UNIVERSIDADE DE TRÁS OS MONTES E ALTO DOURO

2º Ciclo em Engenharia Informática

Plataforma Móvel Para Gestão De Dados Em Vinhas

Dissertação de Mestrado em Engenharia Informática

Filipe José Borges de Carvalho

Orientador: Carlos Manuel José Alves Serôdio

Co-orientador: Pedro Miguel Mestre Alves da Silva

Vila Real, setembro de 2018

UNIVERSIDADE DE TRÁS OS MONTES E ALTO DOURO

2º Ciclo em Engenharia Informática

Plataforma Móvel Para Gestão De Dados Em Vinhas

Dissertação de Mestrado em Engenharia Informática

Filipe José Borges de Carvalho

Orientador: Carlos Manuel José Alves Serôdio

Co-orientador: Pedro Miguel Mestre Alves da Silva

Dissertação submetida à Universidade de Trás-os-Montes e Alto Douro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, elaborada sob a orientação de Carlos Manuel José Alves Serôdio e coorientação de Pedro Miguel Mestre Alves da Silva da Universidade de Trás-os-Montes e Alto Douro.

Vila Real, setembro de 2018

“Recomeça…
Se puderes,
Sem angústia e sem pressa.
E os passos que deres,
Nesse caminho duro
Do futuro,
Dá-os em liberdade.
Enquanto não alcances
Não descanses.
De nenhum fruto queiras só metade.”

Miguel Torga

Resumo

A atividade vitivinícola no Alto Douro Vinhateiro tem feito com que, cada vez mais, existam empresários decididos a apostar na produção de vinho nesta região. Apesar do despovoamento do território e da consequente falta de mão de obra, tem surgido uma nova economia e uma nova forma de olhar para os socalcos do Douro, mais voltada para a utilização da tecnologia, que estabelece novos ritmos de vida e influencia o quotidiano das gentes desta região.

Apesar dos meios rurais serem cada vez mais desertificados, as pessoas que neles habitam têm se vindo a acostumar à utilização de novas tecnologias, como por exemplo os *smartphones*, o que permite que sejam desenvolvidas aplicações móveis para auxiliar essas pessoas na gestão das tarefas diárias das vinhas e desta forma colmatar a falta de mão de obra.

É neste contexto que surge a ideia da criação de uma plataforma móvel para a gestão de dados em vinhas. Um sistema que permite a gestão/monitorização remota das vinhas e da atividade vitivinícola nelas desenvolvida, com a utilização dos *smartphones*. A deteção de anomalias (como por exemplo um bardo caído numa vinha), o agendamento de trabalhos a realizar (como por exemplo a vindima) ou até mesmo a delimitação de parcelas que contém uma determinada casta são tarefas de gestão/monitorização das vinhas que exigem a presença diária de equipas no terreno para registar e garantir que essas tarefas são realizadas.

Com esta aplicação para dispositivos móveis, as pessoas que diariamente trabalham no terreno podem fornecer dados aos responsáveis pelas vinhas e estes poderão tomar decisões de gestão, com base nesses dados, sem terem de se deslocar ao terreno. Por outro lado, esta aplicação dá ainda a possibilidade aos trabalhadores de terem uma visão mais alargada dos trabalhos que têm para desenvolver de modo a poderem coordená-los da melhor forma possível, concentrando os recursos disponíveis nos locais aonde são realmente necessários.

Palavras-chave: Viticultura de precisão, Gestão de dados, Redução de Custos, Plataforma móvel, *Smartphones*.

Abstract

The wine industry in Alto Douro wine region cause that more and more entrepreneurs determined to bet on wine production in this region. Despite the depopulation of the territory and the consequent lack of manpower, a new economy and a new way of looking at the terraces of Douro have emerged, more focused on the use of technology, which establishes new rhythms of life and influences the daily life of people of this region.

Although rural areas are becoming more and more desertified, the people who inhabit them have become accustomed to the use of new technologies, such as smartphones, which allows the development of mobile applications to assist these people in the management of the daily tasks of vineyards and thus to fill the lack of workforce.

It is in this context that emerges the idea of creating a mobile platform for data management for vineyards. A system that allows the remote management/monitoring of vineyards and the winemaking activity developed in them, with the use of smartphones. Detection of anomalies (for exemple, a vine overturned in a vineyard), the scheduling of works to be carried out (for exemple, harvesting) or even the delimitation of parcels that contains a certain kind of wine variety are tasks of management/monitoring of the vineyards that require the daily presence of teams on the ground to record and ensure that these tasks are carried out.

With this application for mobile devices, people who daily work in the vineyards can provide data to those that are responsible for the vineyards and they can make management decisions based on that data without having to go to the ground. On the other hand, this application also gives workers the possibility to take a broader view of the work they have to develop so that they can coordinate them as best as possible, concentrating the available resources in the places where they are really needed.

Key-words: Precision Viticulture; Data Management; Cost Reduction; Mobile Aplication; Smartphones

Agradecimentos

À Universidade de Trás-os-Montes e Alto Douro, por me ter dado a possibilidade de frequentar a Licenciatura em Comunicação e Multimédia e o Mestrado em Engenharia Informática nesta instituição.

Ao meu orientador, professor Carlos Manuel José Alves Serôdio e coorientador, professor Pedro Miguel Mestre Alves da Silva, que me acompanharam na realização deste trabalho, por todas as horas disponibilizadas, por todos os esclarecimentos de dúvidas, pela compreensão, incentivo, motivação e ensinamento transmitidos.

Aos amigos e colegas da UTAD que ao longo desta etapa estiveram presentes, pelo companheirismo, pela amizade, pelo apoio e pelo seu grande contributo para que fosse possível a concretização desta etapa.

Aos meus pais, Modesto e Fernanda, por todo o apoio e esforço em tornar possível a realização de mais uma etapa no meu percurso académico.

Aos meus avós, Manuel e Lídia pelo exemplo de uma vida humilde e de trabalho, por terem sido os meus segundos pais e se terem dedicado à minha educação de forma tão empenhada, transmitindo-me ensinamentos intemporais.

À minha irmã, Inês, por todos os bons e maus momentos passados, que juntos nos fizeram crescer e encarar a vida com outros olhos, por estar sempre presente nos momentos mais importantes da minha vida e pelo apoio incondicional sempre demonstrado.

À minha namorada, Natália, pelo apoio e carinho facultado ao longo deste período.

A toda a minha família e amigos que me acompanharam ao longo de todo o meu percurso.

A todos os que durante a minha vida me transmitiram os valores que fizeram de mim a pessoa que hoje sou. Principalmente, às minhas professoras primárias Lurdes e Elvira, a todos os professores do Colégio de Nossa Senhora da Boavista e da Escola Secundária de São Pedro, bem como às pessoas de Vila Cova que sempre me ajudaram e ensinaram.

A todos, o meu muito obrigado!

Índice Geral

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

AP – Agricultura de Precisão

API – Application Programming Interface

APK – Android Package

CO2 – Dióxido de Carbono

EUA – Estados Unidos da América

GPS – Global Positioning System

HTTP – Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure

IDE - Integrated Development Environment

iOS – iPhone Operating System

JDK – Java Development Kit

JRE – Java Runtime Environment

MVC – Model View Controller

OSM – Open Street Map

PHP – Hypertext Preprocessor

RF – Radio Frequência

SDK – Software Development Kit

SIG – Sistemas de Informação Geográfica

VP – Viticultura de Precisão

VRT – Variable Rate Technology

XML – Extensible Markup Language

Capítulo 1

1. Introdução

1.1. Contexto e Motivação

A região do Alto Douro Vinhateiro é uma das regiões do mundo com potencial para a produção de vinhos de grande qualidade, devido sobretudo ao seu clima e posição geográfica [43]. Apesar disso, existem fatores que dificultam a produção de vinho nesta região e que têm feito com que a agricultura/viticultura seja encarada cada vez mais de uma forma negativa. O facto das vinhas serem plantadas em socalcos impede que o trabalho possa ser feito através da utilização de máquinas, como acontece noutras regiões do mundo, e por isso a mão de obra humana é a única forma de trabalhar as vinhas do Alto Douro Vinhateiro. Contudo, essa mão de obra é cada vez mais escassa nesta região e por isso, quando está disponível, é muito importante aproveitá-la ao máximo de forma a dar resposta às tarefas mais prioritárias ou às tarefas que necessitam de mais recursos humanos.

É na tentativa de auxiliar os produtores de vinho e o desenvolvimento da viticultura nesta região que surge a motivação para a criação de uma plataforma para dispositivos móveis que permita a gestão de dados em vinhas. Pretende-se com este projeto introduzir alguma inovação tecnológica no trabalho desenvolvido nas vinhas do Alto Douro Vinhateiro e desta forma contribuir para uma modernização do setor, com o intuito de atrair mais investimento para esta região.

Gerir as tarefas diárias de uma vinha e concentrar os (poucos) recursos humanos disponíveis nos locais aonde verdadeiramente são necessários, são os principais contributos que se pretendem dar com o desenvolvimento deste projeto, através do qual os produtores de vinho poderão, por exemplo, agendar trabalhos que requerem a concentração de mais recursos humanos num determinado local (ex.: vindima) e os empreiteiros agrícolas poderão organizar a sua agenda, de forma a mobilizar os seus trabalhadores para executarem as tarefas estipuladas, nos locais previamente definidos pelos produtores de vinho. Também os próprios trabalhadores, que diariamente estão no terreno, podem dar o seu contributo para uma gestão

mais eficaz das vinhas através da inserção, através plataforma, de anomalias que detetem enquanto efetuam o seu trabalho.

Desta forma todas as informações que dizem respeito às diferentes vinhas poderão ser vistas nesta plataforma móvel, o que diminuirá a necessidade de deslocações ao terreno e irá permitir que os recursos humanos se concentrem num determinado local apenas quando forem necessários, poupando-se tempo e dinheiro.

1.2. Objetivos

Na presente dissertação é apresentado o sistema "plataforma móvel para gestão de dados em vinhas" que irá permitir verificar todas as anomalias associadas a uma vinha, agendar e acompanhar todos os trabalhos que deverão ser desenvolvidos e ainda verificar as diferentes parcelas/castas que constituem uma determinada vinha, assim como adicionar e visualizar os limites de terreno que constituem essa parcela.

O objetivo deste projeto é desenvolver uma plataforma móvel que permita monitorizar e gerir as atividades do dia-a-dia de uma vinha através da utilização dos *smartphone*s. Para tal, será desenvolvida uma aplicação para dispositivos móveis que permitirá ao utilizador visualizar, adicionar, modificar e eliminar dados referentes às vinhas, dados esses que estarão alojados numa base de dados e que serão partilhados por outras aplicações que possam vir a fazer parte do projeto onde esta dissertação se enquadra.

As vinhas poderão ser visualizadas num mapa, com imagens de satélite, e as informações sobre as anomalias, os trabalhos programados e as parcelas serão acompanhadas de dados de geolocalização (coordenadas) para que a identificação das tarefas seja efetuada de forma mais rápida e mais fácil. Desta forma pretende-se proporcionar uma gestão das vinhas mais eficaz para que a concentração dos meios no terreno seja efetuada de acordo com as necessidades e para que o processo de cultivo de vinhas e colheita das uvas possa ser otimizado, evoluindo-se assim para um processo de viticultura de precisão.

1.3. Organização da Dissertação

Esta dissertação é composta por 6 capítulos sendo o primeiro, o presente, de introdução, onde é feito o enquadramento dos problemas apresentados, a motivação e os objetivos que espoletaram a realização deste projeto.

O segundo capítulo tem como objetivo apresentar a resenha tecnológica e o levantamento de trabalhos científicos alusivos a esta problemática. São abordados temas como a comparação entre a agricultura tradicional e a agricultura de precisão, as redes de sensores em produção agrícola e a arquitetura de sistemas de computação em vinhas. Neste capítulo são ainda apresentadas as tecnologias a implementar e algumas metodologias de desenvolvimento de *software*.

O terceiro capítulo apresenta a conceção do projeto, onde é feita a análise de requisitos, são definidos o fluxo e a estrutura conceptual da aplicação a desenvolver, o modelo exemplificativo dos *layouts* a implementar e a especificação da estrutura da base de dados.

No capítulo seguinte, que diz respeito à implementação, explica-se todo o processo de implementação da aplicação, desde a arquitetura, a criação dos *layouts*, a criação de todas as funcionalidades, a implementação do modelo MVC e do padrão Singleton e a implementação da API do Google Maps na aplicação.

Posteriormente, o quinto capítulo apresenta alguns testes realizados com a aplicação e os resultados obtidos nesses testes, através da utilização da aplicação em contexto real numa vinha do campus da UTAD. São apresentados também neste capítulo testes do envio de imagens da aplicação para um servidor.

Por último, o sexto capítulo é dedicado às conclusões retiradas do desenvolvimento desta dissertação, tendo em conta os objetivos inicialmente traçados, e ao trabalho futuro que poderá vir a ser desenvolvido em torno deste projeto.

No final são ainda apresentadas todas as referências bibliográficas consultadas para a elaboração desta dissertação.

Capítulo 2

2. Enquadramento Tecnológico

Neste capítulo será apresentada uma análise das tecnologias utilizadas para a realização do presente projeto. Serão referidas algumas áreas de aplicação da tecnologia no auxílio da agricultura e ainda um conjunto de aplicações de sucesso, com funcionalidades importantes para o desenvolvimento deste setor de atividade. Também será feita uma abordagem à plataforma Android, ao modelo MVC, ao padrão de desenvolvimento Singleton, às APIs para a inserção de mapas em aplicações móveis e às metodologias de desenvolvimento de *software*.

2.1. Agricultura de Precisão

O conceito de Agricultura de Precisão (AP) surgiu no início da década de 90 nos EUA, tendo os primeiros estudos recaído de forma mais efetiva sobre as culturas cerealíferas da Austrália e dos EUA [37]. Existem diversas definições para AP, sendo que uma das mais completas é citada por Cook & Bramley: "agricultura de precisão é o termo dado aos métodos de gestão de culturas que reconhece a gestão da variabilidade espacial e temporal da parcela". A esta definição pode ainda complementar-se que a AP é um sistema focado em minimizar as entradas de dados (*inputs*) e em melhorar a eficiência, de modo a tornar a agricultura um sistema mais sustentável [19].

A AP é considerada como um sistema de gestão da produção agrícola, que tem como objetivo principal a otimização dos sistemas de produção em função da variabilidade espacial e temporal da produção [32]. Esta gestão orientada tornou-se possível com o desenvolvimento de tecnologias recentes, tais como os Sistemas de Posicionamento Global (GPS), os Sistema de Informação Geográficos (SIG) e os sensores remotos, permitindo assim observar e compreender a variabilidade associada aos respetivos sistemas de produção [4] [13].

2.1.1. Agricultura Tradicional vs Agricultura de Precisão

A agricultura de precisão distingue-se da agricultura tradicional pelo seu nível de gestão, isto é, em vez de se administrar uma área inteira como uma única unidade, a gestão é adaptada para pequenas parcelas [20] [9]. Através da análise da Figura 1 podemos distinguir três formas distintas de agricultura em função da heterogeneidade das parcelas: 1- Agricultura tradicional, antes da mecanização (parcelas pequenas diferenciadas); 2- Agricultura tradicional, com mecanização intensiva (parcelas grandes, consideradas uniformes); 3- Agricultura de precisão (parcelas grandes com diferenciação) [32].

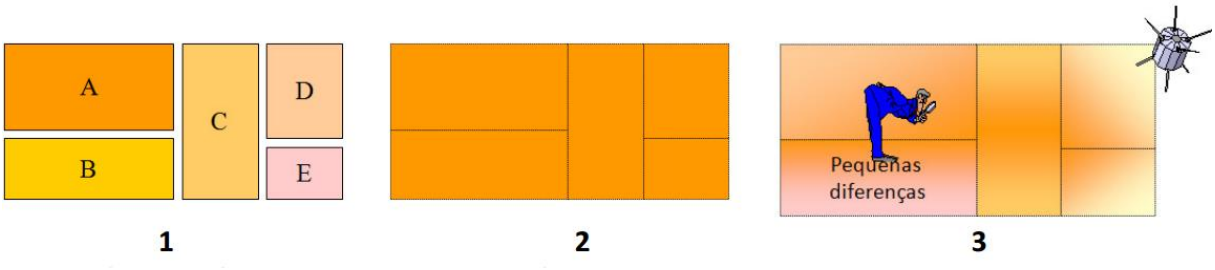

Figura 1 - Diferentes formas de agricultura em função da dimensão e homogeneidade das parcelas [32].

A AP envolve o desenvolvimento e a adoção de técnicas de gestão da variabilidade temporal e espacial das parcelas, com o objetivo principal de otimizar a rentabilidade e a qualidade da unidade de produção, bem como a redução dos impactes ambientais [19] [6].

AP - Sistemas de Posicionamento Global

Segundo Searcy, o sistema de posicionamento global é uma das bases da agricultura de precisão. Uma vez que o conceito de agricultura de precisão baseia-se em informações georreferenciadas, a precisão das coordenadas deve ser tão alta quanto possível [39].

AP - Sistemas de Informação Geográfica

Um sistema de informação geográfica (SIG), integra um *software* de informação espacial que funciona como base de dados para análise e gestão de uma representação espacial, que nos permite ter uma perceção do espaço e que tem a capacidade de manipular, consultar, visualizar, arquivar e modelar os dados [17].

AP – Monitores de Produtividade

Um monitor de produtividade é um equipamento que, em conjugação com um recetor GPS, permite a medição em tempo real da produtividade de uma cultura, possibilitando a apresentação desses dados em forma de carta.

As cartas de produtividade são bastantes importantes em agricultura de precisão já que apresentam de forma clara os padrões de uma das variáveis mais importantes na gestão agrícola, a produtividade [18].

AP – Tecnologia de Taxa Variável

A análise da variabilidade espacial da produtividade assim como da disponibilidade de recursos (solo, nutrientes, etc.) pode resultar na necessidade de aplicação dos fatores de produção (água, fitofármacos, etc.) com taxa variável, no espaço de uma parcela. Assim, ao contrário das distribuições convencionais, em que a taxa a aplicar (determinada para as condições médias de uma parcela) é mantida constante durante toda a aplicação, nas aplicações com taxa variável a taxa a aplicar varia no espaço em função de zonas pré-definidas. Esta possibilidade surge com os distribuidores munidos da tecnologia de taxa variável (VRT – *Variable Rate Technology*) e possibilita, por exemplo, a adequada produtividade e um uso eficiente de nutrientes, com simultânea redução do potencial de poluição ambiental [38] [35].

O objetivo é que em cada local de uma parcela seja aplicada a taxa desejada do fator de produção em causa, de acordo com as necessidades de cada zona, realizando-se uma gestão orientada. Por exemplo, o operador da máquina em vez de calibrar o distribuidor para uma taxa única, fornece através de um cartão de memória, uma carta de prescrição em que para cada zona da parcela é definida uma taxa de aplicação específica [7].

AP – Deteção Remota

As imagens de satélite são exemplos da deteção remota. De acordo com Thenkabail, P., o uso de dados provenientes de satélites com o propósito de quantificar e qualificar a variabilidade dentro e entre parcelas cultivadas é cada vez mais, a mais importante fonte de informação para a agricultura de precisão [41].

Os sistemas de deteção remota podem ser passivos ou ativos. Os sistemas passivos não têm uma fonte artificial de radiação e detetam radiação eletromagnética que é refletida ou emitida pelo objeto a analisar (exemplo: sensores de satélite para radiação visível ou infravermelha). Os sistemas ativos contêm uma fonte de radiação que incide sobre o objeto a analisar (exemplo: micro-ondas de radar ou sonar) [37].

2.2. Viticultura de Precisão

O surgimento dos monitores de produtividade permitiu o desenvolvimento dos primeiros estudos em viticultura de precisão (VP), inicialmente nos EUA [44] e Austrália [13], mais tarde também na Argentina, no Chile, na África do Sul e mais recentemente em Espanha e outros países da Europa, particularmente França e Portugal [4].

A viticultura de precisão, pode ser entendida como a gestão da variabilidade temporal e espacial das parcelas com o objetivo de melhorar o rendimento económico da atividade vitivinícola, quer pelo aumento da produtividade e/ou qualidade, quer pela redução dos custos de produção, reduzindo também o seu impacte ambiental e risco associado, através da aplicação dos fatores de produção de acordo com as necessidades das vinhas, e não de forma homogénea, como se todas as vinhas tivessem as mesmas características e todas as videiras tivessem as mesmas necessidades [8].

Uma questão central na viticultura de precisão é a gestão da informação, uma vez que a maioria das aplicações em viticultura de precisão lidam com grandes volumes de dados que é preciso gerir e converter em informação útil, que possa ser utilizada como base no processo de tomada de decisões no dia-a-dia das explorações vitivinícolas [8]. Tal como se pode observar na Figura 2, o processo de viticultura de precisão é um processo de entradas (*inputs*) e saídas (*outputs*), que permite uma utilização mais eficiente de dados como a rega, fertilização e produtos fitossanitários (*inputs*) de forma a aumentar a qualidade da produção e a rentabilidade do sistema (*outputs*) [19].

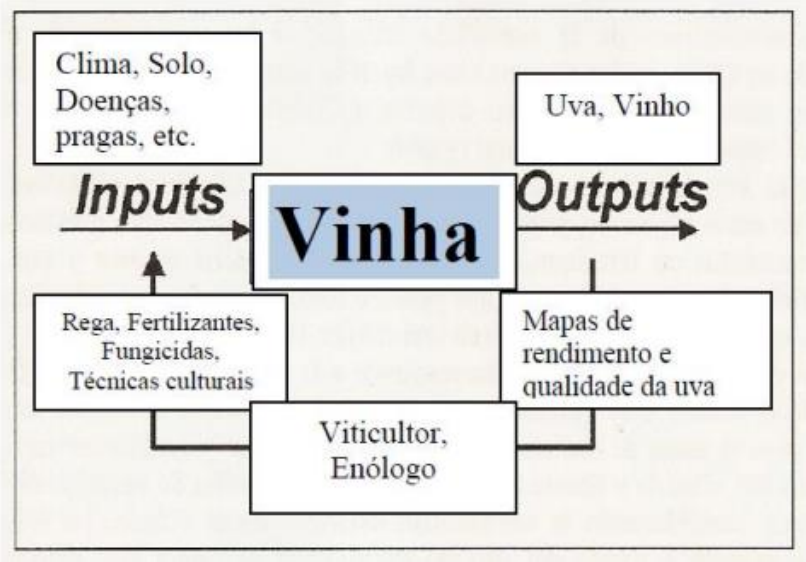

Figura 2 - Processo de entradas (inputs) e saídas (outputs) [19].

