caso seja necessário também é selecionada a conta de fornecedor.

2. Estações disponíveis

Nesta etapa são apresentadas as estações meteorológicas disponíveis da conta selecionada, para que seja possível selecionar a estação a registar, é nesta etapa onde também é selecionada o tipo de informação a recolher.

3. Dados da Estação

Esta etapa final tem dois propósitos, o primeiro é a inserção do nome da estação meteorológica dentro do sistema FitoAgro, e o segundo apresentar um resumo do processo até ao momento para que o utilizador possa confirmar que não cometeu nenhum erro durante o processo.

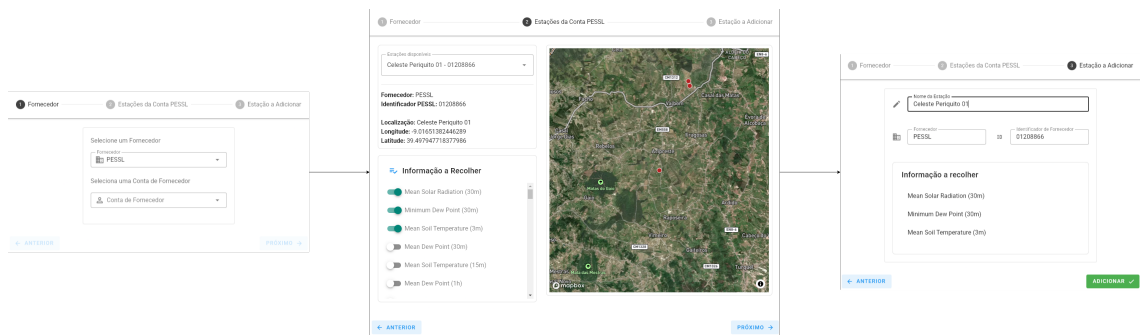


Figura 4.13: Registo de estações meteorológicas

4.2.2.3 Consulta

Através da página de estações meteorológicas da aplicação, todos os utilizadores podem consultar todas as estações meteorológicas registadas no sistema, sendo possível através de dois componentes, uma tabela e um mapa (figura 4.14).

Se pretender localizar uma estação no mapa, precisa de clicar botão azul localizado na coluna "Localização" da tabela, adicionalmente é possível ainda consultar as coordenadas ao passar com o cursor por cima do mesmo.

Estas estações podem ser filtradas através de um campo de pesquisa, onde é possível pesquisar por nome, fornecedor e identificador do fornecedor.

Todos os utilizadores podem visualizar os detalhes das estações meteorológicas, através do botão azul localizado na coluna mais à esquerda da tabela. Após clicar no botão, é apresentado um painel com o nome, fornecedor, o identificador que o fornecedor usa para identificar a estação e o tipo de informação que está a recolher atualmente.

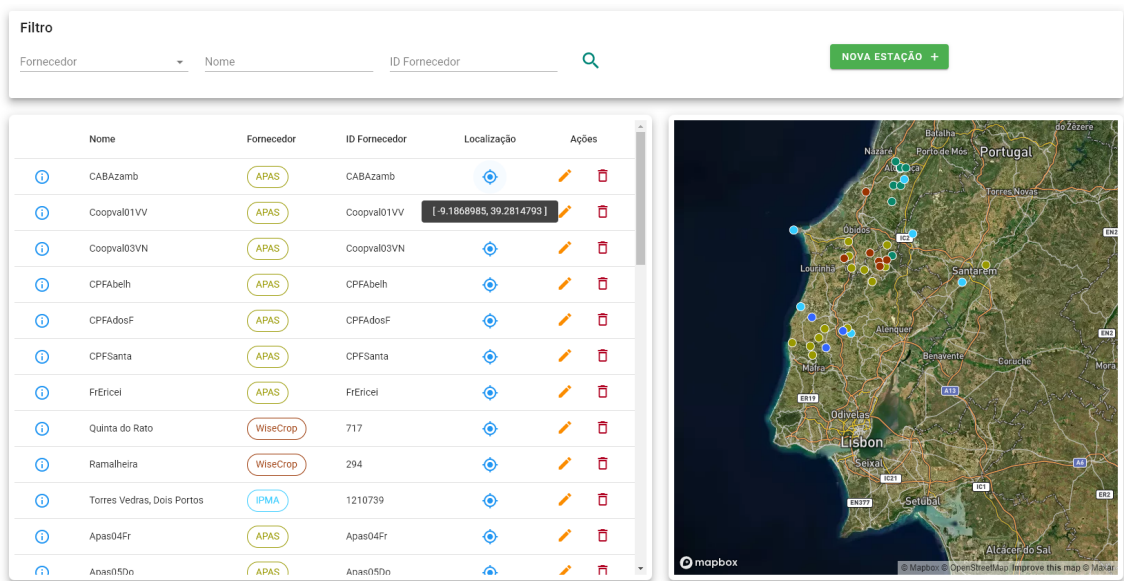


Figura 4.14: Página estações meteorológicas

4.2.2.4 Visualização de dados recolhidos

Para além da informação relativa à estação também é possível visualizar os dados recolhidos por esta. É possível aceder a estes dados através de um botão azul no painel da estação meteorológica junto ao tipo de informação a recolher. Após clicar no botão, é apresentado outro painel onde os dados dos últimos sete dias podem ser visualizados em forma de gráficos de linhas (figura 4.15), podendo ser possível visualizar até 3 tipos diferentes de informação meteorológica ao mesmo tempo, algo útil para efeitos de comparação.



Figura 4.15: Visualização de dados recolhidos

4.2.2.5 Edição e Remoção

Apenas os **administradores** podem editar e remover estações meteorológicas, Sendo possível editar o nome da estação meteorológica e o tipo de informação que esta recolhe (figura 4.16).

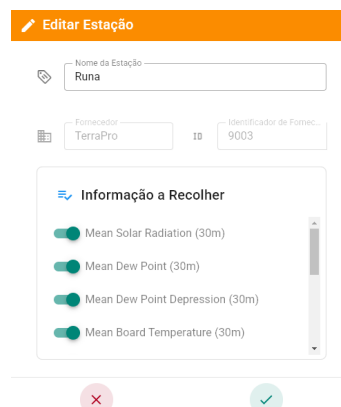


Figura 4.16: Editar estação meteorológica

4.2.3 Parcelas

As parcelas são regiões de terreno, onde são cultivadas culturas de pomóideas, estes elementos são de extrema importância pois são um dos elementos base do sistema, onde estão associados vários outros elementos, como por exemplo, elementos de interesse, estações meteorológicas e períodos, que por sua vez estão associados a outros elementos.

4.2.3.1 Registo

Através da aplicação apenas **produtores** pertencentes a uma organização e **organizações** podem registar novas parcelas, sendo este processo dividido em três etapas, como é possível ver na figura 4.17.

1. Região

Nesta etapa é selecionado o tipo da parcela e é introduzida a informação geográfica da mesma, sendo esta introduzida através de um ficheiro GeoJSON [42] com o formato representado em 4.1.

Listagem 4.1: Exemplo de ficheiro GeoJSON

```
1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {
7         "name": "Parcela teste",
8         "code": "RT",
9         "description": "Descricao da parcela"
10      },
11      "geometry": {
12        "coordinates": [
13          [ [-7.86, 38.00], [-7.86, 38.00], ... ]
14        ],
15        "type": "Polygon"
16      }
17    }
18  ]
19 }
```

2. Detalhes

Nesta etapa são introduzidos os detalhes da parcela, como o seu nome, código, cultura, variedade da cultura, localidade e descrição. A localidade é calculada automaticamente através da localização da parcela.

3. Períodos e Inimigos

Nesta etapa final é possível associar períodos à parcela e indicar a estudar na parcela.

The figure displays three sequential screenshots of the FITOPS web application interface for parcel registration:

- Região:** The first panel shows a map view with a parcel highlighted in blue. It includes a dropdown for 'Tipo de Região*' (set to 'Parcela Agrícola') and a file upload area for 'Ficheiro GeoJSON'.
- Detalhes:** The second panel is a form for entering parcel details. It contains fields for 'Nome*', 'Código*', 'Cultura*', 'Variedade*', 'Localidade*' (set to 'Beja'), and a text area for 'Descrição'.
- Períodos e Inimigos:** The third panel shows fields for 'Período(s)*' (set to 'Campanha Prática 2024/2025') and 'Inimigo(s)*' (set to 'Cecidomia'). It includes navigation buttons for '< ANTERIOR' and a green 'ADICIONAR' button.

Figura 4.17: Registo de parcelas

4.2.3.2 Consulta, Edição e Remoção

Dependendo do tipo de utilizador as permissões de consulta, edição e remoção de parcelas são diferentes.

- **Agentes de Campo** apenas tem permissão para consultar as parcelas que estão associados, e não podem editar ou remover as mesmas.
- **Produtores** podem consultar e editar as parcelas de que são proprietários.
- **Organizações** podem consultar, editar e remover todas as parcelas registadas pela organização ou produtores da organização.
- **Administradores** podem consultar, editar e remover todas as parcelas registadas no sistema.

As parcelas que o utilizador têm acesso são listadas numa tabela, podendo ser filtradas através de vários campos de pesquisa, como por exemplo, nome, código, localidade, e caso o utilizador autenticado seja um **administrador** também é possível pesquisar por organização, como é possível ver na figura 4.18.

Para consultar mais detalhes de uma parcela, é possível clicar no botão azul localizado na coluna mais à esquerda da tabela, que irá abrir um painel com os detalhes da parcela semelhante ao da figura 4.19.

Ao editar uma parcela é possível alterar a cultura, a respetiva variedade da cultura e a descrição, através do formulário apresentado na figura 4.20.

Filtro

Código da Parcela

Nome da Parcela

Tipo de Região

Localidade

Cultura

Organização

Código da Parcela	Nome da Parcela	Tipo	Localidade	Cultura	Organização	
VF	Vale da Fonte	Parcela Agrícola	Reguengo Grande		GRANFER	
VL	Vale Leite	Parcela Agrícola	Reguengo Grande		CPF	
BRC	Barrocalvo	Parcela Agrícola	Bombarral		ECOFRUTAS	
TER	Termas	Parcela Agrícola	Alcobaça		FRUBAÇA	
QR	Quinta do Rato	Parcela Agrícola	Vale Maceira		GRANFER	
SL	São Lourenço	Parcela Agrícola	Ericeira		FRUTOESTE	
BB	Barreiras da Bica	Parcela Agrícola	A-dos-Francos		CPF	
BL	Boiça do Louro	Parcela Agrícola	Boiça do Louro		COOPVAL	
QB	Quinta do Brejo	Parcela Agrícola	Sobrena		FRUTUS	
ENC	Encarnação	Parcela Agrícola	Encarnação		FRUTOESTE	

Figura 4.18: Página de parcelas

Detalhes da Parcela Vale Leite

PARCELA

EOIS

PERÍODOS

UTILIZADORES

VISITAS

Informação Geral

Detalhes

Código da Parcela: VL

Nome da Parcela: Vale Leite

Tipo da Parcela: Parcela Agrícola

Localidade: Reguengo Grande

Cultura: Sem cultura

Organização: CPF

SEM DESCRIÇÃO

Meteorologia

Estação meteorológica

Estação virtual

CABAzamb

Vale Leite

Figura 4.19: Detalhes de uma parcela

Editar Parcela

Cultura

Macieira

Variedade

Royal Gala

Descrição

Figura 4.20: Editar parcela

46

4.2.3.3 Estações Meteorológicas Associadas

Os utilizadores com permissões de edição podem associar a uma parcela dois tipos de estações meteorológicas, uma física e uma virtual. A estação meteorológica física consiste num dispositivo físico de um fornecedor, que está instalado no local, enquanto a estação meteorológica virtual é criada através da combinação da informação, das N estações físicas mais perto da parcela. Ao contrário das estações físicas, os utilizadores com permissões de edição na parcela podem alterar os dados que a estação virtual recolhe, pois esta estação é criada apenas para a parcela em questão.

A adição ou edição das estações pode ser efetuada na área da meteorologia da parcela, onde é possível adicionar ou alterar a estação meteorológica física ou alterar o tipo de informação que a estação virtual recolhe (figura 4.21).



(a) Alterar estação meteorológica física

(b) Editar estação meteorológica virtual

Figura 4.21: Alterar e editar estações meteorológicas da parcela

4.2.3.4 Elementos de Interesse (EOIs)

Os elementos de interesse podem ser elementos que são monitorizados numa parcela como por exemplo, armadilhas e árvores ou podem ser estruturas fixas como por exemplo, entradas e lagos.

Para tal criou-se uma aba na página da parcela dedicada aos elementos de interesse, como é possível ver na figura 4.22.

Estes elementos podem ser registados, consultados, editados e removidos por utilizadores com permissões para tal, as permissões são as mesmas que as das parcelas, ou seja, se o utilizador pode visualizar, editar ou remover a parcela, então pode efetuar as mesmas operações, nos elementos de interesse. Adicionalmente estes podem ser filtrados por nome e período, quando consultados.

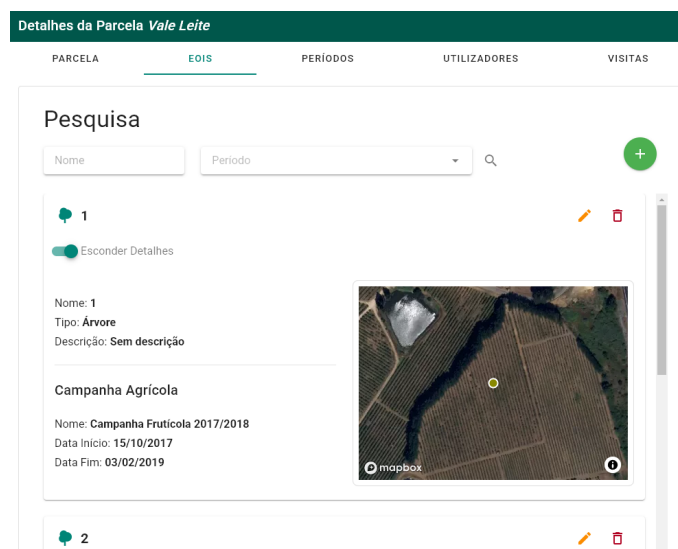


Figura 4.22: Página de elementos de interesse

É possível editar toda a informação do elemento de interesse, exceto a sua localização, através do seguinte formulário 4.23.

Figura 4.23: Editar elemento de interesse

Para criar um novo elemento de interesse existe um processo de três etapas, como é possível ver na figura 4.24, sendo estas as etapas:

1. Período

Caso seja um elemento de interesse que só está presente num determinado período, é necessário seleccionar o respetivo período, caso contrário existe a opção "Permanente" que é usada quando se pretende registar uma estrutura fixa.

2. Tipo e Localização

Nesta etapa é escolhido o tipo de elemento de interesse dos vários tipos presentes no sistema e é introduzida a localização de maneira semelhante à das parcelas, ou seja, através de um ficheiro GeoJSON com um formato semelhante ao referido em 4.1.

3. Dados adicionais

Nesta etapa final são introduzidos os detalhes do elemento de interesse sendo estes, o seu nome e descrição,

O diagrama ilustra o processo de criação de um elemento de interesse (EOI) em três etapas principais:

- Período do EOI:** Seleção de um período da lista (ex: Campanha Frutícola 2017/2018).
- Tipo e Localização do EOI:** Escolha do tipo (Árvore) e localização no mapa.
- Dados adicionais do EOI:** Inserção do nome e descrição do EOI.

Botões de navegação: < ANTERIOR, PRÓXIMO >, e um botão ADICIONAR ✓.

Figura 4.24: Criar elemento de interesse

4.2.3.5 Períodos associados

Os períodos associados são os períodos que a parcela está associada, como por exemplo, campanhas agrícolas. É possível através da interface da aplicação web visualizar, associar e desassociar períodos de uma parcela se o utilizador tiver permissões para tal, permissões estas que são semelhantes às das parcelas.

Detalhes da Parcela <i>Vale Leite</i>																																		
PARCELA	EOIS	PERÍODOS	UTILIZADORES	VISITAS																														
Períodos associados <table> <tr> <th>Nome</th><th>Tipo</th><th>Data de Início</th><th>Data de Fim</th><th></th></tr> <tr> <td>Campanha Frutícola 2017/2018</td><td>Campanha Agrícola</td><td>15/10/2017</td><td>03/02/2019</td><td> </td></tr> <tr> <td>Campanha Frutícola 2018/2019</td><td>Campanha Agrícola</td><td>04/02/2019</td><td>02/02/2020</td><td> </td></tr> <tr> <td>Campanha Frutícola 2019/2020</td><td>Campanha Agrícola</td><td>03/02/2020</td><td>31/01/2021</td><td> </td></tr> <tr> <td>Campanha Frutícola 2020/2021</td><td>Campanha Agrícola</td><td>01/02/2021</td><td>31/01/2022</td><td> </td></tr> <tr> <td>Campanha Frutícola 2021/2022</td><td>Campanha Agrícola</td><td>01/02/2022</td><td>31/01/2023</td><td> </td></tr> </table>					Nome	Tipo	Data de Início	Data de Fim		Campanha Frutícola 2017/2018	Campanha Agrícola	15/10/2017	03/02/2019		Campanha Frutícola 2018/2019	Campanha Agrícola	04/02/2019	02/02/2020		Campanha Frutícola 2019/2020	Campanha Agrícola	03/02/2020	31/01/2021		Campanha Frutícola 2020/2021	Campanha Agrícola	01/02/2021	31/01/2022		Campanha Frutícola 2021/2022	Campanha Agrícola	01/02/2022	31/01/2023	
Nome	Tipo	Data de Início	Data de Fim																															
Campanha Frutícola 2017/2018	Campanha Agrícola	15/10/2017	03/02/2019																															
Campanha Frutícola 2018/2019	Campanha Agrícola	04/02/2019	02/02/2020																															
Campanha Frutícola 2019/2020	Campanha Agrícola	03/02/2020	31/01/2021																															
Campanha Frutícola 2020/2021	Campanha Agrícola	01/02/2021	31/01/2022																															
Campanha Frutícola 2021/2022	Campanha Agrícola	01/02/2022	31/01/2023																															

Figura 4.25: Página de períodos associados

O processo de associar um período é bastante simples e está dividido em duas etapas (figura 4.26), sendo estas as etapas:

1. Tipo de Período

Nesta etapa é selecionado o tipo de período a associar dos presentes no sistema.

