INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Gestão distribuída de ocorrências no espaço público

Jorge Daniel Silva de Magalhães

Mestrado em Engenharia de Telecomunicações e Informática,

Orientador:
Doutor Luís Henrique Ramilo Mota, Professor Auxiliar,
Iscte - Instituto Universitário de Lisboa

Novembro, 2021

TECNOLOGIAS
E ARQUITETURA

Departamento de Ciências e Tecnologias da Informação

Gestão distribuída de ocorrências no espaço público

Jorge Daniel Silva de Magalhães

Mestrado em Engenharia de Telecomunicações e Informática,

Orientador:
Doutor Luís Henrique Ramilo Mota, Professor Auxiliar,
Iscte - Instituto Universitário de Lisboa

Novembro, 2021

Utilizo esta dedicatória para agradecer a ajuda e apoio dos meus pais e do meu irmão neste percurso académico.

Agradecimento

Quero começar por agradecer aos meus pais e ao meu irmão, dado que foram as pessoas mais importantes na ajuda ao meu percurso académico.

Agradecer também o apoio incondicional dos meus amigos mais chegados, que me ajudaram a ultrapassar momentos mais difíceis do curso, não só da licenciatura como também do mestrado.

Muito grato pela instituição ISCTE que forneceu os recursos necessários pelo meu sucesso e aos meus professores por terem-me ajudado nestes anos.

Agradeço, por fim, ao meu orientador Luís Mota pela disponibilidade e orientação durante a realização da dissertação.

Resumo

Ocorrem diariamente vários incidentes na via pública passíveis de serem tratados, não existindo assim uma plataforma para a resolução destes.

Posto isto, foi criada uma aplicação móvel Android para o registo de ocorrências na via pública. Esta aplicação foi feita com base num mapa interativo, onde os utilizadores podem criar e visualizar as ocorrências existentes, podendo ver assim quais os problemas identificados e o estado das ocorrências. O utilizador poderá também dar a sua apreciação e opinião sobre as ocorrências, dando assim mais contexto e informação sobre as mesmas. Poderá também escolher seguir ocorrências, com o objetivo de ser informado sobre futuras atualizações sobre o estado das mesmas. Por fim, tem também a opção de visualizar graficamente as ocorrências mais comuns de cada localidade.

Para a criação desta aplicação móvel foram analisadas várias bases de dados, de modo a escolher a mais adequada para este tipo de aplicação. Esta aplicação levou à escolha da base de dados MongoDB. Foi também criado um servidor NodeJS a fim de receber os pedidos do utilizador, pesquisando na base de dados e devolvendo assim a informação pretendida.

A aplicação foi bem recebida por parte dos utilizadores, porém estes deram várias sugestões para a melhoria da mesma. Algumas destas sugestões foram ainda implementadas, tendo outras ficado para melhorar no futuro.

Em relação a aplicações semelhantes para resolver o problema que esta dissertação visa resolver, já existiam algumas soluções idênticas, no entanto esta dissertação veio melhorar as mesmas.

Palavras-Chave: Ocorrência; Mapas; Internet; Utilizador; Aplicação Móvel.

Abstract

Several treatable incidents occur daily on public roads and there is no platform for resolving these incidents.

That being said, it was built a mobile android application for the registration of occurrences on public roads. The mobile application is built around an interactive map, where users can create and visualize existing occurrences, thus being able to see what the problem is and what is their state. The user can also give his appreciation and opinion to the occurrences, in order to give more relevance and more information about them. Furthermore, the user can also choose to follow events, in order to be informed about future updates of their status. At last, the user also has the option to visualize a pie chart that contains information about the location and their occurences.

For the creation of this mobile application it was studied which database would be more interesting to use for this type of application, having been chosen the MongoDB database. A NodeJS server was also created with the objective of receiving user requests, searching the database and returning the desired information to the user.

The mobile application was well received by users but they gave several suggestions for future improvement, some of which were implemented and others were left for future work.

There were already some solutions to solve this problem, however this dissertation was made to improve the existing options.

Keywords: Occurrence, Maps, Internet, User, Mobile Application

Índice

Lista de Tabelas

Lista de Figuras

Acrónimos

SIG - Sistema de Informação Geográfica
SQL - Structured Query Language
GIST - Generalized Search Tree
BRIN - Block Range Index
SP-GIST - Space-Partitioned Generalized Search Tree
MQL - MongoDB Query Language
JSON - JavaScript Object Notation
IU - Interface do Utilizador
UM - University of Mississippi
USM - University of Southern Mississippi
RAM - Random-Access Memory
MB - Megabyte
HTTP - Hypertext Transfer Protocol
REST - Representational State Transfer
API - Application Programming Interface
OSM - OpenStreetMap

CAPíTULO 1

Introdução

1.1. Motivação

Todos os dias ocorrem incidentes na via pública, tais como problemas de higienização, de iluminação, entre outros. Contudo, não existe uma plataforma que permita a qualquer cidadão alertar as entidades responsáveis para reportar incidentes na via pública (existindo apenas opções limitadas em alguns concelhos), sendo que, geralmente, para tratar destes assuntos é necessário ir à junta de freguesia formular uma queixa, o que se apresenta como uma opção trabalhosa e demorada.