Desta forma, a adoção deste tipo de viticultura deve ser feita tendo em conta os objetivos de cada exploração vitivinícola e o seu contexto no mercado, sendo que a gestão das vinhas pode ser orientada para diversos ramos [8]:

- Garantir uma maior estabilidade do produto produzido de ano para ano;

- A manutenção de um registo de monitorização ambiental de modo a garantir uma maior rastreabilidade do processo produtivo;

- A utilização mais eficiente dos recursos face aos objetivos de produção;

- A identificação de zonas de produção de uvas com potencial para vinhos topo de gama.

2.2.1. Implementação da Viticultura de Precisão

Tal como se pode observar na Figura 3, a adoção da VP como ferramenta de gestão da vinha é um processo cíclico e contínuo, que se inicia com a observação e recolha de dados, seguindo-se a interpretação e avaliação dos mesmos, de modo a implementar uma gestão orientada dos *inputs* [10] [11].

A gestão orientada diz respeito ao tempo e duração dos *inputs* utilizados, nomeadamente da água e dos fertilizantes, bem como a gestão da utilização das máquinas, dos recursos humanos

disponíveis ou qualquer outra operação relacionada com a gestão da vinha, tendo em conta a sua variabilidade [13] [33].

Assim, os benefícios da viticultura de precisão podem demonstrar-se a dois níveis:

- Para o viticultor: melhora o uso dos fatores de produção, reduz os custos e o impacte ambiental e melhora a produtividade do produto sem comprometer a sua qualidade.

- Para os trabalhadores: permite melhorar a logística/organização do trabalho e ter um conhecimento prévio do que está a acontecer na vinha e das tarefas a realizar [4].

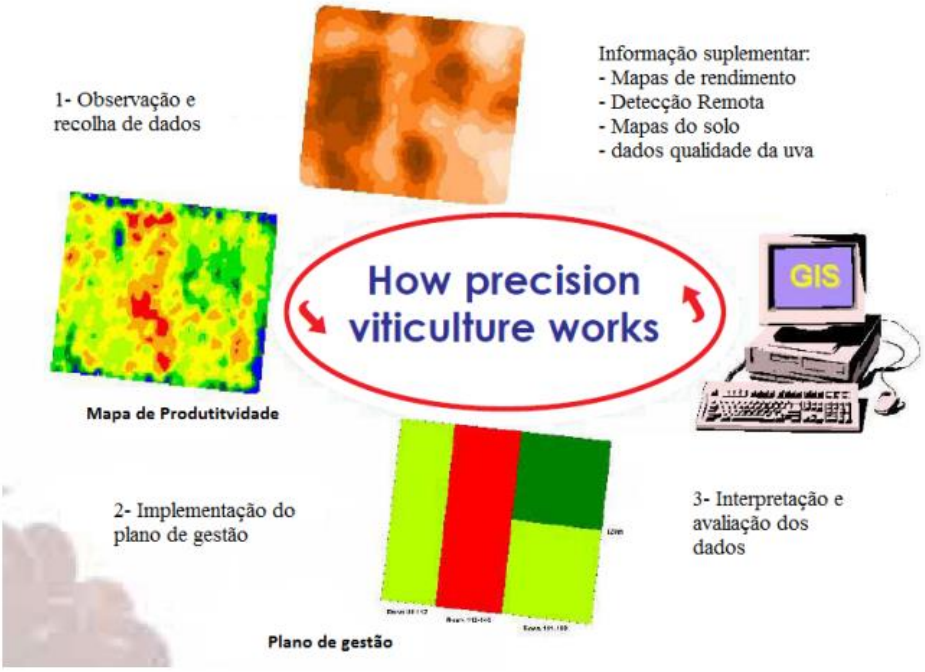

Figura 3 - Processo cíclico da Viticultura de Precisão [12].

2.3. Trabalhos Relacionados

O uso crescente de dispositivos móveis e a ampla difusão de redes sem fios têm vindo a permitir o desenvolvimento de inúmeras aplicações relacionadas com computação ubíqua e pervasiva. Estas tecnologias fornecem as primeiras oportunidades para explorar a computação em ambientes *"out-of-the-box"* e desta forma surge uma enorme variedade de ambientes que investigadores começam a explorar como potenciais para o uso da tecnologia.

Os sistemas de computação pervasiva originais, utilizados em espaços de escritório, são um

ponto de partida para a exploração do conceito [5]. Por outro lado, pensando em ambientes de trabalho *outdoor*, como por exemplo fábricas e quintas, surge uma abordagem diferente que vai de encontro àquilo que se pretende com este trabalho, ou seja, a agricultura vem sendo considerada uma das principais áreas de aplicação destas tecnologias [16].

Desta forma, apresentam-se de seguida alguns trabalhos desenvolvidos neste contexto, trabalhos esses que foram tidos em consideração para a estruturação desta plataforma de gestão de dados em vinhas.

2.3.1. Sistema de Mapeamento do Rendimento de Silagem

Um exemplo de uma aplicação de agricultura de precisão é um sistema de mapeamento do rendimento de silagem que lida com a análise local de um terreno e armazena dados como por exemplo: disponibilidade de água no solo, produção de biomassa, compactação do solo, índice de área foliar, dados climáticos, infestação de insetos e ervas daninhas, rendimento dos grãos, entre outros [31]. Estes dados recolhidos pelas alfaias agrícolas são armazenados e difundidos através da rede local. De seguida, podem ser analisados e transmitidos ao veículo de trabalho (trator por exemplo) através de rede sem fios.

Com base nessas informações o condutor do veículo monitoriza e controla o desempenho das alfaias agrícolas. Este sistema é composto por um sensor de humidade, um módulo GPS, células de carga e um módulo de comunicação sem fios *bluetooth*, para mapeamento do rendimento. O sistema possui ainda sensores de infravermelhos, controladores lógicos programáveis e transmissores/recetores rádio de baixo consumo através dos quais é possível recolher e transmitir os dados para um recetor remoto colocado fora da quinta [31].

2.3.2. Aplicação de Sensores Sem Fios Para Controlo de Estufas

Outro exemplo é a tecnologia de *variable-rate*[1] que serve para determinar a quantidade de fertilizante a aplicar em culturas de árvores em estufa. Neste sistema de monitorização em tempo real, os sensores de aquisição de dados e o GPS recolhem informações que mais tarde

[1] Tecnologia que permite a aplicação de uma quantidade de produtos diferente, nas colheitas em agricultura de precisão.

podem ser fornecidas ao agricultor através de *web servers.* As informações sobre pesticidas, infestações e previsões climáticas podem ser descarregadas diretamente via Internet [16]. Com este tipo de tecnologia de controlo remoto pretende-se melhorar a produtividade e reduzir o custo do trabalho.

A tecnologia na agricultura de precisão ajuda os agricultores a aumentar a qualidade das suas colheitas. O trabalho referido é um projeto de uma aplicação para controlo e monitorização de estufas num cenário de agricultura de precisão [16].

Aplicações deste género utilizam redes de sensores compostas por um grande número de pequenos dispositivos autónomos chamados de nós de sensores. O principal objetivo é controlar os ambientes conforme as exigências das sementeiras. Todos os sensores são reconfiguráveis conforme o estado de crescimento da cultura, as mudanças climáticas, a natureza do solo, entre outros [16].

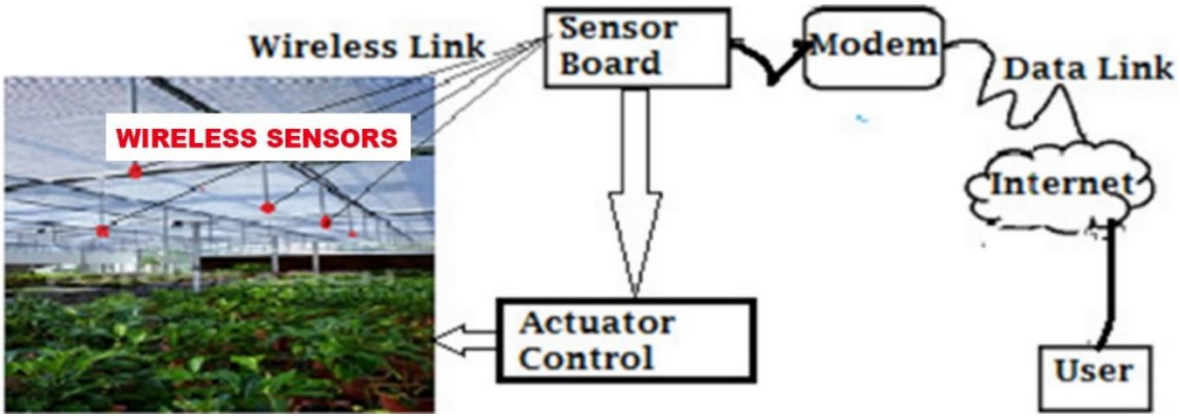

Figura 4 - Estufa típica com controlo remoto do efeito de estufa [16].

Uma estufa típica com nós de sensores é idêntica à representada pela Figura 4. O objetivo principal do sistema é controlar as condições climatéricas conforme as necessidades da sementeira. No exterior da estufa existem sensores configurados para recolher informações sobre o clima externo, tais como: temperatura, pressão atmosférica, luz, humidade, CO2, velocidade e direção do vento, entre outros. Estes parâmetros fornecem informações importantes sobre o clima e através da análise deles decidir-se-á a ação a tomar sobre os controlos, como por exemplo, o controlo do fluxo de ar na estufa para o melhor desenvolvimento das sementeiras [16].

2.3.3. Computação em Vinhas: Redes de Sensores

Uma característica dos sistemas de computação ubíqua é o facto de um único sistema poder ter vários pontos de interação humano-computador. Desta forma, neste estudo aplicado à vinha é importante abordar logo no princípio, os diferentes papéis e responsabilidades de um grupo de pessoas que gerem determinada vinha (trabalhadores esporádicos, produtores de vinho, etc.) para que seja possível definir qual a interface que melhor se adequa a cada tipo de utilizador [14].

Na gestão de uma vinha existem prioridades diferentes e tarefas associadas com funções diferentes. Assim, os dados devem estar relacionados entre si para que seja fácil para um determinado utilizador saber que tarefas foram realizadas por outros utilizadores (por exemplo, é importante para o produtor de vinho saber quando é que os trabalhadores aplicaram o último tratamento, para poder realizar a vindima) [14].

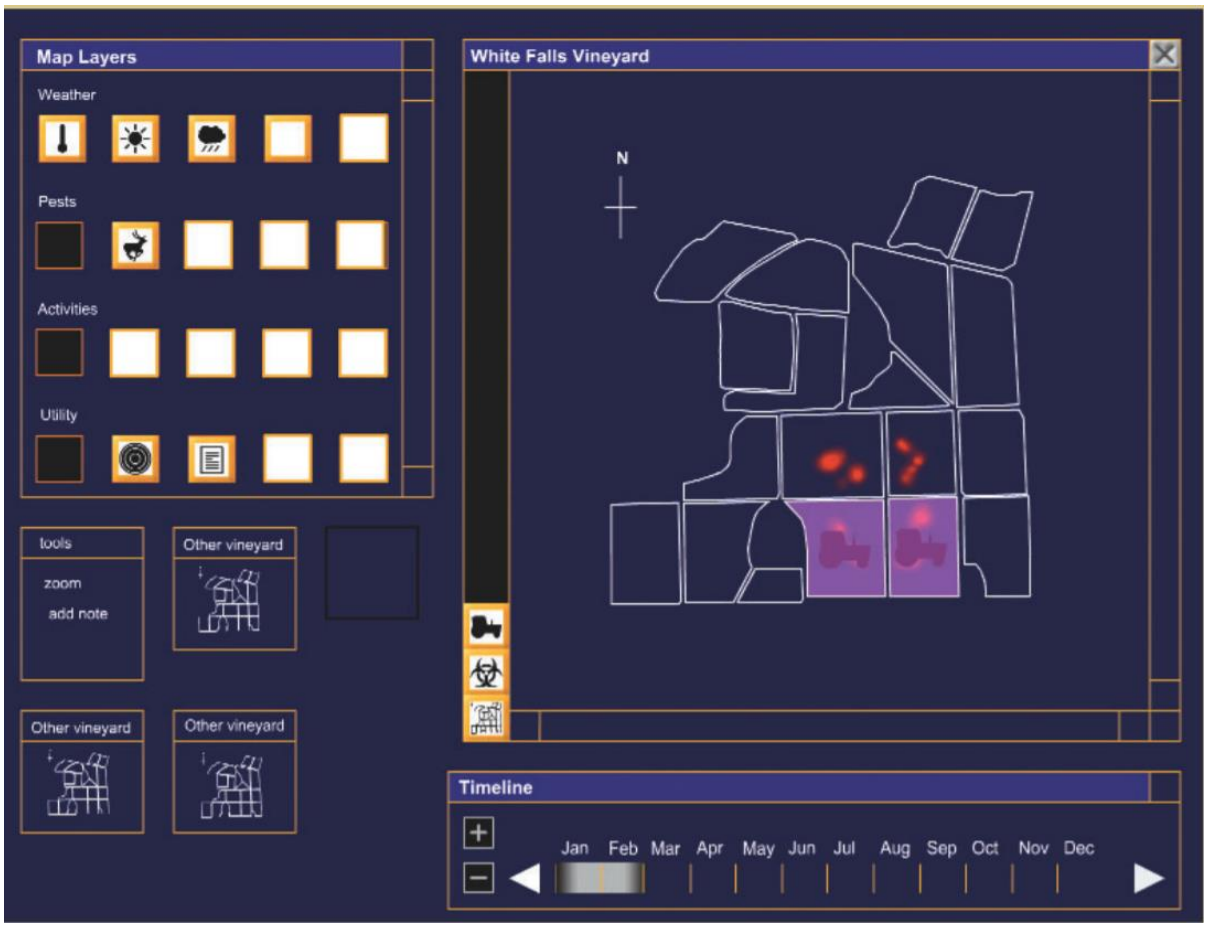

Figura 5 - Interface de gestão de vinha com mapa de uvas na vinha, áreas de risco de infeção e áreas pulverizadas com pesticida [14].

Desta forma, os requisitos essenciais para que um sistema destes funcione são [14]:

- Permitir ao gestor da vinha descarregar os dados de atividade na vinha e visualizá-los;

- Permitir aos trabalhadores dar entrada no sistema das atividades que estão a desenvolver na vinha.

Uma aplicação deste género seria ideal se reunisse ainda dados das atividades do pessoal, das necessidades da vinha e dos equipamentos existentes na vinha incorporando esses mesmos dados automaticamente em orçamentos e previsões de tempo [14].

Os autores deste trabalho tiveram em consideração que muitas vezes os trabalhadores das vinhas não têm muito conhecimento tecnológico, o que faz com que a introdução dos dados seja uma tarefa difícil para eles.

Para resolver esse problema pensaram numa solução em que, por exemplo, se o gestor quiser saber em zona da vinha foi aplicado um determinado pesticida, numa rede de sensores estática, poderia ser colocado um identificador RF (rádio frequência) no pulverizador utilizado pelo trabalhador e desta forma seria possível saber, de modo automático, em que zonas da vinha foi aplicado o pesticida, tal como se pode observar na Figura 5.

Os autores deste trabalho consideram ainda que, da mesma forma, os utensílios utilizados na vinha (pás, tesouras da poda, entre outros) também poderiam ter identificadores RF para monitorizar a sua localização e o seu tipo de atividade. Assim ao trabalhar com estas ferramentas o trabalhador iria fornecer os dados para o sistema de uma forma implícita [14].

As *SON (Self Organized Networks)*[2] são geralmente consideradas as arquiteturas dos sistemas padrão e são também as que fazem mais sentido para redes de sensores em ambientes remotos e inacessíveis, uma vez que ao seguir esta abordagem, os próprios sensores podem criar arquiteturas de rede diferentes, com ligações diferentes, dependendo das necessidades do

[2] – Tecnologia automática projetada para tornar o planeamento, configuração, gestão e otimização de redes mais simples e rápidas.

sistema e desta forma aumentar a vida útil de toda a rede e minimizar os consumos de energia desses sensores. Além disso, nas arquiteturas *self organized* os sensores são capazes de estabelecer fluxos de ligação alternativos, de forma automática, no caso de ocorrerem falhas no fluxo inicialmente estabelecido [14].

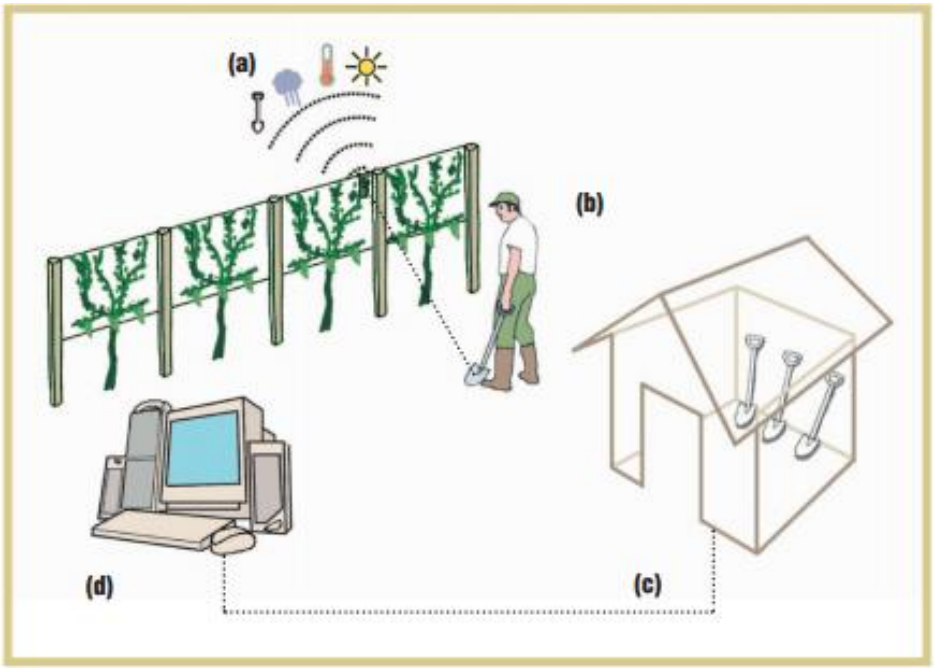

Figura 6 - Exemplo de arquitetura de rede de sensores numa vinha [14].

Por exemplo, uma arquitetura que usa dados adquiridos e transportados pela rede de sensores distribuídos ao longo da vinha como a da Figura 6: em que existem sensores para o registo de dados ambientais (a) e sensores embutidos na ferramenta, que recolhem dados através dos movimentos do utilizador (b), dados esses que posteriormente são enviados para uma base de dados central (d), quando as ferramentas forem colocadas novamente no armazém (c) [14].

2.4. Plataforma de Desenvolvimento Android

O sistema operativo Android é um sistema operativo *open source* para dispositivos móveis que tem como base o *kernel* Linux e foi desenvolvido pela empresa Google com o apoio da Open Handset Alliance [2].

O desenvolvimento de aplicações para este sistema operativo é realizado normalmente na linguagem de programação *Java*, sendo também possível desenvolver bibliotecas de código

que, posteriormente, são integradas nas aplicações desenvolvidas [2].

As aplicações de sistema são executadas sob uma plataforma orientada a objetos, contêm bibliotecas de sistema e fornecem aos programadores acesso às diferentes funcionalidades do mesmo.

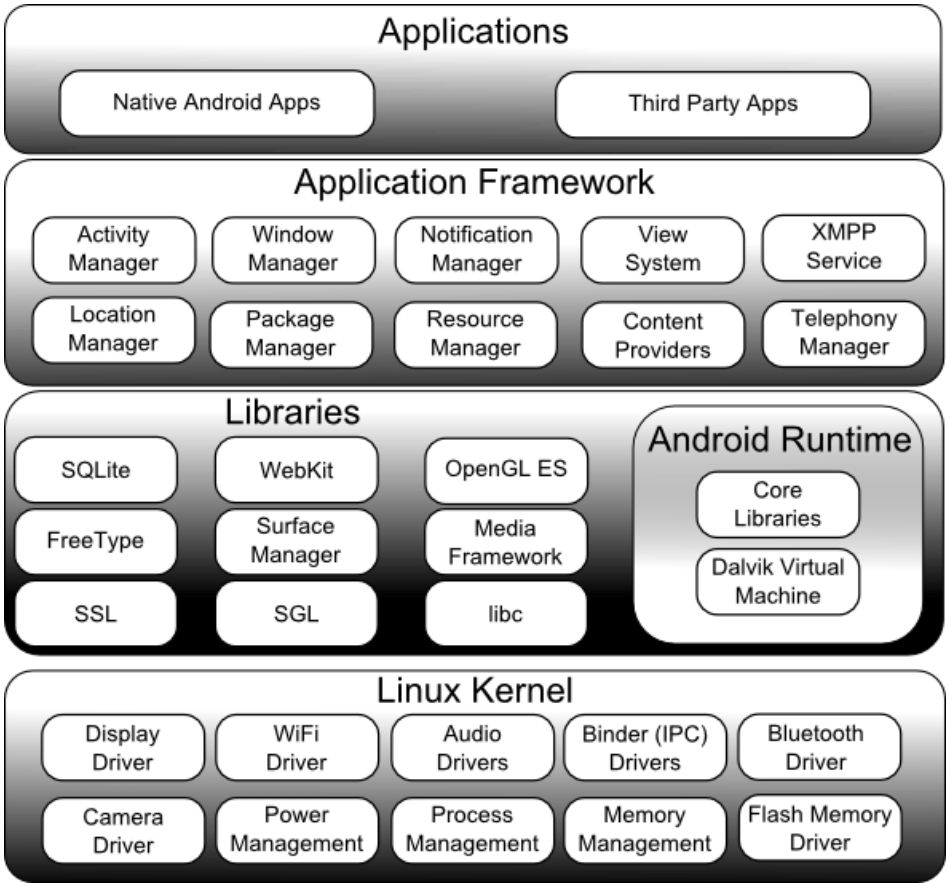

Figura 7 - Arquitetura Android [24].

A plataforma Android fornece o sistema operativo para dispositivos móveis, o ambiente de desenvolvimento e possui uma máquina virtual (*Dalvik Virtual Machine*) construída para executar as aplicações e que funciona também como *middleware* entre o código e o sistema operativo, como se pode observar na Figura 7. As aplicações são distribuídas no formato *Dalvik Executable* (.dex), tendo sido idealizadas para sistemas com recursos limitados (como memória e capacidade de processamento). São distribuídas num formato compactado (.apk), onde se encontram os ficheiros binários e respetivos recursos.

Cada aplicação é executada por uma instância da máquina virtual *Dalvik*, correndo em processos distintos no sistema operativo [21].

No Android os programadores e os utilizadores têm também a possibilidade de consultar fóruns para esclarecimento das suas dúvidas e têm ainda a possibilidade de dar sugestões para melhorias do ambiente de programação, do sistema operativo e da API [24].