2. Período

Nesta etapa é selecionado o período a associar, sendo este escolhido de uma lista de períodos disponíveis cujo o tipo corresponde ao previamente selecionado.

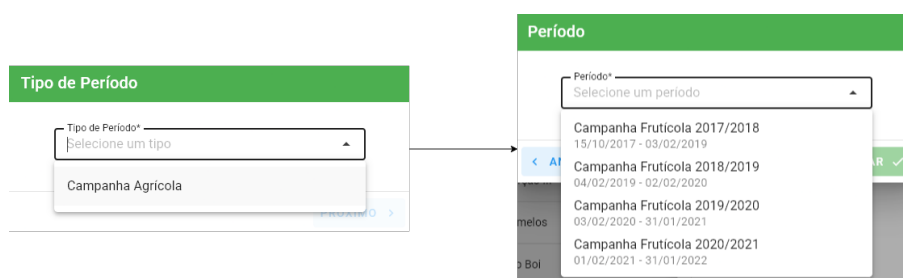


Figura 4.26: Associar período a parcela

Adicionalmente é possível consultar e alterar os inimigos estudados na parcela durante o um certo período (figura 4.27). Esta informação é importante para efeitos de rastreabilidade e para a tomada de decisões futuras.



(a) Editar inimigos estudados

(b) Adicionar inimigos estudados

Figura 4.27: Inimigos estudados durante um período

4.2.3.6 Utilizadores associados

As parcelas podem ter vários utilizadores associados como é possível ver na figura 4.28, como por exemplo, o técnicos de campo que trabalham na parcela ou produtores que são os proprietários da parcela. Apenas as **organizações** podem associar e desassociar utilizadores de uma parcela.

PARCELA	EOIS	PERÍODOS	UTILIZADORES	VISITAS
Utilizadores Associados				
Tipo ↑	Conta	Nome	Data de Início	
Proprietário	Prod Pedro	Pedro Santos	01/02/2025	
Agente de Campo	Agente Miguel	Miguel Baião	01/02/2025	

Figura 4.28: Página de utilizadores associados

O processo de associar um utilizador é bastante simples e está dividido em três etapas, como é possível ver na figura 4.29, sendo estas as etapas:

1. Cargo

Nesta etapa é selecionado o cargo do utilizador a associar, dos presentes no sistema, podendo ser um agente de campo ou um proprietário. Esta informação é importante para que sistema possa filtrar os utilizadores disponíveis para associar.

2. Utilizador

Nesta etapa é selecionado o utilizador a associar, de uma lista de utilizadores disponíveis cujo o cargo corresponde ao previamente selecionado e que sejam pertencentes à organização.

3. Data de Início

Nesta etapa é introduzida a data de início, em que o utilizador começou a desempenhar o cargo na parcela. Achou-se importante introduzir esta informação, para efeitos de rastreabilidade.

Após a associação o utilizador começa a ter permissões para visualizar a parcela se for um agente de campo, ou editar a parcela se for um produtor.

O diagrama ilustra o processo de associação de um utilizador a uma parcela em três etapas sequenciais:

- Cargo do utilizador:** Uma interface com um campo de seleção rotulado 'Escolha o cargo do utilizador'. As opções visíveis são 'Proprietário' e 'Agente de Campo'.
- Utilizador:** Uma interface com um campo de seleção rotulado 'Produtor'. A opção selecionada é 'Prod Pedro'. Abaixo, há uma lista com 'Prod Pedro' e 'Pedro Santos'. Botões '< ANTERIOR' e 'PRÓXIMO >' estão visíveis.
- Data de Início:** Uma interface com um campo de data rotulado 'Data de Início' contendo '01/01/2025'. Abaixo, há botões '< ANTERIOR' e 'ADICIONAR ✓'.

Setas indicam a progressão entre as etapas.

Figura 4.29: Associar utilizador a parcela

É possível editar a data de início do utilizador associado e remover o utilizador associado, como é possível ver na figura 4.30. Na remoção é necessário introduzir a data em que o utilizador deixou de desempenhar o cargo na parcela.

Dois painéis de interface de utilizador são apresentados:

(a) Editar data de início de Pedro Santos: Um painel laranja com um ícone de lápis. Contém um campo de data rotulado 'Data de início' com o valor '02/01/2025'.

(b) Desassociar Produtor: Um painel vermelho com um ícone de lixeira. Contém o texto 'Tem a certeza que quer remover o Proprietário Pedro Santos?' e um campo de data rotulado 'Última data de atuação' com o valor '31/12/2025'.

Abaixo de cada painel, há dois círculos: um vermelho com um 'X' e um verde com um '✓', indicando o sucesso ou falha da ação.

Figura 4.30: Gestão de utilizadores associados

4.2.3.7 Visitas na parcela

Nesta secção é possível aceder a todas as visitas efetuadas na parcela (figura 4.31), sendo possível filtrar por:

- Data e Período
- Nível Económico de Ataque (NEA)
- Estado fenológico da cultura (EF)
- Período
- Se foi possível efetuar a visita
- Se existem dados georreferenciados do trajeto do agente, durante a visita
- Se existem fotos
- Se choveu
- Se ocorreu um tratamento

Esta informação é de extrema importância, para a gestão da parcela, pois permite ter um histórico de todas as visitas efetuadas e a informação recolhida durante as mesmas, a qual pode ser útil para a tomada de decisões futuras. Adicionalmente o utilizador tem a possibilidade de consultar a informação de cada visita e corrigir os dados recolhidos, caso seja necessário.

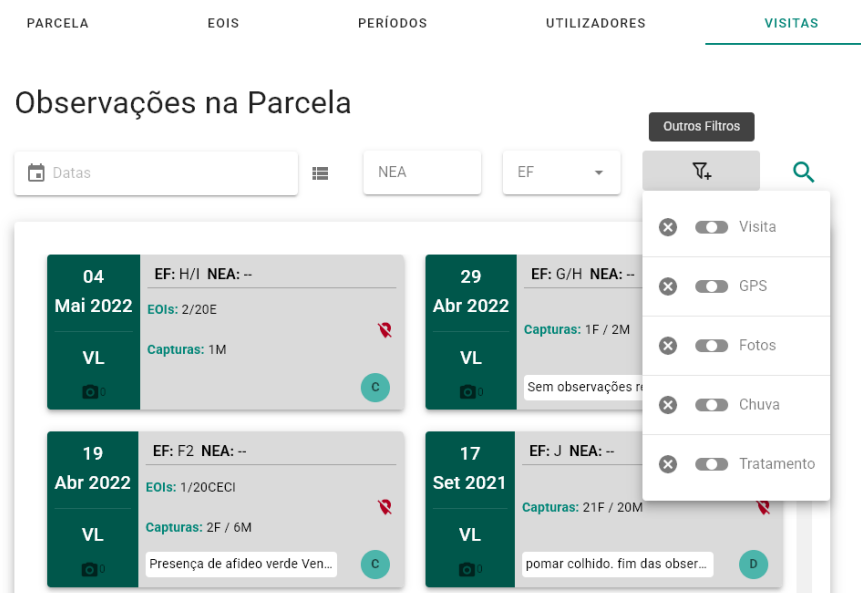


Figura 4.31: Página de visitas

4.2.4 Períodos

Os períodos são as épocas que são importantes, em algum aspeto para a gestão do sistema, como por exemplo, campanhas agrícolas.

4.2.4.1 Criação

A criação de um período é feita apenas por **administradores** e pode ser efetuada de duas maneiras, sendo a primeira um processo de duas etapas (figura 4.32):

1. Tipo de Período

Nesta etapa é selecionado o tipo de período a criar dos presentes no sistema.

2. Período

Nesta etapa é introduzido o nome do período, a sua data de início e fim e a descrição caso seja necessário.

O diagrama ilustra o processo de criação de períodos em duas etapas. A primeira etapa, intitulada 'Tipo', apresenta um menu suspenso 'Tipo de Período' com a opção 'Campanha Agrícola' selecionada. Uma seta indica a transição para a segunda etapa, intitulada 'Datas'. Nesta etapa, o utilizador preenche um formulário com os seguintes campos: 'Nome', 'Data de início', 'Data de fim' e 'Descrição'. Na base do formulário, encontram-se os botões de navegação '< ANTERIOR' e 'ADICIONAR >'.

Figura 4.32: Registo de períodos

Sendo a segunda opção para criação de períodos, um processo de duplicação, onde é possível duplicar um período já existente, copiando assim também todas as parcelas e elementos de interesse associados. Esta funcionalidade é bastante útil, pois evita de ter de se associar as parcelas e elementos de interesse, um a um novamente. Este processo é dividido em duas etapas (figura 4.33):

1. Detalhes

Nesta etapa aparece um formulário preenchido com as informações do período a copiar, para que seja possível alterar os detalhes necessário para o novo período.

2. Parcelas a manter

Nesta etapa são apresentadas as parcelas e o respetivo número de elementos de interesse da parcela que estão associadas ao período a copiar, para que se possa selecionar quais as parcelas e consequentemente os seus elementos de interesse a associar ao novo período.

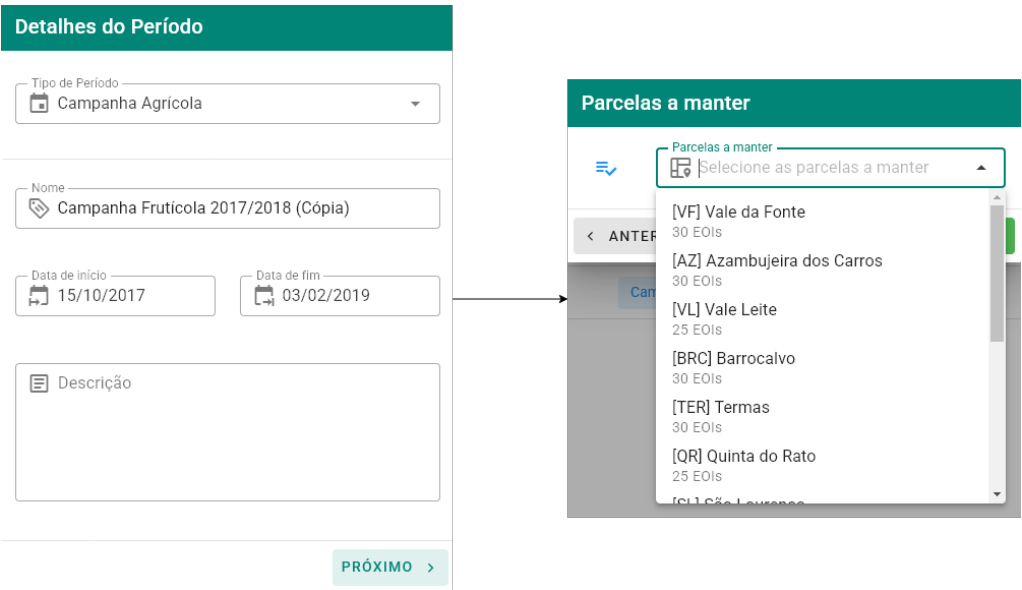


Figura 4.33: Duplicar período

4.2.4.2 Consulta

A consulta dos períodos é feita através de uma tabela, como é possível ver na figura 4.34, onde é apresentada os detalhes de cada período, como o nome, tipo, data de início e fim e descrição. Sendo esta informação disponível para todos os utilizadores autenticados. Esta tabela pode ainda ser filtrada, por nome e tipo do período.

Adicionalmente é possível consultar informação mais detalhada de um período (figura 4.35), clicando no botão azul localizado na coluna mais à esquerda da tabela, esta informação consiste nos detalhes do período e quais as parcelas do utilizador estão associadas ao período em questão, havendo ainda a possibilidade para **organizações, produtores e administradores** desassociar parcelas do período.

Filtro

Nome

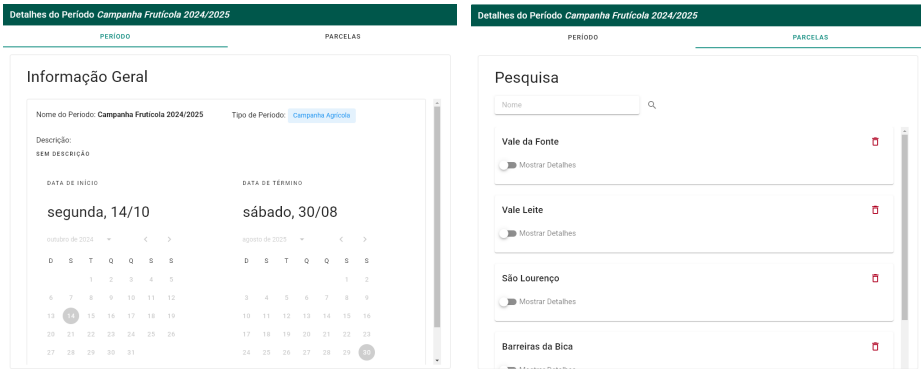
Tipo de Período

NOVO PERÍODO +

	Nome	Tipo	Data de Início	Data de Fim	Descrição	
ⓘ	Campanha Fruticola 2017/2018	Campanha Agrícola	15/10/2017	03/02/2019		   
ⓘ	Campanha Fruticola 2018/2019	Campanha Agrícola	04/02/2019	02/02/2020		   
ⓘ	Campanha Fruticola 2019/2020	Campanha Agrícola	03/02/2020	31/01/2021		   
ⓘ	Campanha Fruticola 2020/2021	Campanha Agrícola	01/02/2021	31/01/2022		   
ⓘ	Campanha Fruticola 2021/2022	Campanha Agrícola	01/02/2022	31/01/2023		   
ⓘ	Campanha Fruticola 2024/2025	Campanha Agrícola	14/10/2024	30/08/2025		   

Figura 4.34: Página de períodos

4.2. FUNCIONALIDADES DA APLICAÇÃO WEB FITOPS



(a) Detalhes gerais do período

(b) Parcelas associadas ao período

Figura 4.35: Detalhes de um período

4.2.4.3 Edição e Remoção

A edição e remoção de períodos só pode ser efetuada por **administradores**, sendo apenas possível para os restantes tipos de utilizadores consultar a sua informação. Em relação à edição do período é possível alterar toda a informação do período, isto é, o nome, o tipo, a data de início e fim e a descrição (figura 4.36).

The image shows a form for editing a period. At the top is an orange button with a pencil icon and the text 'Editar Período'. Below this are several input fields:
1. 'Tipo de Período' with a dropdown menu showing 'Campanha Agrícola'.
2. 'Nome' with a text input field containing 'Campanha Fruticola 2017/2018'.
3. 'Data de início' with a date picker showing '10/15/2017'.
4. 'Data de fim' with a date picker showing '02/03/2019'.
5. 'Descrição' with a large text area.
At the bottom of the form are two circular buttons: a red one with a white 'X' icon and a green one with a white checkmark icon.

Figura 4.36: Editar período

4.2.5 Organizações

As organizações são entidades que têm como objetivo representar as instituições, com interesse e envolvidas no projeto FitoAgro.

4.2.5.1 Registo

O registo de uma organização é feito apenas por **administradores** através de um formulário de registo (figura 4.37). Após o registo, a organização é criada, sendo possível novos utilizadores serem associados a esta.

The figure displays a registration interface. On the left, a 'Tipo de Organização' dropdown menu is shown with a placeholder and the option 'Cooperativa Agrícola'. An arrow points from this menu to a larger form titled 'Detalhes da Organização'. This form contains several input fields: 'Nome' (with a NIF field next to it), 'Email', 'Telefone', 'Morada', and 'Cidade'. At the bottom of the form, there are two buttons: 'ANTERIOR' (with a left arrow) and 'ADICIONAR' (with a checkmark).

Figura 4.37: Registo de organizações

4.2.5.2 Consulta, Edição e Remoção

As organizações são listadas através de uma tabela, como é possível ver na figura 4.38, tabela esta que pode ser filtrada por nome, cidade e tipo de organização. Através da tabela pode-se consultar os contactos de cada organização, como o nome, email, telefone e a cidade da sede da organização, estando esta informação disponível para todos os utilizadores autenticados. Esta tabela pode ser filtrada por campos de pesquisa.

Para visualizar a informação completa da organização é necessário que o utilizador seja membro da organização ou um **administrador**.

The figure shows two views of the organization management interface.
 (a) **Página do administrador**: A table listing organizations. The columns are: Nome, Tipo, Endereço, Email, and Cidade. The table contains several rows of data, including organizations like 'CIP', 'SINAGRO', 'FELIX', 'COOPAL', 'FELIXE', 'SINAGRO', 'SINAGRO', 'SINAGRO', 'SINAGRO', and 'SINAGRO'.
 (b) **Página da conta de administração da organização**: A similar table, but with an additional column for 'NIF'. It also includes a 'Detalhes' button for each organization row.

Figura 4.38: Página de organizações

Adicionalmente existe também a possibilidade de editar toda a informação da organização através do formulário apresentado na figura 4.39. sendo esta funcionalidade apenas disponível para os **administradores** ou a **conta de administração da organização**.

Editar Organização Cooperativa Agrícola

Nome: NIF:

Email: Telefone:

Morada:

Cidade:

Figura 4.39: Editar organização

E por fim, a aplicação também disponibiliza a funcionalidade de remoção da organização. Sendo esta funcionalidade apenas disponível para os **administradores**.

4.2.6 Visitas e Observações

Os agentes de campo efetuam visitas às parcelas com o objetivo de recolher informação fitossanitária na forma de observações sendo de extrema importância que esta informação possa ser acedida de forma rápida e intuitiva e caso seja necessário corrigida. Esta componente é uma das mais importantes do sistema, pois é através dela que é possível monitorizar o estado fitossanitário das parcelas.

