IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE ESTÁGIO

Licenciatura em Engenharia Informática

Daniel Monteiro Maia Pereira

julho | 2020

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE ESTÁGIO

ReportAvarias EDP Distribuição

DANIEL MONTEIRO MAIA PEREIRA

RELATÓRIO PARA A OBTENÇÃO DA LICENCIATURA

EM ENGENHARIA INFORMÁTICA

Julho 2020

Ficha de Identificação

Aluno

Nome: Daniel Monteiro Maia Pereira

Número: 1012117

Curso: Engenharia Informática

Estabelecimento de ensino

Escola Superior de Tecnologia e Gestão – Instituto Politécnico da Guarda

Morada: Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Telefone: 271220120 | **Fax:** 271220150

Duração do projeto

Inicio: 9 de setembro 2019

Fim: 4 de novembro 2019

Orientador do projeto

Nome: Celestino Gonçalves

Supervisor do projeto

Nome: Filipe Nogueira

Agradecimentos

Apesar de a conclusão do meu curso superior não ter sido a mais fácil devido a situações externas, tenho de agradecer à minha família pelo apoio que sempre me deu, em especial aos meus pais e irmãos, por sempre acreditarem em mim. Essas situações contribuíram também para que eu ganhasse consciência do mundo do trabalho e de como por vezes as adversidades fortalecem o carácter de uma pessoa, o que faz com que uma pessoa atinja objetivos que antes considerava impossíveis, com um pouco de foco e dedicação.

Quero agradecer ao meu orientador de estágio, o professor Celestino Gonçalves, por acreditar nas minhas capacidades, e aos demais professores, por serem exigentes de maneira a que o meu conhecimento relativo à área de informática se desenvolvesse de uma maneira produtiva.

Deixo um agradecimento também ao Engenheiro Filipe Nogueira pela ajuda, acompanhamento e orientação disponibilizada no estágio realizado na EDP durante a realização do mesmo e da aplicação desenvolvida.

Um abraço também a todos os meus colegas de curso pelo apoio disponibilizado ao longo da licenciatura em Engenharia Informática, que me permitiu a conclusão da mesma com uma perspetiva da vida mais sorridente.

Resumo

Este relatório tem como objetivo apresentar o trabalho realizado por Daniel Monteiro Maia Pereira no contexto de estágio de licenciatura em Engenharia Informática do Instituto Politécnico da Guarda. Este descreve a aplicação ReportAvarias EDP Distribuição, realizado na empresa EDP (Energias de Portugal).

ReportAvarias EDP Distribuição é uma aplicação móvel para ser disponibilizada aos trabalhadores da empresa EDP, que permite o registo de avarias através de uma interface que fornece um mapa para identificação das mesmas e depois o seu registo numa base de dados online para ser possível a criação de relatórios automáticos. Foi implementado com o intuito de ser aplicada no distrito da Guarda, mas tem possibilidades de ser atualizada para abranger Portugal inteiro com pequenas atualizações da mesma. É uma aplicação que facilita a registo de avarias pois cria um relatório automático e regista as mesmas numa base de dados online que possibilita uma consulta futura que contém dados relativos à mesmas, para consultas posteriores.

Neste relatório estarão sumarizadas as diferentes fases do projeto da aplicação ReportAvarias EDP Distribuição, desde o seu planeamento, desenvolvimento até à fase de implementação, tendo em conta as metodologias utilizadas e o estado de arte que deu suporte para a sua realização.

Palavras Chave:

Aplicação Móvel, Bases de Dados, Sistema de Registos, Registo de Avarias

Abstract

This report aims to present the work done by Daniel Monteiro Maia Pereira in the context of an undergraduate degree in Computer Engineering at the Polytechnic Institute of Guarda. This describes the application ReportAvarias EDP Distribuição, carried out at the company EDP (Energias de Portugal).

ReportAvarias EDP Distribuição is a mobile application to be made available to employees of the EDP company, which allows the recording of malfunctions through an interface that provides a map for their identification and then their registration in an online database to be possible to create automatic reporting. It was implemented in order to be applied in the district of Guarda, but it has the possibility of being updated to cover the whole of Portugal with minor updates. It is an application that facilitates the recording of malfunctions as it creates an automatic report and records them in an online database that allows a future consultation that contains data related to them.

This report summarizes the different project phases of the ReportAvarias EDP Distribuição application from its planning, development to the implementation phase, taking into account the methodologies used and the state of the art that supported its realization.

Key words:

Mobile Application, Database, Records System, Malfunctions Records

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Acrónimos

API - Application Programming Interface

APK - Android application Package

CSS3 – Cascading Style Sheets, versão 3

EDP - Energias de Portugal

ESTG - Escola Superior de Tcnologia e Gestão

GPS - Global Positioning System

HTML5 - Hypertext Markup Language, versão 5

IDE - Integrated Development Environment

IPG - Instituto Politécnico da Guarda

iOS - iPhone Operating System

IPA - iOS App Store Package

SDK - Software Development Kit

UI - User Interface

UML - Unified Modeling Language

XML - Extensible Markup Language

1.Introdução

O relatório apresentado descreve o projeto de fim de curso em contexto de estágio desenvolvido no âmbito da unidade curricular de Projeto de Informática, unidade curricular do 3º ano da licenciatura de Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

1.1. Caracterização da empresa

A empresa Energias de Portugal (EDP) foi criada em 1976 através da fusão de 13 das 14 empresas nacionalizadas do sector elétrico já existente, então com o nome "Eletricidade de Portugal" [7]. O objetivo da empresa era ser uma empresa estatal encarregue da eletrificação de todo o país, a monitorização e extensão das redes de distribuição elétrica, do planeamento e construção do parque electroprodutor nacional, e do estabelecimento de um tarifário para todos os clientes. Durante três décadas, a empresa cresceu, conquistou mercados, alargou a atividade, expandiu negócios e contribuiu para mudar culturas. Inevitavelmente, a marca acompanhou todas estas mudanças. Este período de crescimento demarcou um caminho na internacionalização, que começou no Brasil e que, atualmente, se expandiu a 13 novos países[7].

A EDP é uma operadora de soluções energéticas que desenvolve a sua atividade nas áreas da eletricidade (produção, distribuição e comercialização) e do gás (comercialização e distribuição). É o maior grupo industrial português, um dos maiores operadores no setor da energia e o terceiro maior produtor de energia eólica do mundo[7]. O grupo EDP tem como objetivo ser uma empresa global de energia, líder em criação de valor, inovação e sustentabilidade, apresentando como principais valores: a iniciativa, a confiança, a excelência, a sustentabilidade e a inovação[7].

1.2. Motivação

A escolha deste projeto em contexto de estágio proporcionou-se devido ao facto de que à escala mundial, a utilização de *smartphones* cresceu exponencialmente e, consequentemente, os seus utilizadores utilizam os seus aparelhos para consultar e enviar informação de uma forma mais simples e rápida, pelo que a criação de aplicações móveis para o uso quotidiano teve também um grande crescimento.

1.3. Objetivos do estágio

A empresa EDP propôs-me a criação de uma aplicação móvel com o intuito de registar diferentes tipos de avarias, através de um formulário e seleção de coordenadas geográficas. Torna-se mais fácil a criação dos relatórios automáticos, uma vez que alguns dos dados serão introduzidos automaticamente. É então necessário:

1. **Fazer o levantamento das ferramentas e tecnologias a utilizar**

   Uma vez que o projeto se desenrola num contexto nunca antes lecionado durante o meu percurso académico, é preciso realizar uma pesquisa das ferramentas que mais se adequam ao projeto, tanto para o gestor de conteúdos, como para a aplicação móvel. As tecnologias utilizadas por estas ferramentas são linguagens de programação que foram lecionadas durante a licenciatura, porém o estudo da sua aplicação será necessário.

2. **Conceção e avaliação dos requisitos do projeto**

   Após o levantamento e adaptação das ferramentas a utilizar, o próximo objetivo passa por fazer juntamente com o supervisor da EDP o levantamento dos requisitos funcionais. Esta etapa será a de maior importância, pois a partir deste momento será delineado tudo aquilo que será desenvolvido.

3. **Desenvolvimento da aplicação**

É neste objetivo que se passa grande parte do tempo de estágio, pois consiste no desenvolvimento da aplicação com base nos requisitos levantados no ponto anterior. Segue-se a metodologia Ágil dividindo, em ciclos, o desenvolvimento da aplicação, e o acompanhamento semanal por parte do supervisor da empresa.

4. **Testes**

Pretende-se que sejam feitos testes durante o desenvolvimento da aplicação, para garantir que a implementação está de acordo com os requisitos. Espera-se que na etapa final do estágio, toda a aplicação seja testada por elemento imparcial, para garantir a fiabilidade da aplicação.

1.4. Estrutura do relatório

Em termos de estrutura, este relatório é composto por 6 capítulos. Em cada um destes capítulos vai ser descrito o progresso do projeto da seguinte forma:

- A caracterização da empresa e a motivação na escolha deste projeto;
- Abordagem ao problema, onde serão estudadas as possíveis tecnologias e a sua escolha para este projeto;
- A metodologia que irá ser utilizada e a modelagem para a produção de software e a descrição das tecnologias envolvidas;
- Descrição do processo de desenvolvimento do projeto;
- A validação de ambas as partes do projeto e testes elaborados ao mesmo;
- A conclusão, onde será feita uma análise ao projeto desenvolvido e à importância do estágio.

2. Estado da Arte

Uma vez levantados os requisitos da aplicação a desenvolver é necessário a escolha de ferramentas que permitam a sua construção, com isto é necessária a recolha de informação de ferramentas pertinentes que permitam o desenvolvimento da aplicação.

2.1. Software para desenvolvimento de aplicações móveis

A escolha da plataforma móvel para a qual se pretende desenvolver determinada aplicação é também uma questão que gera muitas dúvidas no início de muitos projetos existindo assim, por vezes, alguma dificuldade na tomada de decisões nesta matéria. Num cenário ideal, a aplicação seria desenvolvida para todas as plataformas de forma a atingir o maior número de utilizadores possível, o que quase nunca é possível.

Como a maior parte dos colaboradores da empresa usam dispositivos que recorrem à tecnologia Android, decidi optar por um software que desenvolvesse apps adaptadas a esses mesmo dispositivos com a possibilidades de posteriormente ser adaptada para iOS.