Algumas câmaras municipais já implementaram plataformas para tratar de incidentes na via pública, porém não existe uma plataforma global para a resolução destes, sendo este um dos problemas que se deseja resolver. Estas aplicações possuem outro problema importante, sendo esse a sua gestão fechada e opaca para com os utilizadores, dado que não possuem uma forma de comunicação com os utilizadores sobre a resolução das ocorrências.

1.2. Enquadramento

Algumas câmaras municipais já avançaram com sítios *web* e aplicações móveis com o intuito de resolver ocorrências nas suas áreas respetivas, assim como a câmara municipal de Sintra e Lisboa com a criação das aplicações móveis Sintra Resolve [**1**] e Na Minha Rua Lx [**2**], respetivamente, em que possuem funcionalidades semelhantes ao objetivo desta aplicação, tais como:

- Criação de ocorrências num mapa interativo.
- Na criação da ocorrência, a possibilidade de carregar uma foto para melhor demonstração da ocorrência.
- Disponibilização de informação sobre o estado da ocorrência.

Este trabalho tem como objetivo criar uma aplicação que permita aos utilizadores reportar qualquer tipo de ocorrência independentemente da sua localização, dando assim oportunidade a qualquer cidadão português de reportar ocorrências através do uso de uma aplicação móvel. Deste modo, a aplicação não terá apenas uma abrangência local, mas sim uma abrangência nacional.

É importante também que a aplicação permita aos utilizadores verificarem quais os critérios, prioridades e estratégias utilizados na gestão das ocorrências por parte das entidades responsáveis, através de um espaço onde estes possam comunicar entre si.

1.2.1. Objetivo

O objetivo desta dissertação é a criação e avaliação de uma aplicação móvel que consiga registar ocorrências em tempo real em todo o território nacional. Também deverá ser possível observar ocorrências criadas por todos os utilizadores através de um mapa interativo e escolher várias formas de visualização das mesmas, como, por exemplo, visualizar apenas ocorrências numa certa área escolhida pelo utilizador.

A aplicação também deverá ser intuitiva de modo a que seja de fácil utilização para qualquer cidadão, mesmo que este não esteja habituado a utilizar tecnologias. Também irá ser possível criar alertas e visualizar estatísticas gerais, ou por entidade, de modo a avaliar a resposta das mesmas.

1.2.2. Especificação da Aplicação

Nesta secção, irão ser descritas as várias funcionalidades pretendidas para a aplicação e como esta irá funcionar no que toca aos utilizadores e ao tratamento das ocorrências.

Quando o utilizador inicia a aplicação, é-lhe apresentado um mapa interativo onde poderá visualizar todos os eventos criados através de ícones intuitivos que identificam cada tipo de ocorrência. Ao carregar no ícone de uma ocorrência, é direcionado para um ecrã onde tem a informação detalhada sobre a mesma, assim como a localização, número de apreciações, descrição, estado e possíveis comentários feitos por outros utilizadores.

Neste ecrã, os utilizadores vão poder dar a sua opinião sobre a ocorrência, o que possibilita a existência de uma perspetiva diferente da do criador. Haverá também uma opção para os utilizadores sinalizarem a resolução da ocorrência sem ação de entidade terceira, se tal de facto se verificar.

As ocorrências têm cinco possíveis estados:

- **Sob Análise** - A ocorrência ainda está sob consideração e precisa de aprovação por parte da entidade que tem responsabilidade sobre a ocorrência.
- **Registado** - A ocorrência já foi aprovada, mas ainda está à espera do início da sua resolução.
- **Em execução** - A ocorrência já começou a ser tratada.
- **Finalizado** - A ocorrência já foi resolvida.
- **Rejeitado** - O evento foi rejeitado pela entidade.

O ciclo da criação duma ocorrência é assim descrito de seguida.

(1) O utilizador cria uma ocorrência: escolhe a localização, o tipo e escreve uma breve descrição sobre a situação.
(2) A entidade apropriada para resolver o problema é automaticamente contactada, ficando a ocorrência no estado "Sob Análise".
(3) De seguida, um supervisor verifica a credibilidade da ocorrência. Se os pressupostos não se verificarem, o supervisor apaga a ocorrência. A entidade pode também rejeitar resolver a ocorrência, mudando o estado da ocorrência para "Rejeitado" e informando os utilizadores interessados dessa decisão.