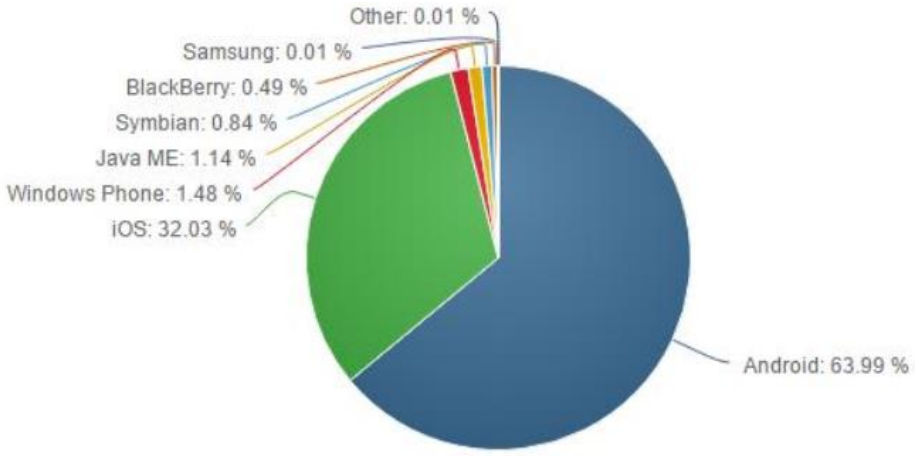

Figura 8 - Estatísticas de mercado das diferentes plataformas móveis em 2017 [45].

Existem ainda outros sistemas operativos para dispositivos móveis, sendo que o desenvolvimento de aplicações é diferente para cada um deles. Cada sistema operativo requer diferentes ferramentas e técnicas de desenvolvimento, como por exemplo a linguagem de programação [42]. Apesar disso, o Android continua a ser a plataforma mais utilizada para o desenvolvimento, tal como se pode observar na Figura 8, uma vez que é executado em 63.99% dos dispositivos móveis que existem no mundo [45]. Com base nestas características, o Android foi o sistema operativo escolhido para o desenvolvimento deste projeto.

2.5. Tecnologias a implementar

Neste capítulo será feita uma abordagem aos padrões de desenvolvimento a utilizar na implementação da aplicação assim como a todas as tecnologias consideradas úteis para a criação de um sistema capaz de permitir aos viticultores ter uma visão geral das suas vinhas através do seu *smartphone*.

2.5.1. Modelo MVC

O padrão de arquitetura MVC (*Model-View-Controller*) é bastante utilizado no desenvolvimento de aplicações para dispositivos móveis pois determina a separação de uma

aplicação em três elementos distintos [30]. Tal como se pode observar na Figura 9, o *Model* é formado por entidades que representam os dados da aplicação. A *View* apresenta os dados e capta as ações do utilizador, sendo representada por janelas de visualização. O *Controller* faz a ligação entre o *Model* e a *View* e é responsável por decidir quais os componentes que devem executar determinada tarefa, atuando sobre o *Model* e alterando os elementos da *View* para representar o novo formato dos dados [30].

A utilização desta arquitetura na implementação deste projeto será uma mais valia, uma vez que a separação da aplicação em elementos distintos permitirá o aperfeiçoamento da mesma no futuro, por exemplo, através do acréscimo de funcionalidades no *Controller* ou mesmo através da implementação de um novo *Controller* com funcionalidades diferentes.

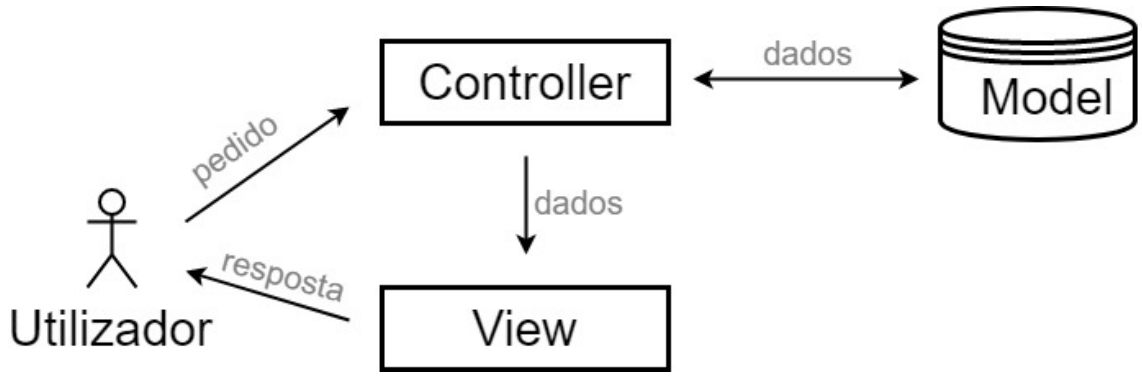

Figura 9 - Arquitetura do Modelo MVC.

Separando o *Controller* do *Model* e da *View*, o fluxo da aplicação é isolado. Desta forma, os programadores podem entender a perspetiva do utilizador olhando apenas para o *Controller*. O fluxo da aplicação pode ser alterado modificando-se o *Controller*, sem alteração (ou com pouca) do restante código [3].

Ao separar o *Model* da *View*, o *Model* é independente e livre de alterações. Desta forma os programadores podem alterar a aparência da interface do utilizador sem ter de mudar o *Model.* Ao separar a *View* do *Model*, esta não tem de lidar com os detalhes de funcionamento do *Model*. O *Model* pode obter os dados do armazenamento local, de um servidor, da memória em cache ou de uma combinação de todas essas fontes [3].

Para que a aplicação seja útil mesmo quando não existe uma conexão à rede, a parte de cliente da aplicação deve decidir sobre quando buscar os dados do servidor e quando buscar os dados

do armazenamento local. As estratégias de dados locais podem ser baseadas em cache ou sincronização para melhorar a responsividade e manter a coerência dos dados. O particionamento destes detalhes no modelo MVC torna a implementação, teste e manutenção mais fácil do que se os mesmos estivessem repartidos pela aplicação uma vez que, quando é necessário efetuar alguma alteração, efetua-se em apenas uma das partes sem comprometer nem ter de alterar as restantes [3].

O MVC pode requerer a escrita de mais código (grande parte no controlador). Contudo, a pior limitação do padrão MVC é que, apesar dele separar o *Model* da *View*, de tal forma que os programadores possam alterar a aparência da interface do utilizador sem ter que mudar o *Model*, o MVC não permite a mudança do *Model* da aplicação em si [3]. A mudança do *Model* é o resultado da adaptação da aplicação quando surgem limitações de recursos (da rede e do sistema) [3].

2.5.2. Padrão Singleton

O padrão Singleton é um dos padrões de *design* mais simples de implementar. Este tipo de padrão de *design* vem sobrepor-se aos padrões que tentam criar objetos da forma mais adequada à situação e que muitas vezes adicionam complexidade ao *design* das aplicações. O Singleton envolve uma única classe que é responsável por criar um objeto, assegurando-se que apenas uma única instância seja criada. A classe fornece ainda uma forma de aceder à única instância do objeto, podendo esta ser acedida diretamente sem necessidade de instanciar explicitamente o objeto da classe. Desta forma, o padrão Singleton não permite que a aplicação gere objetos desnecessários, o que resulta num ganho em termos de desempenho. Com a implementação deste padrão no desenvolvimento deste projeto será possível resolver o problema de replicação de objetos uma vez que, vários botões que executem a mesma função (de diferentes *layouts*), poderão aceder à mesma instância (não havendo necessidade de criar objetos diferentes para produzirem o mesmo resultado) e será possível também que exista partilha de dados entre os vários layouts da aplicação.

2.5.3. APIs Para a Inserção de Mapas em Aplicações Móveis

A utilização de APIs que permitem a utilização de mapas em aplicações para dispositivos móveis é indispensável para integração da plataforma de gestão de dados em vinhas proposta neste documento. Desta forma, foram selecionadas dentre as APIs existentes duas que atendiam os requisitos para o problema em questão, estando elas apresentadas abaixo:

- Google Maps: É um serviço de pesquisa e visualização de mapas e imagens de satélite da Terra fornecido e desenvolvido pela Google, que permite incorporar mapas em aplicações móveis livremente e ainda adicionar dados da própria aplicação desenvolvida sobre os mapas [23]. A Google Maps API é um serviço gratuito, disponível para qualquer aplicação que o público possa usar gratuitamente. Existe uma restrição de uso para empresas que cobram taxas pelo acesso, essas empresas são orientadas a usar a Google Maps API Premier, que fornece suporte técnico, recursos mais avançados e um acordo de nível de serviço [23].

- Open Street Maps: É um projeto colaborativo para criar uma visualização da Terra. Além de ser um serviço *open source*, os mapas são criados usando dados de recetores GPS portáteis, fotografias aéreas e outras fontes gratuitas [36]. Além disso, os mapas são disponibilizados para qualquer uso, incluindo fins comerciais. Tanto as imagens obtidas por processamento dos dados, quanto os dados estão disponíveis sob uma licença C*reative Commons Attribution-ShareAlike 2.0.* e os utilizadores registados podem carregar os históricos dos GPS e editar os dados usando as ferramentas disponíveis [36].

Com o intuito de facilitar a escolha da API apropriada para o desenvolvimento deste projeto criou-se a Tabela 1, que se apresenta de seguida e na qual foram avaliados os critérios de confiabilidade, usabilidade, licença e suporte das duas APIs descritas, numa escala de notas de 0 a 5 [22].

	confiabilidade	usabilidade	licença	suporte	
Google Maps	Qualquer utilizador pode editar as informações dos mapas, contudo uma equipa do Google precisa de aprovar as modificações antes delas serem disponibilizadas, garantindo uma confiabilidade maior das informações.	A usabilidade do mapa da Google é muito intuitiva e não há dificuldades para encontrar o local que se procura.	Todas as informações dadas pelos utilizadores tornam-se propriedade da Google.	O Google Maps possui muita documentação da sua API, além de vídeos e tutoriais.	**Total**
Nota atribuída	5	5	3	5	18
	3	5	5	3	16
Open Street Maps	Qualquer utilizador pode editar as informações dos mapas, e essas informações são disponibilizadas instantaneamente. Há uma pequena equipa que analisa as informações depois delas serem disponibilizadas, mas se a informação estiver errada pode levar cerca de dois dias para deixar de estar disponível.	A usabilidade do mapa do OSM é muito parecida à do Google Maps, bastante intuitiva e sem dificuldades para encontrar informações.	As informações dadas pelos utilizadores são públicas.	O OSM possui uma documentação fraca comparada com a do Google Maps, contudo a sua comunidade é mais ativa e pronta para ajudar.	

Tabela 1 - Comparação entre APIs para a inserção de mapas.

Tendo em conta a análise comparativa efetuada na Tabela 1, a API escolhida para o desenvolvimento deste projeto foi a Google Maps API, contudo pode-se verificar que a Open Street Maps API também é uma solução bastante fiável para a inserção de mapas em aplicações móveis e as diferenças entre as duas são muito reduzidas [22].

2.6. Metodologias de Desenvolvimento de *Software*

Uma metodologia de desenvolvimento de *software* pode ser definida como um conjunto de atividades (e resultados associados) que auxiliam na produção de *software*. Dentre as várias atividades associadas, existem por exemplo a análise de requisitos e a programação. O resultado da implementação de uma metodologia de desenvolvimento de *software* é um produto (*software*) que reflete a forma como essa metodologia foi seguida.

Embora existam várias metodologias para o desenvolvimento de *software* existem atividades fundamentais que são comuns a todas elas [26]:

- Especificação de *software*: consiste em definir as funcionalidades do *software* e as restrições da sua operação.

- Projeto de implementação de *software*: garante que o *software* deve ser produzido de modo a que sejam cumpridas todas as suas especificações.

- Validação de *software*: é feita a validação do *software* para garantir que ele faz o que o cliente solicitou.

- Evolução do *software*: garante a evolução do *software* de modo a atender ás necessidades futuras do cliente.

2.6.1. Metodologia Scrum

O Scrum é uma metodologia de desenvolvimento ágil que tem como objetivo desenvolver e distribuir o *software* com a maior qualidade possível, dentro de etapas compostas por Sprints (intervalos de tempos definidos para o desenvolvimento) [15].

No Scrum os requisitos não precisam de ser detalhados no início do projeto, eles vão sendo definidos com a sua evolução. Com recurso às reuniões diárias são definidas novas tarefas do processo de desenvolvimento. Nas reuniões diárias procura-se identificar problemas referentes ao andamento do projeto o mais cedo possível e propor as respetivas soluções [29]. A metodologia Scrum não possui atividades e práticas que orientam o processo de desenvolvimento, o seu enfoque é para a gestão do projeto. A Figura 10 reproduz graficamente o ciclo do Scrum.

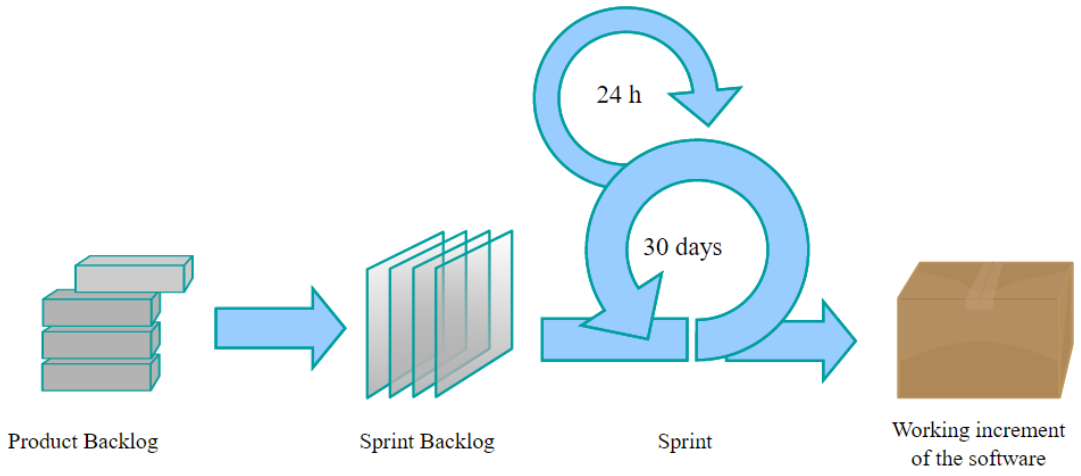

Figura 10 - O processo Scrum [34].

O ciclo do Scrum pode ser dividido, em três partes [15]: pré-sprint, sprint e pós-sprint. De acordo com Highsmith et al [27], o pré-sprint inicia-se com a definição da lista do *backlog* de produto, através de uma reunião de planeamento, onde o utilizador vai definir e classificar as funcionalidades que pretende para o *software*, por nível de prioridade. A fase de sprint

completa as tarefas previamente definidas e desenvolve uma pequena parte funcional do *software*, também conhecida como incremento. O ciclo Scrum termina no pós-sprint, no qual o pedaço executável de *software*, ou incremento, é apresentado ao cliente para que possa ser analisado.

2.6.2. Prototipagem

Um protótipo é uma representação limitada de um *software*, essa representação pode ser um esboço em papel de um *layout* ou conjunto de *layouts* do *software*, um *print screen*, uma simulação em vídeo de uma tarefa, uma maquete tridimensional de papel ou cartolina, ou um simples conjunto de *layouts* ligados por *hyperlinks* [46]. Segundo Sommerville e Sawyer [25], um protótipo pode ser usado como meio de comunicação entre os diversos membros da equipa de desenvolvimento ou mesmo como forma de testar ideias. Quanto mais interativo for o processo de desenvolvimento do protótipo, melhor o será sistema final [46]. Os protótipos podem ser desenvolvidos usando tecnologias que em nada se assemelham com as do sistema final [28] e elaborados recorrendo a diversas técnicas. Na Figura 11 pode-se observar o ciclo de prototipagem, em que a interface de utilizador é o mecanismo através do qual a comunicação entre o *software* e o utilizador é estabelecida.

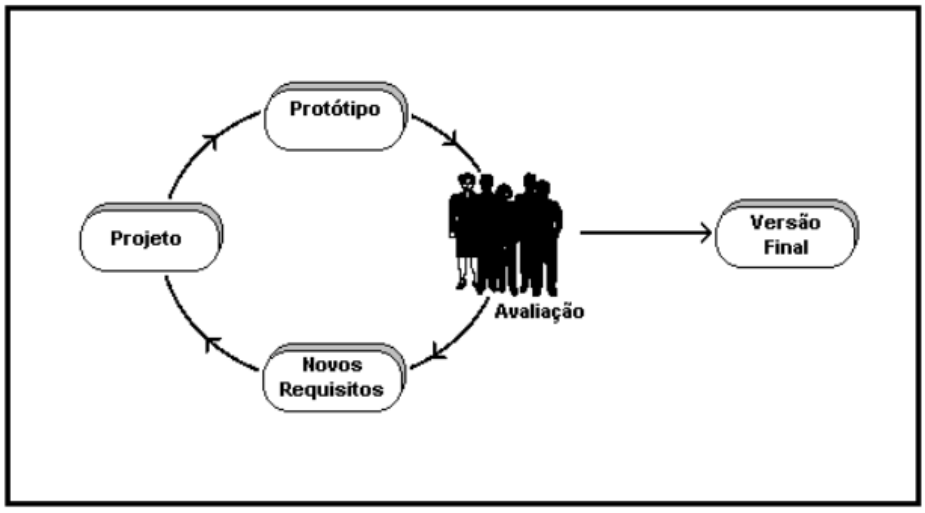

Figura 11 - O processo de Prototipagem.

A metodologia de prototipagem pressupõe o desenvolvimento de protótipos executáveis que são bastante úteis para demonstrar a navegação e testar o uso da interface, por isso, esta será a metodologia a utilizar no desenvolvimento deste projeto uma vez que se pretende

implementar a interface do utilizador para a plataforma móvel para gestão de dados em vinhas.

Capítulo 3

3. Conceção

Este capítulo apresenta a aplicação, caracterizando-a de modo detalhado, assim como a fundamentação das funcionalidades consideradas úteis para o cumprimento dos objetivos inicialmente previstos.

O propósito deste projeto é o desenvolvimento de uma aplicação para dispositivos móveis que permita fazer a gestão das tarefas diárias das vinhas, monitorizando todos os acontecimentos que estejam a ocorrer. Com base nos dados recolhidos pelos utilizadores em contato com o terreno, os produtores de vinho têm a possibilidade de saber quais as anomalias presentes nas suas vinhas, para que possam encontrar soluções para as resolver. Todos os dados serão disponibilizados na plataforma móvel e podem ainda ser alterados pelos utilizadores de forma a manter sempre atualizadas as informações que dizem respeito às vinhas.

3.1. Análise de Requisitos

Nesta secção são descritos detalhadamente os requisitos da plataforma móvel para gestão de dados em vinhas. Tendo em consideração o enquadramento tecnológico previamente analisado e o objetivo deste trabalho que é o desenvolver uma solução capaz de permitir a gestão/monitorização remota das tarefas diárias das vinhas, obteve-se então a seguinte especificação de requisitos:

- A plataforma deve permitir ao utilizador autenticar-se para a poder utilizar;

- Os utilizadores podem ativar a sua localização para saberem em que vinha se encontram e adicionar pontos às parcelas;

- Existem três diferentes tipos de utilizadores (trabalhador, empreiteiro e produtor de vinho) com funções diferentes;

- Os trabalhadores podem ver as suas vinhas, as anomalias, os trabalhos programados

e as parcelas associadas a essas vinhas;

- Os trabalhadores podem adicionar e editar anomalias que encontrem nas vinhas;

- Os empreiteiros, além das funções dos trabalhadores podem também editar e alterar o estado dos trabalhos programados assim como as parcelas que dizem respeito às suas vinhas;

- Os produtores de vinho (responsáveis máximos pela gestão das vinhas) podem executar as mesmas funções dos trabalhadores e dos empreiteiros;

- Os produtores de vinho podem ainda adicionar/eliminar trabalhos e parcelas assim como apagar anomalias;

- A plataforma deve permitir aos utilizadores inserir imagens referentes às anomalias detetadas na vinha para uma melhor identificação das mesmas.

Para complementar a compreensão da especificação de requisitos foi elaborado um diagrama de casos de uso da aplicação, de forma a documentar as ações do sistema do ponto de vista do utilizador, tendo em conta o levantamento de requisitos feito anteriormente. Estão assim representadas na Figura 12 as principais funcionalidades do sistema e a interação dessas funcionalidades com os utilizadores descrevendo a sequência de eventos.

Como se pode observar pela análise do diagrama de casos de uso apresentado na Figura 12, na aplicação móvel existe uma hierarquia de utilizadores. Os três tipos de utilizadores que existem são: trabalhadores, empreiteiros e produtores de vinho.

Os trabalhadores são os utilizadores que diariamente estão no terreno e como tal podem utilizar a aplicação para ver os trabalhos que têm de realizar (trabalhos esses que foram programados pelos produtores de vinho), e para adicionar anomalias que encontrem no decorrer do seu trabalho, contribuindo assim para uma gestão mais eficaz das tarefas das vinhas.

Os empreiteiros são os utilizadores responsáveis por garantir que os trabalhos programados pelos produtores de vinho são executados pelos trabalhadores e como tal, além das funções dos trabalhadores podem ainda editar o estado de execução dos trabalhos (“Executado”, “Em

Execução” ou “Por executar”) e editar as parcelas de uma determinada vinha (a mando dos produtores de vinho).

Figura 12 - Diagrama de casos de uso da aplicação.

Os produtores de vinho são os responsáveis máximos por todo o processo de gestão das vinhas. Estes utilizadores além das funções dos trabalhadores e dos empreiteiros podem também agendar trabalhos para serem executados (com base ou não nas anomalias reportadas) e têm ainda a possibilidade de adicionar parcelas às vinhas que gerem. A função de eliminar o que quer que seja da plataforma também diz respeito a estes utilizadores.

3.2. Fluxo da Aplicação

A plataforma móvel para gestão de dados em vinhas segue um fluxo previamente estruturado que pode ser compreendido pela análise da Figura 13 e que é descrito de seguida: os utilizadores ligam-se à aplicação para visualizar, monitorizar e gerir as suas vinhas e esta efetua pedidos HTTP/HTTPS a um servidor que é responsável por fornecer todos os serviços, tais como a partilha de dados, e enviar respostas novamente para a aplicação. O serviço de base de dados do servidor servirá ainda para guardar toda a informação relativa às vinhas e aos utilizadores da aplicação.

Assim, pretende-se que a plataforma móvel para gestão de dados em vinhas seja uma forma prática de os produtores de vinho e gestores de vinhas receberem informações recolhidas por outros acerca do estado de manutenção das suas vinhas.