Android Studio

O Android Studio[1] é uma plataforma de desenvolvimento para Android, criada pela Google. A plataforma é baseada no IDE IntelliJ IDEA da Jetbrains e é semelhante ao popular Eclipse, com ADT Plugin, ou ao Netbeans, oferecendo as melhores ferramentas e funcionalidades aos programadores. Segundo a própria Google, com o Android Studio a programação para Android é mais simples e rápida. A partir da versão 3.0, a Google focou-se essencialmente em oferecer uma ferramenta que permitisse o desenvolvimento rápido de aplicações.

Este IDE tem tudo o que um programador precisa para programar uma app, incluindo um editor de código bastante completo, ferramentas de análise de código, emuladores, etc.

Xamarin

Com Xamarin[11] é possível desenvolver apps móveis nativas utilizando a linguagem de programação C# (ou F#). Porém, além de simplesmente poder escrever código utilizando C#, é possível utilizar recursos do C# e do .NET no desenvolvimento destes aplicativos, atributos como *async/await*(código assíncrono), entre outros.

O desenvolvimento de apps mobiles na plataforma Xamarin é feito utilizando: Xamarin.Forms, Xamarin.iOS ou Xamarin.Android.

Xamarin. Android é o conjunto de tecnologias que nos permite desenvolver para Android utilizando C#.

Apache Cordova

O Apche Cordova[10] é uma *framework* de código aberto para o desenvolvimento móvel , que permite a utilização de tecnologias *web* padrão como o HTML5, CSS3 e JavaScript para desenvolvimento multiplataforma, que se executam em código encapsulado (*wrappers*) específicos para cada plataforma (Android, iOS, Windows Phone) utilizando APIs para ter acesso às capacidades nativas de cada sistema operativo móvel (camara, sistema de ficheiros, GPS, etc.).

A sua utilização passa pela utilização de um IDE como o Visual Studio que possui as bibliotecas para esta *framework* facilitando a inclusão das mesmas.

Escolha do software

Após uma breve análise dos diferentes tipos de soluções apresentadas por cada um dos softwares apresentados nos pontos anteriores, decidi optar pelo Android Studio, devido à minha maior familiaridade com o software e por haver um maior conjunto de informação online que permite o desenvolvimento do mesmo. Será mais fácil a implementação dos requisitos pedidos pela empresa pois é uma plataforma que é bastante intuitiva e fácil de modificar conforme os eventuais pedidos que sejam feitos pelo supervisor do projeto de maneira a que as respetivas alterações sejam efetuadas.

A plataforma Android Studio fornece diferentes tipos de *frameworks* com diferentes aspetos visuais e tipos de funcionalidades, mas a utilizada será apenas uma em branco onde se terá que desenvolver todo o aspeto e funcionalidades da aplicação, uma vez que não é fornecido qualquer tipo de projeto pré feito onde o programador se pode basear de forma a tornar a aplicação mais interativa e com uma estética mais agradável.

3. Metodologias e Modelação

Neste capítulo irá ser feita a caracterização da metodologia denominada Ágil, e delinear as etapas para a produção do projeto, tendo como valores fundamentais a iteração entre os intervenientes, as melhores ferramentas para realizar a produção, a colaboração do cliente e a resposta às mudanças de plano.

3.1. Metodologias aplicadas

A metodologia Ágil foi escolhida devido ao seu princípio iterativo em que todos os intervenientes do projeto têm um papel fundamental no mesmo.

Em todas as iterações do desenvolvimento, todos os intervenientes se reúnem para fazer a planificação de tarefas, neste caso como a equipa é apenas composta pelo aluno e supervisor, a comunicação é mais recorrente criando um melhor canal de comunicação e compreensão. Torna-se mais fácil então atingir o objetivo pretendido.

Esta metodologia é também importante devido ao facto de saber a evolução do projeto e também indicar novas metas para a próxima fase.

3.2. Desenho e modelação

A aplicação vai utilizar uma base de dados online (Firebase) para guardar os dados inseridos através do utilizador usando a aplicação móvel. Como é apenas um módulo para ser integrado noutra aplicação não foi necessário a criação de permissões para os utilizadores.

Funções do utilizador:

- Registar Ocorrência
- Visualizar lista de ocorrências
- Gerar PDF com total de ocorrências
- Apagar ocorrência

3.2.1. Diagrama de Contexto

O diagrama de contexto (Figura 1) serve para identificar a interação que os atores têm com o sistema, onde os dados vão ser inseridos para se poder obter a lista de casos de uso correspondentes a cada ator.

Figura 1 Diagrama de Contexto

3.2.2. Diagrama de Casos de Uso

O diagrama de casos de uso (Figura 2) é composto por quatro elementos: o cenário, o ator, o caso de uso e a comunicação[12]. Para a aplicação que desenvolvi vamos apenas considerar um único ator que pode executar qualquer um dos casos de uso definidos no cenário, já que não existem restrições de utilizador.

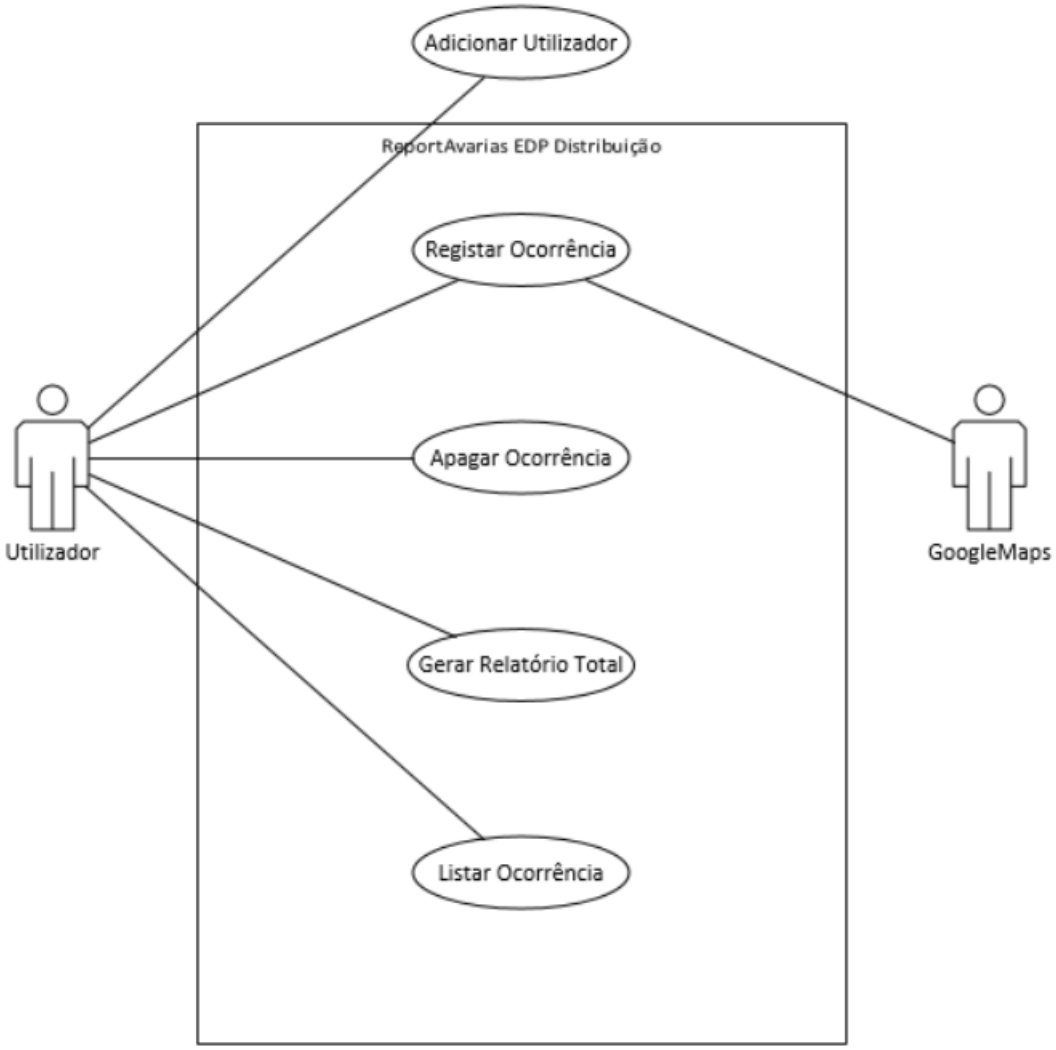

Figura 2 Diagrama de Casos de Uso

3.2.3. Descrição de casos de uso e diagramas de sequência

A descrição dos casos de uso é também importante porque permite definir as interfaces necessárias para o utilizador interagir com a aplicação.

Os quadros seguintes, que descrevem os casos de uso referidos no ponto anterior, estão divididos em sete secções: nome, objetivo, atores envolvidos, pré-condição, fluxo principal, fluxo alternativo e fluxo de exceção pós-condição.

3.2.3.1. Registar Ocorrência

Através deste caso de uso (Tabela 1) sabemos já qual o tipo de interfaces que devemos utilizar e os elementos pertencentes a cada uma delas, tal como botões para proceder para as diferentes interfaces (*activities*), um mapa para registo do marcador, caixas de texto para o formulário, entre outros.

Tabela 1 Registar Ocorrência

Nome	Registar Ocorrência
Objetivo	Registar uma nova ocorrência na base de dados
Atores envolvidos	Utilizador
Pré-condição	Ligação à internet
Fluxo principal	**1.** O ator seleciona a opção "Registar Ocorrência" na interface. **2.** O sistema apresenta um mapa com a localização atual do ator. **3.** O ator insere um marcador no mapa do local da ocorrência e pressiona o botão adicionar. **4.** O sistema apresenta uma nova interface para inserir dados relativos à ocorrência. **5.**O utilizador preenche os campos "tipo", "conc", "descricao, adiciona fotos e clica em "REGISTO +" **6.**O sistema guarda a ocorrência na base de dados.
Fluxo alternativo	**3a)** O ator só pode selecionar um marcador. **5a)** O ator tem um máximo de 3 fotos que pode adicionar.
Pós-condição	O sistema cria uma mensagem que informa que a ocorrência foi adicionada (*Toast*).

De notar também, que no Registo de Ocorrência é possível a criação de um relatório individual da mesma em PDF e o envio automático de e-mail com os dados da mesma. O diagrama de sequência abaixo representado (Figura 3) resulta da descrição do caso de uso.