4.2.6.1 Visualização

É possível consultar visitas e suas observações em dois locais da aplicação: Em cada parcela existe uma secção reservada para as visitas efetuadas na mesma, como já foi referido anteriormente [4.2.3.7](#), que está disponível para todos os utilizadores com acesso à parcela. Adicionalmente existe uma página na aplicação reservada para consultar visitas (figura [4.40](#)), esta página está reservada aos **administradores** e **agentes de campo**, sendo que estes últimos só possuem acesso as visitas efetuadas por eles mesmos, enquanto os **administradores** têm acesso a todas as visitas registadas no sistema.

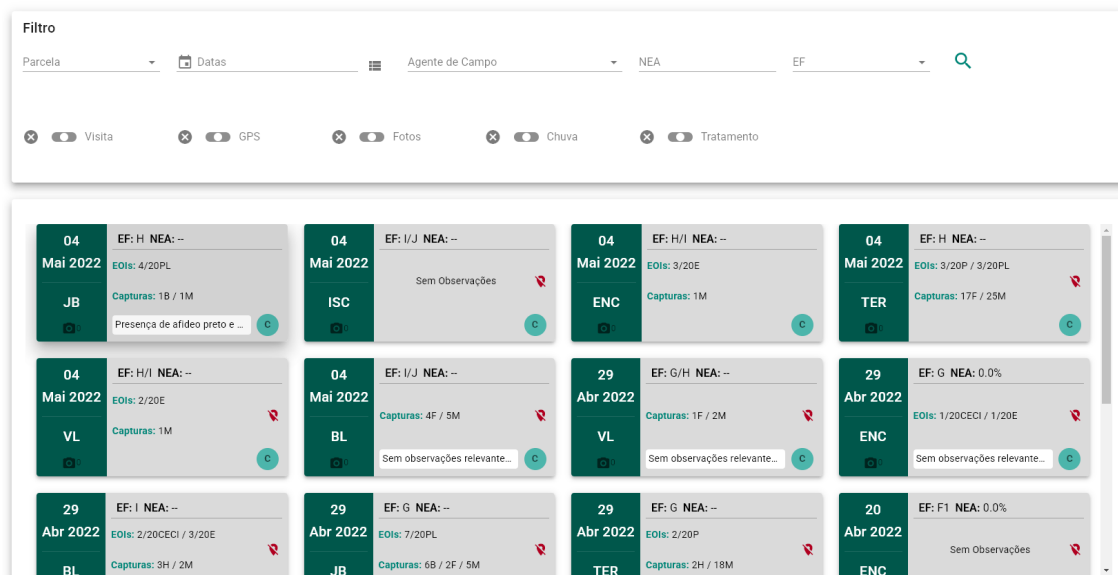


Figura 4.40: Página de visitas

Através do clique na visita, será mostrada mais informação relativa à mesma (figura 4.41), sendo esta informação:

- **Data e hora:** A data em que foi efetuada e a hora de início e fim .
- **Parcela:** A parcela em que foi efetuada.
- **Agente de Campo:** O agente de campo que efetuou a visita.
- **Estado fenológico:** O estado fenológico da cultura na parcela durante a visita.
- **Notas:** As notas tiradas pelo técnico.
- **Observações:** A informação fitossanitária recolhida pelo técnico em forma de tabela ou gráfico (figura 4.42).
- **Fotos:** As fotos tiradas durante a visita (figura 4.43).
- **Trajeto percorrido:** É ilustrado num mapa o trajeto percorrido pelo técnico e onde este tirou as fotos (figura 4.44).

4.2. FUNCIONALIDADES DA APLICAÇÃO WEB FITOPS

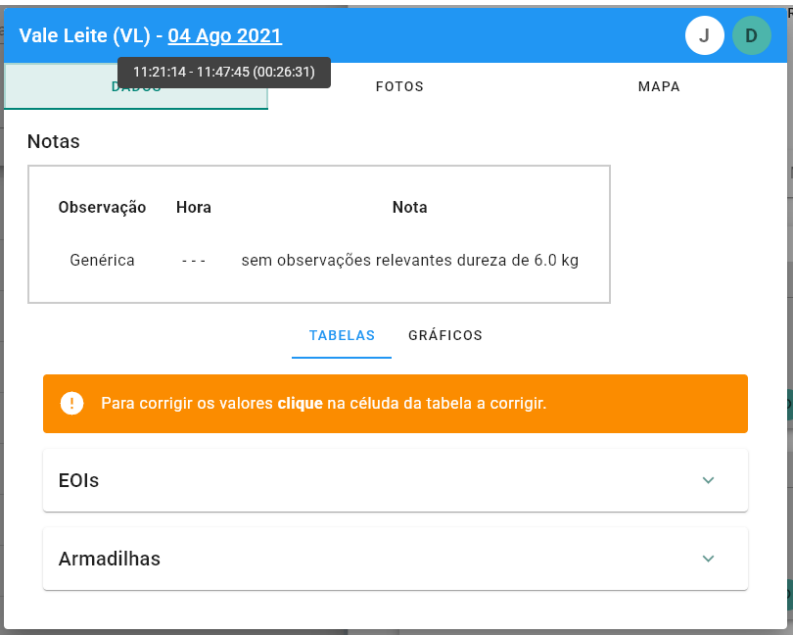


Figura 4.41: Detalhes de uma visita

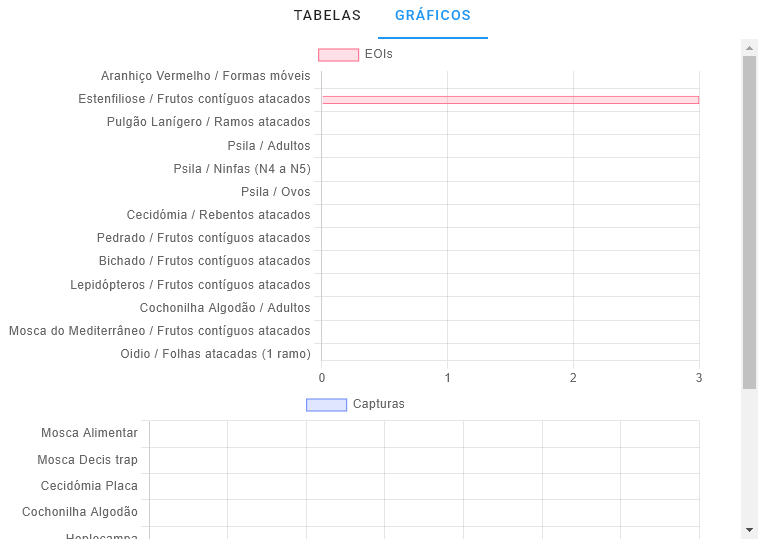
TABELAS

GRÁFICOS

EOIs

Protocolo	Observação	Etapas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Aranhizo Vermelho	Formas móveis	Folhas ocupadas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Aranhizo Vermelho	Ovos	Folhas ocupadas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Estenfiliose	Frutos contíguos atacados	---	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Estenfiliose	Folhas atacadas (1 ramo)	---	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pulgão Lanígero	Ramos atacados	---	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lagartas Mineiras	Folhas atacadas (1 ramo)	---	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a) Dados em forma de tabela



(b) Dados em forma de gráficos

Figura 4.42: Observações de uma visita

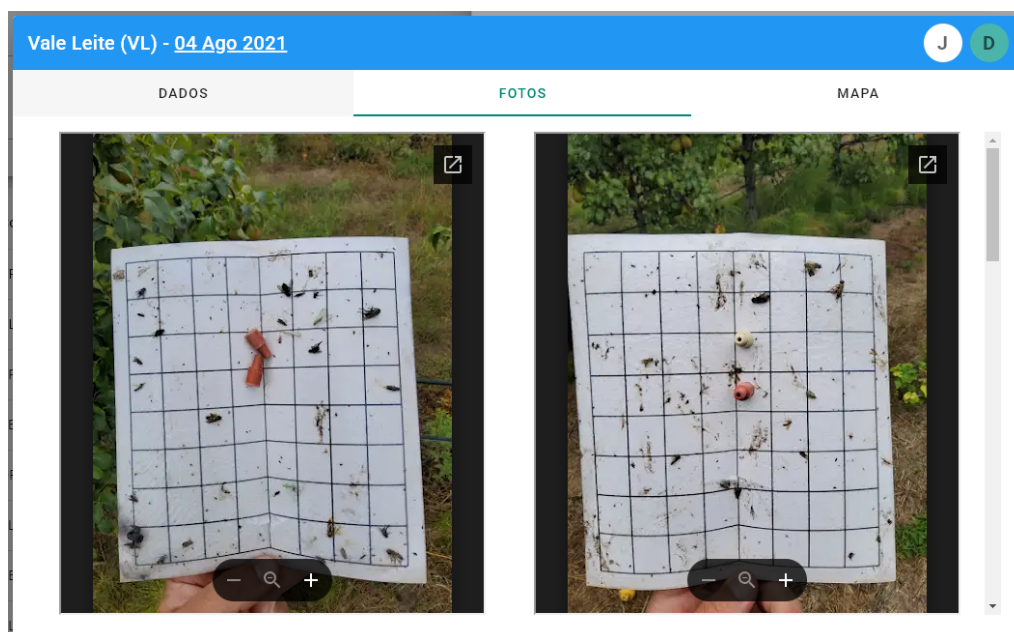


Figura 4.43: Fotos tiradas durante a visita

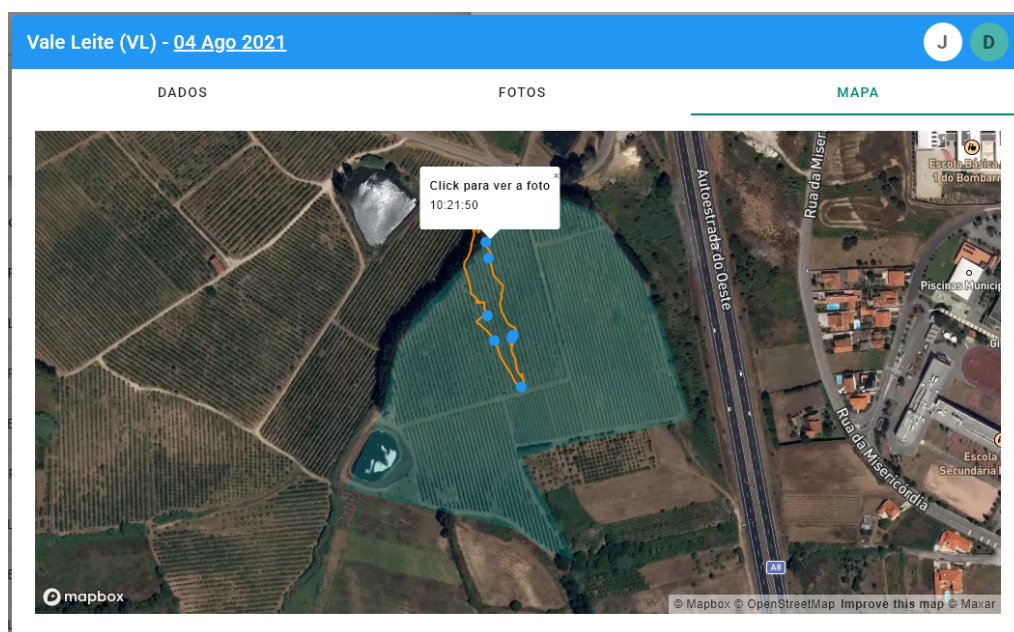


Figura 4.44: Trajeto percorrido durante a visita

4.2.6.2 Correção de valores

Caso seja encontrado algum erro nas observações efetuadas, é possível corrigir os valores recolhidos. Para tal, basta ir à vista em tabelas das observações e clicar no valor a corrigir, de seguida irá aparecer uma janela onde é possível introduzir o novo valor (figura 4.45). Esta correção irá ser registada na base de dados e propagada para a aplicação móvel.



Figura 4.45: Correção de valores

4.2.7 Protocolos de recolha de informação fitossanitária

Estes protocolos são efetuados pelos agentes de campo durante visitas, com o propósito de recolher informação fitossanitária. Sendo efetuados com o auxílio da aplicação móvel, desenvolvida no âmbito da Dissertação do M. Correia [10] pois é nesta que são introduzidos os dados recolhidos e estão descritos os procedimentos a efetuar para a sua recolha.

4.2.7.1 Visualização

Através da aplicação web desenvolvida os **administradores** e **agentes de campo** podem visualizar todos os protocolos de recolha de informação fitossanitária, que podem ser efetuados através da aplicação móvel e os seus respetivos detalhes (figura 4.46).

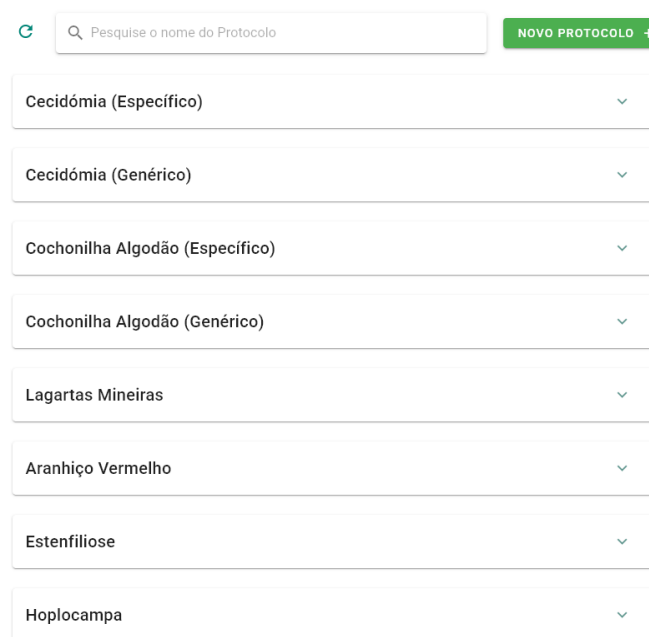


Figura 4.46: Página de protocolos

4.2.7.2 Criação de protocolos

A criação de protocolos de recolha de informação fitossanitária é apenas possível para os **administradores**. Para tal, foi criada uma interface onde é possível introduzir toda a informação necessária para a criação de um protocolo (figura 4.47), sendo esta:

- **Nome e código:** O nome e código do protocolo.
- **Inimigo:** O inimigo que o protocolo visa estudar.
- **Elementos de Interesse:** O tipo e número de elementos de interesse a serem estudados.
- **Observações e Etapas:** As observações a serem efetuadas, as etapas destas observações e o tipo de informação a recolher em cada etapa.

Novo Protocolo

Protocolo

Nome Código Inimigo

EOIs a estudar

Tipo de EOI Nº eois

Observações

Nome da Observação

1 1 Nome da Etapa Tipo de Dado Quantitativo Unidades

+ Quantitativo

+ Qualitativo

+ +

CANCELAR ADICIONAR

Figura 4.47: Registo de protocolos

4.2.8 Controlo de Entidades do Sistema

Existe uma página na aplicação web apenas disponível para **administradores**, reservada para a gestão das entidades bases do sistema (figura 4.48), sendo estas:

- Tipos de Culturas (Ex: Pomóideas)
- Culturas (Ex: Macieira)
- Variantes de Culturas (Ex: Royal Gala)
- Tipos de Inimigos (Ex: Inseto)
- Inimigos (Ex: Filoxera)
- Tipos de Parcelas (Ex: Parcela Agrícola)
- Tipos de Períodos (Ex: Campanha Agrícola)
- Tipos de Organizações (Ex: Cooperativa Agrícola)
- Tipos de Media (Ex: Foto)
- Tipos de Eventos (Ex: Tratamento)

Para cada uma destas entidades é possível criar novas, editar e remover caso não haja dependências.

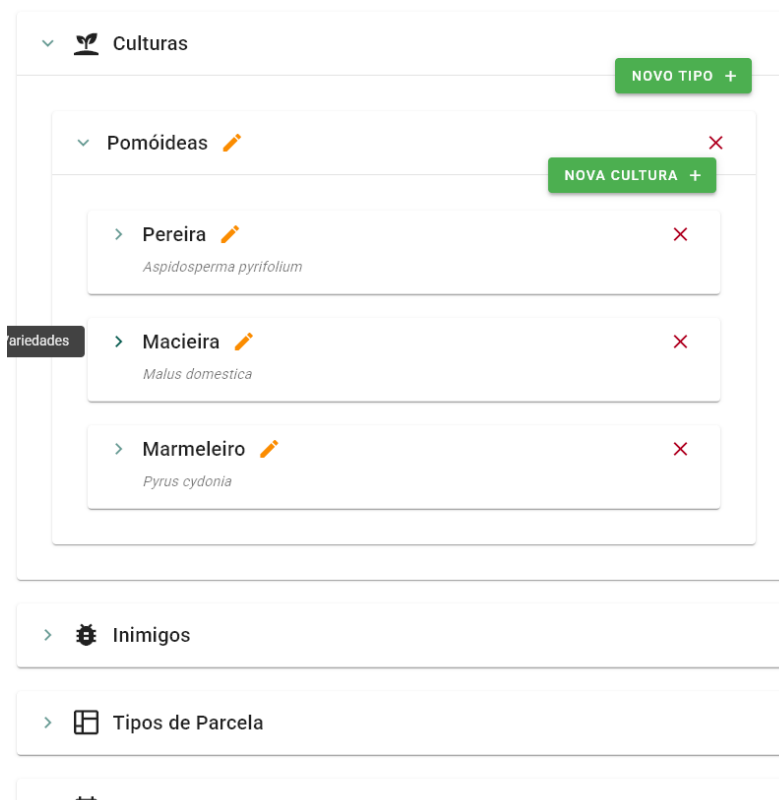


Figura 4.48: Página de entidades do sistema

4.3 Conclusão

Neste capítulo foi apresentada a aplicação e as suas diversas funcionalidades desenvolvidas ao longo desta dissertação. Estas funcionalidades foram planeadas e desenvolvidas para satisfazer especificamente as necessidades dos parceiros do projeto FitoAgro. Visto que, não existia nenhuma solução no mercado que satisfizesse os requisitos do projeto completamente. Sendo um dos principais pontos de destaque, a integração com uma aplicação móvel que permite a recolha de dados em campo, que posteriormente podem ser consultados e corrigidos na aplicação web. Adicionalmente, através da adição de protocolos, é possível configurar a aplicação móvel para suportar novas formas de recolha de dados, conferindo à aplicação móvel a capacidade de evoluir, podendo vir a suportar novos inimigos ou a adaptar-se a possíveis alterações no seu comportamento.