Contudo, esta dissertação centra-se no desenvolvimento da plataforma móvel do ponto de vista da interface do utilizador sendo que, as tecnologias a implementar do lado do servidor não serão objeto de estudo neste trabalho e serão posteriormente desenvolvidas por outras equipas de trabalho que virão a ser inseridas no projeto.

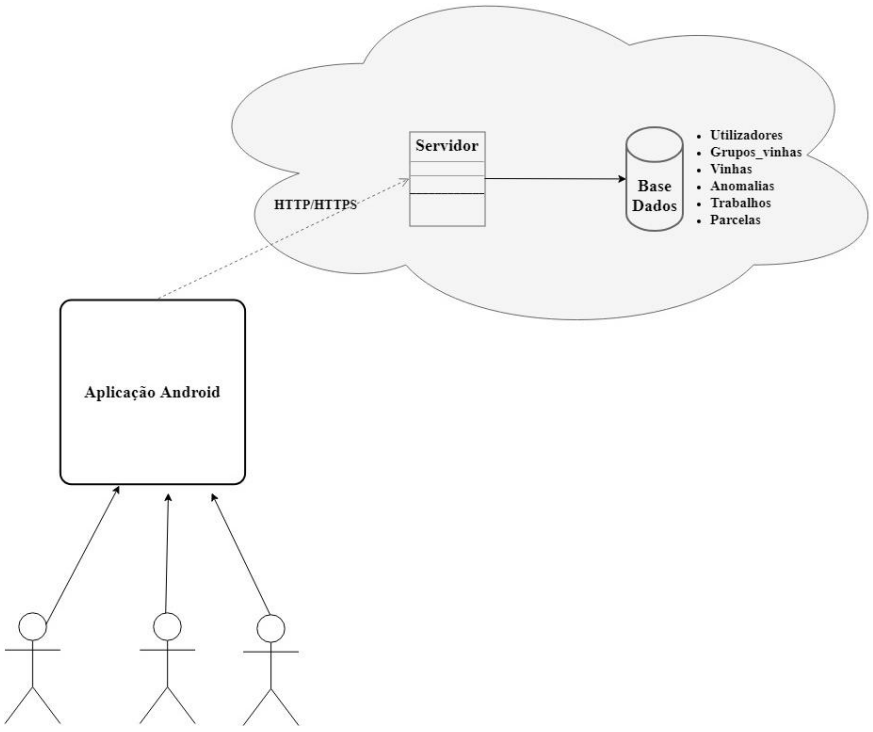

Figura 13 - Esquema de fluxo da aplicação.

3.3. Estrutura Conceptual da Aplicação

A estrutura conceptual desta aplicação será baseada no modelo MVC e no padrão Singleton apresentados anteriormente. Tal como se pode observar na Figura 14 irão ser criados dois tipos de classes: as classes *View* (que dizem respeito aos *layouts* da aplicação: aqueles com que o utilizador interage) e as Classes *Controller* que comandam as anteriores.

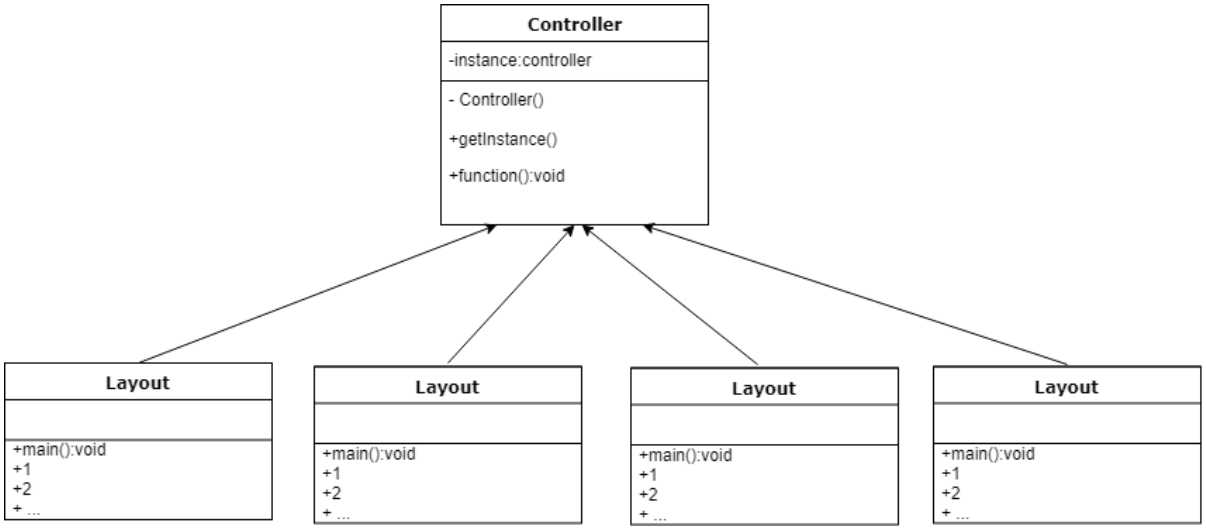

Figura 14 - Diagrama de classes da estrutura conceptual da aplicação.

Como se pode observar na Figura 14, todos os *layouts* que dizem respeito a determinada funcionalidade (anomalias por exemplo) estarão dependentes de uma classe *Controller* que controla o fluxo da aplicação e que recebe e envia os dados para os *layouts*.

Este processo é realizado com recurso ao padrão Singleton através da criação de instâncias no *Controller* que poderão ser acedidas através dos *layouts*, sem a necessidade de criar objetos diferentes para produzirem o mesmo resultado. Esta estrutura será utilizada tanto para as anomalias, como para os trabalhos programados e as parcelas.

3.4. Modelos de *Layouts*

Para uma melhor perceção do fluxo da aplicação de gestão de dados em vinhas apresentam-se alguns modelos dos *layouts* que se pretendem implementar na aplicação e a descrição de cada um deles de forma a transmitir de uma forma mais concreta as funcionalidades que se pretendem implementar.

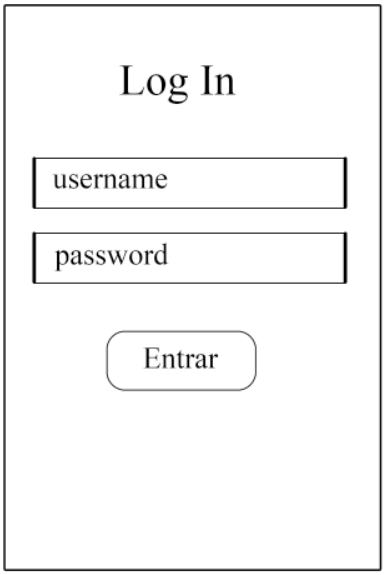

Figura 15 - Modelo do layout inicial da aplicação.

Ao ser executada a aplicação inicia com um *layout*, cujo modelo pode ser observado na Figura 15, no qual o utilizador (trabalhador, empreiteiro, produtor de vinho) pode autenticar-se na aplicação.

Figura 16 - Modelo de layout com menu da aplicação.

Depois de iniciar sessão na aplicação é apresentado um *layout* com um mapa, cujo modelo pode ser observado na Figura 16, no qual é possível pesquisar as diferentes vinhas, visualizar imagens de satélite das mesmas e obter informações (nome, localização, entre outras) da vinha pesquisada. Ainda neste *layout* são apresentadas três opções (Anomalias – permite ao

utilizador ver as anomalias da vinha pesquisada, inserir novas anomalias, saber o estado de resolução das anomalias, entre outros; Trabalhos Programados – permite ao utilizador ver a lista de trabalhos programados para aquela vinha, inserir novos trabalhos, estabelecer uma data limite para efetuar o trabalho, entre outros; Parcelas – permite ao utilizador saber por quantas parcelas é constituída a vinha, visualizar os limites de cada parcela, saber que tipo de castas estão plantadas, entre outros) e através do clique numa delas o utilizador é reencaminhado para um novo *layout* com outras funcionalidades sempre associadas à vinha que pesquisou.

Se optar por selecionar a primeira opção do menu ("Anomalias") o utilizador (trabalhador, empreiteiro ou produtor de vinho) é reencaminhado para um *layout*, cujo modelo pode ser observado na Figura 17, no qual terá uma lista das anomalias que existem na vinha anteriormente pesquisada e o estado de resolução dessas mesmas anomalias ("Resolvido" ou "Por Resolver"). Neste *layout* o utilizador pode ver e atualizar as informações relativas às anomalias existentes e pode ainda adicionar uma nova anomalia à lista. Para ver e atualizar as informações de uma das anomalias o utilizador terá de selecionar a respetiva anomalia.

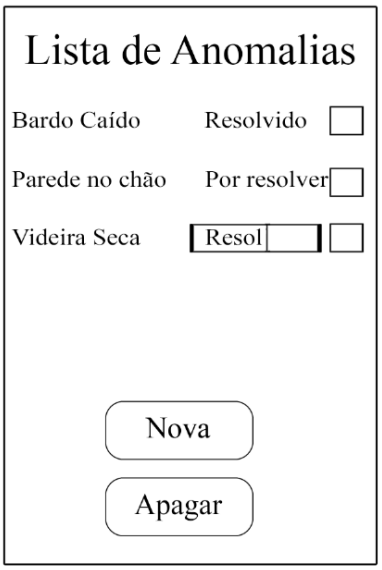

Figura 17 - Modelo de layout da lista de anomalias.

O produtor de vinho, responsável máximo pela gestão das vinhas, pode ainda eliminar uma anomalia que já não tenha interesse que esteja presente na lista, selecionando-a através da c*heckbox* e de seguida clicando no botão "Apagar". O produtor de vinho pode também, com base na informação disponibilizada, programar um novo trabalho para resolução da respetiva

anomalia.

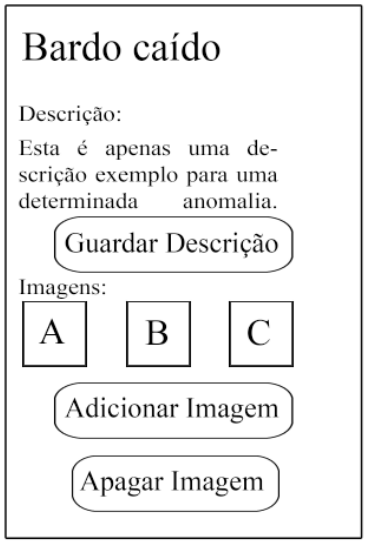

Figura 18 - Modelo de layout para visualizar as informações de uma anomalia.

Ao selecionar uma anomalia, o utilizador será mais uma vez reencaminhado para outro *layout*, cujo modelo pode ser observado na Figura 18. Neste *layout* pode ver e atualizar a descrição da referida anomalia, pode ver imagens que ajudarão a perceber melhor a anomalia e pode ainda eliminar ou adicionar imagens para que outros utilizadores possam, mais tarde, ter uma melhor perceção da anomalia.

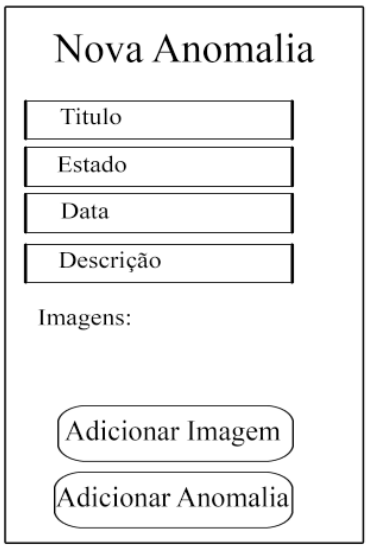

Figura 19 - Modelo de layout para a inserção de uma nova anomalia.

Para adicionar uma anomalia o utilizador terá de selecionar a opção “Nova”, na Figura 17, e

será desta vez reencaminhado para um *layout*, cujo modelo pode ser observado na Figura 19, no qual terá um formulário que deve preencher com as diferentes informações que dizem respeito à anomalia que está prestes a adicionar e no qual poderá ainda adicionar imagens referentes à anomalia.

Sempre que o utilizador pretender adicionar uma imagem a uma anomalia (quer seja ao criar uma nova ou a atualizar uma existente) é redirecionado para a câmara do seu dispostito, de forma a poder capturar a imagem que pretende adicionar.

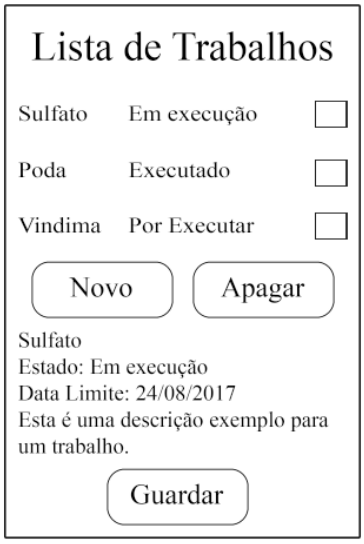

Figura 20 - Modelo de layout da lista de trabalhos.

Se, no *layout* do menu (Figura 16) o utilizador selecionar a segunda opção ("Trabalhos Programados") é reencaminhado para um *layout*, cujo modelo pode ser observado na Figura 20, no qual terá uma lista dos diferentes trabalhos que estão programados para a vinha anteriormente selecionada, o estado ("Executado", "Em execução" ou "Por executar") e a data limite de execução de cada um desses trabalhos. Quando um novo trabalho é programado pelo produtor de vinho, o seu estado é "Por executar", sendo que, este pode ser alterado para "Em execução" ou "Executado" pelo empreiteiro responsável pela coordenação dos trabalhos no terreno.

Tal como acontece com as anomalias também neste *layout* o produtor de vinho pode eliminar um trabalho que já não tenha interesse que esteja presente na lista, pode ver e atualizar os trabalhos existentes e pode também adicionar um novo trabalho à lista. Para eliminar um

trabalho o produtor de vinho deve proceder do mesmo modo que foi anteriormente descrito para eliminar as anomalias assim como para ver e atualizar as informações de um dos trabalhos, sendo que, no caso dos trabalhos, ao selecionar um dos itens da lista o utilizador não é encaminhado para um novo *layout*, uma vez que as informações respeitantes ao trabalho selecionado são exibidas no mesmo *layout*, por baixo da lista de trabalhos, tal como se pode observar na Figura 20.

Neste *layout* os trabalhadores e os empreiteiros poderão ver os trabalhos programados e as informações relativas a esses mesmos trabalhos para procederem à sua execução com eficácia, contudo não podem alterar essas mesmas informações. Os empreiteiros podem apenas alterar o estado de execução dos trabalhos para que os produtores de vinho possam estar a par da execução dos mesmos.

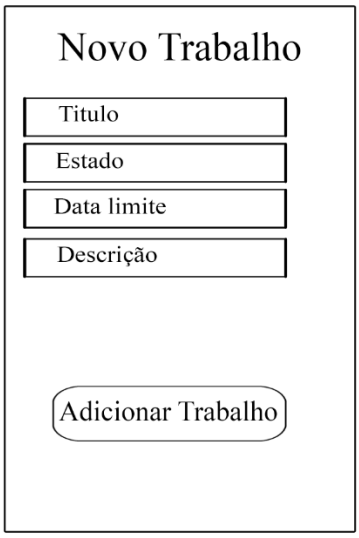

Figura 21 - Modelo de layout para a inserção de um novo trabalho.

Para adicionar um novo trabalho o produtor de vinho deve selecionar a opção "Novo" e, tal como acontecia com as anomalias, é direcionado para outro *layout* (Figura 21) no qual terá um formulário para preencher com as diferentes informações que dizem respeito ao trabalho que está prestes a adicionar.

Ainda no *layout* do menu (Figura 16), se o utilizador (trabalhador, empreiteiro ou produtor de vinho) optar por selecionar a terceira opção ("Parcelas") é reencaminhado para um *layout* (Figura 22) no qual terá: uma lista das diferentes parcelas que constituem a vinha anteriormente selecionada e um mapa no qual é possível visualizar (através de imagens de

satélite) os limites das parcelas que o utilizador selecionar e outras informações respeitantes a essas parcelas.

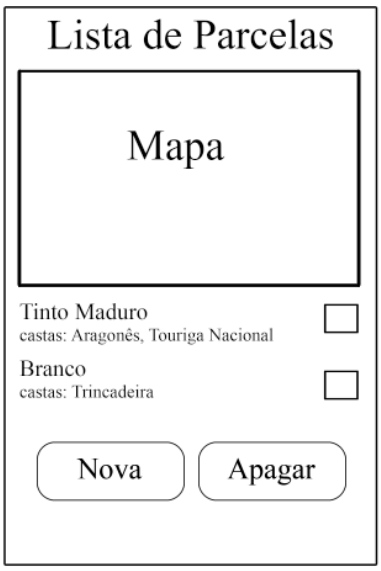

Figura 22 - Modelo de layout da lista de parcelas.

Neste *layout* o produtor de vinho pode também eliminar uma parcela ou adicionar uma nova.

Ao selecionar a opção "Nova" o produtor de vinho é direcionado para outro *layout* (Figura 23) no qual terá um pequeno formulário para preencher com as informações que dizem respeito à parcela que está prestes a adicionar e no qual poderá delimitar a parcela no mapa.

Figura 23 - Modelo de layout para a inserção de uma nova parcela.

Para delimitar a parcela, o produtor de vinho deve ativar a localização do seu dispositivo e à medida que se move no terreno vai clicando sobre o botão “Adicionar Ponto”, para que as coordenadas pelas quais o produtor de vinho se move sejam adicionadas no mapa. Assim que todos os pontos que delimitam a parcela estiverem no mapa, o produtor de vinho pode ligá-los através do botão “Desenhar” e desta forma obter no mapa os limites da nova parcela. Caso se tenha enganado a adicionar os pontos, ou não pretenda adicionar aqueles pontos por onde passou pode também selecionar a opção “Limpar Mapa” e eliminar do mapa os pontos que tinha adicionado.

Depois de adicionar os limites da parcela e as informações no formulário o produtor de vinho pode adicionar a nova parcela à lista do *layout* anterior (Figura 22) através do botão “Adicionar Parcela”.

3.5. Base de Dados

O objetivo deste projeto é o desenvolvimento da plataforma móvel em si e não o desenvolvimento da base de dados de suporte à aplicação. Contudo apresenta-se aqui um modelo de uma base de dados que cumpre os requisitos fundamentais para o bom funcionamento da aplicação, sendo que, este modelo não será o único possível para o funcionamento da plataforma.

A Figura 24 ilustra uma visão geral do modelo de base de dados proposto, que permite armazenar os dados essenciais para o projeto em questão.

A tabela “vinhas” será a tabela que possuirá maior relacionamento entre a maioria das tabelas e terá os seguintes atributos:

- Id_grupo (identificação do grupo a que pertence a vinha);

- Id_vinha (identificação da vinha);

- Nome (nome da vinha);

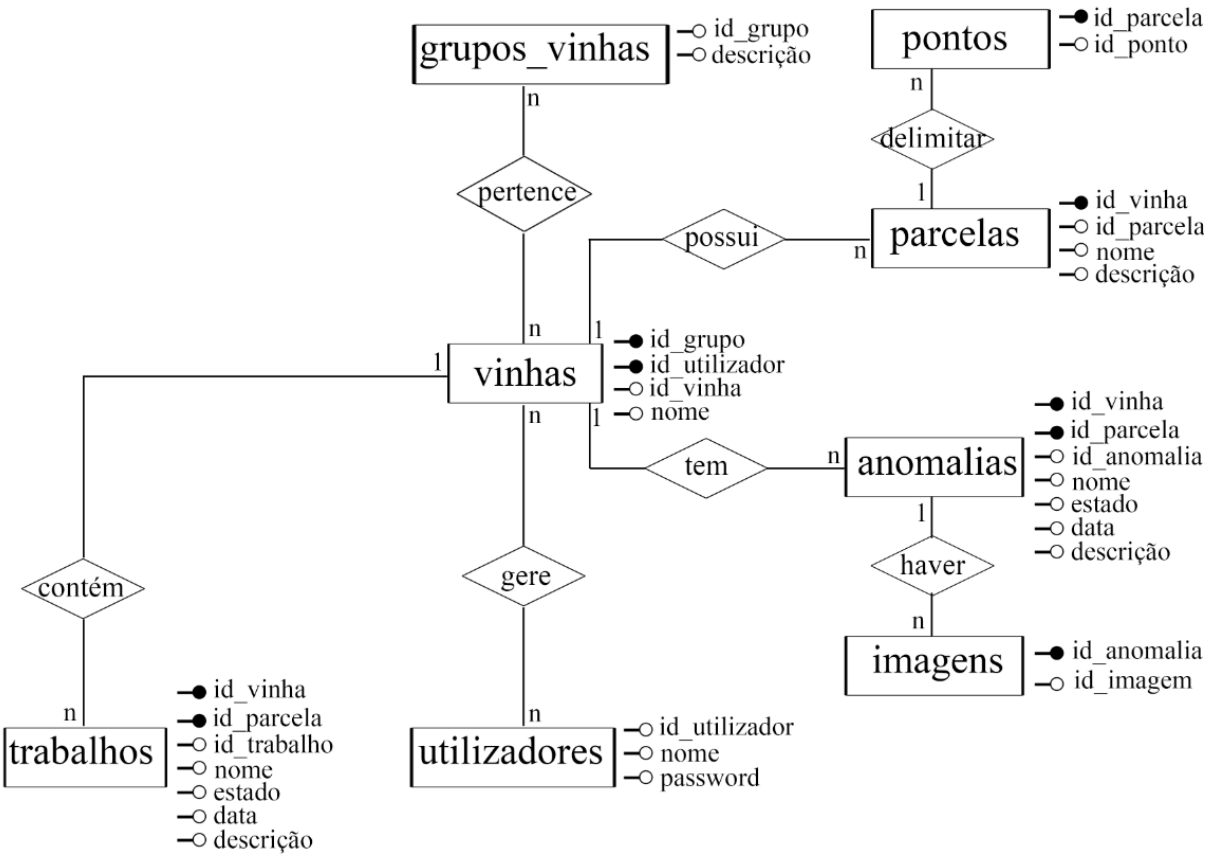

Figura 24 - Diagrama E-R da base de dados.

Exemplo de Pesquisa: A vinha "Faijo", nº1 pertence ao grupo 5.

Cada vinha pertencerá a um grupo, portanto criar-se-á a tabela "grupos_vinhas" com os seguintes atributos:

- Id_grupo (identificação do grupo);

- Descrição (breve descrição sobre o grupo).

Exemplo de Pesquisa: As vinhas "Faijo" e "Louredo" pertencem ao grupo 5.

Todas as vinhas são constituídas por diferentes parcelas, portanto criar-se-á a tabela "parcelas" com os seguintes atributos:

- Id_vinha (identificação da vinha à qual corresponde a parcela em causa);

- Id_parcela (identificação da parcela);

- Nome (nome da parcela);

- Descrição (breve descrição da parcela);

- Pontos (coordenadas dos pontos que delimitam a parcela).