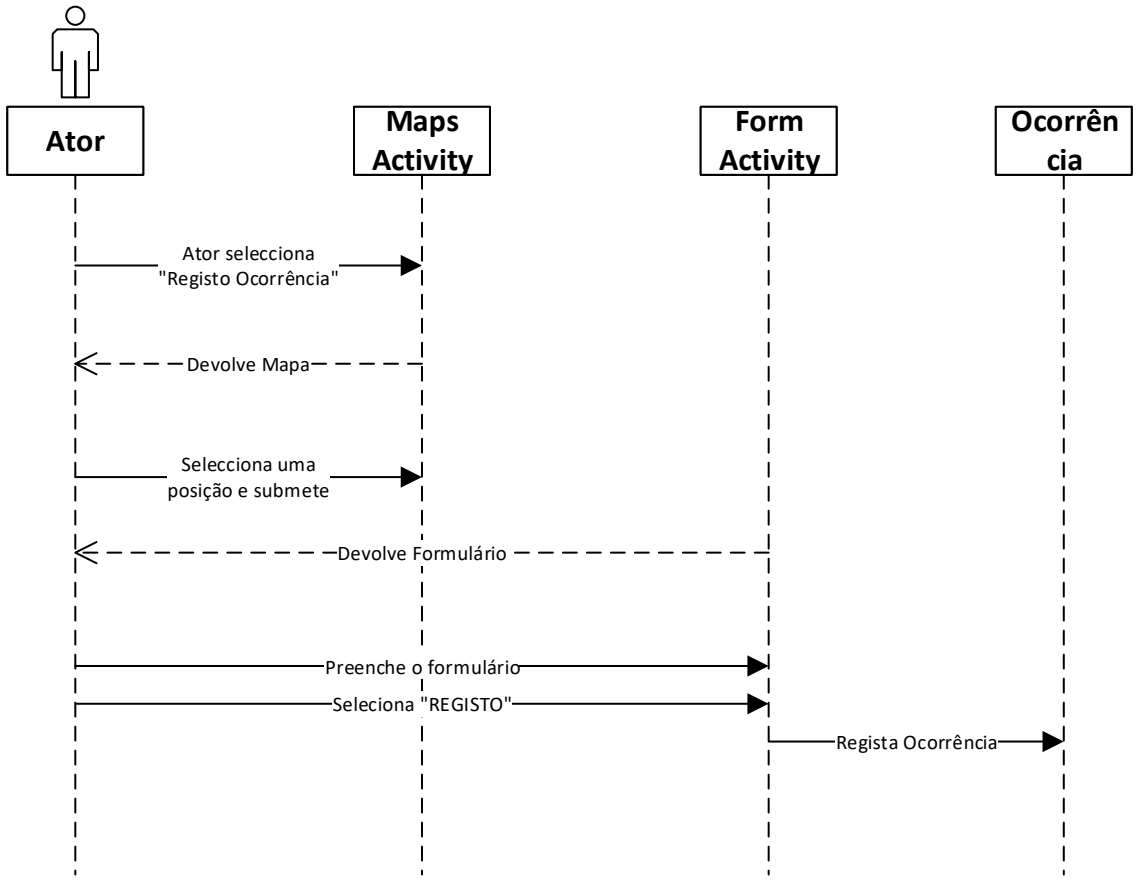

Figura 3 Diagrama de Sequência "Registar Ocorrência"

Interfaces presentes no cenário:

MapsActivity- Apresenta um mapa do qual se retiram valores para serem usados no FormActivity

FormActivity-Usa esses valores para uma implementação no preenchimento do formulário

3.2.3.2. Listar Ocorrência

Neste caso de uso (Tabela 2), vamos obter a listagem das ocorrências já adicionadas à base de dados da Firebase. Vão ser mostradas através de uma *RecyclerView* com a possibilidade de serem ordenadas por data, por tipo ou por concelho de incidência. Podem ser ainda vistas individualmente premindo a *CardView* de cada uma.

Tabela 2 Listar Ocorrência

Nome	Listar Ocorrências
Objetivo	Lista os dados relativos a uma ocorrência inserida na base de dados
Atores envolvidos	Utilizador
Pré-condição	Ligação à internet
Fluxo principal	**1.** O ator seleciona a opção "Lista Ocorrência" na interface. **2.** O sistema apresenta a interface onde existe uma *RecyclerView* com todas as ocorrências registadas na base de dados. **3.** O ator seleciona qual a ocorrência que quer ver. **4.** O sistema apresenta a interface com os dados correspondentes da ocorrência.
Fluxo alternativo	**3a)** Não existem ocorrências adicionadas
Pós-condição	Não tem

O diagrama de sequência abaixo representado (Figura 4) resulta da descrição do caso de uso.

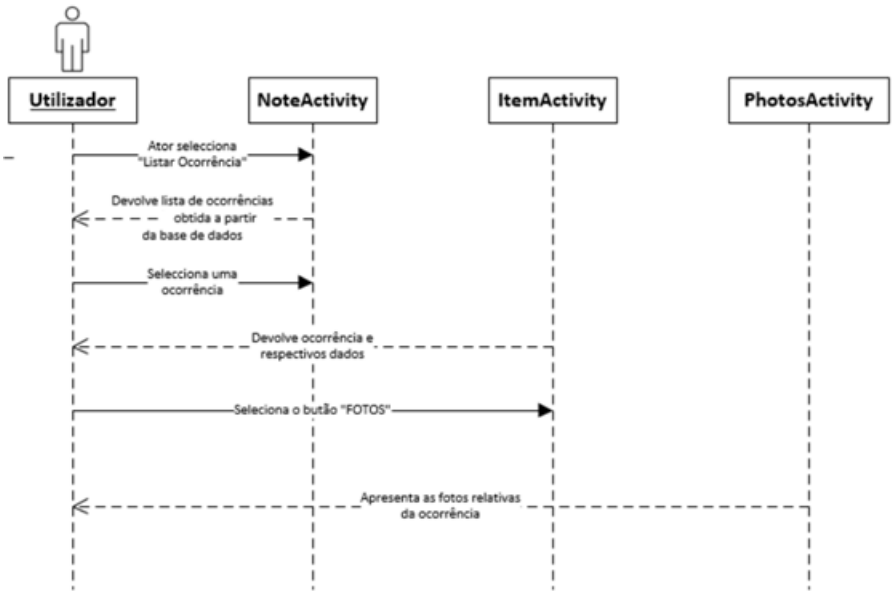

Figura 4 Diagrama de Sequência "Listar Ocorrência"

Interfaces presentes no cenário:

NoteActivity- Apresenta a lista obtida a partir da consulta de uma base de dados

ItemActivity-Apresenta um único item da lista com os valores detalhados

PhotoActivity-Apresenta as fotos pertencentes á Item Activity seleccionada

3.2.3.3. Gerar Relatório Total

Este caso de uso (Tabela 3) permite gerar um PDF com o total de ocorrências menos detalhado que o individual, mas mais focado para identificar o tipo de ocorrências de diferentes tipos que existem por concelho.

Tabela 3 Gerar Relatório Total

Nome	Gerar Relatório Total
Objetivo	Gera um PDF com os dados de todas as ocorrências
Atores envolvidos	Utilizador
Pré-condição	Ligação à internet
Fluxo principal	**1.** O ator seleciona a opção "Lista Ocorrência" na interface. **2.** O sistema apresenta uma interface. **3.** O ator seleciona a opção no topo da página "+" **4.** O sistema recolhe os valores relativos a todas as ocorrências. **5.** O ator seleciona a opção "💾" **6.** O sistema gera um relatório os valores relativos das ocorrências que são guardadas no dispositivo móvel.
Fluxo alternativo	**6a)** Não existem ocorrências registadas na base de dados
Pós-condição	O sistema informa que o PDF foi criado através de um Toast

O diagrama de sequência abaixo representado (Figura 5) resulta da descrição do caso de uso. Neste caso o utilizador seleciona a opção Listar Ocorrência que vai mostrar a lista completa de ocorrência registadas na base de dados, vai premir o botão "+" para fazer uma leitura do total de ocorrências guardadas na base de dados e depois preme o botão "💾" para gerar o relatório com essa informação total.

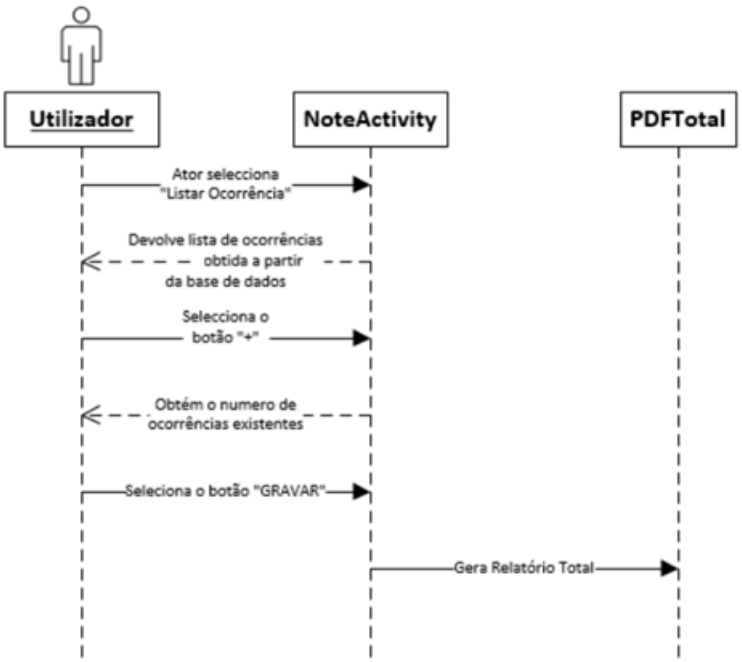

Figura 5 Diagrama de Sequência "Gerar Relatório Total"

3.2.3.4. Apagar Ocorrência

Este caso de uso (Tabela 4) corresponde ao método para apagar ocorrência da base de dados que já não sejam uteis estarem guardadas na base de dados por já terem sido resolvidas e não ser necessário que se encontrem online já que o PDF da mesma com os dados foi criado para consultas posteriores não sendo necessário um registo on-line.

Tabela 4 Apagar Ocorrência

Nome	Apagar Ocorrência
Objetivo	Apaga a ocorrência da base de dados
Atores envolvidos	Utilizador
Pré-condição	Ligação à internet
Fluxo principal	**1.** O ator seleciona a opção "Lista Ocorrência" na interface. **2.** O sistema apresenta a interface onde existe uma *RecyclerView* com todas as ocorrências registadas na base de dados. **3.** O ator excuta o movimento de deslizar para qualquer um dos lados premindo a ocorrência pretendida. **4.** O sistema apaga a ocorrência da base de dados.
Fluxo alternativo	**2a) N**ão existem registos na base de dados
Pós-condição	O sistema informa que o registo foi apagado através de um Toast

O diagrama de sequência abaixo representado (Figura 6) resulta do caso de uso.