Em conclusão as principais vantagens da aplicação FitOps em relação às outras alternativas exploradas em 3.1 são o facto da aplicação ter sido concebida especificamente para satisfazer os requisitos dos parceiros do projeto e de esta estar integrada com uma aplicação móvel configurável, projetada para a recolha de informação de campo.

IMPLEMENTAÇÃO

Neste capítulo são apresentados os detalhes de implementação da aplicação web, nomeadamente as tecnologias usadas, a arquitetura e a segurança da aplicação. Adicionalmente são apresentadas as alterações efetuadas no sistema, para que este possa acomodar a aplicação.

5.1 Tecnologias Usadas

Nesta secção são descritas as tecnologias usadas para o desenvolvimento na aplicação web. Primeiramente, são descritas as tecnologias usadas na construção do *frontend* da aplicação 5.1.1, que consiste na interface em que o utilizador visualiza e interage. De seguida são descritas as tecnologias usadas na construção do *backend* da aplicação 5.1.2, que consiste no servidor web, o qual permite a comunicação com a base de dados e com o *frontend* através de uma API REST.

5.1.1 *Frontend*

Nesta secção são descritas as tecnologias para construir a camada cliente da aplicação, ou seja, a componente da aplicação web que está em execução no browser do utilizador. Esta componente é composta pela interface com que os utilizadores interagem e a lógica responsável por processar a informação proveniente do *backend*.

5.1.1.1 **Vue.js**

No desenvolvimento do *frontend* da aplicação web foi utilizado o *framework* Vue.js, mais especificamente a versão 3 (Vue 3). Foi escolhido este framework, devido ao facto de adotar uma arquitetura baseada em componentes. Este tipo de arquitetura promove a reutilização do código, acelerando assim o processo de desenvolvimento e facilitando a manutenção.

5.1.1.2 **Vuetify**

Juntamente com o framework Vue.js recorreu-se à biblioteca de componentes Vuetify [44], que oferece um conjunto de componentes (botões, formulários, tabelas, entre outros) prontos a usar e altamente personalizáveis e projetados para serem responsivos, criando assim uma interface de utilizador consistente e intuitiva. A biblioteca possibilita também a fácil personalização de temas, possibilitando aos programadores que ajustem as cores,

fontes e outros aspetos visuais que adequem à identidade visual do projeto. E para finalizar, o Vuetify, tem uma documentação bastante completa e detalhada, com exemplos práticos que facilitam a aprendizagem e a utilização da biblioteca.

5.1.1.3 Pinia

Para guardar o estado global da aplicação foi utilizado o Pinia [30], que consiste numa versão moderna e mais simples de implementar do tradicional Vuex. Tendo este um design modular mais leve e flexível com melhor performance, facilitando a manutenção e a escalabilidade da aplicação. Optou-se por esta biblioteca, devido ao facto de ser modular, ou seja, é possível dividir o estado da aplicação em módulos, facilitando a organização do código. e devido ao facto de ter sido criada com o Vue 3 em mente, o que permite tirar partido das novas funcionalidades do Vue 3 e oferecer melhor performance.

5.1.1.4 Mapbox

Para a visualização de mapas foi utilizado a biblioteca Mapbox [25], que é uma biblioteca pública, para mapas interativos. Esta biblioteca permite a criação de mapas personalizados, com a possibilidade de adicionar camadas, marcadores, entre outros. A biblioteca é bastante flexível e fácil de usar, tem uma documentação bastante completa e exemplos práticos que facilitam a aprendizagem e a utilização da biblioteca.

5.1.1.5 Chart.js

Para a visualização de gráficos foi utilizado a biblioteca Chart.js [8], que é uma biblioteca de gráficos flexível e fácil de implementar. Esta biblioteca permite a criação de vários tipos de gráficos, como por exemplo, gráficos de barras, linhas, radar, entre outros, importantes para a visualização de grandes quantidades de dados, como por exemplo dados meteorológicos.

5.1.1.6 Axios

Para a comunicação com o servidor foi utilizado a biblioteca Axios [4], que é uma biblioteca que permite efetuar pedidos *HTTP* sendo baseada em promessas, ou seja, permite fazer pedidos assíncronos ao servidor. Sendo os motivos principais para a escolha desta biblioteca o facto de ser bastante simples de implementar e oferecer diversas funcionalidades convenientes para desenvolvimento web, sendo algumas destas funcionalidades: a fácil alteração e manipulação das diversas componentes do pedido *HTTP* (método *HTTP*, *headers*, parâmetros, *payload*), conversão automática de dados JSON para objetos JavaScript e vice versa. Assim como a captura de erros.

5.1.1.7 JSON Web Token (JWT)

JWT [20] é um método *open source* usado para transmitir informações de maneira segura entre duas componentes como um cliente e um servidor. Essas informações são transmitidas como um objeto JSON que é comprimido e encriptado para garantir a segurança. Para além de encriptação de informação, o *token* é assinado usando uma chave secreta contida no servidor e é enviado para o cliente. Este, por sua vez, pode então enviar o *token* de volta ao servidor para autenticação e autorização, sendo que o servidor verifica a assinatura do *token* e, se for válida, aceita o pedido. O JWT é bastante utilizado em aplicações web modernas, pois é fácil de usar, seguro e eficiente.

5.1.1.8 GitHub Pages

Para a publicação da aplicação foram utilizadas as GitHub Pages [15], que é um serviço de alojamento de páginas web estáticas, oferecido pela plataforma GitHub, que permite a publicação de páginas web diretamente, a partir de um repositório GitHub ou um *branch*.

5.1.2 Backend

Nesta secção são descritas as tecnologias para construir a camada servidor da aplicação. Esta componente é composta pela base de dados, onde é armazenada a informação do sistema, e pelo servidor web que é responsável pela comunicação do *frontend* com a base de dados.

5.1.2.1 NGINX

NGINX (pronunciado como "Engine-X") é um servidor *open-source*, de alta performance, usado maioritariamente para servir aplicações web. Oferece bastantes funcionalidades deste balanceamento de carga a atuar como *proxy* reverso. Ele foi criado por Igor Sysoev em 2004 e tornou-se popular devido à sua capacidade de lidar eficientemente com uma quantidade enorme de conexões concorrentes.

5.1.2.2 Node.js

Node.js é um ambiente de execução que permite correr código JavaScript em ambiente de servidor, sem ser necessário o *browser*. Ele é responsável por compilar JavaScript diretamente para código de máquina, tornando a execução rápida e eficiente. Node.js é conhecido por seu modelo I/O não bloqueante e orientado a eventos, o que o torna ideal para construir aplicações escaláveis, especialmente em sistemas que exigem operações de entrada e saída frequentes, como servidores web com APIs.

5.1.2.3 Express.js

Para implementar o servidor da aplicação web foi utilizado o *framework* Express.js, que é um framework web minimalista e flexível para Node.js. Ele facilita o desenvolvimento de aplicações web e APIs, pois fornece funcionalidades como *routing*, *middleware*, algo que é importante para a segurança do servidor, pois permite fazer um pré processamento dos pedidos recebidos, e tratamento de pedidos *HTTP*. Express.js é amplamente utilizado devido à sua simplicidade, extensibilidade e integração com o ecossistema Node.js.

5.1.2.4 Sequelize

Para comunicar com a base de dados foi usada a biblioteca Sequelize [36], que é um ORM (*Object-Relational Mapping*) para Node.js. Esta biblioteca permite a comunicação com a base de dados de forma simples, segura e intuitiva, pois possibilita utilizar objetos JavaScript, denominados de modelos de dados, para interagir com a base de dados, ao invés de escrever as operações diretamente em SQL.

5.1.2.5 *Google Drive API v3*

Para comunicar com a pasta da Google Drive da aplicação móvel foi usada a Google Drive API v3 [16], que é uma API fornecida pela Google, para interagir com a Google Drive, sendo as principais funcionalidades usadas:

- **Gestão de pastas e ficheiros:** A criação, leitura, atualização e remoção de pastas e ficheiros na Google Drive.
- **Permissões e Partilha:** A adição e remoção de permissões de escrita ou leitura de pastas e ficheiros, com outros utilizadores.
- **Pesquisa de ficheiros:** A pesquisa de ficheiros na Google Drive com base em critérios como nome, tipo, data de criação, entre outros.

Esta API foi usada tanto para alterar os ficheiros de configuração da aplicação móvel, como para consultar os ficheiros com informação referente às visitas efetuadas nas parcelas pelos técnicos, usando a aplicação móvel.

Para que se pudesse usar esta API foi necessário criar um projeto na Google Cloud Console, pois é aqui que se pode ativar a mesma e obter as credenciais de autenticação necessárias para a sua utilização.

5.1.2.6 **PostgreSQL**

Para armazenar a informação do sistema foi utilizada a base de dados PostgreSQL, que é um sistema de gestor de bases de dados relacionais. Foi usada o PostgreSQL devido ao facto da base de dados do sistema de informação já existente ser PostgreSQL, sendo apenas necessário fazer a extensão da base de dados, para acomodar os novos tipos de dados, gerados pela aplicação FitOps.

5.2 Arquitetura

A aplicação FitOps foi desenvolvida com base numa arquitetura cliente-servidor, na qual o cliente 5.2.1 corresponde à interface gráfica com que o utilizador interage, permitindo a visualização e manipulação dos dados. O servidor 5.2.2 é responsável pela comunicação com as bases de dados e o principal responsável pela segurança e controlo de acessos do sistema, como ilustrado na figura 5.1.

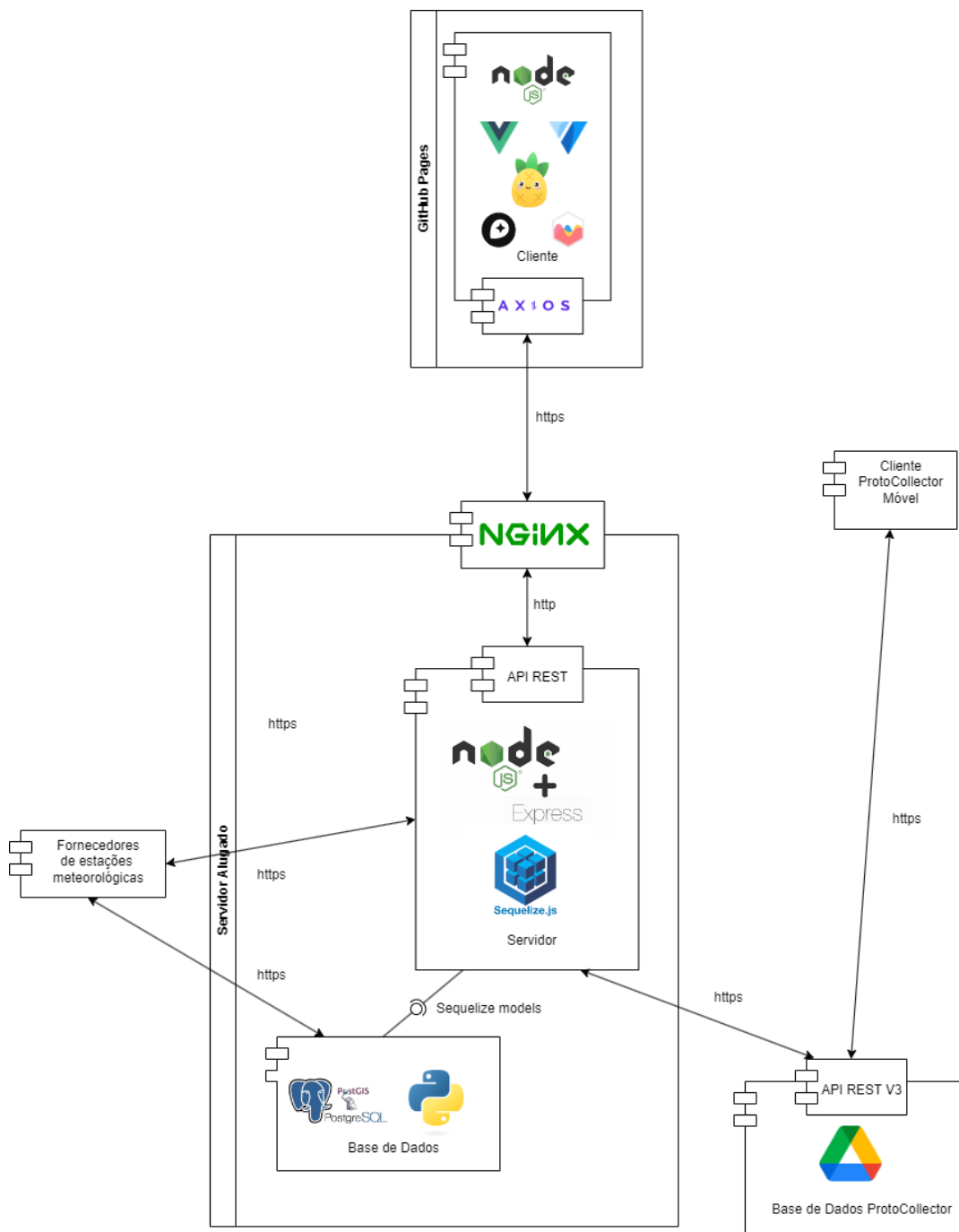


Figura 5.1: Arquitetura da aplicação FitOps.

5.2.1 Cliente

Nesta secção irão ser descritas as várias componentes do cliente da aplicação, nomeadamente como é efetuada a comunicação com o servidor [5.2.1.1](#), como é armazenado o estado global da aplicação e qual o propósito da informação armazenada [5.2.1.2](#), como é feito o *routing* dentro da aplicação para que seja possível navegar entre as várias páginas da aplicação dinamicamente [5.2.1.3](#), e finalmente é explicado como é efetuado o controlo de permissões dos utilizadores e como a interface se adapta consoante as estas permissões [5.2.1.4](#).

5.2.1.1 Comunicação com o Servidor

A comunicação entre o cliente e o servidor é realizada através de pedidos *HTTP*, sendo que o cliente envia pedidos ao servidor usando a biblioteca *Axios*, que permite fazer pedidos *HTTP* de forma assíncrona, ou seja, sem bloquear o fluxo de execução da aplicação. Os pedidos são encriptados através do protocolo *HTTPS*, garantindo assim a segurança da informação transmitida. Os pedidos são compostos por:

- **Método *HTTP*:** O tipo de ação que o pedido deve realizar, sendo os principais métodos usados: **GET** para obter dados, **POST** para inserir nova dados, **PATCH** para alterar dados e **DELETE** para remover dados.
- **URL:** O *endpoint* do servidor para onde o pedido deve ser enviado.
- **Corpo do pedido:** Os dados que devem ser enviados para o servidor, como por exemplo, os dados de um formulário.
- **Cabeçalho do pedido:** Informações adicionais que devem ser enviadas com o pedido, como por exemplo, o token de autenticação.

5.2.1.2 Armazenamento do estado da aplicação

Para armazenar o estado da aplicação foi utilizado o *Pinia*, que é uma biblioteca de gestão de estado global para *Vue*. O *Pinia* permite dividir o estado da aplicação em módulos chamados *stores* e cada um destes módulos é composto pelo *state*, que é o estado da aplicação, e por *actions*, que são métodos que permitem alterar ou retornar o estado da aplicação.

Foram então criadas várias *stores* para armazenar e manipular o estado da aplicação, sendo as principais:

- **accountStore:** Armazena informação relativa ao utilizador autenticado, sendo esta: o nome, organização a que pertence, o tipo de conta e o *token* de autenticação. Adicionalmente, esta *store* também contém métodos usados no controlo de permissões, como pode ser observado em [5.1](#), algo que é explicado em mais detalhe em [5.2.1.4](#).

Listagem 5.1: accountStore.

```
1 export const useAccountStore = defineStore("accountStore", {
2   state: () => ({
3     account: null,
4   }),
5   // persiste apos refresh da pagina
```

```

6      persist: true,
7      actions: {
8          // Devolve utilizador atual
9          getAccount() {
10             return this.account;
11          },
12          // Altera utilizador atual
13          setAccount(account) {
14             this.account = account;
15          },
16          // Apaga utilizador atual
17          resetAccount() {
18             this.account = null;
19             this.role;
20          },
21          // Verifica se o utilizador atual tem alguma das roles
22          hasRole(roles) {
23             if (this.account) {
24                 return roles.includes(this.account.role);
25             }
26             return false;
27          },
28          // Devolve a role do utilizador atual
29          Role() {
30             if (this.account) {
31                 return this.account.role;
32             }
33             return null;
34          },
35      },
36  });

```

- **generalStore:** Esta *store* foi criada para complementar as funcionalidades de transporte de informação de renderização do *Vue*, ou seja, permite aos componentes, que estão localizados em baixo na árvore de componentes alterar a informação que está em componentes, no topo da árvore de componentes, como por exemplo, a barra de navegação e o componente responsável pela mensagem de *pop up*.