Exemplo de Pesquisa: A parcela "Tinto" nº 4 pertence à vinha nº7 e tem como extremos os pontos a,b,c,d,e,f e g.

Também em todas as vinhas poderão ser identificadas anomalias, muitas vezes só numa parte da vinha, portanto criar-se-á também a tabela "anomalias" com os seguintes atributos:

- Id_vinha (identificação da vinha em que foi registada a anomalia);

- Id_parcela (identificação das parcelas em que foi registada a anomalia);

- Id_anomalia (identificação da anomalia);

- Nome (nome da anomalia);

- Estado (estado de resolução da anomalia);

- Data (data de registo da anomalia);

- Descrição (descrição da anomalia);

- Imagens (imagens fotográficas da anomalia).

Exemplo de Pesquisa: Na vinha 5 foi registada a anomalia "queda de bardo" no dia 23-6-2017.

Podem existir ainda em todas as vinhas trabalhos para realizar e, portanto, criar-se-á a tabela "trabalhos" com os seguintes atributos:

- Id_vinha (identificação da vinha onde deve ser efetuado o trabalho);

- Id_parcela (identificação das parcelas onde deve ser efetuado o trabalho);

- Id_trabalho (identificação do trabalho);

- Nome (nome do trabalho);

- Estado (estado do trabalho);

- Data (data limite para a realização do trabalho);

- Descrição (descrição do trabalho a realizar).

Exemplo de Pesquisa: Na vinha 5 é necessário realizar o trabalho “vindima” até ao dia 1-10-2017.

Estas são as principais tabelas que devem ser criadas para o funcionamento da aplicação e estes são exemplos de pesquisa semelhantes às pesquisas que serão feitas pelos utilizadores aquando da utilização da aplicação, portanto esta base de dados proposta parece ir de encontro a todas as necessidades da aplicação no âmbito do armazenamento dos dados.

Capítulo 4

4. Implementação

Neste capítulo são descritos os pormenores de implementação da plataforma móvel de gestão de dados em vinhas, que foi desenvolvida com a ferramenta de programação Android Studio e com recurso à linguagem de programação *Java*. Na implementação deste projeto foram criadas várias classes baseadas no modelo de desenvolvimento MVC, no padrão Singleton e para a exibição de mapas na aplicação foi utilizada a API do Google Maps.

4.1. Arquitetura da Aplicação

Antes de descrever os pormenores de implementação da plataforma móvel de gestão de dados em vinhas é importante explicar a sua arquitetura, portanto nesta secção será descrito o funcionamento da plataforma bem como as tecnologias utilizadas.

Depois da análise de requisitos feita anteriormente neste documento, foi desenvolvido um diagrama de arquitetura, que se pode observar na Figura 25 e que demonstra como foi implementado o protótipo da plataforma.

A plataforma móvel de gestão de dados em vinhas possui duas fases principais de funcionamento. A primeira fase, que é a que diz respeito a esta dissertação, incide sobre o desenvolvimento de uma aplicação para dispositivos móveis Android que permite aos seus utilizadores registar anomalias, programar trabalhos e adicionar parcelas numa vinha. A segunda fase, que não diz respeito a esta dissertação, por isso será desenvolvida por outras equipas que fazem parte do projeto, incide sobre a parte do servidor que irá receber e enviar os dados para a plataforma.

Para a implementação da primeira fase deste protótipo foi utilizado o IDE (*Integrated Development Environment*) Android Studio, bem como os componentes de suporte à linguagem *Java*, o JDK (*Java Development Kit*) e o JRE (*Java Runtime Environment*). Foi utilizado também o Android SDK, que é o conjunto de ferramentas de desenvolvimento de aplicações móveis para a plataforma Android.

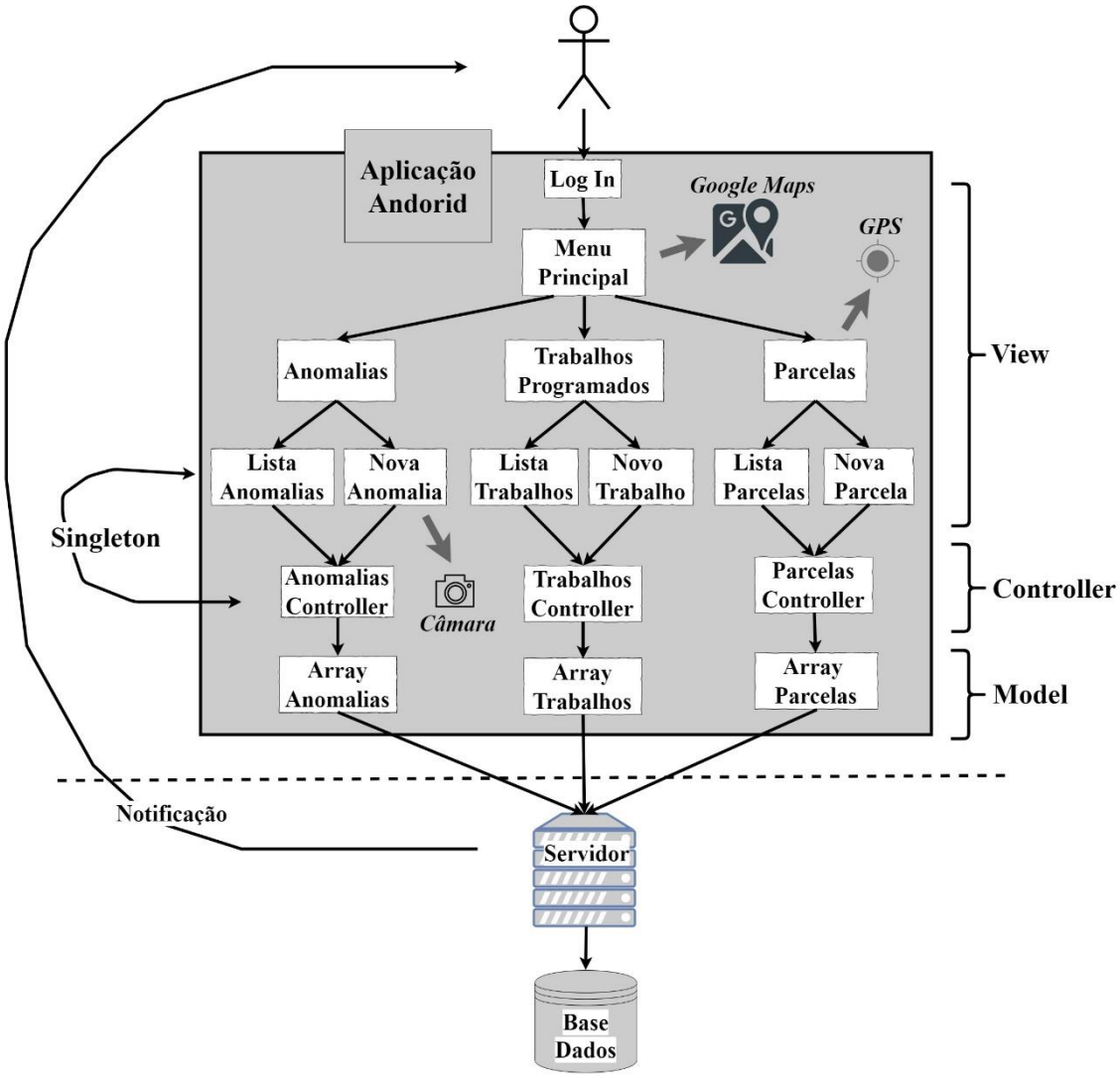

Figura 25 - Diagrama de Arquitetura da Aplicação.

Para a troca dos dados entre as *Views* e os *Controllers* foram criadas instâncias, tal como sugere o padrão de desenvolvimento Singleton. Esses dados, antes de serem enviados para o servidor, são guardados na aplicação sob a forma de *Arrays*.

O desenvolvimento deste protótipo foi todo ele baseado no modelo MVC. Desta forma o *Controller* faz a ligação entre o *Model* (dados da aplicação) e a *View* (apresenta os dados e capta as ações do utilizador). Para melhor perceber a utilização do modelo MVC no desenvolvimento deste projeto apresentam-se de seguida, na Figura 26, Figura 27 e Figura 28, os diagramas de classes referentes às funcionalidades mais relevantes da plataforma.

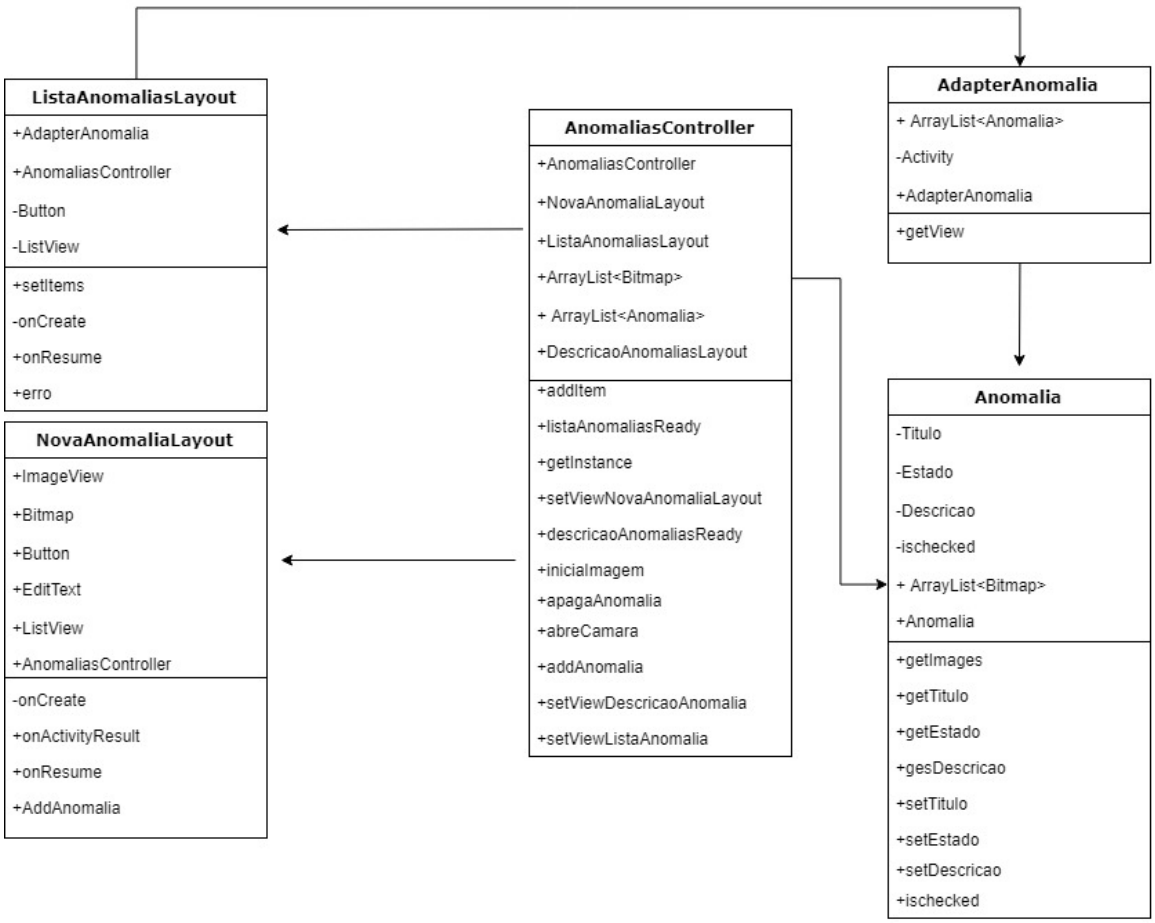

Figura 26 - Diagrama de classes da funcionalidade anomalias.

Para adicionar uma nova anomalia a classe *NovaAnomaliaLayout* disponibiliza um formulário (que o utilizador terá de preencher com os dados referentes à anomalia) e possibilita ainda a captura de imagens com a câmara do dispositivo móvel. Depois de submetidos os dados (através do botão Adicionar Anomalia) a classe *AnomaliasController* recolhe-os e guarda-os num *Array*. A classe *AnomaliasController* é ainda responsável pelo envio dos dados guardados, para a classe *ListaAnomaliasLayout* (para que estes possam ser exibidos ao utilizador).

Para a exibição dos dados através da classe *ListaAnomaliasLayout*, é ainda utilizada uma outra classe (*AdapterAnomalia*) que permite que os dados armazenados no *Array* sejam exibidos na interface desenvolvida.

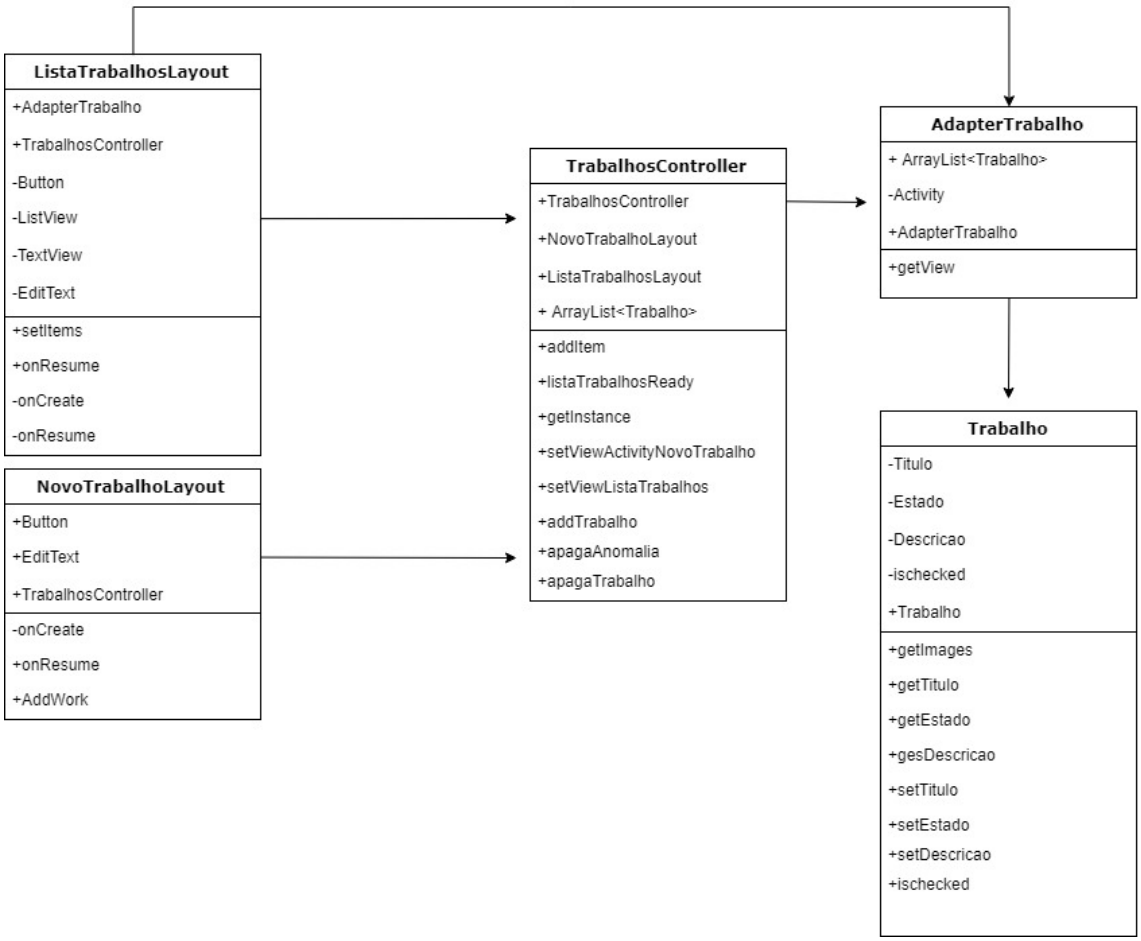

Figura 27 - Diagrama de classes da funcionalidade trabalhos programados.

Para programar um novo trabalho a classe *NovoTrabalhoLayout* disponibiliza um formulário (que o utilizador terá de preencher com os dados referentes ao trabalho).

Depois de submetidos os dados (através do botão Adicionar Trabalho) a classe *TrabalhosController* recolhe-os e guarda-os num *Array*. A classe *TrabalhosController* é ainda responsável pelo envio dos dados guardados, para a classe *ListaTrabalhosLayout* (para que estes possam ser exibidos ao utilizador).

Para a exibição dos dados através da classe *ListaTrabalhosLayout* é ainda utilizada uma outra classe (*AdapterTrabalho*) que permite que os dados armazenados no *Array* sejam exibidos na interface desenvolvida.

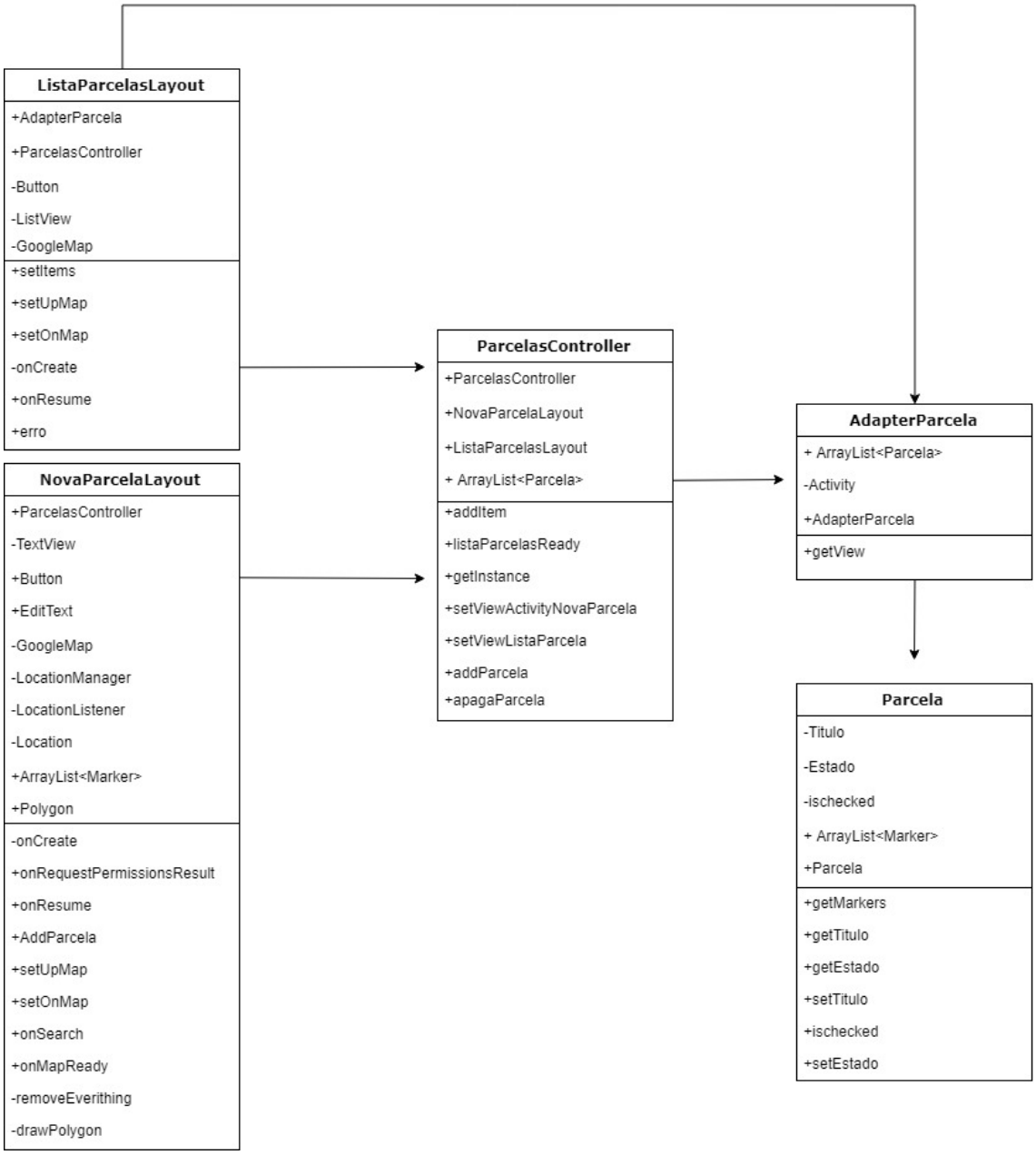

Figura 28 - Diagrama de classes da funcionalidade parcelas.

Para adicionar uma nova parcela a classe *NovaParcelaLayout* disponibiliza um formulário (que o utilizador terá de preencher com os dados referentes à parcela), possibilita a adição dos pontos que delimitam a parcela (através das coordenadas do GPS do dispositivo móvel) e possibilita ainda a ligação dos pontos adicionados através do desenho de um polígono.

Depois de submetidos os dados (através do botão Adicionar Parcela) a classe *ParcelasController* recolhe-os e guarda-os num *Array*. A classe *ParcelasController* é ainda responsável pelo envio dos dados guardados para a classe *ListaParcelasLayout* (para que

estes possam ser exibidos ao utilizador).

À semelhança do que acontecia com as funcionalidades anteriores, para a exibição dos dados, através da classe *ListaParcelasLayout*, é utilizada uma outra classe (*AdapterParcela*) que permite que os dados armazenados no *Array* sejam exibidos na interface desenvolvida.

A título de exemplo, a Figura 29 demonstra as etapas que ocorrem quando é registada uma nova anomalia sendo que, esta sequência de etapas é semelhante noutras funcionalidades da aplicação.