(4) Após o registo, a ocorrência mantém-se neste estado até a entidade responsável começar a resolver o problema ou até ser resolvido, como pode acontecer, sem a intervenção dos responsáveis. Quando começar a sua resolução, a entidade altera o estado da ocorrência para "Em Execução", e os utilizadores interessados recebem uma notificação pelo endereço eletrónico em como a ocorrência irá começar a ser tratada.

(5) Por fim, quando a ocorrência for tratada, o utilizador responsável pela entidade altera o estado para "Finalizado", notificando automaticamente todos os utilizadores.

Isto pode ser verificado também com o auxílio do fluxograma da figura 1.

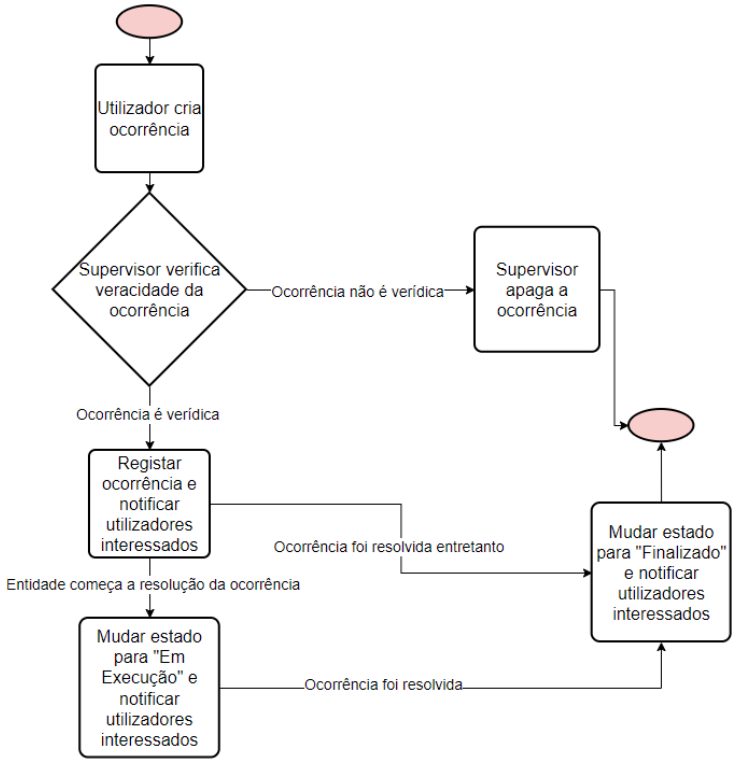

FIGURA 1. Fluxograma - Ciclo da Ocorrência

O utilizador tem a possibilidade de filtrar as ocorrências por vários critérios. Como se pode observar pelo diagrama de casos (figura 2), o utilizador pode escolher filtrar essas ocorrências pela sua categoria, de modo a facilitar a procura. Por exemplo, o utilizador pode querer apenas procurar ocorrências sobre "Higiene Pública", que concerniriam apenas aquelas que possam comprometer a salubridade do espaço público.

Posteriormente, quando todos os eventos estiverem a ser mostrados ao utilizador, estes podem ser ordenados pelo número de apreciações. Isto tem como intuito dar ao utilizador acesso aos eventos mais relevantes.

O utilizador poderá dar uma apreciação às ocorrências, atribuindo-lhes maior ou menor relevância. O utilizador poderá também escolher seguir uma determinada ocorrência, de

modo a receber notificações quando a ocorrência mudar de estado, ficando assim sempre a par do desenvolvimento da resolução da ocorrência em questão.

O utilizador poderá igualmente escolher seguir uma área de interesse como, por exemplo, a cidade de Lisboa, ou uma zona mais pequena como um bairro, para receber notificações sempre que alguma ocorrência é criada e sempre que muda o estado. Por fim, o utilizador vai ter também a possibilidade de seguir utilizadores, para poder receber informação sempre que os mesmos criem ocorrências e sempre que estas sejam atualizadas.

Todos os utilizadores deverão estar registados para poder criar ocorrências na aplicação. Tal é necessário para que o utilizador não tenha de fornecer os seus dados pessoais sempre que crie uma ocorrência, ficando essa informação gravada no seu perfil.

Se o utilizador escolher utilizar a aplicação sem criar uma conta pessoal, pode utilizá-la enquanto anónimo, ficando limitado na utilização, podendo apenas visualizar as ocorrências existentes.

Para criar uma ocorrência, o utilizador terá de:

(1) Escolher a localização do evento, diretamente no mapa fornecido.
(2) Escolher o tipo de ocorrência.
(3) Descrever a ocorrência, podendo adicionar uma fotografia como ajuda à caracterização da mesma.