Listagem 5.2: generalStore.

```

1  export const useGeneralStore = defineStore("generalStore", {
2      state: () => ({
3          navDrawerOpen: true,
4          snackbar: {
5              show: false,
6              color: "",
7              text: "",
8          },
9      }),

```



```
10     actions: {
11         // Abre ou colapsa barra de navegacao
12         toggleNavDrawer() {
13             this.navDrawerOpen = !this.navDrawerOpen;
14         },
15         // Gera um pop up de sucesso com texto personalizado
16         successSnackbar(text) {
17             this.showSnackbar({ color: "success", text });
18         },
19         // Gera um pop up de erro com texto personalizado
20         errorSnackbar(text) {
21             this.showSnackbar({ color: "error", text });
22         },
23         // Gera um pop up de aviso com texto personalizado
24         warningSnackbar(text) {
25             this.showSnackbar({ color: "warning", text });
26         },
27         // Gera um pop up com cor e texto personalizado
28         showSnackbar({ color, text }) {
29             this.snackbar = { show: true, color, text };
30         },
31     },
32 });
```

5.2.1.3 Routing

Routing refere a capacidade de navegar entre diferentes páginas da aplicação, sendo que na nossa aplicação, o *routing* é feito através do Vue Router, que é um *plugin* oficial do Vue para gestão de rotas, dentro de uma *Single Page Application* (SPA), o que significa que é possível ter múltiplas páginas virtuais, dentro de uma única página HTML. Com o Vue Router, a navegação entre essas páginas é feita de maneira dinâmica, ou seja, não é necessário recarregar a página inteira, pois cada rota é mapeada para um componente, que é renderizado ,quando a rota é acionada. Cada rota é composta por:

- **Caminho:** O caminho do URL que a rota deve corresponder, por exemplo: */login*.
- **Nome:** O nome da rota, que é usado para navegar para a rota, por exemplo: *Login*.
- **Componente:** O componente que deve ser renderizado quando a rota é acionada, por exemplo: *LoginView*.
- **Meta:** Informações adicionais que devem ser guardadas com a rota. Na nossa aplicação esta informação é usada para controlo de acesso, por exemplo, certas rotas só estão disponíveis para utilizadores autenticados (*requiresAuth: true*), ou para utilizadores de um certo tipo (*requiresRole: [...]*). Estas verificações são feitas através de um *middleware* 5.4 que corre antes de cada rota ser acionada.

Listagem 5.3: Exemplo de rotas.

```
1 const routes = [  
2   // Rota para pagina de autenticacao  
3   {  
4     path: "/login",  
5     name: "Login",  
6     component: () =>  
7       import(/* webpackChunkName: "login" */ "../views/  
         LoginView.vue"),  
8   },  
9   // Rota para pagina de registo de conta  
10  {  
11    path: "/register",  
12    name: "Register",  
13    component: () =>  
14      import(/* webpackChunkName: "register" */ "../views/  
        RegisterView.vue"),  
15  },  
16  // Rota base das paginas protegidas por autenticacao  
17  {  
18    path: "/",  
19    component: () =>  
20      import(/* webpackChunkName: "Index" */ "../views/Index  
        .vue"),  
21    meta: { requiresAuth: true },  
22    children: [  
23      // Rota da pagina inicial, disponivel utilizadores autenticados  
24      {  
25        path: "/",  
26        name: "Home",  
27      },  
28      // Rota da pagina dos utilizadores, disponivel para Admins e Orgs  
29      {  
30        path: "/users",  
31        name: "Utilizadores",  
32        meta: { requiresRole: ["Administrador", "  
          Organizacao"] },  
33        component: () =>  
34          import(/* webpackChunkName: "users" */ "../views  
            /AgentsView.vue"),  
35      },  
36      ...  
37    ],  
38  },  
39  ...  
40 ];
```

Listagem 5.4: *Vue Router Middleware.*

```
1 router.beforeEach(async (to, from, next) => {
2   // Se a rota esta protegida por autenticacao
3   if (to.meta.requiresAuth) {
4     // Valida o token
5     let token_valid = await axios
6       .request({
7         url: "api/token_validate",
8         method: "post",
9         headers: {
10           Authorization: "Bearer " + useAccountStore().account?.token,
11         },
12       })
13       .then((response) => {
14         return response.data;
15       })
16       .catch(() => {
17         return false;
18       });
19     // Se o token for invalido redireciona para pagina de autenticacao
20     if (!token_valid) {
21       next({ name: "Login" });
22     }
23     // Se o utilizador nao tem role apropriado redireciona para pagina
    inicial
24     if (to.meta.requiresRole) {
25       useAccountStore().hasRole(to.meta.requiresRole)
26         ? next()
27         : next({ name: "Home" });
28     } else {
29       next();
30     }
31   } else {
32     next();
33   }
34 });
```

5.2.1.4 Controlo de Permissões

O controlo de permissões, na camada cliente, é realizado ao nível do tipo da conta do utilizador autenticado. Para isso foi criada a função *hasRole* na *store accountStore* que é possível ver em 5.1, a qual verifica se o tipo da conta do utilizador autenticado é um dos tipos autorizados recebidos como argumento. Esta função é usada tanto no *middleware* do Vue Router 5.4, como também é usada em componentes específicos, permitindo ou não, que estes sejam renderizados, como é possível ser observado em 5.5, que consiste num caso no qual o componente só é renderizado, se o utilizador autenticado, for do tipo Administrador.

Listagem 5.5: Exemplo de uso da função *hasRole*

```
1 <v-col  
2   v-if="accountStore.hasRole(['Administrador'])"  
3   cols="5"  
4   class="d-flex justify-center"  
5 >  
6   <AddWeatherStation @updateWS="getWSfromFitoagro()" />  
7 </v-col>
```

5.2.2 Servidor

Nesta secção, são descritas as várias componentes da camada servidor da aplicação e suas respectivas funções, nomeadamente o *proxy* reverso, que consiste num servidor web que serve de intermediário entre o cliente e o servidor Node.js 5.2.2.1. As rotas que mapeiam os pedidos *HTTP* recebidos para as funções que devem ser executadas 5.2.2.2. O *middleware* constituído por funções que fazem verificações de segurança antes de executar as funções das rotas 5.2.2.3. Os controladores que são responsáveis por executar a lógica de negócio da aplicação 5.2.2.4. Os modelos que representam a estrutura da base de dados e suas tabelas 5.2.2.5 para que esta possa ser manipulada em JavaScript. E finalmente, é explicado em detalhe os vários mecanismos de segurança usados no servidor 5.2.2.6.

5.2.2.1 Proxy Reverso

Decidiu-se recorrer a um servidor web NGINX para atuar como *proxy* reverso para o servidor Node.js devido aos seguintes motivos:

- **Já em uso na máquina alugada**

O NGINX já está a ser usado na máquina alugada do FitoAgro, para um propósito semelhante, sendo apenas necessário configurá-lo para servir a nova aplicação.

- **Encriptação dos pedidos**

Pretende-se que a comunicação entre o cliente e o servidor seja encriptada por motivos de segurança, e para que tal seja possível é necessário usar o protocolo *HTTPS*. Inicialmente pensou-se em usar o servidor Node.js para este fim, mas descobriu-se que poderiam ocorrer problemas na máquina alugada, pois o servidor Node.js teria de reservar a porta 443 (porta *HTTPS*) da máquina, impossibilitando assim a utilização da mesma para outros serviços. O NGINX resolve este problema, pois atua como *proxy* reverso e recebe os pedidos *HTTPS* que chegam à porta 443 e redireciona-os para o servidor Node.js, que está a correr noutra porta diferente e posteriormente devolve as respostas do mesmo, ao cliente.

- **Escalabilidade e balanceamento de carga**

Com esta configuração é possível escalar a aplicação, para tal basta criar novas instâncias dos servidor Node.js e indicar ao NGINX onde estas se localizam, e o NGINX através de mecanismos próprios de balanceamento de carga distribui os pedidos pelas várias instâncias.

5.2.2.2 Routes

As rotas são responsáveis por mapear os pedidos *HTTP* recebidos pelo servidor, para as funções que devem ser executadas. Cada rota é composta por:

- **Método *HTTP*:** O tipo de ação que a rota deve tratar, como *GET*, *POST*, *PUT*, *PATCH* ou *DELETE*. Cada método *HTTP*, representa uma intenção diferente.
- **Caminho ou URL:** A parte da URL que o servidor usa para identificar qual ação deve ser executada. Pode incluir parâmetros dinâmicos por exemplo, */agents/:agent_id*.
- **Função de *callback* (controlador):** Função que é executada, quando a rota é acionada. Geralmente, essa função recebe três parâmetros principais: **req** (*request*), **res** (*response*) e **next** (próxima função a ser executada, se necessário).

Todas as rotas da API do servidor estão listadas no apêndice [B](#).

5.2.2.3 Middleware

Middleware é constituído por funções que têm acesso aos objetos: **req**, **res** e **next**. Sendo os principais objetivos do *middleware* no nosso servidor, fazer verificações de segurança, sendo estas de:

- **Autenticação:** Verificar se o utilizador está autenticado através da validação do *token* de autenticação, recebido no pedido.
- **Autorização:** Verificar se o utilizador a fazer o pedido tem permissões para aceder à respetiva rota.

5.2.2.4 Controllers

Os controladores são responsáveis por executar a lógica de negócio da aplicação, ou seja, são os métodos responsáveis por processar os pedidos recebidos pelo servidor. Os controladores são chamados pelas rotas e recebem os parâmetros necessários para executar a ação pretendida. No nosso servidor, por norma existe um controlador para cada entidade principal do sistema, facilitando, deste modo, a manutenção e a escalabilidade do servidor. Contudo, existem exceções a esta regra, nomeadamente os controladores criados com o propósito de procurar informação do fornecedor de estações meteorológicas.

5.2.2.5 Models

Os modelos de dados são responsáveis por representar a estrutura da base de dados e suas tabelas. Sendo estes os modelos que permitem a conversão das linhas das tabelas da base de dados PostgreSQL para objetos JavaScript e vice versa, facilitando assim a manipulação de base dados baseadas em SQL, numa aplicação com base em JavaScript.

5.2.2.6 Segurança

- **Jason Web Token (JWT):** No nosso sistema os JWTs são utilizados para autenticação e autorização dos utilizadores, sendo que o *token* é gerado no servidor após a autenticação do utilizador e é enviado para o cliente. Após se autenticar perante o sistema o cliente recebe o *token* e armazena localmente, para que posteriormente

possa enviar no cabeçalho de cada pedido *HTTP*, permitindo ao servidor autenticar o pedido e saber quem é o utilizador responsável pelo pedido.

- **HTTPS:** A comunicação entre o cliente e o servidor é feita através do protocolo *HTTPS*, cujo o objetivo é garantir a segurança da informação transmitida, pois toda a informação trocada entre estes é encriptada durante o seu envio.
- **Controlo de permissões:** O controlo de permissões é feito ao nível das rotas e dos controladores.

Nas rotas este controlo de permissões é conseguido através da utilização de um *middleware* que recebe como argumento os tipos de contas que podem aceder à rota em questão. Este *middleware* através da descodificação do *token* de autenticação, descobre quem é o utilizador que está a fazer o pedido e verifica se o tipo de conta deste está autorizado, ou seja, corresponde a um dos tipos recebidos como argumento, se sim o pedido é processado, caso contrário é devolvido um erro.

Nos controladores o controlo de permissões é feito a nível do utilizador, ou seja, verifica se o utilizador autenticado tem permissões, para executar a ação pretendida no recurso indicado.

O exemplo a seguir demonstra porque é necessário fazer o controlo de permissões, nos controladores e nas rotas: Um utilizador do tipo produtor tenta editar os dados de uma parcela **que não é sua** usando a rota 5.6, este pedido não seria bloqueado pelo *middleware*, pois todos os produtores tem permissões para aceder a esta rota, mas seria bloqueado pelo controlador, pois como é possível observar em 5.7 este verifica se o utilizador autenticado é o dono da parcela que está a tentar editar.

Listagem 5.6: Exemplo de uso do *middleware* numa rota.

```
1 // Rota da API so disponivel para Admins, Produtores e Orgs
2 router.patch(
3     "/regions/:region_id(\\d+)",
4     verifyTokenWithRole(["Administrador", "Produtor", "Organizacao"]),
5     controllers.regionsController.region_update
6 );
```

Listagem 5.7: Controlo de permissões no controlador.

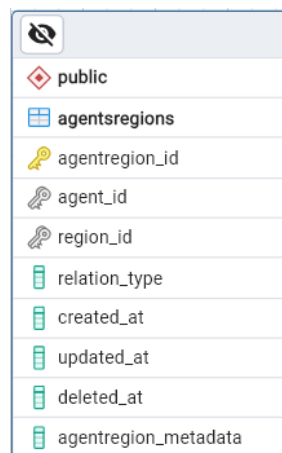
```
1 exports.region_update = async function (req, res, next) {
2     // Obter o utilizador autenticado
3     let principal = utils.getPrincipal(req);
4     // Obter o tipo de conta do utilizador
5     let role = await utils.getRole(principal.agent_id);
6
7     models.regions.findPk(req.params.region_id).then(async (region) => {
8         if (!region) {
9             return next(new createError.NotFound());
10        }
11        // Verificar se e o dono da parcela
12        let isPrincOwner = await isOwner(principal.agent_id, region.
            region_id);
```

```
13 // Verificar se e administrador da organizacao a que a parcela
    pertence
14 let isPrincOrg = await orgsMemb.isAgentAdminOfOrganization(
15     principal.agent_id,
16     region.organization_id
17 );
18 if (!isPrincOwner && !isPrincOrg && role !== "Administrador") {
19     return next(new createError.Unauthorized("Not authorized"));
20 }
21 ...
22 });
23 };
```

5.2.3 Base de Dados

Em relação às extensões efetuadas na base de dados, foi criada uma tabela nova chamada **agentsregions**, que tem como objetivo guardar e representar relações entre utilizadores e parcelas, sendo esta uma relação de muitos para muitos. Esta tabela é composta por oito colunas como é possível observar na figura 5.2, sendo estas:

- **agentregion_id**: Chave primária da tabela.
- **agent_id**: Chave estrangeira que referencia a tabela **agents** e representa o utilizador.
- **region_id**: Chave estrangeira que referencia a tabela **regions** e representa a parcela.
- **relation_type**: Tipo de relação entre o agente e a parcela.
- **created_at**: Data de criação da relação.
- **updated_at**: Data da última atualização da relação.
- **deleted_at**: Data da remoção da relação.
- **agentregion_metadata**: Nesta coluna é guardada informação adicional sobre a relação, como por exemplo, a data de início e fim da relação, que podem ser diferentes das datas em que esta foi criada ou removida do sistema.



	public
	agentsregions
	agentregion_id
	agent_id
	region_id
	relation_type
	created_at
	updated_at
	deleted_at
	agentregion_metadata

Figura 5.2: Schema da tabela **agentsregions**.

Foram também criados *triggers*, que são funções que estão associadas a tabelas, que só são executadas quando um certo comportamento é detetado dentro da tabela, estes foram criados para simular o efeito *delete cascade* que consiste em remover todos os elementos dependentes de um elemento que foi removido, algo que não possível ser efetuado na nossa base de dados, pois quando um elemento é removido da base dados não é removido de verdade, mas sim marcado como removido, esta prática denominasse de "*soft delete*". Para tal foi necessário criar *triggers* que detetem quando um elemento foi marcado como removido e marque como removidos todos os elementos dependentes deste.

Foi também criada uma vista chamada **visits_info**, que tem como objetivo facilitar a visualização da informação mais importante das visitas, através de uma única tabela. Esta vista é composta por 18 colunas, sendo estas:

- **visit_id**: Chave primária da visita.
- **date**: Data da visita.
- **plot**: Código da parcela em que a visita foi efetuada.
- **plotName**: Nome da parcela em que a visita foi efetuada.
- **tBegin**: Hora de início da visita.
- **tEnd**: Hora de fim da visita.
- **duration**: Duração da visita.
- **ef**: Estado fenológico da cultura.
- **nea**: Nível económico de ataque.
- **rain**: Se choveu durante a visita.
- **treatment**: Se estivessem a ser efetuados tratamentos fitossanitários que impossibilitaram a visita.
- **visited**: Se o técnico conseguiu efetuar a visita.
- **numPhotos**: Número de fotos tiradas durante a visita.
- **numVoices**: Número de mensagens de voz gravadas durante a visita.
- **gps**: Se existe informação georreferenciada da visita.
- **notes**: Notas adicionais da visita.
- **agent**: Nome do técnico que efetuou a visita.
- **fileId**: Id da pasta da visita na Google Drive.

E por fim a última extensão efetuada foi a adição da coluna **datapprovider_config** à tabela **datapproviders**, que é a tabela que guarda a informação dos fornecedores de estações meteorológicas. Nesta coluna, é guardada o mapeamento dos tipos de informação meteorológica, provenientes dos fornecedores para os tipos presentes na base de dados do sistema, sendo esta informação crítica para a transformação dos dados recebidos, para um formato padrão que possa ser armazenado na base de dados do sistema. Estes

mapeamentos já existiam na máquina alugada do FitoAgro, mas estavam guardados em ficheiros JSON de configuração, cujo o nome era a estação da qual a informação era recolhida, impossibilitando a adição de novas estações sem a intervenção de um programador, pois seria necessário criar manualmente um novo ficheiro para cada nova estação, algo imprático quando se pretende que a adição de estações possa ser efetuada através da aplicação.

5.2.4 Scripts de Recolha de informação Meteorológica

A recolha de informação meteorológica é feita através de *scripts*, escritos na linguagem Python. Estes *scripts* são executados diariamente e foram criados no âmbito da dissertação de A. Malafaia [24] com o propósito de recolher informação meteorológica de 59 estações distintas, provenientes de vários fornecedores.

Durante o desenvolvimento da aplicação FitOps, foi necessário alterar e atualizar estes *scripts* por duas razões. Sendo a principal razão, o facto dos dados necessários para a recolha de informação destas estações estarem inseridos diretamente no código dos *scripts*, tornando impossível a adição de novas estações meteorológicas, sem a recorrente alteração do código dos mesmos. Para resolver o problema mencionado, foi necessário adaptar os *scripts* de modo a que estes obtivessem os dados necessários para a recolha de informação diretamente a partir da base de dados. Sendo a outra razão para a atualização dos *scripts*, a alteração dos serviços por parte dos fornecedores, tornando assim necessário efetuar uma correção.