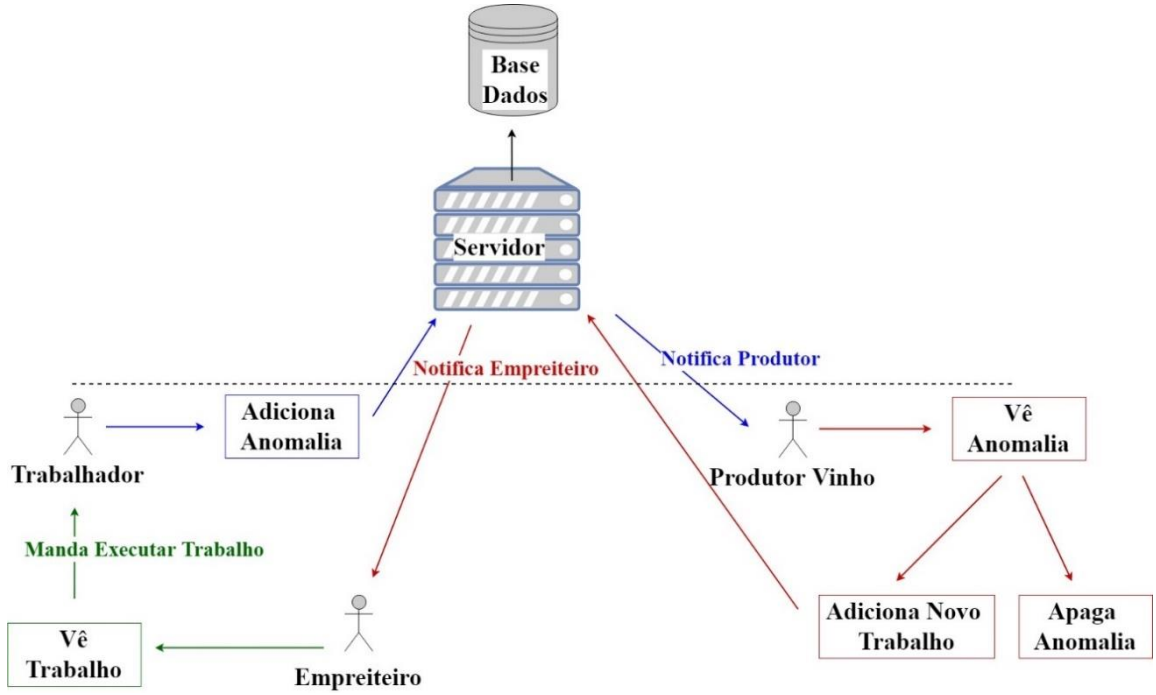

Figura 29 - Workflow do registo de uma anomalia.

Tal como se pode observar na Figura 29, o trabalhador adiciona uma nova anomalia, que posteriormente é enviada para um servidor. Depois disso, o servidor envia uma notificação ao produtor de vinho indicando que uma nova anomalia foi adicionada pelo trabalhador. Assim que visualiza a anomalia, o produtor de vinho pode considerá-la irrelevante (e optar por eliminá-la) ou pode agendar um novo trabalho para resolução da respetiva anomalia. Se decidir agendar um novo trabalho, os dados relativos a esse trabalho são enviados para o servidor que notifica o empreiteiro responsável pelos trabalhos naquela vinha. Assim que o empreiteiro visualiza o trabalho agendado pelo produtor de vinho, manda os seus trabalhadores executá-lo. Quando o trabalho estiver terminado o empreiteiro pode alterar o

seu estado de execução, sendo novamente notificado o produtor de vinho de que o trabalho que agendou foi executado.

É de referir mais uma vez que as funcionalidades relacionadas com o servidor não são objeto de estudo desta dissertação e por isso, até ao momento, os dados são armazenados localmente pela aplicação.

4.2. Implementação dos *Layouts* da Aplicação

Na primeira fase de desenvolvimento foram criados os *layouts* de acordo com os modelos definidos na conceção deste projeto, tal como se pode verificar na Figura 30.

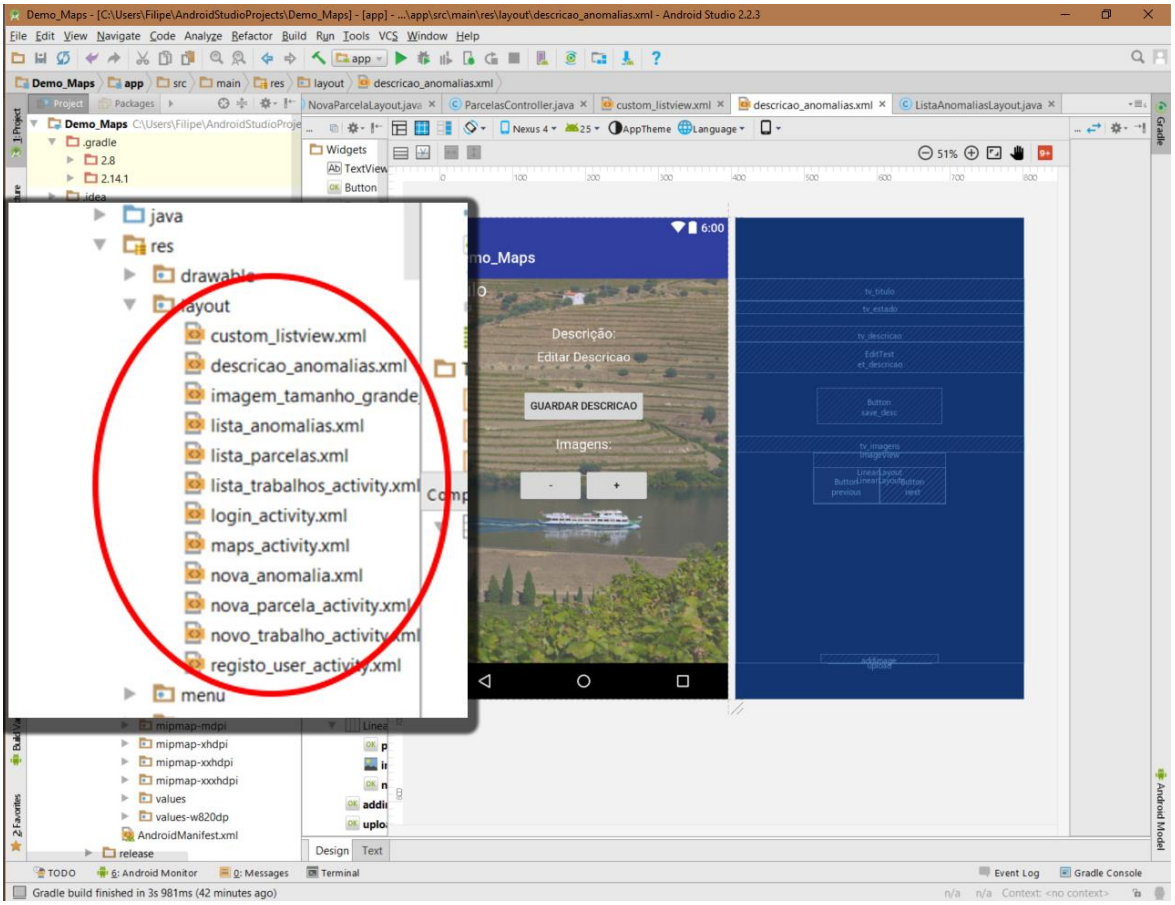

Figura 30 - Criação dos layouts da aplicação.

Os *layouts* criados, seguiram a especificação de *layouts* "*Linear Layout*" do Android uma vez que esta especificação permite que um *layout* contenha sub-elementos alinhados de uma forma sequencial, quer na orientação vertical, quer na orientação horizontal do ecrã, tal como

se pode observar na Figura 31 e permite ainda que o tamanho e posição dos elementos sejam definidos pelo programador. Estas caraterísticas da especificação "*Linear Layout*" facilitam os testes no emulador porque independentemente do conteúdo dos elementos do *layout* (botões, caixas de texto, etc.) estes mantêm-se fixos no lugar em que foram colocados inicialmente.

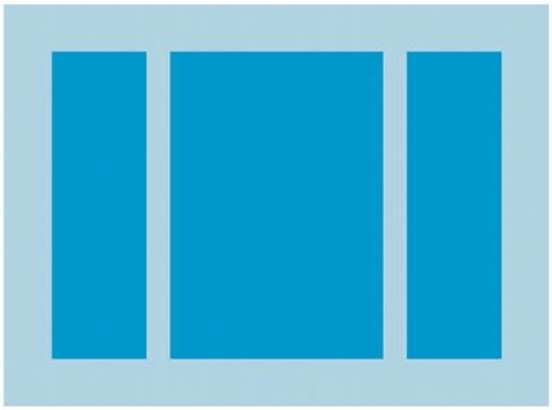

Figura 31 - Sub-elementos organizados na orientação horizontal [1].

Para inserir o mapa do Google foi necessário criar um *fragment* (módulo) dentro do próprio *layout* de modo a suportar o mapa, tal como se pode ver na Figura 32.

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="261dp"
        tools:context="com.example.filipe.demo_maps.Blayouts.MapsLayout"/>

</LinearLayout>
```

Figura 32 - Código do módulo que suporta o mapa dentro do layout.

Para utilizar a API do Google Maps foi ainda necessário registar o projeto no Google API Console e obter uma chave de utilização da API do Google Maps. Essa chave teve de ser adicionada no *"AndroidManifest.xml"* do projeto, tal como se pode observar na Figura 33, e de seguida teve de se recompilar o projeto.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyBeQGvEOb36agKMj64nTxn2ILBaSFoOgew" />
```

Figura 33 - Chave da API do Google no AndroidManifest.xml.

4.3. Implementação do Modelo MVC

Depois de criar os *layouts* e os elementos necessários para a parte gráfica da aplicação foi necessário proceder à programação das funcionalidades desses mesmos elementos. Só a partir daqui é que a aplicação começa a executar algumas funções e torna-se possível interagir com ela.

Em primeiro lugar começou-se por programar os botões para alternar entre os diferentes *layouts*, sendo que, este código ficou sempre em classes do tipo *view* (do modelo MVC), tal como se pode observar na Figura 34. A alternância entre *layouts* não é uma funcionalidade que envie ou receba dados inseridos pelo utilizador, por isso, sendo uma funcionalidade que diz respeito aos *layouts* (*views*) deve ficar numa classe do tipo *view*.

Figura 34 - Código de alternância entre layouts.

Seguidamente foram programadas as funcionalidades que envolvem a transferência de dados (texto, imagens, etc.), como por exemplo a inserção de anomalias, de trabalhos e de parcelas. Para a implementação destas funcionalidades, as classes do tipo *controller* foram separadas das classes do tipo *view*, tal como se pode ver na Figura 35, para que os dados recolhidos pelas *views* fossem processados pelo *controller*. Desta forma as *views* ficam apenas com a responsabilidade de recolher os dados inseridos pelo utilizador e de os enviar para o *controller*, que de seguida os guarda em *arrays*. Com a implementação deste modelo é possível que no futuro o *controller* possa ser alterado, para realizar outras ações com os dados e a *view* se mantenha independente dessas ações.

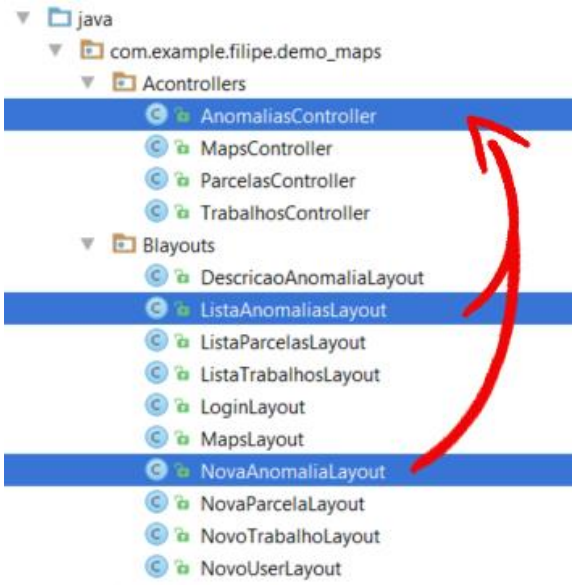

Figura 35 - Separação das classes controller das classes view.

4.4. Implementação do Padrão Singleton

O padrão Singleton permite que várias classes do tipo *view* (que executem funcionalidades sobre os mesmos dados) sejam comandadas pelo mesmo *controller*, tal como se pode observar na Figura 36.

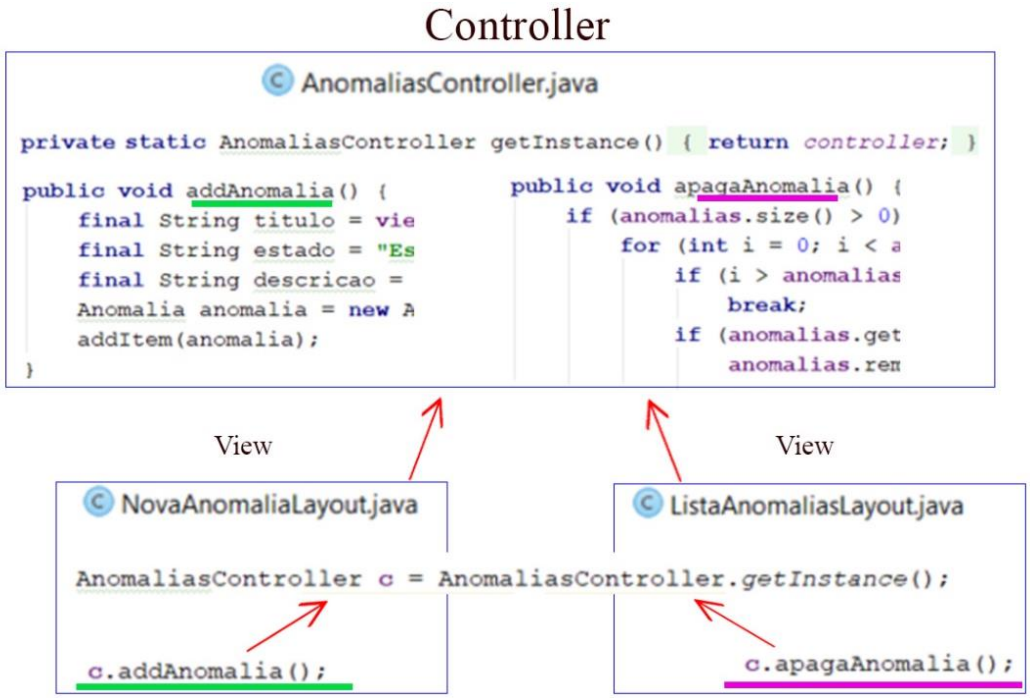

Figura 36 - Exemplo de duas views comandadas pelo mesmo controller.

Por exemplo: na *view* "*NovaAnomaliaLayout*", são enviados dados de uma anomalia para o *controller* "*AnomaliasController*" e esses dados podem mais tarde ser eliminados na *view* "*ListaAnomaliasLayout*".

Para que isso seja possível, em primeiro lugar, é preciso criar uma classe *AnomaliasController*, tal como se pode observar na Figura 37. A classe *AnomaliasController* tem o seu construtor privado e tem uma instância de si própria, fornecendo ainda um método estático que permite exibir a sua instância para o mundo exterior.

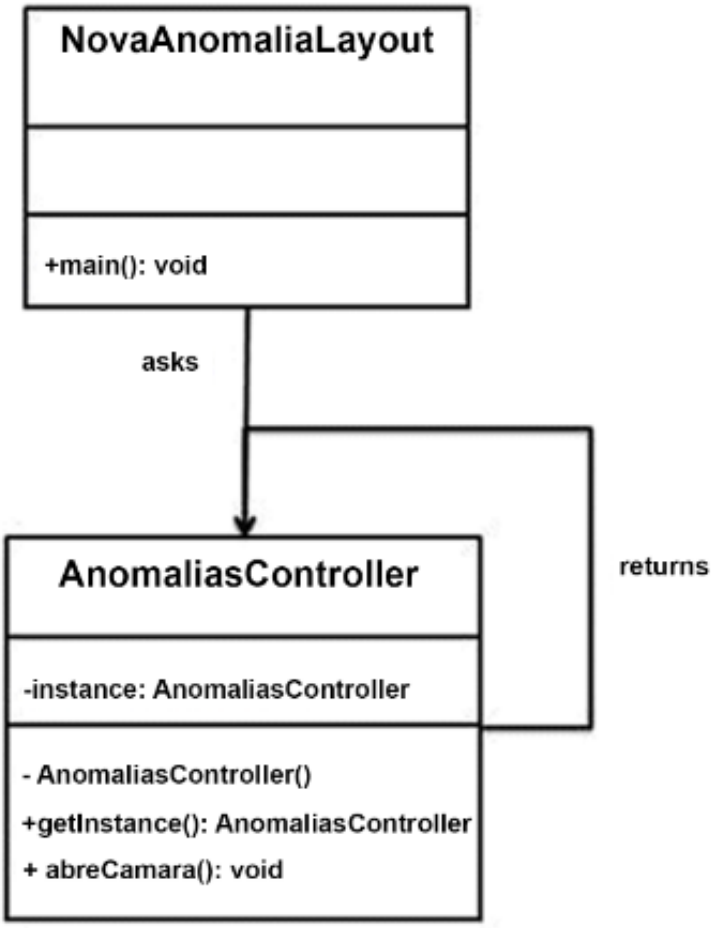

Figura 37 - Fluxo de implementação do padrão Singleton.

Para melhor perceber a implementação deste padrão apresenta-se na Figura 38 e Figura 39 o código das duas classes implementadas no desenvolvimento deste projeto.

```
public class AnomaliasController {

        //criar um objeto de AnomaliasController
 private static AnomaliasController controller = new AnomaliasController();

        //tornar o construtor privado de modo que esta classe não possa ser
        //instanciada
 private AnomaliasController(){}

       //Obter o único objeto disponível
 public static AnomaliasController getInstance() {
        return controller;
 }

 public void abreCamara() {
       Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

       if (intent.resolveActivity(viewNovaAnomaliaLayout.getPackageManager()) != null) {
       viewNovaAnomaliaLayout.startActivityForResult(intent, viewNovaAnomaliaLayout.REQUEST_IMAGE_CAPTURE);
       }

    }
}
```

Figura 38 - Excerto de código da classe AnomaliasController.

A classe *NovaAnomaliaLayout*, que se pode observar na Figura 39, é uma classe exemplo que irá usar a classe *AnomaliasController*, exibida na Figura 38, para obter um objeto *AnomaliasController*.

```
public class NovaAnomaliaLayout {

          //Obter o único objeto disponível
          AnomaliasController c = AnomaliasController.getInstance();

        // Abrir a câmara
        buttoncamara.setOnClickListener(new View.OnClickListener() {

                public void onClick(View v) {

                c.abreCamara();

                }
         });

}
```

Figura 39 - Excerto de código da classe NovaAnomaliaLayout.

4.5. Deteção da Localização do Utilizador

Uma outra funcionalidade desta aplicação é a deteção da localização do utilizador para que seja possível marcar pontos no mapa e delimitar os limites de uma parcela. Para isso foi necessário fornecer à aplicação permissões para aceder à localização do *smartphone*, tal como se pode ver na Figura 40.

```
(ActivityCompat.checkSelfPermission
(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission
(this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
```

Figura 40 - Código de permissão de acesso à localização GPS do smartphone.

Depois de ter as permissões necessárias para aceder à localização, a aplicação mostra no mapa a posição atual do utilizador e permite que este adicione um ponto nesse local através do clique num botão ("Adicionar Ponto"). Sempre que o utilizador efetuar essa ação a aplicação recolhe as coordenadas de latitude e longitude do *smartphone* e adiciona um ponto ("*Marker*") no mapa, tal como se pode observar na Figura 41.

```
buttonponto.setOnClickListener((view) → {
        LatLng latLng2 = new LatLng(lastLocation.getLatitude(), lastLocation.getLongitude());
        MarkerOptions options = new MarkerOptions()
                .position(latLng2)
                .snippet(" "+ latLng2)
                .draggable(true)
                .title("Marker1");
        markers.add(mMap.addMarker(options));


});
```

Figura 41 - Código para adicionar um marker no mapa.

Como se pode ainda observar na Figura 42, assim que existirem pelo menos três pontos marcados no mapa é possível desenhar um polígono.

```
buttondesenhar.setOnClickListener((view) → {
        if (markers.size() == 0) {
            Toast.makeText(getApplicationContext(), "Adicione Pontos", Toast.LENGTH_SHORT).show();

        } else if (markers.size() >= 3) {
            PolygonOptions options = new PolygonOptions()
                    .fillColor(0x330000FF)
                    .strokeWidth(3)
                    .strokeColor(Color.RED);
            for (int i = 0; i < markers.size(); i++) {
                options.add(markers.get(i).getPosition());
            }
            shape = mMap.addPolygon(options);
        } else {
            Toast.makeText(getApplicationContext(), "Adicione Pontos", Toast.LENGTH_SHORT).show();
        }
});
```

Figura 42 - Código para desenhar um polígono no mapa.

Desta forma, a delimitação da parcela, é o resultado do desenho de um polígono com três ou mais pontos, tal como se pode observar na Figura 1Figura 43.

Figura 43 - Resultado de um polígono com três "markers".

Capítulo 5

5. Testes e Resultados

Este capítulo serve para apresentar os testes e resultados da aplicação elaborada e retirar conclusões a partir dos mesmos. De realçar que todos os testes aqui apresentados foram testes de prova de conceito, não se tratando portanto, de ensaios efetuados por pessoal especializado em trabalhos na vinha.

5.1. Utilização da Aplicação em Contexto Real

Para uma melhor perceção do funcionamento da aplicação foram efetuados testes de usabilidade em contexto real numa vinha do campus da Universidade de Trás os Montes e Alto Douro. Para a realização destes testes foi pedida a ajuda de uma colaboradora para simular um trabalhador da vinha e utilizou-se um *smartphone* com o sistema operativo Android para correr a aplicação.

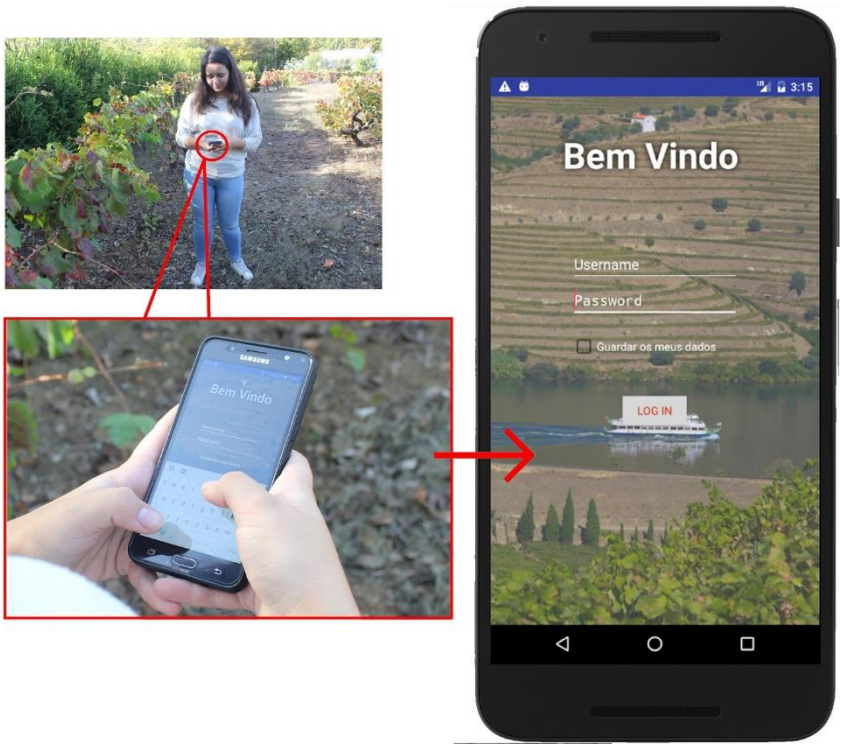

Figura 44 - Log in na aplicação.