Na figura 2 pode ver-se um diagrama que descreve as ações que podem ser executadas nesta aplicação, e, na figura 3, está representado um diagrama de componentes, cujo objetivo é mostrar a relação que os diferentes componentes do sistema têm entre si.

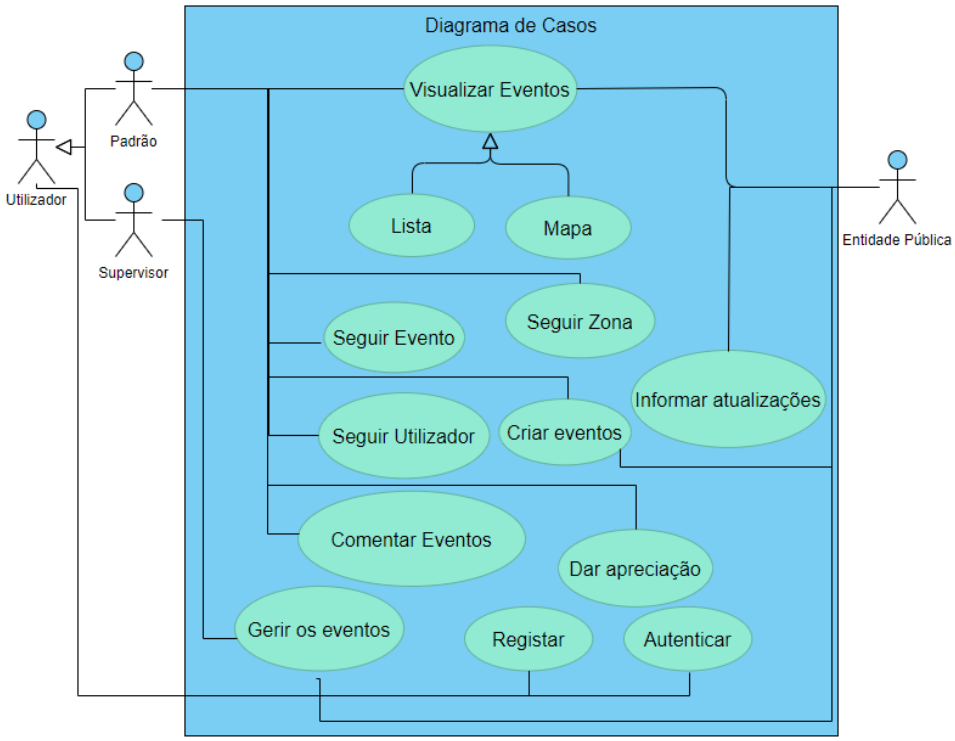

FIGURA 2. Diagrama de Casos

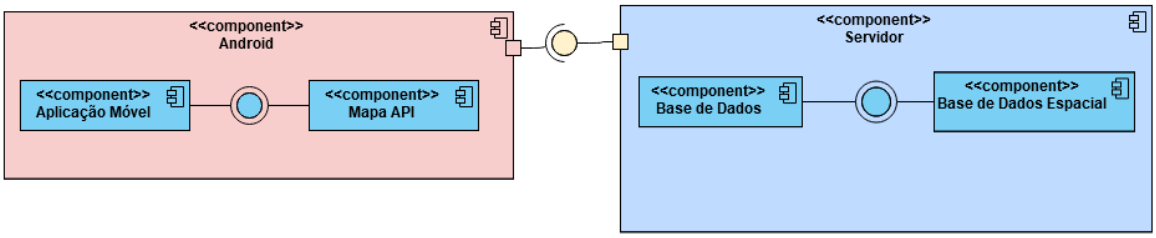

FIGURA 3. Diagrama de Componentes

1.2.3. Tipos de Utilizadores

Vão existir 4 categorias de utilizadores nesta aplicação, o utilizador "Padrão", "Supervisor", "Entidade" e "Anónimo".

- **Utilizador Padrão** - Estes utilizadores possuem os privilégios básicos de utilização da aplicação. Estes privilégios básicos incluem a criação de ocorrências, a verificação de ocorrências já existentes, e o resto das funcionalidades básicas da aplicação.
- **Supervisor** - Estes utilizadores terão capacidades acrescidas e deverão verificar a veracidade e exatidão das ocorrências criadas. Este utilizador vai possuir autoridade de poder limitar as funcionalidades de contas de utilizadores em caso de utilização inapropriada da aplicação.
- **Utilizador Entidade** - Estes utilizadores representam entidades públicas junto do sistema, sendo os responsáveis por determinar se a ocorrência já começou a ser tratada, informando assim os utilizadores interessados da sua atualização.
- **Utilizador Anónimo** - Todos os utilizadores da aplicação poderão utilizar a aplicação neste modo, podendo apenas visualizar as ocorrências criadas por outros utilizadores.