5.3 Conclusão

A implementação das duas componentes da aplicação web (cliente e servidor) foi onde se concentrou a maior parte do esforço. Porque ao contrário da base de dados e *scripts*, estas tiveram de ser implementadas de raiz.

A integração com a aplicação móvel, provou-se mais complexa do que esperado. Isso deve-se ao facto da aplicação móvel não comunicar diretamente com o sistema de informação, ou seja, a aplicação móvel armazena os dados em formato JSON numa pasta Google Drive, que por sua vez são recolhidos pelo sistema através de *scripts*. Para além da recolha é necessário processar os ficheiros de forma que estes possam ser armazenados na base de dados. Então para cada funcionalidade que implicasse a alteração da configuração da aplicação móvel, como a adição de protocolos e adição de parcelas, é necessário fazer alterações tanto na base de dados como nos ficheiros de configuração da aplicação móvel que estavam guardados na Google Drive. E com a existência de dois locais de armazenamentos, tornou-se necessário, garantir a sincronização adequada entre ambos, evitando inconsistências entre eles.

AVALIAÇÃO

Neste capítulo são descritos os métodos de avaliação utilizados para avaliar e validar as duas componentes da solução desenvolvida. Primeiramente, serão apresentados os métodos de avaliação utilizados para avaliar a usabilidade da interface da componente cliente e os respectivos resultados obtidos 6.1; posteriormente, serão apresentados os métodos de avaliação, utilizados para avaliar o desempenho sob carga e a segurança e o controlo de acessos do servidor da aplicação FitOps 6.2.

6.1 Avaliação da interface

Para avaliar a interface gráfica da aplicação, foram conduzidos testes de usabilidade com os futuros utilizadores da aplicação, participando no total 5 técnicos pertencentes ao projeto FitoAgro.

Para a realização destes testes, foi necessário conceder aos técnicos acesso à aplicação, distribuindo-os conforme o tipo de utilizador que se aplicava ao seu caso de uso. Em seguida foi pedido que testassem as funcionalidades chaves, relacionadas ao seu tipo de utilizador. Após a conclusão dos testes, foi pedido aos participantes, que respondessem a um inquérito que se divide em três partes, sendo a primeira parte direcionada à avaliação geral da usabilidade da interface através do sistema SUS 6.1.1, enquanto o foco da segunda é avaliar a facilidade de uso e utilidade das principais funcionalidades da aplicação 6.1.2, a última parte do inquérito, contem duas afirmações, sendo a primeira para avaliar a satisfação dos participantes com a solução fornecida para a gestão do sistema FitoAgro, enquanto a segunda é uma questão de resposta aberta, onde os participantes podem indicar sugestões para funcionalidades futuras.

6.1.1 SUS

Para avaliação da usabilidade da interface da aplicação foi utilizada a *System Usability Scale* (SUS) [6]. Para um participante avaliar a interface perante a escala, é necessário que este expresse a sua concordância com dez afirmações, apresentadas no início do inquérito. A concordância com cada afirmação é expressa através da escolha de uma opção, numa escala de *Likert* de cinco valores, onde o primeiro valor corresponde a "discordo totalmente" e o último a "concordo totalmente". As afirmações apresentadas aos participantes foram:

1. Gostaria de usar esta aplicação regularmente.
2. A aplicação é mais complexa que o necessário.
3. Considero a aplicação fácil de usar.
4. Necessito da ajuda de um profissional para utilizar a aplicação.
5. Considero que as várias funcionalidades foram bem integradas.
6. Considero que aplicação contém demasiadas inconsistências.
7. Penso que a maioria das pessoas aprenderia a usar esta aplicação rapidamente.
8. Achei a aplicação muito complicada de usar.
9. Senti-me confiante a usar a aplicação.
10. Preciso de aprender sobre vários assuntos antes de estar apto a utilizar a aplicação.

Para calcular a pontuação SUS de um participante é necessário converter cada resposta em um valor numérico de 1 a 5 e de seguida efetuar as seguintes conversões: Para as afirmações com índice ímpar (1, 3, 5, 7, 9), subtrai-se 1 do valor numérico obtido na etapa anterior. Para afirmações com índice par (2, 4, 6, 8, 10), subtrai-se o valor numérico obtido na etapa anterior a 5. Após essas conversões, soma-se o valor das dez afirmações e multiplica-se pela constante 2,5 obtendo-se um valor compreendido entre 0 e 100. Após calcular os valores de todos os participantes e para terminar, é calculada a média, sendo este o resultado final da avaliação da interface na *System Usability Scale* (SUS).

Após ser obtido o resultado final é possível interpretá-lo com a ajuda do gráfico representado na figura 6.1.

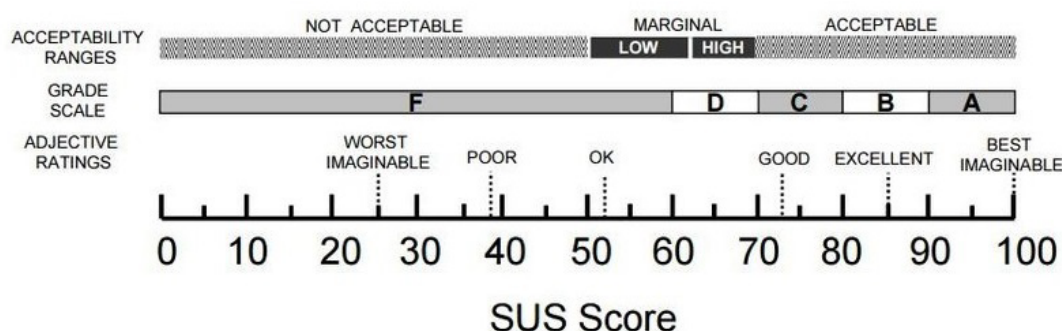


Figura 6.1: Gráfico SUS

Os resultados das respostas dos inquéritos encontram-se apresentados em 6.2.

Após calcular a média dos resultados dos participantes, chegou-se ao valor 73. Sendo classificada segundo o gráfico apresentado em 6.1 com uma nota C, sendo considerada uma solução aceitável e "boa".

Afirmção/Utilizador	1	2	3	4	5
Gostaria de usar esta aplicação regularmente.	3	5	5	4	4
A aplicação é mais complexa que o necessário.	3	2	1	5	4
Considero a aplicação fácil de usar.	3	5	5	5	4
Necessito da ajuda de um profissional para utilizar a aplicação.	4	1	2	4	2
Considero que as várias funcionalidades foram bem integradas.	4	4	5	4	4
Considero que aplicação é contém demasiadas inconsistências.	2	2	2	2	2
Penso que a maioria das pessoas aprenderia a usar esta aplicação rapidamente.	3	5	4	4	4
Achei a aplicação muito complicado de usar.	2	1	1	1	2
Senti-me confiante a usar a aplicação.	3	5	5	4	3
Preciso de aprender sobre vários assuntos antes de estar apto a utilizar a aplicação.	4	2	1	4	2
	52,5	90	92,5	62,5	67,5

Figura 6.2: Resultados SUS

6.1.2 Usabilidade das funcionalidades

Para avaliar individualmente a usabilidade das várias funcionalidade da aplicação, foram adicionadas ao inquérito afirmações, em relação à facilidade de utilização e utilidade de cada funcionalidade. Tal como nas afirmações para a avaliação SUS, estas também são avaliadas usando a mesma escala de *Likert* de cinco valores.

As primeiras vinte e nove afirmações são referentes as funcionalidades comuns a todos os tipos de utilizadores.

1. Considero fácil listar e filtrar as estações meteorológicas.
2. Considero útil listar e filtrar as estações meteorológicas.
3. Considero fácil consultar a informação de uma estação meteorológica.
4. Considero útil consultar a informação de uma estação meteorológica.
5. Considero fácil listar e filtrar as parcelas que tenho acesso.
6. Considero útil listar e filtrar as parcelas que tenho acesso.
7. Considero fácil consultar os detalhes de uma parcela.
8. Considero útil consultar os detalhes de uma parcela.
9. Considero fácil consultar os elementos de interesse de uma parcela.
10. Considero útil consultar os elementos de interesse de uma parcela.
11. Considero fácil consultar os períodos de uma parcela.
12. Considero útil consultar os períodos de uma parcela.
13. Considero fácil consultar os inimigos estudados, durante um período, numa parcela.
14. Considero útil consultar os inimigos estudados, durante um período, numa parcela.
15. Considero fácil consultar os utilizadores associados à parcela.
16. Considero útil consultar os utilizadores associados à parcela.

17. Considero fácil consultar as visitas efetuadas na parcela.
18. Considero útil consultar as visitas efetuadas na parcela.
19. Considero fácil corrigir os dados de uma visita.
20. Considero útil corrigir os dados de uma visita.
21. Considero fácil listar e filtrar os períodos do sistema.
22. Considero útil listar e filtrar períodos do sistema.
23. Considero fácil consultar os detalhes de um período.
24. Considero útil consultar os detalhes de um período.
25. Considero fácil listar e filtrar organizações.
26. Considero útil listar e filtrar organizações.
27. Considero fácil consultar a informação de uma organização.
28. Considero útil consultar a informação de uma organização.

De seguida, é apresentada uma questão onde o participante tem de seleccionar o tipo de utilizador que interpretou, durante os testes de usabilidade. Conforme a resposta, as seguintes afirmações do inquérito serão diferentes, havendo tipos de utilizador que terão questões em comum, devido ao facto de ambos terem acesso a funcionalidades, que não são comuns a todos os utilizadores da aplicação. As respostas a estas afirmações repetidas, serão analisadas em conjunto.

Se o participante agiu como um **Agente de Campo** foram-lhe apresentadas as seguintes afirmações:

1. Considero fácil listar e filtrar os protocolos.
2. Considero útil listar e filtrar os protocolos.
3. Considero fácil consultar os detalhes de um protocolo.
4. Considero útil consultar os detalhes de um protocolo.
5. Considero fácil consultar as visitas efetuadas por mim.
6. Considero útil consultar as visitas efetuadas por mim.

Se o participante agiu como um **Produtor** foram-lhe apresentadas as seguintes afirmações:

1. Considero fácil gerir (adicionar/editar) parcelas.
2. Considero útil gerir (adicionar/editar) parcelas.
3. Considero fácil gerir (adicionar/editar/remover) os elementos de interesse de uma parcela.
4. Considero útil gerir (adicionar/editar/remover) os elementos de interesse de uma parcela.

5. Considero fácil gerir (adicionar/remover) os períodos de uma parcela.
6. Considero útil gerir (adicionar/remover) os períodos de uma parcela.
7. Considero fácil gerir (adicionar/remover) os inimigos estudados durante um período em uma parcela.
8. Considero útil gerir (adicionar/remover) os inimigos estudados durante um período em uma parcela.

Se o participante agiu como um **Organização** foram lhe apresentadas as seguintes afirmações:

1. Considero fácil gerir (adicionar/editar) parcelas.
2. Considero útil gerir (adicionar/editar) parcelas.
3. Considero fácil gerir (adicionar/editar/remover) os elementos de interesse de uma parcela.
4. Considero útil gerir (adicionar/editar/remover) os elementos de interesse de uma parcela.
5. Considero fácil gerir (adicionar/remover) os períodos de uma parcela.
6. Considero útil gerir (adicionar/remover) os períodos de uma parcela.
7. Considero fácil gerir (adicionar/remover) os inimigos estudados durante um período em uma parcela.
8. Considero útil gerir (adicionar/remover) os inimigos estudados durante um período em uma parcela.
9. Considero fácil gerir (adicionar/remover) os utilizadores de uma parcela.
10. Considero útil gerir (adicionar/remover) os utilizadores de uma parcela.
11. Considero fácil editar os detalhes da minha organização.
12. Considero útil editar os detalhes da minha organização.
13. Considero fácil gerir (adicionar/aceitar/rejeitar/remover) os membros da minha organização.
14. Considero útil gerir (adicionar/aceitar/rejeitar/remover) os membros da minha organização.

Se o participante agiu como um **Administrador** foram lhe apresentadas as seguintes afirmações:

1. Considero fácil gerir (adicionar/editar/remover) as estações meteorológicas.
2. Considero útil gerir (adicionar/editar/remover) as estações meteorológicas.
3. Considero fácil gerir (adicionar/editar/remover) as contas de fornecedores.

4. Considero útil gerir (adicionar/editar/remover) as contas de fornecedores.
5. Considero fácil consultar as estações de uma conta de fornecedor.
6. Considero útil consultar as estações de uma conta de fornecedor.
7. Considero fácil gerir (editar/remover) parcelas.
8. Considero útil gerir (editar/remover) parcelas.
9. Considero fácil gerir (adicionar/editar/remover) os elementos de interesse de uma parcela.
10. Considero útil gerir (adicionar/editar/remover) os elementos de interesse de uma parcela.
11. Considero fácil gerir (adicionar/remover) os períodos de uma parcela.
12. Considero útil gerir (adicionar/remover) os períodos de uma parcela.
13. Considero fácil gerir (adicionar/remover) os inimigos estudados durante um período em uma parcela.
14. Considero útil gerir (adicionar/remover) os inimigos estudados durante um período em uma parcela.
15. Considero fácil gerir (adicionar/duplicar/editar/remover) períodos.
16. Considero útil gerir (adicionar/duplicar/editar/remover) períodos.
17. Considero fácil gerir (adicionar/editar/remover) organizações.
18. Considero útil gerir (adicionar/editar/remover) organizações.
19. Considero fácil consultar todas as visitas.
20. Considero útil consultar todas as visitas.
21. Considero fácil consultar os protocolos.
22. Considero útil consultar os protocolos.
23. Considero fácil adicionar protocolos.
24. Considero útil adicionar protocolos.
25. Considero fácil gerir (consultar/editar/remover) utilizadores.
26. Considero útil gerir (consultar/editar/remover) utilizadores.
27. Considero fácil gerir (adicionar/editar/remover) entidades base do sistema.
28. Considero útil gerir (adicionar/editar/remover) entidades base do sistema.

As respostas dos participantes, às afirmações anteriormente listadas foram divididas em dois grupos de forma a avaliar, a facilidade de utilização e utilidade das funcionalidades implementadas.

Após a análise das respostas da facilidade 6.3, concluiu-se que, os técnicos concordam que as funcionalidades são fáceis de utilizar. Através do cálculo da média de todas as funcionalidades, concluiu-se que o valor médio da facilidade é 4,23 numa escala de 1 a 5, o que indica que, de um modo geral, a aplicação é fácil de utilizar. Durante os testes os participantes expressaram, a necessidade da aplicação ter uma explicação prévia das suas funcionalidades. Referiram ainda que, após compreender o seu funcionamento, as funcionalidades se tornam fáceis de reproduzir.

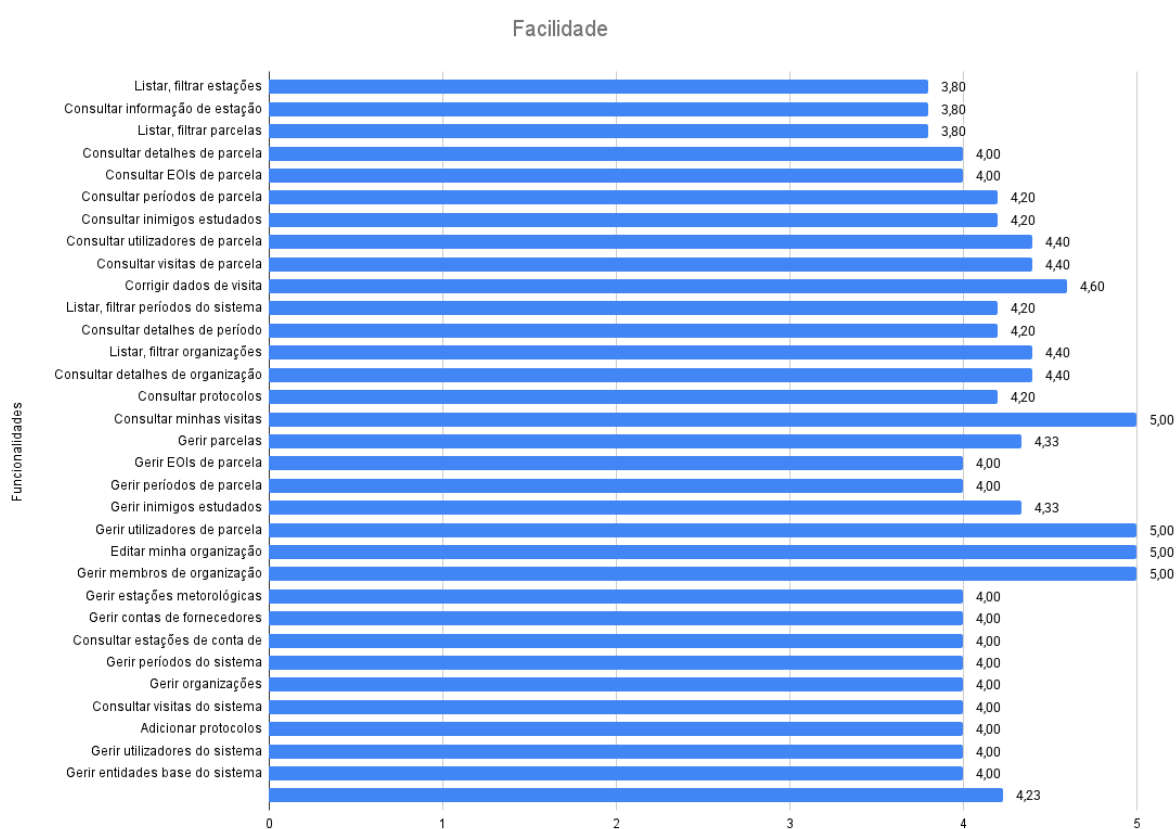


Figura 6.3: Resultados de facilidade

Após a análise das respostas da utilidade 6.4, concluiu-se que, as funcionalidades tem uma utilidade elevada. E através do cálculo da média de todas as funcionalidades dos valores da utilidade, chegou-se ao valor de 4,33 numa escala de 1 a 5, o que indica que, a aplicação desenvolvida é bastante útil e relevante para o projeto FitoAgro.