Assim que acede à aplicação o utilizador tem de efetuar a autenticação de forma a poder utilizar as funcionalidades da mesma, tal como se pode observar na Figura 44.

O utilizador pode ainda selecionar se deseja que a aplicação guarde os seus dados de autenticação para passar à frente este passo em acessos futuros à aplicação.

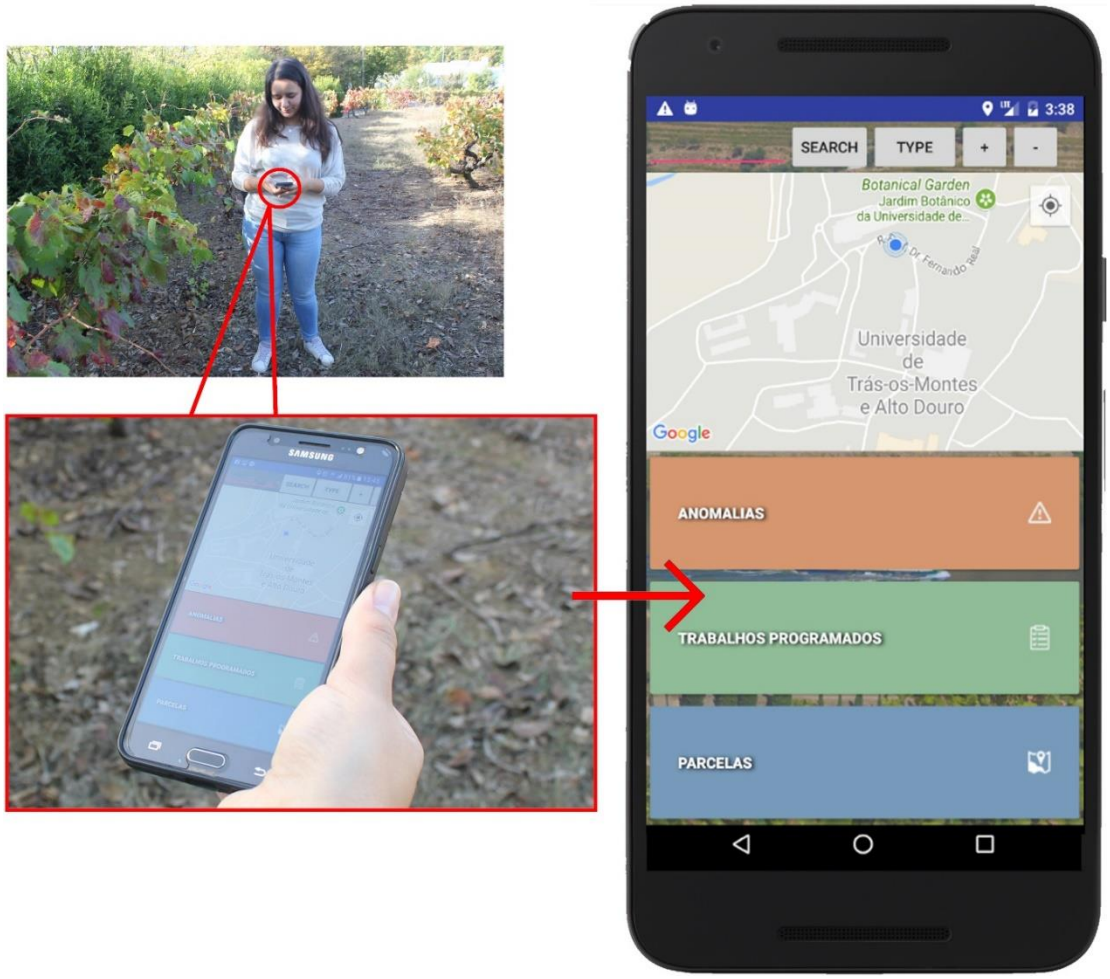

Figura 45 - Menu principal da aplicação.

Depois de se autenticar, a localização do utilizador é automaticamente detetada pela aplicação, através do GPS do *smartphone*, e desta forma o utilizador sabe que todas as suas ações serão efetuadas para a vinha da UTAD, tal como se pode observar na Figura 45.

Apesar disso, o utilizador pode escolher no mapa uma outra vinha e aceder aos dados da vinha que desejar. É importante também referir que antes de utilizar a aplicação o utilizador deve ativar o GPS do seu *smartphone*, caso contrário a sua localização não será detetada.

Figura 46 - Inserção de uma nova anomalia.

Ao percorrer a vinha o utilizador depara-se com um bardo caído e então decide adicionar essa anomalia para que mais tarde alguém a resolva. Para isso acede ao separador anomalias selecionando “Nova Anomalia”, tal como se pode observar na Figura 46.

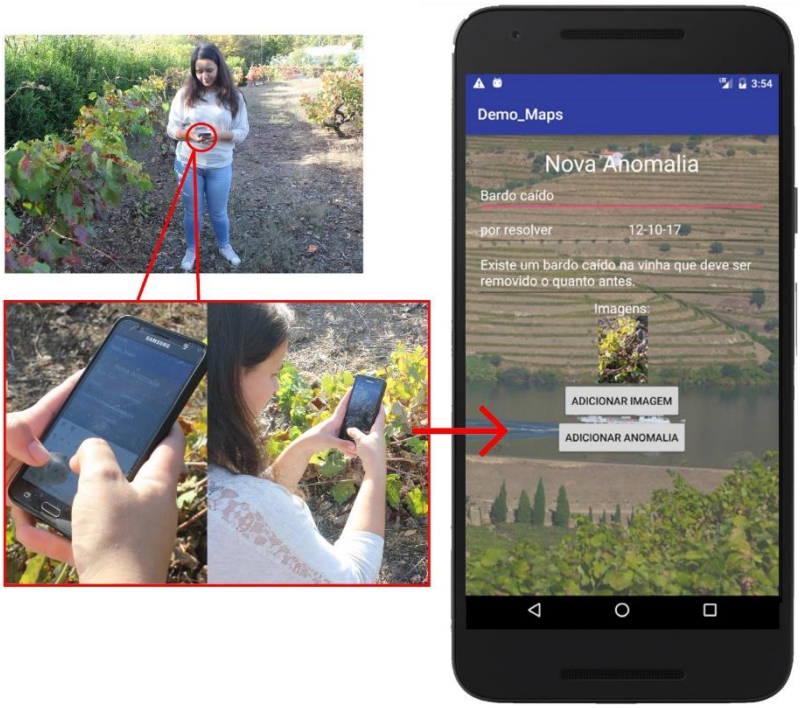

Figura 47 - Preenchimento das informações respeitantes à anomalia.

Depois de selecionar “Nova Anomalia”, no *layout* anterior, o utilizador é reencaminhado para um *layout* no qual tem de preencher um formulário, com as informações relativas à anomalia que deteta no seu percurso e no qual adiciona algumas imagens da mesma, para uma melhor perceção da anomalia em causa, tal como se pode observar na Figura 47.

Para adicionar uma ou mais imagens relativas à anomalia o utilizador deve selecionar a opção “Adicionar Imagem” de forma a poder capturar uma imagem com a câmara do seu dispositivo.

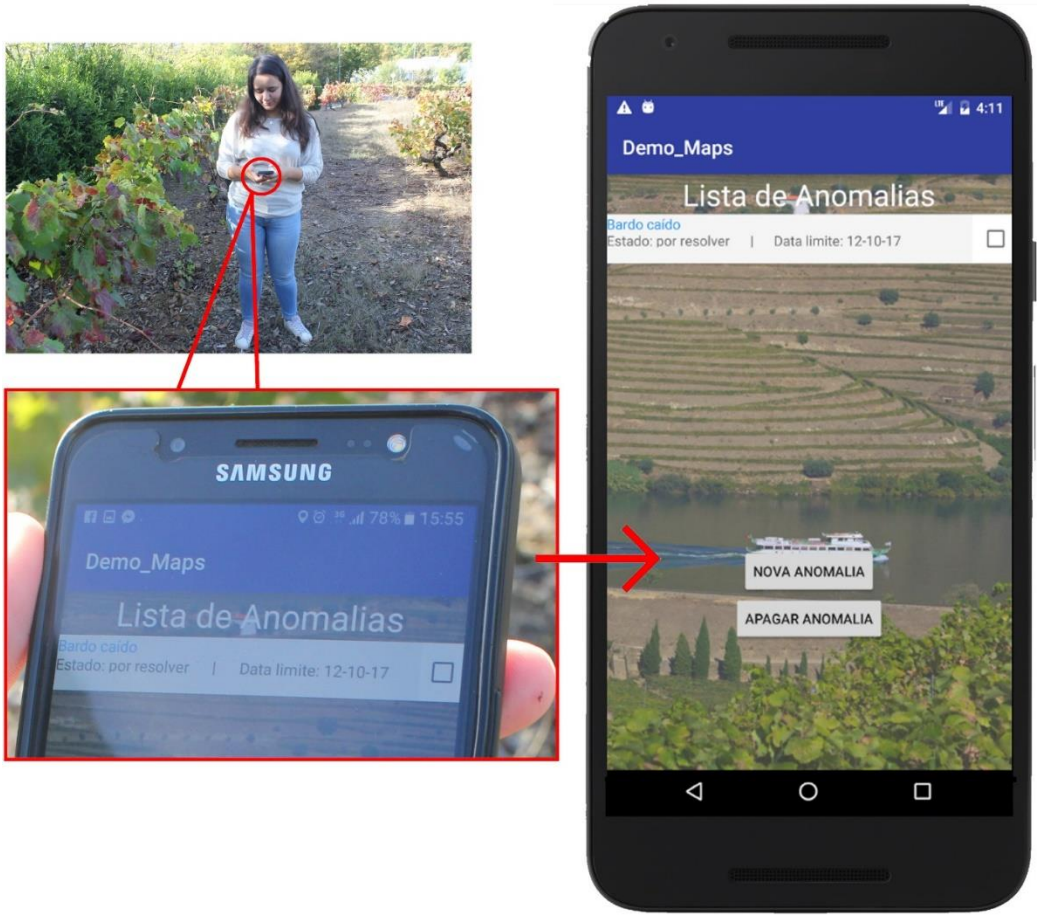

Figura 48 - Lista de anomalias.

Depois de preencher todas as informações relativas à anomalia e de selecionar “Adicionar Anomalia” o utilizador adiciona a anomalia à lista de anomalias da vinha da UTAD, tal como se pode observar na Figura 48.

Ao continuar o seu percurso pela vinha o utilizador verifica que é necessário proceder ao corte do mato rasteiro presente entre os bardos e para isso decide programar um novo trabalho.

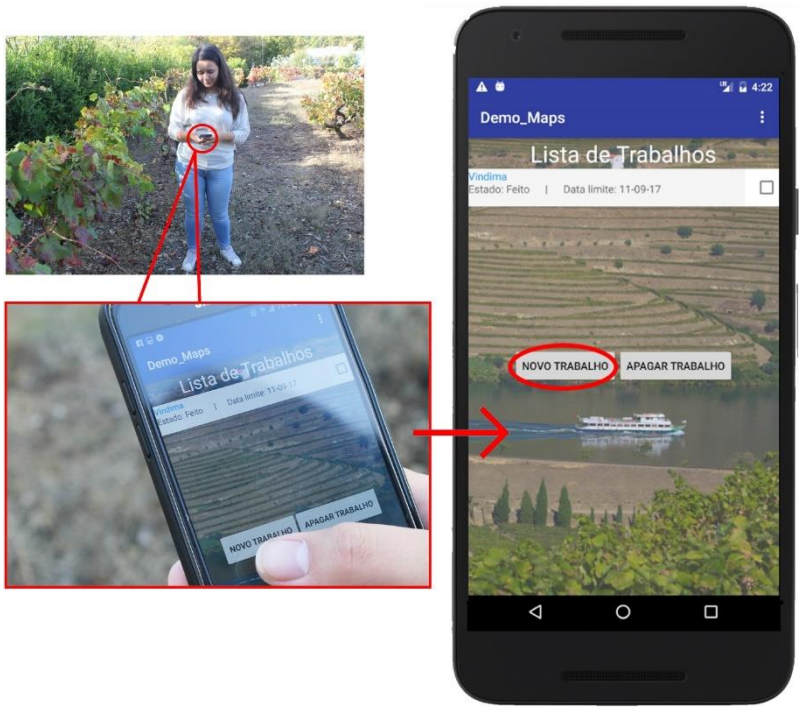

Figura 49 - Inserção de um novo trabalho.

O utilizador acede ao menu "Trabalhos Programados" e seleciona "Novo Trabalho" para adicionar as informações relativas ao trabalho que pretende que seja efetuado, tal como se pode observar na Figura 49.

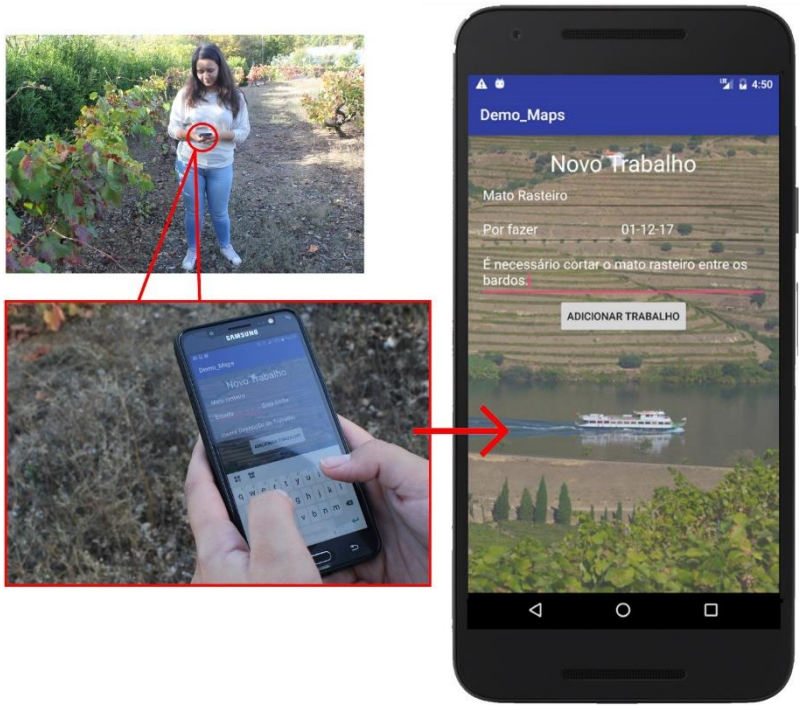

Figura 50 - Preenchimento das informações respeitantes ao trabalho.

Ao efetuar essa ação o utilizador é reencaminhado para uma outra página na qual preenche um formulário com a descrição do trabalho a efetuar e depois disso adiciona-o à lista de trabalhos programados, tal como se pode observar na Figura 50.

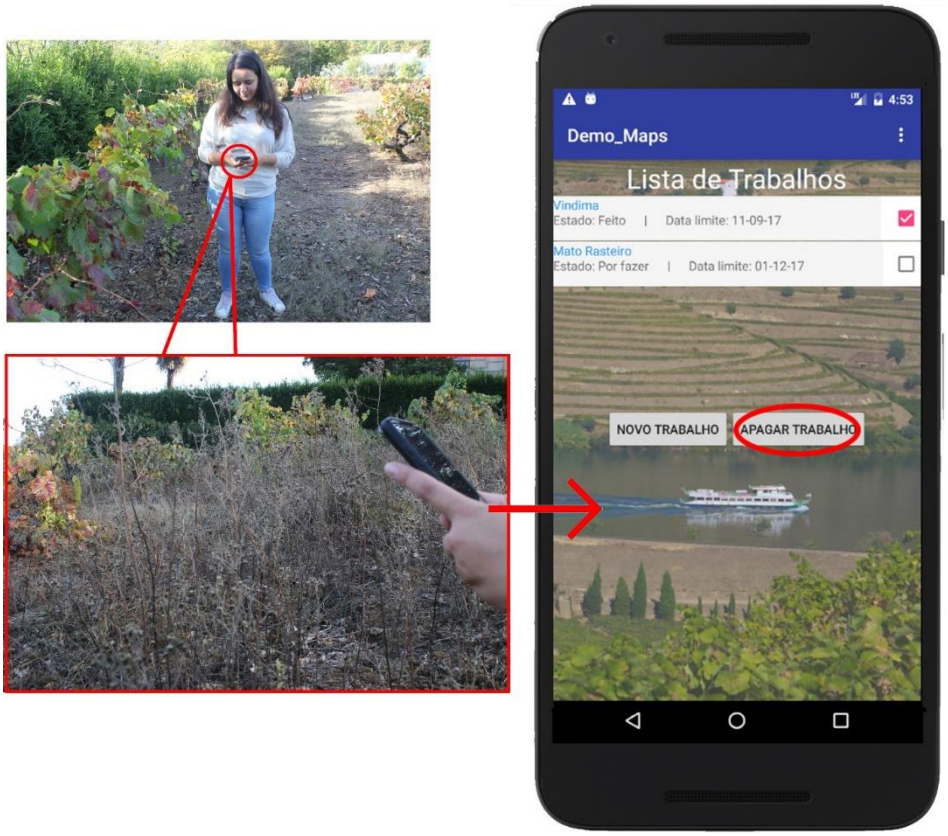

Figura 51 - Lista de trabalhos.

Depois de adicionar o trabalho à lista de trabalhos programados, o utilizador verifica que existe na lista um trabalho que já foi executado (Vindima) e por isso decide removê-lo da lista através do botão "Apagar Trabalho", tal como se pode observar na Figura 51.

Tanto nas listas de anomalias como nas listas de trabalhos programados é possível eliminar itens através da sua seleção na c*heckbox* e do clique no botão "Apagar (Anomalia ou Trabalho)". Ainda nestas listas, se o utilizador selecionar um dos itens, pode voltar à página de formulário do respetivo item (Anomalia ou Trabalho) e editar, adicionar ou eliminar informações que anteriormente foram adicionadas relativamente a esse mesmo item.

Ainda antes de abandonar a vinha o utilizador decide delimitar os limites da mesma através do menu parcelas.

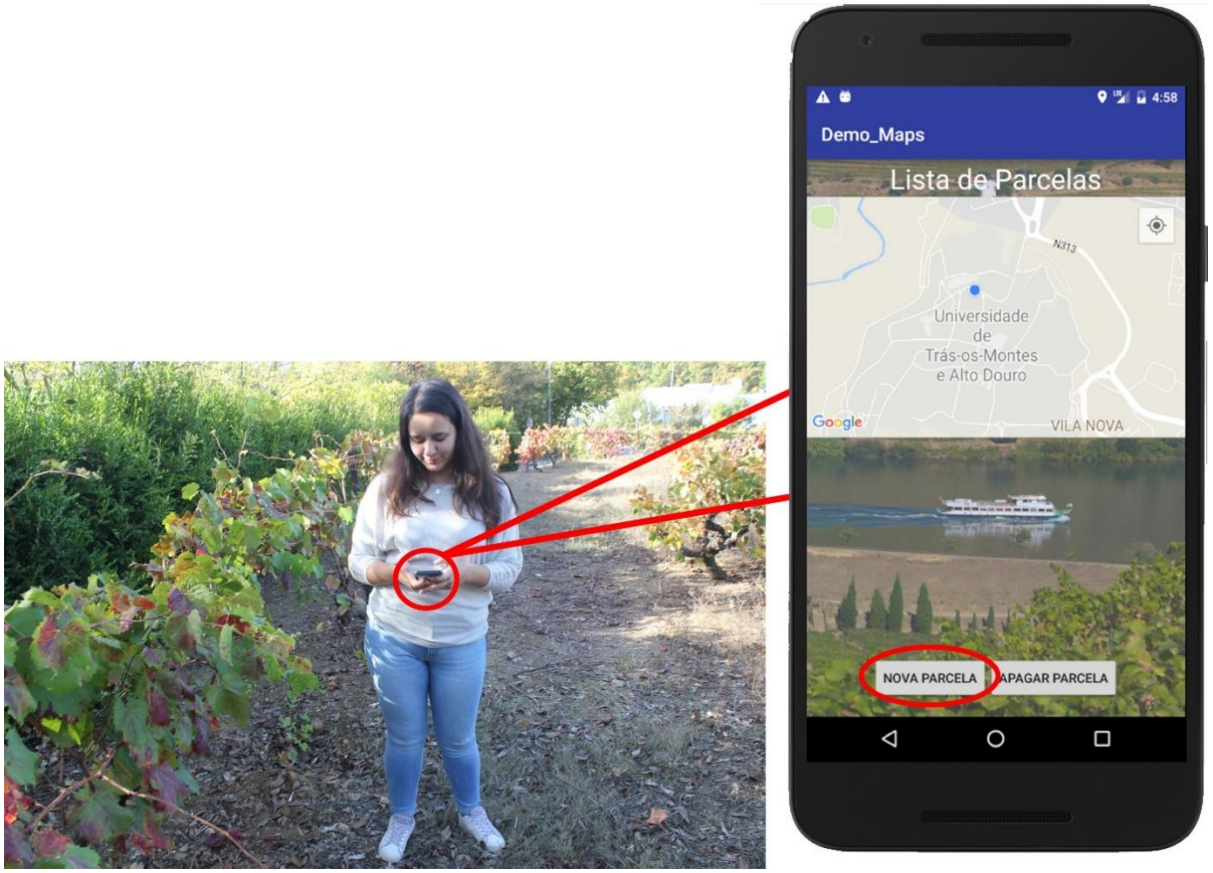

Figura 52 - Inserção de uma nova parcela.

Ao selecionar "Parcelas", no menu inicial, o utilizador é reencaminhado para a lista de parcelas da vinha da UTAD.

Sendo esta uma vinha com uma área pequena e com o mesmo tipo de uvas em toda a sua extensão o utilizador decide criar apenas uma parcela para delimitar toda a vinha. Para isso seleciona a opção "Nova Parcela", tal como se pode observar na Figura 52.

Depois de efetuar essa ação o utilizador é direcionado para um *layout*, no qual tem de adicionar as informações relativas à parcela que está prestes a adicionar e onde tem também de adicionar os pontos que delimitam a parcela que vai adicionar.

Para adicionar os pontos que delimitam a parcela o utilizador move-se ao longo dos limites da mesma e através das suas coordenadas (que são detetadas pelo GPS do seu *smartphone*) vai adicionando pontos relevantes para delimitar os limites da parcela, tal como se pode observar na Figura 53.