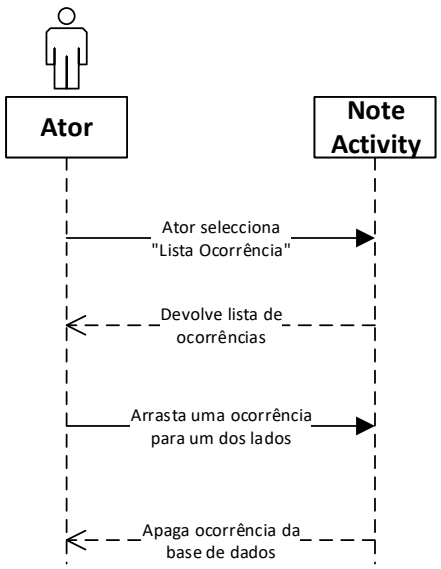

Figura 6 Diagrama de Sequência "Apagar Ocorrência"

3.2.4. Diagrama de classes

O objetivo dos pontos anteriores serviu para identificar as interfaces e classes necessárias. Após a identificação das classes, foi definida a criação de uma classe: a classe "DadosOcorrência". A referida classe contém os atributos necessários para a manipulação de dados e para a criação do modelo ER (Figura 7).

Figura 7 Diagrama de Classes

3.2.5. Modelo ER

O modelo entidade relacionamento, criado a partir do diagrama de classes, surge com o sentido de clarificar o relacionamento entre classes, identificar chaves primárias e estrangeiras, os tipos de dados e os seus tamanhos. Como o Firebase se trata de uma plataforma que emprega uma base de dados *noSQL* não vai haver nenhum relacionamento entre as tabelas nem uma limitação do tamanho de dados que vão ser utilizados. Por isso, a figura acima representada, trata-se apenas de uma representação teórica da maneira como a informação pretende ser armazenada.

3.2.6. Dicionário de dados

Nesta secção vamos identificar os campos das duas classes apresentando o tipo de dados, valores válidos e restrições. Também se irão descrever os métodos disponíveis em cada uma das classes. A tabela 5 corresponde aos dados da classe DadosOcorrência e as operações representadas na tabela 6.

Tabela 5 Dicionário de dados da classe DadosOcorrência

Nome	Tipo de dados	Descrição	Restrições	Tamanho	Formato
conc	string	Concelho de Avaria	Não nulo	40	Até 40 caracteres
data	datetime	Data da avaria	Não nulo	data	dd/mm/yyyy hh:mm:ss
descricao	string	Descrição do material danificado	Não nulo	256	Até 256 caracteres
lati	float	Latitude da ocorrência	Pode ser nulo	Max	Máximo possível de caracteres
longi	float	Longitude da ocorrência	Pode ser nulo	Max	Máximo possível de caracteres
morada	string	Local da Avaria	Não nulo	100	Até 100 caracteres
tipo	string	Tipo de avaria	Não nulo	20	Até 20 caracteres
url	string	Foto do Mapa	Não nulo	100	Até 100 caracteres
Foto1	image	Foto 1 da avaria	Pode ser nulo.	-	Apenas uma imagem
Foto2	image	Foto 2 da avaria	Pode ser nulo.	-	Apenas uma imagem
Foto3	image	Foto 3 da avaria	Pode ser nulo.	-	Apenas uma imagem

Tabela 6 Operações da classe DadosOcorrência

Operações: Classe "DadosOcorrência"	
Inserir()	Conjunto de operações (set) que permitem inserir nos campos da classe.
Ver()	Conjunto de operações (get) que permitem ver os dados dos campos da classe.
Eliminar()	Conjunto de operações (get e set) que permitem remover dados dos campos da classe.

3.3. Ferramentas e tecnologias utilizadas

Neste capítulo serão detalhadas as tecnologias para o desenvolvimento do projeto, sendo elas distribuídas gratuitamente e utilizadas em diversos campos. Deste modo há documentação especifica de forma a ajudar no desenvolvimento.

3.3.1. Android Studio

O Android Studio [1] (Figura 8) é um ambiente de desenvolvimento integrado (IDE) para desenvolver para a plataforma Android. Foi anunciado em 16 de maio de 2013. Android Studio é disponibilizado gratuitamente sob a Licença Apache 2.0.

É baseado no software IntelliJIDEA de JetBrains, e foi feito especificamente para o desenvolvimento para Android. Está disponível para download em Windows, Mac OS X Linux, e substituiu Eclipse Android Development Tools (ADT) como a IDE primária do Google de desenvolvimento nativo para Android.

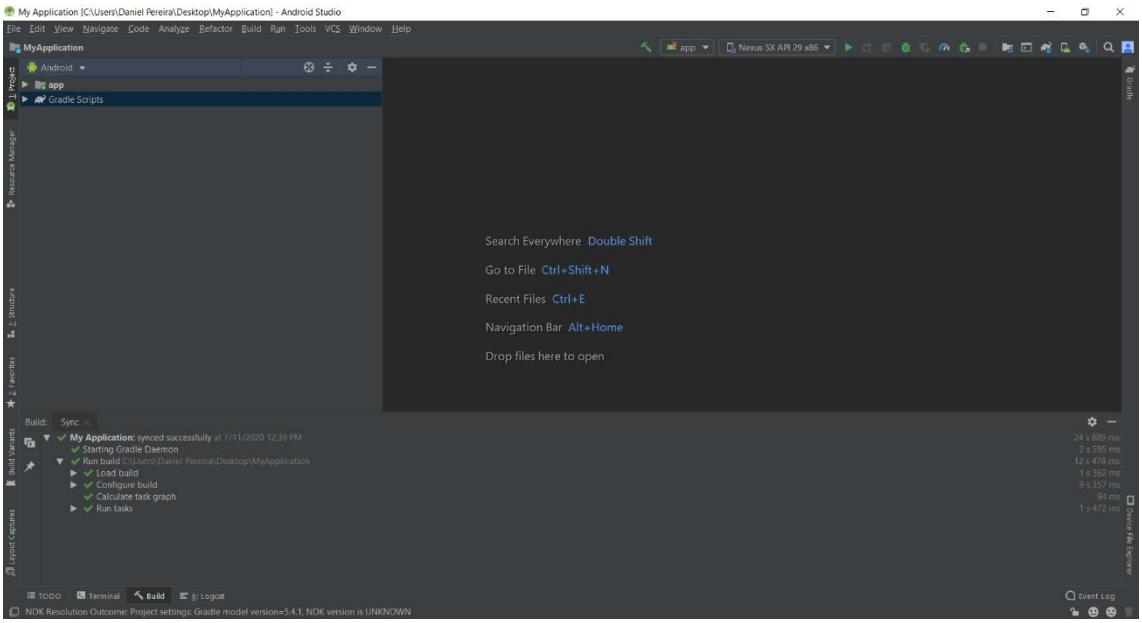

Figura 8 Android Studio

Características:

- Suporte para compilações baseadas em Gradle.
- Refatoração específica para Android e reparações rápidas.
- Ferramentas de Lint para capturar performance, usabilidade, compatibilidade de versão e outros problemas.
- Integração com Proguard e capacidade de assinatura de aplicativo.
- Um assistente baseado em predefinições com designs e componentes comuns de Android.
- Um editor de layout que permite que usuários arrastem componentes de interface de utilizador, opção de pré-visualizar em várias configurações de ecrã.
- Suporte para a criação de apps para Android Wear.
- Suporte nativo para a Google Cloud Platform, permitindo a integração com Google Cloud Messaging e App Engine.

3.3.2. Java

Java[8] é uma linguagem de programação orientada a objetos desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems. Em 2008 o Java foi adquirido pela empresa Oracle Corporation. Diferente das linguagens de programação modernas, que são compiladas para código nativo, a linguagem Java é compilada para um *bytecode* que é interpretado por uma máquina virtual (Java Virtual Machine, mais conhecida pela sua abreviação JVM).

3.3.3. XML

XML[3] significa Extensible Markup Language. XML é uma linguagem de marcação semelhante ao HTML usado para descrever dados. *Tags* XML não são predefinidas em XML. Devemos definir nossas próprias *tags*. O XML é bem legível tanto por humanos quanto por máquinas. Além disso, é escalável e simples de desenvolver. No Android, usamos o XML para projetar os nossos layouts, pois o XML é uma linguagem leve e, portanto, não torna o layout pesado.

3.3.4. Firebase

O Firebase [2] é uma plataforma de desenvolvimento de aplicações que fornece múltiplas ferramentas e serviços para permitir a criação de aplicações de alta qualidade. Os serviços da Firebase (Figura 9) podem ser divididos em duas partes: desenvolvimento e expansão. Como neste projeto se tratou do desenvolvimento de uma aplicação utilizei apenas a parte de desenvolvimento.

Figura 9 Serviços Firebase

Para armazenamento de dados utilizei o serviço Realtime Database. O Firebase Realtime Database trata-se de uma base de dados *noSQL* hospedada em nuvem que permite armazenar dados em tempo real. Um benefício da Realtime Database é que é fornecido com SDKs integrados que permitem a construção de aplicativos sem a necessidade de servidores.

Para o armazenamento das imagens utilizei o Firebase Storage. È uma solução para fazer upload de conteúdos(imagens, vídeos) para aparelhos iOs e Android, assim como a Web. O armazenamento usa um simples sistema de pasta/ficheiros para estruturar os dados.

4. Desenvolvimento da Aplicação

Nos pontos anteriores foram determinadas as ferramentas e tecnologias a utilizar para a elaboração do projeto proposto, de forma a que os intervenientes estejam de acordo com os procedimentos a efetuar.

Neste capítulo vai ser demonstrada a maneira como o projeto foi desenvolvido utilizando as ferramentas descritas.

4.1. Desenvolvimento dos Módulos Funcionais

Neste ponto vamos abordar o desenvolvimento do projeto tanto na sua vertente em Java como em XML. Vamos abordar pequenos exemplos de código utilizados na criação das diferentes *activities*. A classe *Activity* é um componente importante da aplicação para Android, sendo uma parte fundamental do funcionamento da mesma. Diferente das outras linguagens de programação em que as aplicações são lançadas através de um método *main*, no Android o código é iniciado numa instância *Activity* invocando métodos *callback* que correspondem a estágios específicos do ciclo de vida. O desenvolvimento das interfaces vai recair na maioria, sobre a programação em Java e em XML. De notar que para cada *Activity* em Java tem a sua parte correspondente em XML. Na parte do XML vamos identificar os elementos mais comuns que são empregues na realização do projeto.