1.3. Questões de Investigação

Esta dissertação visa responder as seguintes questões de investigação:

(1) Como é possível a criação de uma aplicação de ocorrências que consiga cobrir todo o território nacional português?
(2) Será possível a implementação de estatísticas interessantes através do uso de mapas, assim como, qual o tipo de ocorrência mais recorrente numa certa área?
(3) Como será tratado o grande número de utilizadores que a aplicação possa vir a ter?
(4) Qual o valor acrescentado da interface do utilizador ser baseada em mapas?

1.4. Método de Investigação

Para a realização desta dissertação foi escolhido o método de investigação "Design Science Research Method" [**3**]. Existem quatro abordagens distintas que podem ser utilizadas, tendo sido escolhida a abordagem "Design and Development-Centered", visto que

esta abordagem é utilizada quando já existe uma tecnologia semelhante, que porém não possui todas as qualidades necessárias para satisfazer o objetivo final.

Assim, este método de investigação possui quatro etapas:

(1) **Design e Desenvolvimento -** O objetivo nesta etapa é fazer o *design* e o desenvolvimento da aplicação final.

(2) **Demonstração -** Nesta etapa é realizada a demonstração das funcionalidades da aplicação e o seu uso numa situação real.

(3) **Avaliação -** São realizados testes na aplicação com o objetivo de verificar potenciais problemas que a aplicação possa vir a apresentar.

(4) **Comunicação -** Por fim, é apresentado o produto final e respetivas conclusões obtidas durante a realização desta dissertação.

CAPíTULO 2

Revisão da Literatura e Estado da Arte

Esta secção visa descrever os mais recentes desenvolvimentos relacionados com a resolução de ocorrências no espaço público, utilizando um sistema de informação geográfica (SIG). Primeiro, far-se-á uma exposição do que é um SIG, e as aplicações que podem ser implementadas sobre este tipo de sistemas. De seguida, serão estudados algumas bases de dados espaciais, comparando-as entre si, contribuindo assim para a escolha final da base de dados mais apropriada para o presente trabalho.

2.1. Sistema de Informação Geográfica

Um SIG é um "sistema espacial que obtém, gere, analisa e apresenta todo o tipo de dados" [**4**]. Estes dados espaciais guardam informação geográfica, mas também podem guardar informação adicional sobre estes (por exemplo, as suas dimensões ou pose) [**5**].

Um SIG tem a função de obter os dados espaciais necessários e fazer a sua análise, de modo a remover possíveis erros que possam existir sobre estes. Tem também a função de armazenar e apresentar os dados aos utilizadores através da utilização de mapas [**6**].

Um SIG pode ser utilizado em múltiplas áreas como a saúde, *marketing*, prevenção de acidentes rodoviários, ou gestão de ocorrências no espaço público, que é o caso de estudo desta dissertação, entre outras [**4**].

2.2. Bases de Dados Espaciais

Para manter a informação obtida dos utilizadores, é necessária a utilização de um sistema de informação. Sendo assim, esta secção vai focar-se na descrição de várias bases de dados espaciais, mostrar os seus benefícios e fraquezas, e escolher a mais indicada para a realização desta dissertação.

Uma base de dados espacial é uma base de dados que consegue armazenar e gerir dados espaciais. Existem três aspetos importantes sobre bases de dados espaciais: o tipo de dados, a indexação, e as funções espaciais [**7**].

Tipo de dados espaciais

Os tipos de dados espaciais têm a função de representar dados geográficos. Pode-se verificar na seguinte imagem os diferentes tipos de dados espaciais que tipicamente existem.