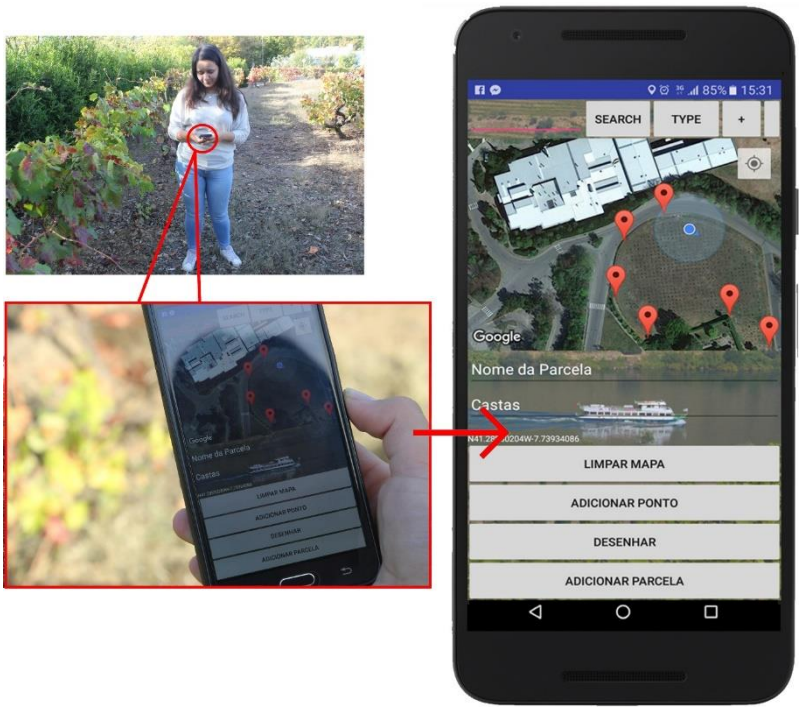

Figura 53 - Preenchimento das informações respeitantes à parcela.

Depois de ter os pontos suficientes para delimitar os limites da parcela clica sobre o botão "Desenhar" e obtém assim uma forma geométrica que delimita a parcela, tal como se pode observar na Figura 54.

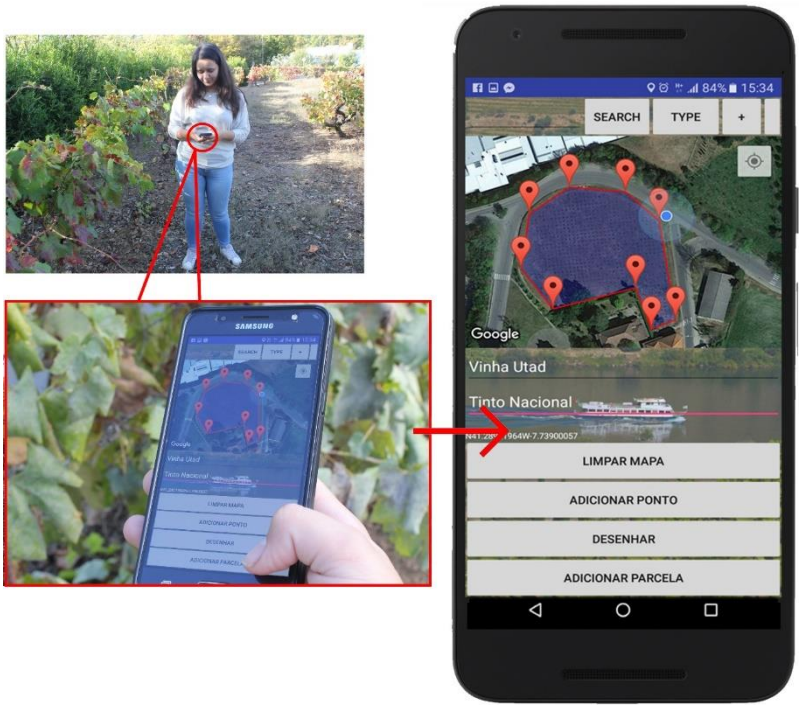

Figura 54 - Delimitação dos limites da parcela.

Depois de completar todas as informações relativas à parcela seleciona “Adicionar Parcela” e a sua parcela é adicionada à lista das parcelas da vinha UTAD, tal como se pode observar na Figura 55.

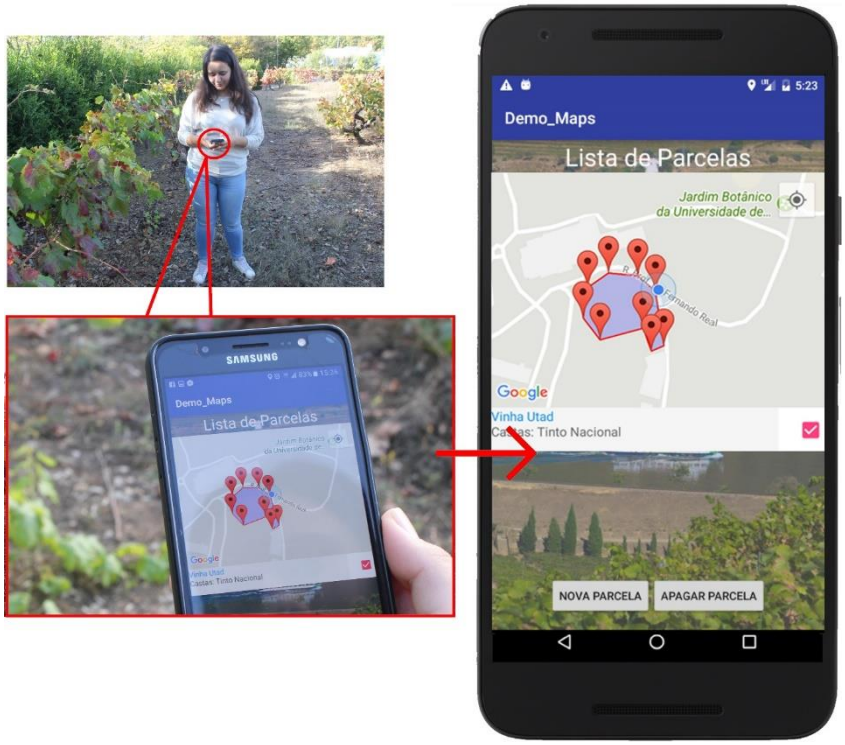

Figura 55 - Lista de parcelas.

5.2. Envio de Imagens para um Servidor

No desenvolvimento desta aplicação foi também testado o envio de imagens relativas às anomalias para um servidor local. Essa funcionalidade acabou por não ser implementada uma vez que só permitia o *upload* de uma imagem por anomalia, contudo apresenta-se aqui o teste dessa funcionalidade com vista à sua integração neste projeto em trabalho futuro.

Para o envio de imagens para o servidor foi utilizada a biblioteca Picasso do Android, uma vez que sem ela seria necessário escrever uma grande quantidade de código e lidar com o armazenamento das imagens em cache dentro da própria aplicação [40].

Tal como se pode observar na Figura 56, o utilizador pode captar uma imagem com a câmara do seu *smartphone* (como é feito na plataforma móvel para gestão de dados em vinhas) e de seguida fazer o *upload* da mesma para um servidor.

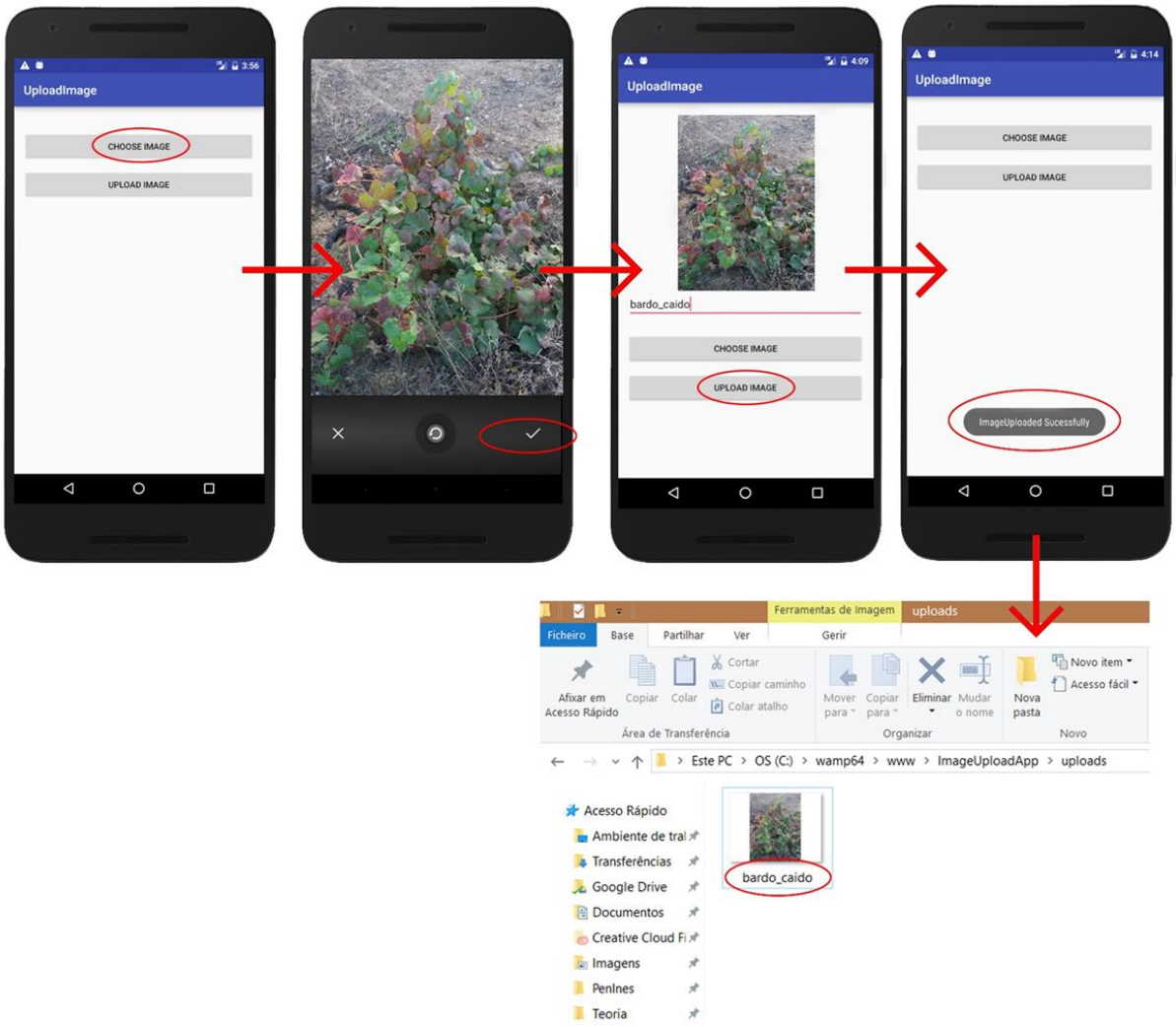

Figura 56 - Envio de imagem para um servidor.

O *upload* da imagem é efetuado com recurso a uma *script* PHP que faz a ligação entre a aplicação e o servidor.

Capítulo 6

6. Conclusões e Trabalho Futuro

Tal como esperado, os objetivos inicialmente pretendidos foram atingidos e conseguiu-se desenvolver o protótipo de uma aplicação capaz de gerir e monitorizar as atividades do dia-a-dia de uma vinha, através da utilização dos *smartphones*.

A plataforma móvel para gestão de dados em vinhas poderá ser uma mais valia para aqueles que todos os dias estão no terreno a efetuar trabalhos nas vinhas, assim como para os proprietários das mesmas, que através da sua utilização evitarão deslocações e poderão concentrar os recursos humanos disponíveis, nos locais aonde fazem mais falta, poupando tempo e consequentemente poupando dinheiro.

Esta aplicação permite registar anomalias, programar trabalhos e delimitar as parcelas existentes numa vinha, permite ainda que as vinhas sejam visualizadas num mapa (com imagens de satélite) e é capaz de detetar a localização do utilizador, para uma mais fácil identificação da vinha (e dos dados a analisar) e para a delimitação das parcelas.

Considera-se que a utilização desta aplicação virá colmatar o problema de falta de mão de obra na região do Alto Douro Vinhateiro uma vez que, através da sua utilização, os poucos recursos humanos existentes poderão mais facilmente identificar os problemas nas vinhas e poderão mais rapidamente dedicar-se à sua resolução.

A aplicação foi desenvolvida com um *design* simples para que o seu acesso seja intuitivo e para que os potenciais utilizadores da mesma possam interagir com ela facilmente.

Apesar de terem sido testadas soluções para o envio de dados para o servidor, nenhuma dessas soluções se mostrou suficientemente robusta para que pudesse ser implementada, como tal, essa funcionalidade será implementada como trabalho futuro, para que exista interligação entre a informação adicionada pelos diferentes utilizadores da aplicação. Pretende-se ainda, em trabalho futuro, aumentar a diferenciação entre os diferentes tipos de utilizadores (produtores de vinho, empreiteiros e trabalhadores) para que cada um possa ter um leque de tarefas associadas mais restritas e mais relacionadas com as suas qualificações.

Para complementar a utilidade da aplicação seria interessante que esta gerasse relatórios dos trabalhos realizados durante a campanha de produção (cartas de produtividade), para que o produtor de vinho pudesse obter um histórico de todas as tarefas e da data de realização das mesmas. A criação de uma lista de empreiteiros e trabalhadores, organizados de acordo com as suas competências e com o seu preço de trabalho, seria também uma mais valia para que o produtor de vinho pudesse mais facilmente encontrar o pessoal adequado para realizar determinada tarefa.

Todas estas são questões importantes, que poderão vir a acrescentar valor à plataforma de gestão de dados em vinhas e que a poderão tornar mais atrativa num futuro mercado de comercialização.

Referências Bibliográficas

[1] Android Developers. "*User Interface*". Obtido em 21 de dezembro de 2017, de Android Developers: https://developer.android.com/guide/topics/ui/layout/linear.html

[2] Android Developers. "*What is Android?*". Obtido em 08 de junho de 2017, de Android Developers: http://developer.android.com/guide/basics/what-is-android.html

[3] Araújo, R. B. (2003). Computação Ubíqua: Princípios, Tecnologias e Desafios. In *XXI Simpósio Brasileiro de Redes de Computadores*, pp.70-71.

[4] Arnó, J.; Martinez Casas Novas, J. A.; Ribes Dasi, M.; Rosell, J. (2009). "*Review. Precision viticulture.research topics, challenges and opportunities in site specific vineyard manegement*". Spanish Journal of Agricultural Research, v. 7, n. 4, p. 779-790

[5] Barbosa, J., Hahn, R., Rabello, S., Pinto, S. C. C., & Barbosa, D. N. F. (2007). *Computação Móvel e Ubíqua no Contexto de uma graduação de Referência.* Brazilian Journal of Computers in Education, 15(3).

[6] Blackmore, B. S. (1999). "*Developing the principles of precision farming. Agrotech 99"*. Barretos, Brazil, Barretos Insitute of Technology.

[7] Braga, R. (2009). "*Base de funcionamento e casos de estudo de VRT: Gestão intra-parcelar da densidade de plantas e taxa de aplicação de fertilizantes*". In Agricultura de Precisão. (J. P. Coelho e J. R. Silva). Inovação e Tecnologia na Formação Agrícola, AJAP, Lisboa, pp. 54 – 72.

[8] Braga, R. (2009). *Inovação e Tecnologia na formação agrícola: Viticultura de Precisão*. Obtido em 05 de junho de 2017, de Agrinov: http://agrinov.ajap.pt/diapositos/viticprecisao_final/Viticultura/Diapositivos_Viticultura_de_Precisao.pdf

[9] Bramley, R.; Lamb, D.; Proffittt, T.; Winter, E. (2006). "*Precision Viticulture: A new era in vineyard management and wine production*". Winetitles.

[10] Bramley, R.G.V. & Lamb, D.W. (2001). "*Making sense of vineyard variability in*

Austrália". Occasional report nº14. Fertilizer and lime research centre, Massey University, Palmerston Nort.

[11] Bramley, R.G.V. & Lamb, D.W. (2003). "*Making sense of vineyard variability in Australia*". In: Ortega, R.; Esser, A.. Precison Viticulture. Proceedings of an internacional symposium held as part of the IX Congreso Latinoamericano de Viticultura e Enologia, Chile. Centro de Agricultura de Precisión (CAPUC), Pontificia Universidad Católica de Chile.

[12] Bramley, R.G.V. (2000). "*Measuring within vineyard variability in yield and quality attributes*". Vineyard monitoring and management beyond. Wagga.

[13] Bramley, R.G.V. and Proffittt, A.P.B. (1999). "*Managing variability in viticultural production*". The Australian Grapegrower and Winemaker 427, 11-16.

[14] Burrell, J., Brooke, T. & Beckwith, R., 2004. "*Vineyard Computing: Sensor Networks in Agricultural Production*", IEEE Pervasive Computing, vol. 3, no., pp. 38-45, doi:10.1109/MPRV.2004.1269130

[15] C. Reis. "*Caracterização de um Modelo de Processo para Projetos de Software Livre*". 2001.

[16] Chaudhary, D.D., Nayse, S.P., Waghmare, L.M., 2011. *Application of Wireless Sensor Networks for Greenhouse Parameter Control In Precision Agriculture*. International Journal of Wireless & Mobile Networks. Vol. 3, No. 1, pp.144 - 145.

[17] Clark, R.L. e McGucken, R.L. (1996). "*Variable rate application technology: An overview*". Proceedings of the Third International Conference on Precision Agriculture. Minneapolis, MN, June 23-26, 1996. Robert, P. C., Rust, R. H. e Larson, W. E. (Eds.). ASA Miscellaneous Publications, ASA, CSSA, e SSSA, Madison, WI, 651-662.

[18] Coelho, J. C., Silva, J. M. (2009). "*1ª Edição Agricultura de Precisão"*. Associação de Jovens Agricultores de Portugal, Lisboa

[19] Cook, S. & Bramley, R. (1998). "*Precision agriculture: opportunities, benefits and pitfalls*". Australian Journal of Experimental Agriculture, 38, 753-763.

[20] Cook, S.E. & Bramley, R.G.V. (2001). "*Is agronomy being left behind by precision agriculture?"* In Australian Agronomy Conference, 10.

[21] Ehringer, D. (2010). *The dalvik virtual machine architecture*. Techn. report (March 2010), 4, 8.

[22] Ferreira, A. A. J., & Gonçalves, W. O. (2014). "*Rastreamento veicular com auxílio de dispositivos móveis"*.

[23] Google maps. Disponível em: <https://developers.google.com/maps/licensing?hl=pt-r>. Acesso em 28/07/2014.

[24] Gronli, T., Hansen, J., Ghinea, G. & Younas, M. "*Mobile application platform heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS*" in IEEE 28th International Conference on Advanced Information Networking and Applications, 2014. doi: 10.1109/AINA.2014.78

[25] I. Sommerville, e P. Sawyer. *"Requirements Engineering - A good practice guide*" - Wiley 1997.

[26] I. Sommerville. "*Engenharia de Software, 9. Edição*", São Paulo, Pearson Prentice Hall, 2011.

[27] J. Highsmith, and K. Cockburn. "*Extreme Programming*". E-Business Application Delivery, Feb., pp 4-17, 2000.

[28] K. Gerald e I. "*Somerville, Requirements Engineering: Processes and Techniques*", Wiley, 1998.

[29] K. Schwaber, and M. Beedle. "*Agile Software Development with Scrum*", NJ, Prentice-Hall, 2002.

[30] Krasner, G. & Pope, S. (1998). A cookbook for using the model view controller user interface paradigm in smalltalk-80. In *Journal of Object-Orientated Programming*, volume 1(3), pp. 26–49.

[31] Lee, W. S., Burks, T. F., & Schueller, J. K. (2002). *Silage yield monitoring system*. In

2002 ASAE Annual Meeting (p. 1). American Society of Agricultural and Biological Engineers.

[32] Marquez, L. (2009). *Presente y futuro de la Agricultura de Precisión*. Obtido em 05 de junho de 2017, de Vinificatum: http://vinificatum.blogspot.pt/2012/06/presente-y-futuro-de-la-viticultura-de.html

[33] Martínez-Casasnova, J.A., Bordes, X. (2005). "*Viticultura de precisión: predicción de cosecha a partir de variables del cultivo e índices de vegetación*". Revista de la Asociación Española de Teledetección 24, 67-71.

[34] Massitela, I. I. R. (2016). "*Aplicação do formalismo SAN para avaliação de desempenho de uma equipe de desenvolvimento de software baseada no modelo Waterfall"* (Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul).

[35] Mulla, D.J. & Schepers, J.S. (1997). "*Key processes and properties for site-specific soil and crop management*". In: PIERCE, F.J., SADLER, E.J. (Eds.). The state of site-specific 93 management for agriculture. Madison: American Society of Agronomy, Crop Science Society of America, Soil Science Society of America, p. 1-18. Citado por Bernardi, A. et al.,, 2004

[36] Open Street Map. *Open street map*. [S.l.], 2012. Disponível em: <http://www.openstreetmap.org/>. Acesso em:02 jul. 2012.

[37] Ortega, R., & Esser, A. (2002). "*Viticultura de Precisión: Fundamentos, aplicaciones y oportunidades en Chile*". Pontificia Universidad Católica de Chile; 1-10

[38] Robert, P.C. (1993). "*Characterization of soil conditions at the field level for soil specific management*". Geoderma, Amsterdam, v. 60, n. 1, pp. 57-72. Citado por Bernardi, et al., 2004

[39] Searcy, S. W. (1997). *Precision Farming: "A New Approach to Crop Management*". Texas Agricultural Extension Service, The Texas A&M University System, College Station, TX

[40] Simplified Coding. "*Picasso Android Tutorial – Android Picasso Image Loader Library*". Obtido em 08 de novembro de 2017, de Simplified Coding: https://www.simplifiedcoding.net/picasso-android-tutorial-picasso-image-loader-library.

[41] Thenkabail, P. S. (2003). "*Biophysical and yield information for precision farming from nearreal-time and historical Landsat TM images*". International Journal of Remote Sensing, pp. 24:14, 2879-2904.

[42] Tracy, K. W. (2012). *Mobile application development experiences on Apple's iOS and Android OS*. Ieee Potentials, 31(4), 30-34.

[43] Utad. *Região Demarcada do Douro.* Obtido em 02 de junho de 2017, de Utad: http://home.utad.pt/~rfvr/reg_dem_douro.html

[44] Wample R.L., Mills L., Davenport J.R. (1999). "*Use of precision farming practices in grape production. Proc IV International Conference on Precision Agriculture*". St Paul, MN, USA, July 19-22. pp. 897-905.

[45] Windows Team. "*Market Share do Windows Phone em queda e iOS avançou sobre o Android*". Obtido em 26 de dezembro de 2017, de Windows Team: https://www.windowsteam.com.br/market-share-do-windows-phone-em-queda-e-ios-avancou-sobre-o-android/

[46] Y. Rogers, H. Sharp, J. Preece, "*Interaction Design - Beyond human-computer interaction".* Wiley , 2002.