4.1.1. Desenvolvimento em Java

Na parte de desenvolvimento em Java foi necessário a criação de múltiplas funções conforme os requisitos que eram necessários para o correto funcionamento da aplicação. Uma das partes mais importantes da programação em Java é o import das diferentes bibliotecas que vão ser posteriormente implementadas no código. A cada *activity* vai corresponder um conjunto de bibliotecas especificas.

WelcomeActivity

Esta *activity* consiste de um simples *gif* inicial que é carregado com um tempo de execução de 4000ms antes de ser encaminhada para a próxima *activity*.

Exemplo:

Foi necessário fazer o import da biblioteca *handler* para poder correr uma animação com um valor em milissegundos correspondente ao SPLASH_TIME_OUT antes de iniciar a nova *activity*.

```
import android.os.Handler;


new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        Intent mainIntent = new Intent(WelcomeActivity.this,
OptionsActivity.class);
        startActivity(mainIntent);
        finish();
    }
},SPLASH_TIME_OUT);
```

OptionsActivity

Esta *Activity* consiste de dois botões para serem selecionados de acordo com o objetivo do colaborador. “Registo Ocorrência” vai encaminhar para um formulário com diferentes campos e “Lista Ocorrência” corresponde à parte onde as ocorrências estão registadas.

MapsActivity

Activity para selecionar a posição da ocorrência num mapa através de um marcador que depois vai ser interpretado na *Activity* seguinte (FormActivity).

Exemplo:

É preciso a biblioteca *SupportMapFragment* para a criação do *fragment* onde o mapa vai ser mostrado, e assim que estiver pronto para ser utilizado vai ser notificado e inicializado na função `public void onMapReady(GoogleMap googleMap)`.

```
import com.google.android.gms.maps.SupportMapFragment;

SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
        .findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
```

FormActivity

Activity para adicionar dados específicos da ocorrência que vão ser guardados numa base de dados. Principais funções:

Principais Funções:

Estes são exemplos de bibliotecas que permitem a formatação de um ficheiro PDF através da aplicação.

```
import com.itextpdf.text.Document;
import com.itextpdf.text.Element;
import com.itextpdf.text.Font;
import com.itextpdf.text.PageSize;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.BaseColor;
```

Criação do documento e atribuição do seu tamanho:

```
Document document = new Document(PageSize.A4);
```

Criação de um parágrafo:

```
paragraph= new Paragraph();
```

Para se poder fazer o alinhamento relativo:

```
paragraph.setAlignment(Element.ALIGN_CENTER);
```

Formatação da *font* onde pode ser utilizado o BaseColor:

```
Font font = new Font();
```

NoteActivity

Nesta *Activity* vão estar representadas as ocorrências registadas na base de dados através de uma lista e diferentes funções para filtrar as que mais lhe interessam no momento. Possui um gerador de relatório PDF semelhante à FormActivity mas neste caso necessita de identificar o total de ocorrências no distrito separando-as por tipo e por concelho.

Exemplo:

Para a visualização da lista recorri ao *RecyclerView* com o qual podemos fazer uma escolha mais especifica utilizando um certo conjunto de parâmetros que vão retornar os valores pretendidos. Para isso usei a biblioteca abaixo identificada, que permite reconstruir o *RecyclerView* com as opções selecionadas.

```
import com.firebase.ui.firestore.FirestoreRecyclerOptions;

FirestoreRecyclerOptions<Note> options = new
FirestoreRecyclerOptions.Builder<Note>()
        .setQuery(query, Note.class)
        .build();
```

RegisterActivity

Esta *Activity* tem como objetivo mostrar um mapa com o total das atividades registadas na base de dados.

ItemActivity

É uma *Activity* que tem como única função mostrar os dados individuais de cada ocorrência.

PhotosActivity

Activity acedida através da ItemActivity que mostra as fotos relativas á ocorrência através de uma *RecyclerView*.

Em anexo, podemos encontrar funções mais detalhadas com o código utilizado na realização do estágio.

4.1.2. Desenvolvimento em XML

O XML que vou abordar nesta parte corresponde ao de layout. É o XML que é implementado em cada *activity*. Estes são usados para definir a UI da aplicação. Contêm todos os elementos(*views*) ou ferramentas que queremos usar na nossa aplicação. Existem diferentes tipos de layouts que podem ser implementados, tais como o *ConstraintLayout*, o Linear Layout, o *RelativeLayout*, entre outros. Decidi usar na aplicação a maior parte das vezes o *RelativeLayout* pois permite a flexibilidade de posicionar os nossos elementos relativos a outros de uma maneira certa permitindo que as diferenças entre diferentes tipos de ecrãs sejam impercetíveis. Principais elementos utilizados.

Fragments

Em Android os fragments fazem parte de uma Activity. É como se fossem um tipo de sub activity. Representam comportamentos ou parte de interface de uma atividade. É possível combinar vários numa Activity para se criar uma interface com múltiplos painéis e reutilizar o fragment em várias activities.

TexteView

O TextView mostra o texto para o utilizador e permite que este o consiga editar através de programação. É um editor de texto completo , no entanto, a classe básica está configurada para não permitir a edição, mas podemos editá-la.

EditText

Trata-se de uma sobreposição sobre o *TextView* que se configura para ser editável. É usado na aplicação para fornecer um campo de entrada que vai ser utilizado para descrever o tipo de ocorrência no formulário.

Button

Como o próprio nome indica trata-se de um botão. É premido para que ocorra uma acção, seja esta mudar de *activty* ou executar um procedimento. Existem vários tipos de botões mas decidi utilizar apenas o regular.

ImageView

É usada para exibir arquivos de imagem no aplicativo. Torna-se algo difícil de utilizar devido aos diferentes tamanhos de ecrã dos dispositivos com software Android. Existem vários *widgets* que permitem criar aplicativos mais atrativos. Exemplo de *Widgets*: Glide[9].

Spinner

O *Spinner* fornece uma maneira rápida de selecionar um valor de um conjunto de valores. Trata-se de uma lista suspensa. No estado padrão mostra o valor seccionado no momento. Implementei este elemento no formulário pois permite a poupança de tempo.

DatePicker

É usado para selecionar datas. Permite a escolha por dia, mês e ano. Na aplicação utilizei através de uma caixa de diálogo pelo que foi preciso usar uma classe *DatePickerDialog* para o efeito.

4.2. Aplicação da Firebase

Esta parte da metodologia corresponde à maneira como foi feito o upload e download dos dados através da plataforma de base de dados online Firebase. Vão ser apresentados tanto os procedimentos para dados como para as fotos.

Upload de arquivos

Primeiro é necessário criar uma referência ao destino do arquivo que se quer fazer upload que no exemplo do projeto que desenvolvi não passou de uma pequena linha de código com referência a uma instância associada ao nosso projeto:

```
CollectionReference reportRef =
FirebaseFirestore.getInstance().collection("Reports");
```

Depois é necessário fazer o upload concreto em que realmente se vão enviar os dados para a plataforma de registo on-line Firebase através da seguinte linha de código:

```
reportRef.add(new Note(data, descricao, morada, url, latit, longit, tip,
conc));
```

De notar que Note é uma classe criada com o intuito de se poder fazer o devido upload com a informação, por exemplo:

```
public class Note {
    private String data;
    private String descricao;
    private String morada;

    public Note() {
        //empty constructor needed
    }

    public Note(String data, String descricao, String morada, String url,
Double lati, Double longi, String tipo, String conc) {
        this.data = data;
        this.descricao= descricao;
        this.morada = morada;
    }

    public String getData() {
        return data;
    }

    public String getDescricao() {
        return descricao;
    }

    public String getMorada() {
        return morada;
    }
    }
```

Download de arquivos

Para fazer o download dos arquivos é necessário primeiro fazer uma *query* daquilo que pretendemos extrair da Firebase. Para isso criamos uma linha de código com a seguinte informação:

```
Query query = reportRef.orderBy("data", Query.Direction.DESCENDING);
```

A partir dessa linha vamos adicionar à *RecyclerView* os valores extraídos usando o seguinte código:

```
FirestoreRecyclerOptions<Note> options = new
FirestoreRecyclerOptions.Builder<Note>()
        .setQuery(query, Note.class)
        .build();

adapter = new NoteAdapter(options);
adapter.startListening();
RecyclerView recyclerView = findViewById(R.id.recycler_view);
recyclerView.setHasFixedSize(true);
recyclerView.setLayoutManager(new LinearLayoutManager(this));
recyclerView.setAdapter(adapter);
```

O *adapter* corresponde a uma *class* criada chamada NoteAdapter que contém funções para o uso na *RecyclerView* tais como:

onBindViewHolder – para adicionar a informação visível a cada *ViewHolder*.

onCreateViewHolder – para adicionar novos *ViewHolders* sempre que forem adicionados.

deleteItem – para apagar os que não são mais necessários.

4.3. Configurações iniciais

O ficheiro AndroidManifest.xml ajuda a declarar as permissões que uma aplicação necessita para aceder a dados de outras aplicações. Também identifica o nome do pacote da aplicação que facilita o AndroidSDK enquanto monta a aplicação. O ficheiro AndroidManifest.xml contém informação acerca das *activities*, serviços, entre outros. A aplicação tem de ter certas permissões para aceder a dados de outras apps. Certas permissões são adicionadas por *default* tais como ACESS_NETWORK_STATE e INTERNET. Outras é preciso serem adicionadas.

ACCESS_FINE_LOCATION - Permite que uma aplicação aceda a uma localização precisa.

ACCESS_NETWORK_STATE - Permite que uma aplicação aceda a informações sobre a rede.

INTERNET - Permite que uma aplicação abra sockets de rede.

READ_GSERVICES - Permite que uma aplicação modifique o mapa de serviço do Google.

ACCESS_COARSE_LOCATION - Permite que uma aplicação aceda a uma localização aproximada.

WRITE_EXTERNAL_STORAGE - Permite que uma aplicação escreva no armazenamento externo.

READ_EXTERNAL_STORAGE - Permite que uma aplicação leia no armazenamento externo.

MAPS_RECEIVE - Permite que uma aplicação aceda a uma localização precisa.

CAMERA - Permite que uma aplicação aceda à camera do dispositivo.