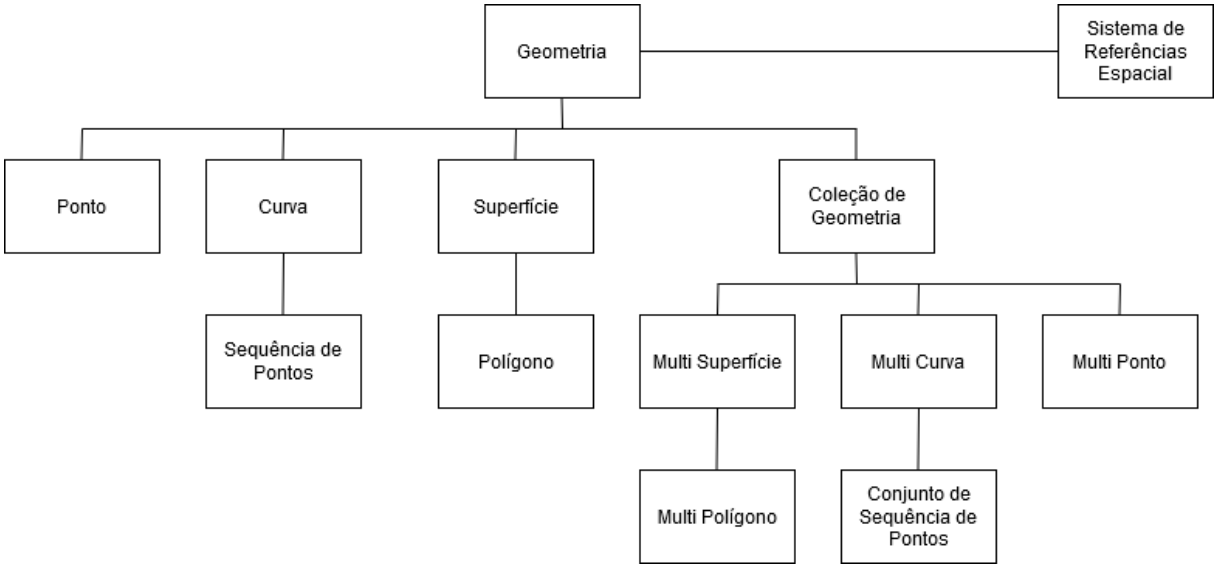

FIGURA 1. Hierarquia dos Dados Espaciais

Como se pode visualizar pela figura 1[1], estes tipos de dados estão organizados de uma forma hierárquica levando a que cada subtipo herde todos os atributos e métodos do supertipo a que está relacionado [**7**].

Índices Espaciais

"Índice Espacial é uma estrutura de dados que visa aceder a um objeto espacial de forma eficiente" [**8**]. Esta solução é muito usada nas bases de dados sendo especialmente importante para bases de dados espaciais, dado que aumenta significativamente o desempenho da mesma. Isto deve-se ao facto de, se não existir indexação, "qualquer procura exige uma "busca sequencial" de cada registo na base de dados" [**9**].

Funções Espaciais

Por fim, as funções espaciais têm a finalidade de fazer operações neste tipo de dados, podendo ser agrupadas em cinco categorias: **funções de conversão**, que servem para converter os dados para outro tipo de formatos, **funções de gestão**, que têm como objetivo gerir a informação sobre as tabelas espaciais, **funções de recuperação**, cuja finalidade é devolver propriedades e medidas dos dados espaciais, **funções de comparação**, que servem para comparar dois dados espaciais entre si, e, por fim, **funções de geração** que geram novos dados a partir de outros [**7**].

Agora que se tem uma noção do que são bases de dados espaciais, segue-se a comparação de algumas bases de dados espaciais estudadas.

[1]Imagem adaptada de: `https://postgis.net/workshops/postgis-intro/introduction.html`

2.2.1. PostgreSQL

PostgreSQL é uma base de dados relacional de código aberto que utiliza SQL (*Structured Query Language*) como linguagem. Por si só não é uma base de dados espacial, mas pode ser complementada com a extensão PostGIS que adiciona os recursos para a utilização desta base de dados como base de dados espacial [**7**]. Suporta os tipos padrão de dados geométricos: "Ponto", "Sequência de Pontos", "Polígono", "Multiponto", "Conjunto de Sequência de Pontos", "MultiPolígono" e "Coleção de Geometria", e ainda dois tipos de dados geométricos mais complexos, "Rede Irregular Triangulada" e "Superfícies Poliédricas".

Estes dois dados geométricos servem para representar dados que possuem três dimensões. De acordo com informação obtida através do site oficial do PostGis [**10**], o tipo geométrico "Rede Irregular Triangulada permite modelar redes triangulares como linhas na base de dados" e o tipo "Superfícies Poliédricas permite modelar objetos volumétricos na base de dados". Suportam também três categorias de índices espaciais, o GiST (*Generalized Search Tree*), BRIN (*Block Range Index*), e, por fim, SP-GiST (*Space-Partitioned Generalized Search Tree*), que, como referido anteriormente, procuram acelerar a execução de consultas espaciais [**11**].

Em termos de funções espaciais, oferece mais de mil funções deste tipo [**12**]. Funções exemplo deste tipo são as funções de comparação "ST_Intersects", que, ao fornecer dois tipos geométricos como argumento, retorna verdadeiro se os argumentos da função se intercetarem e falso se forem disjuntos [**13**], e "ST_DWithin", que, ao fornecer dois tipos geométricos como argumento, retorna verdadeiro se os argumentos da função estiverem a uma certa distância [**14**].

2.2.1.1. *CARTO* - "CARTO é uma ferramenta de código aberto que permite o armazenamento e visualização de dados geoespaciais na *web*" [**15**]. Este utiliza o PostgreSQL com a extensão PostGIS como base de dados espacial para de seguida proceder à sua visualização no mapa [**16**].