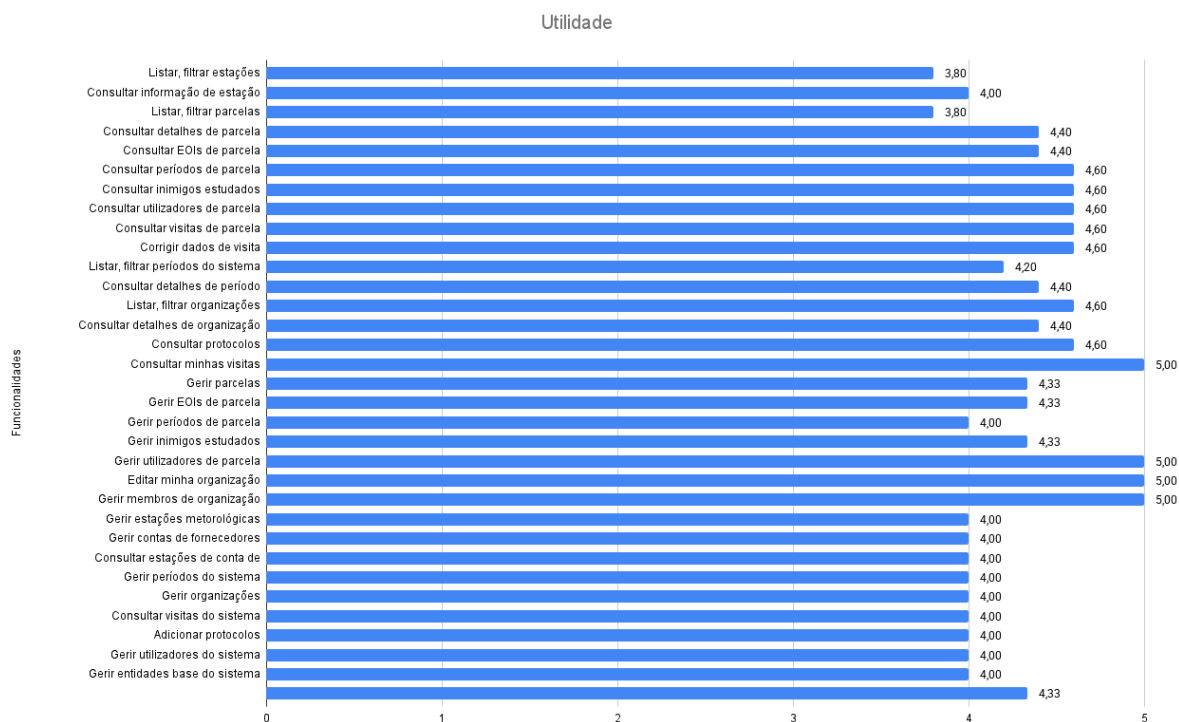


Figura 6.4: Resultados de utilidade

6.2 Avaliação do Servidor

Nesta secção são apresentados os métodos de avaliação e discutidos os resultados obtidos da componente servidor da aplicação, sendo avaliado o seu desempenho sob carga e a sua segurança e o controlo de acessos.

Foi usada a ferramenta *Postman* [34] para efetuar os pedidos *HTTP* necessários para a realização dos testes. Foi escolhida esta ferramenta, porque é bastante intuitiva de usar e oferece funcionalidades, como realização de **scripts** que podem ser executados antes de realizar o pedido ou depois de receber a resposta do mesmo. É possível também agrupar pedidos em coleções, onde se pode definir constantes e variáveis para uso nos pedidos, como por exemplo o endereço base do servidor. Podemos ainda combinar as duas funcionalidades referidas e criar um *script* que armazene o *token* recebido na resposta de um pedido de autenticação nas variáveis da coleção, para que este possa ser enviado em pedidos futuros, com o intuito de autenticar estes.

Para além das funcionalidades anteriormente referidas, a ferramenta *Postman* tem uma funcionalidade dedicada para testes de carga. Nesta funcionalidade é possível definir a duração do teste, o de número processos a realizar pedidos concorrentemente (utilizadores virtuais) e o perfil de carga, que consiste na variação de utilizadores virtuais, ao longo do tempo do teste. No final do teste, a ferramenta fornece um relatório completo, com vários valores pertinentes para o estudo do desempenho.

6.2.1 Testes de Carga

Para avaliar o desempenho do servidor quando está sob carga foi criado um cenário hipotético, em que vários utilizadores tentam aceder aos dados recolhidos do último mês de três estações meteorológicas diferentes e tentam aceder aos detalhes de três parcelas diferentes. Estas funcionalidades foram escolhidas porque o pedido que devolve os dados recolhidos por uma estação meteorológica é aquele que retorna a maior quantidade de dados, enquanto o pedido que apresenta os detalhes de uma parcela é o que combina o maior número de tabelas, da base de dados.

Com o objetivo de identificar as limitações do servidor desenvolvido, foram realizados testes com uma duração de um minuto, utilizando diferentes números de utilizadores virtuais, especificamente, 20, 50 e 100, que realizariam os pedidos mencionados de forma contínua durante a duração do teste.

<i>Users</i>	<i>Lat (ms)</i>	<i>Min (ms)</i>	<i>Max (ms)</i>	<i>Error %</i>	<i>Throughput (Req/s)</i>	<i>Total (Req)</i>
20	1960	81	5484	0.00	8.33	568
50	4980	114	11905	0.00	8.25	564
100	8977	182	22803	0.00	8.06	544

Tabela 6.1: Métricas retiradas dos testes de carga efetuados com **uma** instância do servidor

Observados os resultados obtidos (tabela 6.1) é possível verificar que o aumento da latência (*Lat*) é proporcional ao aumento do número de utilizadores virtuais, indicando que o servidor teve problemas em lidar com a carga. Adicionalmente o máximo tempo de resposta (*Max*) chegou aos vinte e dois segundos com cem utilizadores, um valor demasiado alto. É também de notar que dos cinquenta para os cem utilizadores, existiu uma diminuição considerável no número total de pedidos processados, podendo ser indício que o servidor estava a ter maior dificuldade em suportar o aumento na carga. Por fim, é importante salientar que a taxa de erro manteve-se em zero por cento ao longo dos testes, indicando que apesar do servidor se tornar mais lento, este priorizou responder corretamente aos pedidos recebidos.

Os testes apresentados anteriormente, foram efetuados com uma única instância do servidor a correr, sendo de esperar o aumento no tempo de resposta deste, pois não era possível dividir a carga com mais nenhuma instância. Para que fosse possível avaliar o impacto da escalabilidade horizontal e melhorar o desempenho do servidor sob carga, decidiu-se escalar horizontalmente através da técnica *clustering* que permite criar múltiplas instâncias do servidor, para aproveitar melhor o poder de processamento de processadores com múltiplos núcleos. Permitindo ao servidor escalar horizontalmente dentro de uma única máquina, através da criação de uma instância por cada núcleo. É importante salientar que não é recomendado ocupar todos os núcleos do processador, para que outros processos dentro da máquina não sejam afetados.

<i>Users</i>	<i>Lat (ms)</i>	<i>Min (ms)</i>	<i>Max (ms)</i>	<i>Error %</i>	<i>Throughput (Req/s)</i>	<i>Total (Req)</i>
20	1032	78	4285	0.00	14.28	980
50	2682	89	7120	0.00	15.06	1038
100	5058	92	13564	0.00	15.26	1045

Tabela 6.2: Métricas retiradas dos testes de carga efetuados com **duas** instâncias do servidor

Infelizmente, devido ao número limitado de quatro núcleos do processador da máquina alugada é apenas possível manter duas instâncias em execução. Contudo, mesmo com apenas mais uma instância, os resultados obtidos (tabela 6.2) são consideravelmente melhores. Isto pôde ser constatado, pela redução significativa da latência e tempo máximo de resposta e também pelo aumento para o dobro do *throughput* e número de pedidos processados. É importante realçar que estes valores não diminuíram, pelo contrário, continuam a aumentar com o número de utilizadores virtuais.

Estas melhorias foram um indicador positivo da robustez e capacidade de paralelização do servidor. Era de esperar um aumento mais acentuado se fosse viável manter um maior número de instâncias em execução.

6.2.2 Testes de Segurança

A segurança e o controlo de acessos do sistema são avaliados focando-se no servidor web da aplicação, uma vez que é neste componente que são realizadas as principais verificações. A metodologia de teste consiste na realização de múltiplos pedidos *HTTP* para cada rota da API do servidor, listadas no apêndice B. Estes pedidos tiveram como objetivo de a simulação dos diversos cenários, em que as políticas de acesso podem ser violadas. Os cenários e as respetivas respostas esperadas foram:

- Um utilizador **não autenticado** tenta realizar uma ação no sistema.
A resposta esperada é um erro do tipo *401 Unauthorized*.
- Um utilizador autenticado tenta realizar uma ação que o seu **tipo de conta não permite**.
A resposta esperada é um erro do tipo *403 Forbidden*.
- Um utilizador autenticado tenta realizar uma ação num **recurso que não tem acesso**.
A resposta esperada é um erro do tipo *403 Forbidden*.

Após a realização dos pedidos e análise das suas respostas, foi possível validar o bom funcionamento da segurança e do controlo de acessos do servidor. Sendo o único ponto de preocupação, o caso em que um *token* de acesso é copiado, pois qualquer pessoa com o *token* pode imitar o seu dono e autenticar-se perante o sistema. Para combater parcialmente esta situação, foi definido um período de validade de vinte e quatro horas para cada *token*.

6.3 Conclusão

A diversidade dos testes realizados, tanto para a componente cliente, como para a componente servidor, teve como objetivo validar a aplicação desenvolvida no âmbito desta dissertação.

Para validar a componente cliente, foram realizados testes de utilizador, nos quais os participantes tiveram de descobrir como utilizar as funcionalidades da aplicação sem a ajuda dos desenvolvedores. Para recolher os dados dos testes os participantes tinha de responder a um inquérito sobre a sua experiência com a aplicação. Este inquérito permitia não só avaliar a usabilidade da interface de um modo geral, como permitia avaliar individualmente a utilidade e facilidade de utilização de cada funcionalidade. Após a análise das respostas do inquérito, concluiu-se que a aplicação apresenta uma usabilidade

aceitável e "boa" e que as suas funcionalidades de um modo geral são úteis e fáceis de utilizar. No entanto, no início dos testes, foram detetadas algumas dificuldades no uso da aplicação, estas dificuldades estavam relacionadas com o funcionamento das funcionalidades e as dependências entre estas, e não com a interface em si. Estas dificuldades, eram de esperar dado o elevado número de funcionalidades e dependências entre estas.

Para avaliar a componente servidor, foi testado o desempenho do servidor quando está sob carga, a segurança e o controlo de acessos.

Para testar o desempenho do servidor quando está sob carga, foram escolhidos dos pedidos que a aplicação realiza ao servidor, os pedidos que requerem mais recursos computacionais. E de forma a identificar as limitações do servidor, foram realizados testes em que estes pedidos foram efetuados repetidamente, através de um número variado de processos concorrentes, de forma a simular vários utilizadores, a utilizar a aplicação. Os testes foram efetuados duas vezes, na primeira vez só existia uma instância do servidor, enquanto na segunda existiam duas instâncias ativas. Este segundo grupo de testes teve como objetivo verificar se o escalonamento horizontal do servidor traria vantagens significativas, algo que se confirmou.

Para testar a segurança do servidor, foram enviados pedidos sem o *token* de autenticação para todas as rotas da API que exigissem autenticação, com o objetivo de simular um utilizador não autenticado a tentar utilizar a aplicação. Adicionalmente, para testar o controlo de acessos, para cada rota da API que apresentasse controlo de acessos, foram enviados pedidos que violassem as diversas políticas de controlo de acessos. Após verificar que todos os pedidos realizados receberam um erro de resposta, foi possível confirmar o bom funcionamento do servidor.

Após a realização dos testes às duas componentes da aplicação, pode-se concluir, que a aplicação desenvolvida, atende os requisitos não funcionais inicialmente definidos e listados no apêndice [A](#).

CONCLUSÕES E TRABALHO FUTURO

Este capítulo sumariza os objetivos atingidos e o trabalho efetuado durante esta dissertação 7.1. Adicionalmente, são apresentadas algumas sugestões de funcionalidades a adicionar à aplicação, que podem vir a ser uma mais valia para o sistema 7.2.

7.1 Conclusões

O objetivo principal desta dissertação, foi desenvolver uma aplicação web de *back-office* para o sistema criado, no âmbito do projeto FitoAgro, o qual foi concretizado com sucesso. A aplicação permite a gestão das principais componentes dos sistema, nomeadamente a componente de recolha de informação meteorológica, através da gestão das estações fornecidas pelos diferentes fornecedores e a componente de recolha de informação fitossanitária, através da configuração da aplicação móvel, dos agentes de campo. Adicionalmente através da aplicação FitOps é possível consultar e manipular toda a informação armazenada no sistema de informação do FitoAgro.

A arquitetura da aplicação foi concebida com base em dois grandes objetivos. Sendo o primeiro, a aplicação tem de ser facilmente expandida. Para tal foram escolhidas tecnologias que promovem a escalabilidade, como por exemplo o *Node.js* que possui a funcionalidade de *clustering*, permitindo o fácil escalonamento horizontal do servidor. Foram também selecionadas tecnologias que promovem a integração contínua, como a *framework* *Vue*, cuja sua arquitetura é baseada em componentes, facilitando a sua manutenção e expansão. O segundo objetivo prende-se com a premissa de que o código da aplicação deve ser de fácil compreensão para um novo programador. Para tal, tanto o *framework* do servidor (*Express*), como o *framework* do cliente (*Vue*) são baseados em *Node.js* e utilizam a mesma linguagem de programação (*JavaScript*). É de salientar também, que todas as tecnologias utilizadas beneficiam de uma extensa comunidade de suporte e apresentam uma documentação detalhada.

O principal desafio enfrentado durante o desenvolvimento da solução, foi a integração com a aplicação móvel. A razão para tal prendeu-se com o facto de esta não comunicar com a base de dados, encontrando-se todos os seus dados armazenadas numa pasta da Google Drive. Esta situação complicou o desenvolvimento das várias funcionalidades que influenciam e configuram a mesma. Pois foi necessário garantir a consistência de informação entre dois sistemas distintos, que guardam informação em formatos diferentes.

Para concluir, considero que a solução desenvolvida foi um sucesso, pois cumpre o objetivo proposto e está pronta a ser expandida para que possa satisfazer futuros

requisitos. Espera-se que a aplicação seja usufruída pelas várias entidades que integram o projeto FitoAgro, e presume-se que a aplicação continue a crescer, com a adição de novas componentes, para que melhor sirva o seu propósito. Este crescimento será um testemunho da qualidade do trabalho desenvolvido, durante esta dissertação.

7.2 Trabalho Futuro

Com a evolução do estado da aplicação foram tomadas decisões para priorizar o desenvolvimento de certas funcionalidades, consequentemente houve outras que ficaram em segundo plano. Adicionalmente os utilizadores do projeto sugeriram também algumas funcionalidades. Dessa forma nesta secção serão apresentadas funcionalidades, cuja sua implementação irá aumentar o interesse e utilidade da aplicação.

7.2.1 Integração de dados de satélite

Para complementar os dados recolhidos pelas estações meteorológicas, deveria também ser feita a integração de dados de satélite. Porque a partir destes é possível fazer um estudo mais detalhado das pragas e doenças, pois os satélites têm a capacidade de recolher informação relativa a fatores, que as estações não têm a capacidade de recolher. Fatores estes que influenciam diretamente o desenvolvimento das pragas e doenças.

Também, através destes dados, deverá ser possível melhorar a precisão na análise do estado das culturas, proporcionando uma visão mais detalhada das condições ambientais, que afetam as culturas.

Adicionalmente, foi referido pelos técnicos que as estações meteorológicas são inconsistentes, pois estas avariavam com bastante frequência. No entanto, com a integração dos dados de satélite, os técnicos não estariam dependentes das estações, pois teriam mais uma fonte de informação meteorológica.

7.2.2 Gestão de tratamentos e respetivos resíduos

Para efeitos de rastreabilidade e controlo, deveria ser possível manter um histórico de tratamentos fitossanitários e respetivos resíduos produzidos.

Para cada tratamento, deveriam ser inseridas informações referentes ao tratamento em si e ao responsável pelo procedimento.

Em relação aos resíduos produzidos, seria importante registar o tipo, a quantidade e como estes foram processados. Deste modo tornava-se possível assegurar a conformidade com as normas ambientais e facilitar auditorias, ou inspeções futuras.

Este histórico, ajudaria também na prevenção de aplicação de dosagens excessivas ou insuficientes, permitindo assim aos utilizadores da aplicação, manter um maior controlo sobre os seus tratamentos.

7.2.3 Alertas

Seria interessante implementar um sistema que permitisse aos utilizadores emitir alertas, dando a capacidade de utilizadores e organizações avisarem os seus parceiros, no projeto FitoAgro.

Este sistema poderia trazer grandes benefícios para os utilizadores em termos de prevenção de problemas.

No entanto, a permissão de envio de alertas sem qualquer tipo de controlo e moderação poderia ser arriscado, uma vez que existe a possibilidade de serem enviados alertas falsos.

Para controlar esta funcionalidade, uma opção seria a verificação dos alertas por uma entidade moderadora, antes de estes serem propagados para os outros utilizadores. No entanto, isso poderia causar atrasos na transmissão de informação, o que possivelmente seria prejudicial no caso de o alerta ser verdadeiro. Em alternativa, em vez de ser necessário a aprovação dos alertas, podemos dar a opção de reportar alertas possivelmente nocivos, que por sua vez irão ser marcados para serem revistos, por uma entidade moderadora.

Outra alternativa seria a implementação de um sistema de *publish & subscribe*, ou seja, os utilizadores só receberiam alertas de entidades que confiam e que tenham subscrito.