4.4. Configuração do Construtor Automático de Dependências

O Gradle [6] é um sistema de automatização de *builds*. Todos os projetos criados no Android Studio já vêm por padrão estruturados para usar o Gradle, sendo assim, temos um arquivo de configuração para o projeto principal e um para cada modulo. Os arquivos de configuração são chamados de *build.gradle*, e são arquivos de texto simples que usam a sintaxe da linguagem Groovy para configurar a compilação com os elementos fornecidos pelo plugin do Gradle para Android. Na maioria dos casos, é só necessário editar os arquivos de configuração de cada modulo.

No projeto realizado foi necessário a utilização de certas dependências externas estando definidas no código em baixo todas as que foram utilizadas na realização:

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.annotation:annotation:1.0.2'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.navigation:navigation-fragment:2.0.0'
    implementation 'androidx.navigation:navigation-ui:2.0.0'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.1.0'
    implementation 'com.google.firebase:firebase-firestore:17.1.2'
    implementation 'com.firebaseui:firebase-ui-firestore:4.1.0'
    implementation 'com.google.firebase:firebase-database:16.0.4'
    testImplementation 'junit:junit:4.12'
    implementation 'androidx.arch.core:core-runtime:1.1.1'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.google.android.gms:play-services-location:17.0.0'
    implementation 'com.polyak:icon-switch:1.0.0'
    implementation 'com.itextpdf:itextg:5.5.10'
    implementation 'pl.droidsonroids.gif:android-gif-drawable:1.1.17'
    implementation 'com.squareup.picasso:picasso:2.5.2'
    implementation 'com.zolad:zoominimageview:1.0.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'com.android.support:multidex:1.0.3'
    implementation 'com.google.firebase:firebase-storage:16.0.4'
    implementation 'com.google.firebase:firebase-auth:16.0.5'
    implementation 'com.firebaseui:firebase-ui-storage:4.3.1'
    implementation 'com.github.bumptech.glide:glide:3.5.2'
}
```

4.5. Interface Gráfica

A interface gráfica foi criada de maneira a ser intuitiva de ser utilizada com informação necessária para o utilizador. Vou demonstrar nesta parte exemplos das interfaces criadas usando a linguagem de programação XML utilizada para desenvolvimento de aplicações móveis com o sistema operativo Android. Vamos apresentar a MapsActivity (Figura 10) que tem como objetivo a seleção da posição da ocorrência, a FormActivity (Figura 11) para adicionar os dados a uma base de dados ou a criação do relatório em PDF, NoteActivity (Figura 12) para ver as ocorrências registadas em forma de lista, e a RegisterActivity (Figura 13) para ver todas as ocorrências que são identificadas com diferentes cores dependendo do tipo da avaria num GoogleMaps [4].

Figura 10 MapsActivity

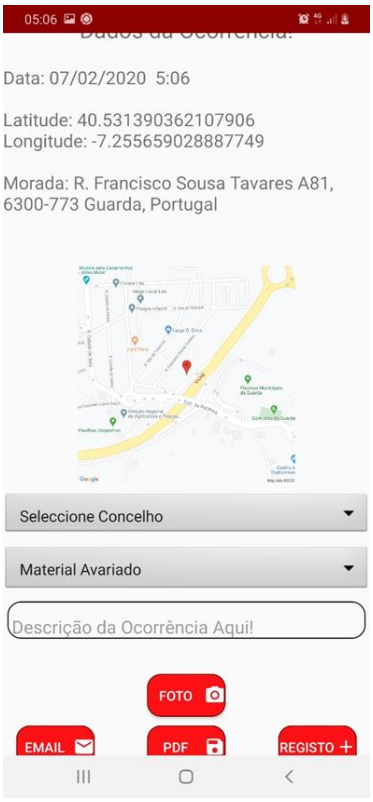

Figura 11 FormActivity

Figura 12 NoteActivity

Figura 13 RegisterActivity

5. Verificação e validação

Neste capítulo será descrito os testes efetuados à aplicação tendo em conta uma aplicação normal por parte do utilizador e da sua interface. Vamos analisar também o conteúdo da Firebase para verificar se os conteúdos foram guardados devidamente.

5.1. Validação das Interfaces da Aplicação

Como a aplicação se destinada ao trabalhador da EDP, esta foi testada de maneira a retratar a utilização normal de uma aplicação móvel a partir da sua instalação, e funcionamento dos seus requisitos principais.

A instalação nas plataformas móveis apenas foi feita definida para ser instalada em dispositivos com um *minSDKVerison 16* que corresponde à plataforma Android 4.2, mas corre com maior eficiência em dispositivos que têm a *SDKversion 29* que corresponde à plataforma Android 10. Devido à falta de material físico não foi possível a execução e adaptação do projeto para o sistema iOS, já que para compilar uma IPA é obrigatório a utilização de um computador Mac com sistema operativo macOS, o IDE XCode e uma conta de *Apple Developer*. Assim, não foi feita a compilação para iPhone por não se cumprir nenhum destes requisitos. Por causa disto a aplicação foi apenas testada em ambiente *Android* a partir da compilação de um ficheiro *Android Application Package* (APK). Após a instalação da aplicação num *smartphone* foi testada em termos de design e funcionalidades, onde toda a interface cumpriu os requisitos necessários nos testes realizados (Criar uma simulação antes-depois). A informação exibida correspondia aos dados registados sempre que um novo caso era adicionado. Houve certas funcionalidades que não ficaram tão otimizadas, tais como certas formatações de tamanhos para diferentes dispositivos ou definir restrições para os campos que eram preciso serem preenchidos, podendo serem inseridos dados sem qualquer limitação: trata-se apenas de uma interface que não se adapta a todas as resoluções de dispositivos.

5.2. Verificação da Firebase

Para a Firebase os testes efetuados foram baseados na sua utilização normal, para a verificação dos dados na base dados. A inserção de dados na interface da aplicação móvel tinha de gerar ficheiros correspondentes aos dados relativos ao caso registados e imagens das fotos que eram tiradas pelos funcionários. Em ambos os resultados foram satisfatórios, sendo os dados registados a partir do smartphone registados corretamente sempre que este se encontrasse ligado à rede. Foi necessário verificar o que ocorria aos dados sempre que estes eram apagados na interface, obtendo-se o resultado pretendido e apagando também da base de dados estes mesmos. Com as imagens o resultado foi o mesmo.

Todos os aspetos da aplicação foram verificados e validados pelo supervisor da EDP, que sempre que tinha dúvidas acerca da aplicação e fazia pedidos extras para a aplicação, estes eram corretamente explicados e as mudanças efetuadas.

5.3. Teste de registo de dados

Nesta parte será demonstrado como o registo de dados é realizado no Firebase. Mostramos primeiro que não existe qualquer tipo de registo guardado na base de dados (Figura 14).

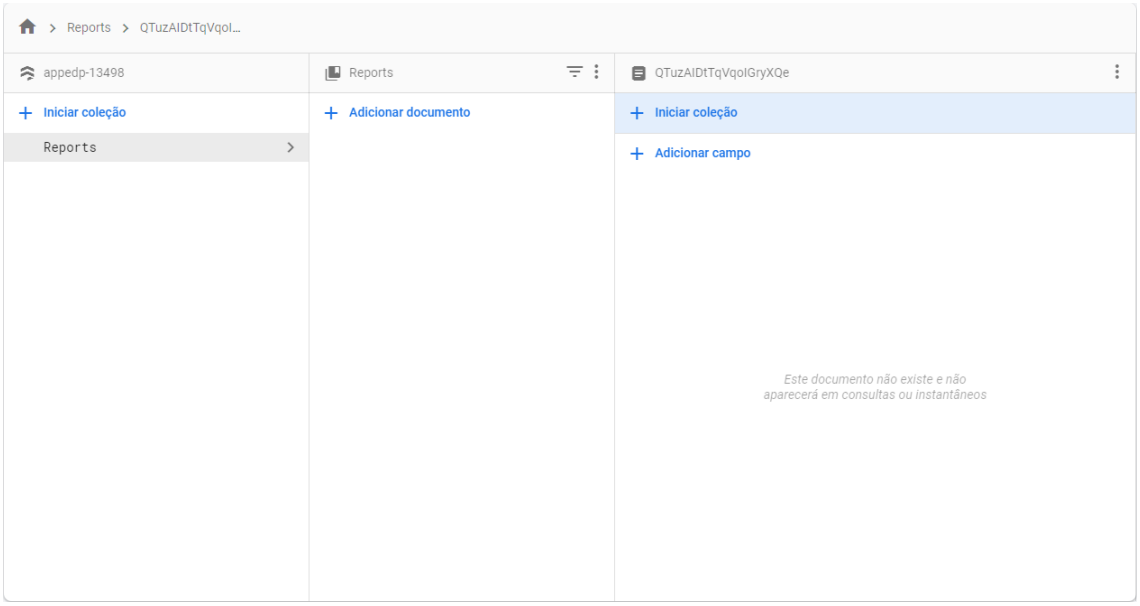

Figura 14 Pré-registo Firebase

Depois através do aplicativo móvel vai ser inserida uma ocorrência, com os valores correspondentes (Figura 15). Trata-se apenas de valores simples sem a inserção de fotos para análise posterior das alterações que causa no Firebase (Figura16). Como se pode observar os valores que foram inseridos correspondem aos valores obtidos não havendo perda de dados. Mostra como a plataforma Firebase é de fácil utilização e fornece uma interface que permite analisar facilmente sempre que algo é adicionado, apagado ou alterado da mesma.

Figura 15 Valores inseridos no aplicativo

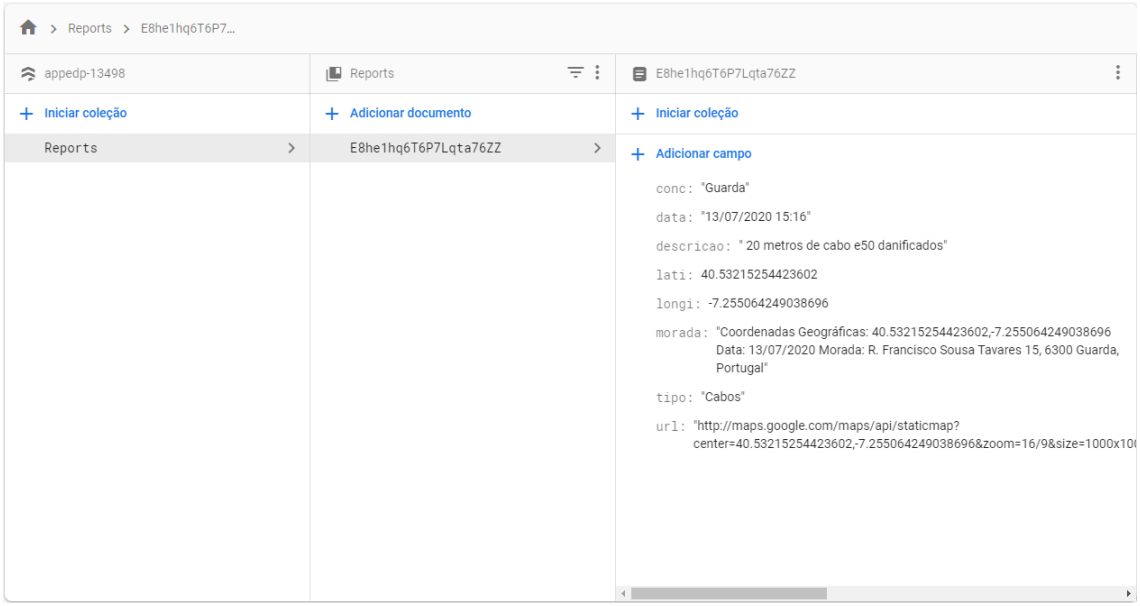

Figura 16 Pós-registo Firebase

6. Conclusão

Este projeto consistiu na realização de um protótipo que tinha como objetivo o registo de avarias por parte dos funcionários da EDP através de uma aplicação móvel com ligação à rede. Para isso foi necessário utilizar conhecimentos adquiridos através das tecnologias lecionadas durante o percurso académico entre outras descobertas independentemente através de pesquisa e com "*trial and error*" para a utilização correta das mesmas. Torna-se uma vantagem pois existe uma vasta expansão de conhecimento através dessas novas tecnologias. Apesar de no início parecer algo complexo, com o passar do tempo, a realização de certas funções tornou-se intuitivo, com o objetivo de ser possível ser usado na vida profissional que se segue ao mundo académico.

Apesar do contentamento pela parte do Engenheiro supervisor, a aplicação não ficou totalmente otimizada pois a parte de estudo para a realização da mesma demorou mais que o previsto pois foi necessário a implementação de uma tecnologia que nunca antes tinha utilizado. No entanto, o estudo realizado durante este estágio permitiu a evolução como potencial engenheiro informático e deu para perceber que o trabalho numa empresa pode ser moroso e com alguns percalços, mas com motivação e esforço torna-se possível a realização dos objetivos propostos.

Este estágio foi fundamental para a aprendizagem de final de curso pois trata-se de uma grande empresa tanto a nível nacional como global e permitiu saber como é o trabalho efetuado numa empresa de grande renome. Apesar de o trabalho ter ficado um pouco aquém das espectativas consegui demonstrar um grande empenho durante a realização do mesmo. Foram adquiridos valores de ética do trabalho, conhecimentos a nível de programação entre outras "*soft-skills*" que irão ser úteis na próxima fase.

De realçar a ajuda prestada pelo Engenheiro Filipe Nogueira que deu indicações precisas para uma maior facilidade de execução do projeto. Puxou pelo estagiário da maneira certa, não pressionando muito para uma compreensão mais cuidada daquilo que queria aplicar na aplicação.

Foi um bom desfecho do percurso académico, uma vez que permitiu criar a oportunidade de trabalhar numa empresa de topo, bem como a aquisição de conhecimentos acerca de tecnologias que estão a ser bastante utilizadas no mundo de hoje. A tecnologia de desenvolvimento de aplicações para dispositivos móveis é uma vertente muito importante da programação pois atualmente o consumidor comum não utiliza com tanta frequência o computador pessoal, sendo o principal método para utilização de internet os dispositivos móveis como tablets e smartphones. Em suma, mostrou a importância da constante aprendizagem num mundo empresarial e pessoal.

Referências Bibliográficas

[1] Android Studio, *Android Studio Documentation.* Disponível em: https://developer.android.com/docs (data da consulta: setembro de 2019)

[2] Firebase, *Firebase Documentation*. Disponível em: https://firebase.google.com/docs (data da consulta: setembro de 2019)

[3] XML. Disponível em: https://abhiandroid.com/ui/xml (data da consulta: setembro de 2019)

[4]GoogleMaps. *Google Maps Documentation*. Disponível em: https://developers.google.com/maps/documentation

[5] Android Tutorial Point. Disponível em: https://www.androidtutorialpoint.com/ (data da consulta: setembro de 2019)

[6] Gradle. Disponível em: https://docs.gradle.org/current/userguide/userguide.html (data da consulta: setembro de 2019)

[7] EDP. Disponível em: https://www.fep.up.pt/docentes/cbrito/Marca%20EDP.pdf (data da consulta: setembro de 2019)

[8] Java. Disponível em: https://docs.oracle.com/javase/7/docs/api/ (data da consulta: setembro de 2019)

[9] Glide. Disponível em: https://bumptech.github.io/glide/ (data da consulta: setembro de 2019)

[10] Apache Cordova. *Apache Cordova Documentation.* Disponível em: https://cordova.apache.org/docs/en/latest/ (data da consulta: setembro de 2019)

[11] Xamarin. *Xamarin Documentation.* Disponível em: https://docs.microsoft.com/en-us/xamarin/ (data da consulta: setembro de 2019)

[12] Silveira, C. (2019) *Apontamentos da Cadeira de Engenharia de Software II*. Texto não publicado, Instituto Politécnico da Guarda, Guarda

Anexos

Código MapsActivity

IconSwitch – serve para modificar o tipo de visualização do mapa.