2.2.2. MongoDB

MongoDB é uma base de dados orientada a documentos de código aberto, que ao invés de utilizar a linguagem de consulta SQL utilizada por bases de dados relacionais, utiliza a linguagem MQL (*MongoDB Query Language*) [**17**]. Esta base de dados armazena informação em documentos similares a JSON (*JavaScript Object Notation*), como se pode verificar na seguinte figura[2] [**18**].

[2] `https://webassets.mongodb.com/_com_assets/cms/1-lwnlfl1ryn.png`

FIGURA 2. Documentos na Base de Dados MongoDB

Segundo o manual do MongoDB [**19**], esta base de dados oferece várias características, como alto desempenho, devido à utilização de dados embutidos e índices, e alta disponibilidade, visto que a MongoDB disponibiliza vários servidores com a informação replicada, fazendo com que esta fique redundante e, em caso de um erro de um deles, a informação possa ser sempre obtida noutro. Oferece também escalabilidade horizontal, utilizando um processo de *sharding*, que é o "processo de garantir que os dados são distribuídos uniformemente pelo grupo de servidores" [**20**].

Relativamente aos tipos de dados, o MongoDB suporta "Ponto", "Sequência de Pontos", "Polígono", "Multiponto", "Conjunto de Sequência de Pontos", "MultiPolígono" e "Coleção de Geometria" [**21**]. Suporta também dois tipos diferentes de indexação espacial, "2dsphere" e "2d". Por fim, suporta um conjunto pequeno de funções espaciais, suportando apenas 4 funções distintas: "geoIntersects", "geoWithin", "near" e "nearSphere" [**22**].

2.2.3. MariaDB

MariaDB é uma base de dados relacional de código aberto que utiliza SQL como linguagem e tem como base o MySQL, sendo um desenvolvimento independente do mesmo [**23**]. Suporta os tipos geométricos padrão assim como "Ponto", "Sequência de Pontos", "Polígono", "Multiponto", "Conjunto de Sequência de Pontos", "MultiPolígono" e "Coleção de Geometria" [**24**]. Em termos de funções espaciais, oferece cerca de uma centena deste tipo de funções [**25**]. Por fim, oferece também a função de indexação espacial "R-Tree" para acelerar o processo de consulta [**26**].

2.2.4. SQLite

SQLite é uma base de dados relacional, que ao invés de comunicar de maneira cliente-servidor como as bases de dados demonstradas anteriormente, tem um servidor embutido no próprio programa [**27**]. À semelhança da base de dados PostgreSQL, o SQLite também precisa de uma extensão para guardar dados espaciais, neste caso designada de Spatialite.

Em termos de tipos de dados, suporta "Ponto", "Sequência de Pontos", "Polígono", "Multiponto", "Conjunto de Sequência de Pontos", "MultiPolígono" e "Coleção de Geometria". Suporta dois tipos diferentes de indexação, "R-Tree" e "MbrCache" [**28**]. Em relação às funções geoespaciais, oferece cerca de uma centena deste tipo de funções, assim como funções de comparação como "Equals", que verifica se dois dados geométricos são idênticos, "Touches" que verifica se os pontos em comum dos dois dados se encontram apenas na união dos seus limites, e "Distance" que retorna a distância entre dois dados geométricos [**29**].

2.2.5. Comparação das Bases de Dados

Depois de estudadas as quatro bases de dados espaciais, comparar-se-ão agora as características de cada uma na seguinte tabela.

	PostgreSQL	MariaDB	MongoDB	SQLite
Tipos de Dados Espaciais	9 tipos de dados	7 tipos de dados	7 tipos de dados	7 tipos de dados
Funções Espaciais	+1000 funções	+100 funções	4 funções	+100 funções
Índices Espaciais	GIST, BRIN, SP-Gist	R-Tree	2dsphere, 2d	R-Tree, MbrCache

Tabela 1. Especificações das Base de Dados

Como se pode verificar pela tabela, a base de dados PostgreSQL [**30**] apresenta mais tipos de dados espaciais (Rede Irregular Triangulada e Superfícies Poliédricas) comparativamente às restantes bases de dados (MariaDB [**31**], MongoDB [**32**] e SQLite [**33**]).

Em relação às funções espaciais, o PostgreSQL é também a base de dados mais completa, fornecendo assim um número consideravelmente maior de funções espaciais em comparação às restantes bases de dados.

Por fim, o PostgreSQL é também a base de dados que apresenta uma maior variedade de índices espaciais.

2.3. Aplicações Similares

Este tipo de aplicações que utilizam sistemas de informação geográfica podem ser utilizados em vários contextos diferentes. Descrevem-se assim aqui várias aplicações com pontos comuns a esta dissertação, com o propósito de demonstrar os avanços feitos sobre este tema.