7.2.4 Sistema de mensagens

Após reunir com os técnicos tornou-se claro que estes sentem necessidade de comunicar entre si, para discutir diversos assuntos relacionados ao projeto. Para tal, sugere-se a implementação de um sistema que permita enviar mensagens entre utilizadores ou a criação de grupos, com vários utilizadores.

No entanto, permitir o envio de mensagens e a criação de grupos sem qualquer moderação ou controlo pode ser perigoso para o sistema, pois facilita a ação de utilizadores mal-intencionados que poderiam abusar desta funcionalidade. Estes utilizadores poderiam fazê-lo de várias formas. Tal como através da disseminação de spam, que é algo incómodo para os utilizadores e poderia sobrecarregar o sistema em casos extremos, a partilha de conteúdos inapropriados, promoção de publicidade indesejada e a disseminação de desinformação. Para evitar esses riscos, seria essencial implementar medidas de prevenção.

Como a limitação de frequência de envio de mensagens em grupos, onde cada utilizador, só é permitido o envio de um certo número de mensagens, num determinado intervalo de tempo.

Como a existência de sistemas de denúncia, onde os utilizadores, podem reportar mensagens ou outros utilizadores.

Como um sistema de reputação de utilizador, onde é atribuída a cada utilizador uma pontuação inicial no momento da criação da conta, pontuação esta que pode ser afetada com base nas suas ações. E caso esta pontuação se torne suficientemente baixa, o utilizador ficará com certas ações no sistema restringidas.

Em conclusão, um sistema de mensagens seria sem dúvida uma mais valia para o projeto, pois permitiria contribuir para a construção de uma comunidade. No entanto, seria crucial que fosse cuidadosamente planeado e desenvolvido, uma vez que poderia ficar vulnerável, a diferentes formas de abuso e utilização indevida.

BIBLIOGRAFIA

- [21] V. Agafonkin. *Leaflet*. URL: <https://leafletjs.com/> (acedido em 2024-04-30) (ver p. 27).
- [1] Agroptima. *Agroptima*. URL: <https://www.agroptima.com/en/> (acedido em 2024-04-25) (ver p. 20).
- [4] Axios. URL: <https://axios-http.com/> (acedido em 2024-10-13) (ver p. 66).
- [6] J. Brooke. «SUS: A quick and dirty usability scale». Em: *Usability Evaluation in Industry*. Ed. por P. W. Jordan et al. Taylor & Francis, 1996, pp. 189–194 (ver p. 81).
- [27] O. Corporation. *MySQL*. URL: <https://www.mysql.com/> (acedido em 2024-04-27) (ver p. 26).
- [9] L. C. Correia. *Modelos Adaptativos para Pragas*. 2020 (ver p. 13).
- [10] M. Â. S. Correia. *Registo de dados baseado em protocolos de recolha de informação*. 2021 (ver pp. 2, 10, 38, 61).
- [7] COTHN. *Cecidómia (Dasineura pyri Bouché)*. URL: <https://infoagro.cothn.pt/portal/index.php?id=1307> (acedido em 2024-04-15) (ver p. 6).
- [13] COTHN. *Filoxera (Aphanostigma pyri Chol.)*. URL: <https://infoagro.cothn.pt/portal/index.php?id=1320> (acedido em 2024-04-15) (ver p. 5).
- [43] COTHN. *Traça Oriental (Grapholita molesta Busck)*. URL: <https://infoagro.cothn.pt/portal/index.php?id=1879> (acedido em 2024-04-15) (ver p. 7).
- [37] Cropin. *SmartFarm*. URL: <https://www.cropin.com/intelligent-agriculture-cloud/cropin-apps/farm-management-solution> (acedido em 2024-04-25) (ver p. 21).
- [8] N. Downie. *Chart.js*. URL: <https://www.chartjs.org/> (acedido em 2024-04-30) (ver pp. 27, 66).
- [11] European Commission. *Farm to Fork Strategy: For a fair, healthy and environmentally-friendly food system*. 2020. URL: https://food.ec.europa.eu/horizontal-topics/farm-fork-strategy_en (ver p. 3).
- [12] M. d'Amaral Ferreira. *Visualização Interativa de Dados para Projetos de Monitorização Fitossanitária*. 2021 (ver pp. 2, 14).
- [2] Google. *Angular*. URL: <https://angular.io/> (acedido em 2024-04-29) (ver p. 29).
- [14] Google. *Flutter*. URL: <https://flutter.dev/> (acedido em 2024-04-29) (ver p. 29).

- [16] Google. *Google Drive API v3*. URL: <https://developers.google.com/drive/api/v3/about-sdk> (acedido em 2024-10-13) (ver p. 68).
- [33] P. G. D. Group. *PostgreSQL*. URL: <https://www.postgresql.org/> (acedido em 2024-04-27) (ver p. 25).
- [39] D. R. Hipp. *SQLite*. URL: <https://www.sqlite.org/index.html> (acedido em 2024-04-27) (ver p. 26).
- [34] P. Inc. *Postman*. URL: <https://www.postman.com/> (acedido em 2024-04-29) (ver p. 88).
- [17] *Inovação para Agricultura, FitoAgro*. URL: <https://inovacao.rederural.gov.pt/2/87-fitoagro> (ver pp. 2, 5).
- [29] P. Instruments. *Pessl Instruments*. URL: <https://metos.at/en/> (acedido em 2024-04-15) (ver p. 13).
- [20] *JWT*. URL: <https://jwt.io/> (acedido em 2024-10-13) (ver p. 66).
- [22] D. A. S. Lopes. *Sistema de Informação para Monitorização de Faixas de Gestão de Combustível de Incêndios*. 2023 (ver p. 8).
- [23] J. M. Lourenço. *The NOVAtesis L^AT_EX Template User's Manual*. NOVA University Lisbon. 2021. URL: <https://github.com/joaomlourenco/novathesis/raw/main/template.pdf> (ver p. i).
- [19] J. Ltd. *JSCharting*. URL: <https://jscharting.com/> (acedido em 2024-04-30) (ver p. 28).
- [24] A. Malafaia. *Sistema de Informação para Projetos de Monitoração Fitossanitária*. 2019 (ver pp. 2, 13, 14, 80).
- [38] G. Manghi. *SpatiaLite*. URL: <https://www.gaia-gis.it/fossil/libspatialite/index> (acedido em 2024-04-27) (ver p. 26).
- [25] Mapbox. *Mapbox*. URL: <https://www.mapbox.com/> (acedido em 2024-04-30) (ver pp. 26, 66).
- [18] I. P. do Mar e da Atmosfera (IPMA). *IPMA*. URL: <https://www.ipma.pt/pt> (acedido em 2024-04-15) (ver p. 13).
- [26] A. J. A. Martins. *As cochonilhas-algodão da vinha (Hemiptera:Pseudococcidae) na região do Oeste*. 2009. URL: <https://www.repository.utl.pt/handle/10400.5/2053> (ver p. 6).
- [35] Meta. *React*. URL: <https://react.dev/> (acedido em 2024-04-29) (ver p. 29).
- [28] MetaCarta. *OpenLayers*. URL: <https://openlayers.org/> (acedido em 2024-04-30) (ver p. 27).
- [15] Microsoft. *Github Pages*. URL: <https://pages.github.com/> (acedido em 2024-10-13) (ver p. 67).
- [30] E. S. M. Morote. *Pinia*. URL: <https://pinia.vuejs.org/> (acedido em 2024-10-13) (ver p. 66).
- [5] M. Otto e J. Thornton. *Bootstrap*. URL: <https://getbootstrap.com/> (acedido em 2024-10-13) (ver p. 28).
- [31] F. Pinto. *Sistema para acompanhamento da cultura do brócolo*. 2017 (ver p. 17).

- [36] T. Porter. *Sequelize*. URL: <https://sequelize.org/> (acedido em 2024-10-13) (ver p. 67).
- [3] A. dos Produtores Agrícolas da Sobrena (APAS). *Associação dos Produtores Agrícolas da Sobrena (APAS)*. URL: <https://apas.com.pt/> (acedido em 2024-04-15) (ver p. 13).
- [40] Syngenta. *Cropwise Operations*. URL: <https://ua.cropwise.com/en/operations> (acedido em 2024-04-25) (ver p. 18).
- [32] P. D. Team. *PostGIS*. URL: <https://postgis.net/> (acedido em 2024-04-27) (ver p. 25).
- [41] T. Technologies. *TerraPro*. URL: <https://terra-pro.net/> (acedido em 2024-04-15) (ver p. 13).
- [42] *The GeoJSON Format Specification*. URL: <https://geojson.org> (acedido em 2024-07-26) (ver p. 44).
- [44] Vuetify. *Vuetify*. URL: <https://vuetifyjs.com/> (acedido em 2024-10-13) (ver p. 65).
- [45] WiseCrop. *WiseCrop*. URL: <https://www.wisecrop.com/> (acedido em 2024-04-25) (ver pp. 13, 19).
- [0] E. You. *Vue.js*. URL: <https://vuejs.org/> (acedido em 2024-04-29) (ver p. 28).

LISTAGEM DOS REQUISITOS

De seguida são apresentados os requisitos funcionais e não funcionais da aplicação.

Requisitos Funcionais

- Registo e autenticação de utilizadores;

Os seguintes requisitos consistem em ações que só podem ser efetuadas se o utilizador tiver permissão para efetuar certa ação (criar, ver, editar, apagar) sobre certo recurso. Quanto é usado o termo "gerir" são implicadas os quatro tipos de ação.

- Visualizar informações de utilizador;
- Atualizar informações de utilizador;
- Criar/Apagar organizações;
- Visualizar organizações;
- Atualizar informação de organizações;
- Criar ou apagar parcelas;
- Visualizar parcelas;
- Atualizar dados de parcela;
- Criar ou apagar campanha para parcela;
- Associar/Desassociar utilizador a organização;
- Associar/Desassociar parcela a utilizador;
- Criar estações meteorológicas virtuais;
- Atualizar a estação meteorológica de uma parcela;
- Carregar dados de estação meteorológica;
- Configurar estação meteorológica;
- Atualizar protocolo a ser efetuado numa parcela;

-
- Criar protocolos novos;
 - Ativar/Desativar dispositivos móveis de técnicos remotamente;
 - Atribuir/remover parcelas de um técnico;
 - Visualizar visitas de campo;
 - Atualizar dados da visita de campo;

Requisitos Não Funcionais

- Plataforma altamente adaptável e configurável;
- Componente *Cloud* robusta, escalável, gerível;
- Mecanismos de visualização intuitivos;
- Segurança através de níveis de informação e confidencialidade;

ROTAS DA API

De seguida são listadas todas as rotas da API do servidor web da aplicação FitOps.

- **Segurança**

- **POST** *api/token_validate*
- **GET** *api/verifyEmail/:token*
- **POST** *api/forgotPassword*
- **POST** *api/resetPassword*

- **Visitas**

- **GET** *api/visits*
- **GET** *api/visit_detail*
- **PATCH** *api/observation_update/eois*
- **PATCH** *api/observation_update/traps*
- **PATCH** *api/observation_update/labs*

- **Protocolos**

- **GET** *api/protocols*
- **GET** *api/protocols/:protocol_name*
- **POST** *api/protocols*

- **Utilizadores**

- **GET** *api/agents*
- **GET** *api/freeagents*
- **GET** *api/agents/:agent_id*
- **POST** *api/agents*
- **POST** *api/agents/administrador*
- **PATCH** *api/agents/me*
- **PATCH** *api/agents/change_password*
- **PATCH** *api/agents/:agent_id*

-
- DELETE *api/agents/:agent_id*
 - **Contas de fornecedor**
 - GET *api/providersaccounts*
 - POST *api/providersaccounts*
 - PATCH *api/providersaccounts/:agent_id*
 - DELETE *api/providersaccounts/:agent_id*
 - **Tipos de utilizadores**
 - GET *api/agenttypes/roles*
 - GET *api/agenttypes*
 - GET *api/agenttypes/:agenttype_id*
 - POST *api/agenttypes*
 - **Utilizadores em parcelas**
 - GET *api/agentsregions*
 - POST *api/agentsregions*
 - PATCH *api/agentsregions/:agentregion_id*
 - DELETE *api/agentsregions/:agentregion_id*
 - **Culturas**
 - GET *api/crops*
 - GET *api/croptypes/:croptype_id/crops*
 - POST *api/crops*
 - PATCH *api/crops/:crop_id*
 - DELETE *api/crops/:crop_id*
 - **Famílias de culturas**
 - GET *api/croptypes*
 - POST *api/croptypes*
 - PATCH *api/croptypes/:croptype_id*
 - DELETE *api/croptypes/:croptype_id*
 - **Fornecedores**
 - GET *api/dataproviders*
 - GET *api/dataproviders/:dataprovider_id*
 - **Estações meteorológicas**
 - GET *api/dataproviders/:dataprovider_id/datasources*
 - GET *api/closedatasources*

- GET *api/datasources/:datasource_id*
- GET *api/datasources/:datasource_id/data*
- POST *api/virtualdatasources*
- POST *api/dataproviders/:dataprovider_id/datasources*
- PATCH *api/datasources/:datasource_id*
- PATCH *api/virtualdatasources/:datasource_id*
- DELETE *api/datasources/:datasource_id*
- **Inimigos**
 - GET *api/enemies*
 - POST *api/enemies*
 - PATCH *api/enemies/:enemy_id*
 - DELETE *api/enemies/:enemy_id*
- **Tipos de inimigo**
 - GET *api/enemytypes*
 - POST *api/enemytypes*
 - PATCH *api/enemytypes/:enemytype_id*
 - DELETE *api/enemytypes/:enemytype_id*
- **Inimigos estudados, no decorrer de um período, numa parcela**
 - GET *api/regionsinperiods/:regioninperiod_id/enemiesmonitored*
 - POST *api/enemiesmonitored*
 - DELETE *api/enemiesmonitored/:enemymonitored_id*
- **Elementos de interesse (EOIs)**
 - GET *api/regions/:region_id/eois*
 - GET *api/eois/:eoi_id*
 - POST *api/regions/:region_id/eois*
 - PATCH *api/eois/:eoi_id*
 - DELETE *api/eois/:eoi_id*
- **Tipos de EOIs**
 - GET *api/eoitypes*
- **Tipos de eventos**
 - GET *api/eventtypes*
 - POST *api/eventtypes*
 - PATCH *api/eventtypes/:eventtype_id*
 - DELETE *api/eventtypes/:eventtype_id*

- **Tipos de media**

- GET *api/mediatypes*
- POST *api/mediatypes*
- PATCH *api/mediatypes/:mediatype_id*
- DELETE *api/mediatypes/:mediatype_id*

- **Membros de uma organização**

- GET *api/membertypes*
- POST *api/membertypes*
- GET *api/organizations/:organization_id/organizationmembers*
- GET *api/organizations/:organization_id/organizationproducers*
- GET *api/organizations/:organization_id/organizationfieldagents*
- POST *api/organizationmembers*
- PATCH *api/organizationmembers/:organizationmember_id/status*
- DELETE *api/organizationmembers/:organizationmember_id*
- POST *api/organizationmembers/subscribe*
- DELETE *api/organizationmembers/subscribe*

- **Organizações**

- GET *api/organizations*
- GET *api/organizations/:organization_id*
- POST *api/organizations*
- PATCH *api/organizations/:organization_id*
- DELETE *api/organizations/:organization_id*

- **Tipos de organização**

- GET *api/organizationtypes*
- POST *api/organizationtypes*
- PATCH *api/organizationtypes/:organizationtype_id*
- DELETE *api/organizationtypes/:organizationtype_id*

- **Períodos**

- GET *api/periods*
- GET *api/periods/:period_id*
- POST *api/periods*
- POST *api/periods/duplicate*
- PATCH *api/periods/:period_id*
- DELETE *api/periods/:period_id*

- **Tipos de período**
 - GET *api/periodtypes*
 - POST *api/periodtypes*
 - PATCH *api/periodtypes/:periodtype_id*
 - DELETE *api/periodtypes/:periodtype_id*
- **Estados fenológicos**
 - GET *api/phenologicalstages*
- **Protocolos a executar, no decorrer de um período, numa parcela**
 - GET *api/regionsinperiods/:regioninperiod_id/protocolsinregions*
 - POST *api/protocolsinregions*
 - DELETE *api/protocolsinregions/:protocolonregioninperiod_id*
- **Parcelas**
 - GET *api/regions*
 - GET *api/regions/:region_id*
 - GET *api/regions/geometry/:region_id*
 - GET *api/regions/geometry/:region_code*
 - POST *api/regions*
 - PATCH *api/regions/:region_id*
 - PATCH *api/regions/:region_id/closestdatasource*
 - DELETE *api/regions/:region_id*
- **Parcelas em períodos**
 - GET *api/regions/:region_id/regionsinperiods*
 - GET *api/periods/:period_id/regionsinperiods*
 - GET *api/regionsinperiods/:regioninperiod_id*
 - POST *api/regionsinperiods*
 - PATCH *api/regionsinperiods/:regioninperiod_id*
 - DELETE *api/regionsinperiods/:regioninperiod_id*
- **Tipos de parcela**
 - GET *api/regiontypes*
 - POST *api/regiontypes*
 - PATCH *api/regiontypes/:regiontype_id*
 - DELETE *api/regiontypes/:regiontype_id*
- **Tipo de informação meteorológica fornecida por um fornecedor**
 - GET *api/dataprovider/:dataprovider_id)/seriesvariables*

- **Variedades de cultura**

- GET *api/crops/:crop_id/varieties*
- POST *api/varieties*
- PATCH *api/varieties/:variety_id*
- DELETE *api/varieties/:variety_id*

- **Comunicar com os sistemas dos fornecedores**

- GET *api/apas/:agent_id/stations*
- POST *api/wisecrop/auth*
- GET *api/wisecrop/:agent_id/stations*
- GET *api/pessl/:agent_id/stations*
- POST *api/pessl/:agent_id/addstation*
- POST *api/terrapro/auth*
- GET *api/terrapro/:agent_id/stations*





2025 Operação de Plataforma Colaborativa para Agricultura: Miguel Baião