```
IconSwitch iconSwitch = findViewById(R.id.icon_switch);
iconSwitch.setCheckedChangeListener(new IconSwitch.CheckedChangeListener()
{
    @Override
    public void onCheckChanged(IconSwitch.Checked current) {
        switch (current){
            case RIGHT:
                mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
                break;
            case LEFT:
                mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
                break;
        }
    }
});
```

onMapReady – assim que o mapa é criado este vai ser centrado na posição atual do utilizador utilizando o seguinte código.

```
public void onMapReady(GoogleMap) {
    mMap = googleMap;

    if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            buildGoogleApiClient();
            mMap.setMyLocationEnabled(true);
        }
    }
    else {
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {

LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
        mMap.getUiSettings().setZoomControlsEnabled(true);
    }
    mMap.setOnMapClickListener(this);
    mMap.setOnMapLongClickListener(this);
}
```

onLocationChanged – Sempre que a localização atual for alterada o marcador com as coordenadas da posição atual vai mudar de posição, atribuindo a nova localização. Vai ser obtido através do seguinte código:

```
public void onLocationChanged(final Location location) {

    mLastLocation = location;
    if (mCurrLocationMarker != null) {
        mCurrLocationMarker.remove();
    }
    LatLng latLng = new LatLng(location.getLatitude(),
location.getLongitude());

    CameraUpdate yourLocation = CameraUpdateFactory.newLatLngZoom(latLng,
16);
    mMap.animateCamera(yourLocation);

    if (mGoogleApiClient != null) {

LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
this);
    }
}
```

onMapLongClick – esta função vai server para adicionar um marcador (diferente do marcador com a posição atual) com uma localização aproximada e fixa da ocorrência que queremos identificar na ocorrência.

```
public void onMapLongClick(final LatLng latLng) {
    mMap.clear();

    MarkerOptions markerOptions =
            new MarkerOptions().position(latLng).title(latLng.toString());
    mMap.addMarker(markerOptions);

    btn = findViewById(R.id.saveBtn);
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            openForm(latLng.latitude, latLng.longitude);
        }
    });
}
```

Código FormActivity

openCamera() – Para Abrir a camara do dispositivo e tirar fotos da ocorrência

```
private void openCamera() {
    ContentValues values = new ContentValues();
    values.put(MediaStore.Images.Media.TITLE, "New Picture");
    values.put(MediaStore.Images.Media.DESCRIPTION, "From the Camera");
    image_uri =
getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
values);

    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, image_uri);
    startActivityForResult(cameraIntent, IMAGE_CAPTURE_CODE);
}
```

uploadPhoto() – Para adicionar as photos à base de dados

```
private void uploadPhoto(Uri image_uri) {
    if (image_uri != null) {
        final StorageReference fileReference =
mStorageRef.child(System.currentTimeMillis()
                + "." + getFileExtension(image_uri));

        mUploadTask = fileReference.putFile(image_uri)
                .addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {
fileReference.getDownloadUrl().addOnCompleteListener(new
OnCompleteListener<Uri>() {
                            @Override
                            public void onComplete(@NonNull Task<Uri>
task) {

profileImageUrl=task.getResult().toString();
                                Toast.makeText(FormActivity.this, "URL: "
+ profileImageUrl, Toast.LENGTH_SHORT).show();
                                Toast.makeText(FormActivity.this, "Upload
successful", Toast.LENGTH_LONG).show();
                                Upload = new Upload(profileImageUrl);
                                String uploadId =
mDatabaseRef.push().getKey();

mDatabaseRef.child(uploadId).setValue(upload);
                            }
                        });
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(FormActivity.this, e.getMessage(),
Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onProgress(UploadTask.TaskSnapshot
```

```
taskSnapshot) {
                        mProgress.dismiss();
                    }
                });
    } else {
        Toast.makeText(this, "No file selected",
Toast.LENGTH_SHORT).show();
    }
}
```

sendEmail() – para enviar emails com dados relativos da ocorrência

```
public void sendEmail(String msg, String tempo, String descricao, String
morada, String tip){

    if (tempo.trim().isEmpty() || descricao.trim().isEmpty() ||
morada.trim().isEmpty() || tip.trim().equals("Material Avariado") ||
conc.trim().equals("Seleccione Concelho")) {
        Toast.makeText(this, "Preencha todos os campos",
Toast.LENGTH_SHORT).show();
        return;
    }

    Intent intentEmail = new Intent(Intent.ACTION_VIEW,
Uri.parse("mailto:" + "ocorrenciasedpguarda@gmail.com"));
    intentEmail.putExtra(Intent.EXTRA_SUBJECT, "Ocorrência " + tempo);
    intentEmail.putExtra(Intent.EXTRA_TEXT, msg + "\n\nTipo:\n" + tipo +
"\n\nDescrição:\n" + et.getText().toString());

    startActivity(intentEmail);
}
```

addToList() – para adicionar os dados à base de dados online

```
public void addToList(String data, String descricao, String morada, String
url, Double latit, Double longit, String tip, String conc){

    if (data.trim().isEmpty() || descricao.trim().isEmpty() ||
morada.trim().isEmpty() || tip.trim().equals("Material Avariado") ||
conc.trim().equals("Seleccione Concelho")) {
        Toast.makeText(this, "Preencha todos os campos",
Toast.LENGTH_SHORT).show();
        return;
    }

    CollectionReference reportRef = FirebaseFirestore.getInstance()
            .collection("Reports");
    reportRef.add(new Note(data, descricao, morada, url, latit, longit,
tip, conc));
}
```

createPDF() – para criar um ficheiro PDF

```
public void createPDF(String currentDate, String descricao, String morada,
String tip ) throws IOException, DocumentException {

    if (currentDate.trim().isEmpty() || descricao.trim().isEmpty() ||
morada.trim().isEmpty() || tip.trim().equals("Material Avariado") ||
conc.trim().equals("Seleccione Concelho")) {
        Toast.makeText(this, "Preencha todos os campos",
Toast.LENGTH_SHORT).show();
        return;
    }

    if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.KITKAT) {

        File pdfFolder= new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_D
OCUMENTS),"REP: ");
        if(!pdfFolder.exists()){
            pdfFolder.mkdir();
        }

        Date date = new Date();

        String timeStamp = new SimpleDateFormat("dd-MM-yyyy
HH:mm").format(date);

        File myFile = new File(pdfFolder + timeStamp + ".pdf");

        OutputStream output = new FileOutputStream(myFile);

        Document document = new Document(PageSize.A4);

        imageView.buildDrawingCache();
        Bitmap map = imageView.getDrawingCache();
        photo1.buildDrawingCache();
        Bitmap bmap1 = photo1.getDrawingCache();
        photo2.buildDrawingCache();
        Bitmap bmap2 = photo2.getDrawingCache();
        photo3.buildDrawingCache();
        Bitmap bmap3 = photo3.getDrawingCache();

        PdfWriter.getInstance(document, output);

        document.open();

        generateReport(document, currentDate, bmap1, bmap2, bmap3, map);

        document.close();

    }


}
```

generateReport() – para adicionar os dados ao ficheiro PDF

```
public void generateReport(Document document, String data, Bitmap bmap1,
Bitmap bmap2, Bitmap bmap3, Bitmap map) throws DocumentException {
    Font fTitle = new Font(Font.FontFamily.TIMES_ROMAN, 20, Font.BOLD);
    Font fSubTitle = new Font(Font.FontFamily.TIMES_ROMAN, 18, Font.BOLD);
    Font fNText = new Font(Font.FontFamily.TIMES_ROMAN, 11, Font.BOLD);
    Font fHighText = new Font(Font.FontFamily.TIMES_ROMAN, 15, Font.BOLD,
BaseColor.RED);


    paragraph= new Paragraph("Ocorrência "+data,fTitle);
    paragraph.setSpacingAfter(30);
    paragraph.setAlignment(Element.ALIGN_CENTER);
    document.add(paragraph);
    paragraph= new Paragraph("Dados da Avaria",fSubTitle);
    paragraph.setSpacingAfter(30);
    paragraph.setAlignment(Element.ALIGN_CENTER);
    document.add(paragraph);
    paragraph= new Paragraph("Dados da Avaria",fSubTitle);
    paragraph= new Paragraph("Localização da Avaria:",fHighText);
    paragraph.setSpacingAfter(15);
    document.add(paragraph);

    paragraph = new Paragraph();
    if (map!=null){
        ByteArrayOutputStream stream1 = new ByteArrayOutputStream();
        map.compress(Bitmap.CompressFormat.JPEG, 100 , stream1);
        Image Map = null;
        try {
            Map = Image.getInstance(stream1.toByteArray());
        } catch (IOException e) {
            e.printStackTrace();
        }
        Map.scaleAbsolute(200, 200);
        Map.setAlignment(Image.MIDDLE);
        paragraph.add(Map);
        paragraph.setSpacingAfter(15);
        document.add(paragraph);
    } else {
        paragraph.add("Nenhuma foto!");
        document.add(paragraph);
    }

    /*paragraph= new Paragraph("URL do Mapa: " + imagem,fNText);
    document.add(paragraph);*/
    paragraph= new Paragraph("Latitude: " +lat,fNText);
    document.add(paragraph);
    paragraph= new Paragraph("Longitude: " +lon,fNText);
    document.add(paragraph);
    paragraph= new Paragraph("Endereço: " +address,fNText);
    paragraph.setSpacingAfter(15);
    document.add(paragraph);
    paragraph= new Paragraph("Tipo de Avaria:",fHighText);
    paragraph.setSpacingAfter(15);
    document.add(paragraph);
    paragraph= new Paragraph("Tipo de  Material: " + tipo,fNText);
    document.add(paragraph);
    paragraph= new Paragraph("Descrição: " +
et.getText().toString(),fNText);
    paragraph.setSpacingAfter(15);
    document.add(paragraph);
    paragraph= new Paragraph("Imagens:",fHighText);
```

```
        paragraph.setSpacingAfter(15);
        document.add(paragraph);

        paragraph = new Paragraph();
        if (bmap1!=null){
            ByteArrayOutputStream stream1 = new ByteArrayOutputStream();
            bmap1.compress(Bitmap.CompressFormat.JPEG, 100 , stream1);
            Image myImg1 = null;
            try {
                myImg1 = Image.getInstance(stream1.toByteArray());
            } catch (IOException e) {
                e.printStackTrace();
            }
            myImg1.scaleAbsolute(300, 300);
            myImg1.setAlignment(Image.MIDDLE);
            paragraph.add(myImg1);
            paragraph.setSpacingAfter(15);
            document.add(paragraph);
        } else {
            paragraph.add("Nenhuma foto!");
            document.add(paragraph);
        }

        paragraph = new Paragraph();
        if(bmap2!=null) {
            ByteArrayOutputStream stream2 = new ByteArrayOutputStream();
            bmap2.compress(Bitmap.CompressFormat.JPEG, 100 , stream2);
            Image myImg2 = null;
            try {
                myImg2 = Image.getInstance(stream2.toByteArray());
            } catch (IOException e) {
                e.printStackTrace();
            }
            myImg2.scaleAbsolute(300, 300);
            myImg2.setAlignment(Image.MIDDLE);
            paragraph.add(myImg2);
            paragraph.setSpacingAfter(15);
            document.add(paragraph);
        } else {
            document.add(paragraph);
        }

        paragraph = new Paragraph();
        if(bmap3!=null) {
            ByteArrayOutputStream stream3 = new ByteArrayOutputStream();
            bmap3.compress(Bitmap.CompressFormat.JPEG, 100 , stream3);
            Image myImg3 = null;
            try {
                myImg3 = Image.getInstance(stream3.toByteArray());
            } catch (IOException e) {
                e.printStackTrace();
            }
            myImg3.scaleAbsolute(300, 300);
            myImg3.setAlignment(Image.MIDDLE);
            paragraph.add(myImg3);
            paragraph.setSpacingAfter(15);
            document.add(paragraph);
        } else {
            document.add(paragraph);
        }
    }
```

Código NoteActivity

setUpRecyclerView() – para popular a RecyclerView com as ocorrências registadas na base de dados. Pode-se alterar os resultados modificando a query.

```
private void setUpRecyclerView() {
    Query query = reportRef.orderBy("data", Query.Direction.DESCENDING);

    FirestoreRecyclerOptions<Note> options = new
FirestoreRecyclerOptions.Builder<Note>()
            .setQuery(query, Note.class)
            .build();

    adapter = new NoteAdapter(options);
    adapter.startListening();
    RecyclerView recyclerView = findViewById(R.id.recycler_view);
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    recyclerView.setAdapter(adapter);

    new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0,
            ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
        @Override
        public boolean onMove(RecyclerView recyclerView,
RecyclerView.ViewHolder viewHolder, RecyclerView.ViewHolder target) {
            return false;
        }
        @Override
        public void onSwiped(RecyclerView.ViewHolder viewHolder, int
direction) {
            adapter.deleteItem(viewHolder.getAdapterPosition());
        }
    }).attachToRecyclerView(recyclerView);

    adapter.setOnItemClickListener(new NoteAdapter.OnItemClickListener() {
        @Override
        public void onItemClick(DocumentSnapshot documentSnapshot, int
position) {
            Note note = documentSnapshot.toObject(Note.class);
            String id = documentSnapshot.getId();
            String path = documentSnapshot.getReference().getPath();
            String data = note.getData();
            Double lat = note.getLati();
            Double lon = note.getLongi();
            String desc = note.getDescricao();
            String url = note.getUrl();
            String conc = note.getConc();
            openRegister(lat, lon, data, desc, morada, tipo, url, conc);
        }    });
}
```

Código RegisterActivity

geoLocate() – Através do valor em **mSearchText** vai centrar o mapa nessa localização

```
private void geoLocate() {
    String searchString = mSearchText.getText().toString();
    Geocoder geocoder = new Geocoder(this);

    List<Address> list = new ArrayList<>();
    try{
        list = geocoder.getFromLocationName(searchString, 1);
    } catch (IOException e) {
        e.printStackTrace();
    }
    if(list.size()>0){
        Address address = list.get(0);
        Double lat = address.getLatitude();
        Double lon = address.getLongitude();
        LatLng latLng = new LatLng(lat, lon);
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(13));
    }
}
```

addMarkers() – adiciona marcadores com diferentes cores dependendo do tipo da avaria

```
private void addMarkers() {
    reportRef.get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if(task.isSuccessful()){
                for(QueryDocumentSnapshot documentSnapshot:
task.getResult()){

                    float color = BitmapDescriptorFactory.HUE_RED;;
                    Double lati = documentSnapshot.getDouble("lati");
                    Double longi = documentSnapshot.getDouble("longi");
                    String tipo = documentSnapshot.getString("tipo");
                    String desc =documentSnapshot.getString("descricao");

                    LatLng latLng = new LatLng(lati, longi);


                    if(tipo.equals("Postes")){
                        color = BitmapDescriptorFactory.HUE_BLUE;
                    } else if (tipo.equals("TP")) {
                        color = BitmapDescriptorFactory.HUE_GREEN;
                    } else if (tipo.equals("Coluna IP")) {
                        color = BitmapDescriptorFactory.HUE_MAGENTA;
                    }
                    mMap.addMarker(new
MarkerOptions().position(latLng).title(tipo).snippet(desc).icon(BitmapDesc
riptorFactory.defaultMarker(color)));
                }
            } else {
                Toast.makeText(RegisterActivity.this, "Erro!!",
Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```