2.3.1. *1st Incident Reporting*

1st Incident Reporting é uma aplicação, tanto em versão móvel como em plataforma *web* direcionada a empresas, a fim de gerir ocorrências que possam acontecer no meio

empresarial, como ocorrências sobre dano de ativos, reportes de patrulha, entre outros. As funcionalidades desta aplicação podem ser divididas em três tipos [**34**].

Captura da Ocorrência - O utilizador pode criar uma ocorrência, podendo adicionar conteúdo multimédia para caracterizar o acontecimento, tal como uma foto, um vídeo ou um áudio, como se pode verificar na seguinte figura 3[3] [**35**].

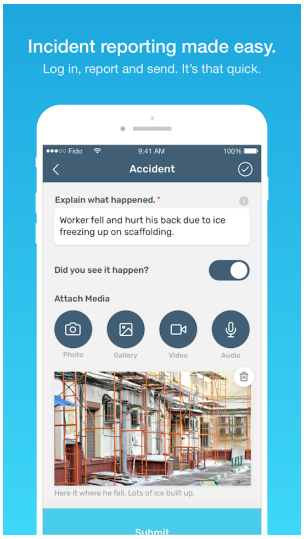

FIGURA 3. 1st Incident Reporting - Reportar Ocorrência

Gerir a Ocorrência - Os incidentes podem ser visualizados em tempo real, tanto em forma de lista, como num mapa. Tanto o mapa como a lista possuem vários ícones que permitem distinguir a diferença entre cada tipo de incidente facilmente. No separador "Lista", o utilizador consegue verificar rapidamente os incidentes mais recentes, aparecendo, para além do ícone, um título a especificar o problema, a data da ocorrência, e por quem foi criado. Estas notificações, ao invés de serem enviadas para todos os trabalhadores numa empresa, são enviadas apenas para aqueles que precisem de ser informados dessa ocorrência.

Sumário das Ocorrências - A aplicação cria relatórios sobre as ocorrências, possibilitando o seu estudo para prevenção futura.

2.3.2. *Everbridge*

Everbridge é uma empresa que oferece serviços com o objetivo de fornecer informação de situações críticas a empresas ou indivíduos. Este serviço pode ser acedido através da instalação da aplicação móvel. Após o seu registo, o utilizador pode receber notificações de ocorrências situadas na área onde ele se encontra, como se pode verificar na seguinte figura 4[4] [**36**].

[3] `https://play-lh.googleusercontent.com/7YUll24J08ODo91ysbDspxsJO3L8rm_ _2zLyL5kiw-moGSpZsvCRl9dcTW7Pbh3B7w=w1920-h966`
[4] `https://play-lh.googleusercontent.com/8rE8FSHF1-7z2FEOKI_gbas0qja6Sy58yoo5_ FPslp2pZuITI37GRy1xR7-Jx9PwUA=w2560-h955`

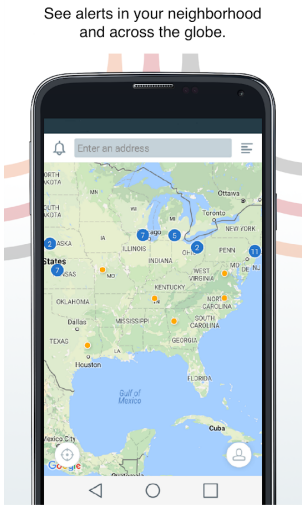

FIGURA 4. Everbridge - Alertas no Mapa

Estas ocorrências podem ser emergências, alertas de crime ou atualizações da comunidade.

Também é possível conectar-se com certas empresas ou universidades, para receber informação importante de algum tipo de evento que esteja a decorrer ou para gerir a resposta a incidentes que possam ocorrer [**37**].

Esta aplicação é também utilizada por universidades com o intuito de proteger os seus estudantes da atual epidemia Covid-19, como a UM (universidade de Mississippi) e USM (University of Southern Mississippi) [**38**]. Os alunos são submetidos a um questionário sobre os sintomas que tenham e, dependendo das suas respostas, recebem um *token* de aprovação ou rejeição, que têm de mostrar à entrada da instituição [**39**].

2.3.3. *iAuditor*

iAuditor é uma aplicação móvel que visa gerir incidentes em empresas. Isto é conseguido através da instalação da aplicação por parte de toda a equipa da empresa. Quando é reportado um incidente, é escolhido um conjunto de *tags* com o propósito de informar apenas as pessoas que precisam de tomar conhecimento do incidente. Também é feita uma descrição do acontecimento [**40**]. A aplicação também oferece resumos das ocorrências, mostrando quais foram as *tags* mais usadas e qual a severidade dos acontecimentos mais comuns.

2.3.4. Sintra Resolve

Sintra Resolve [**1**] é uma aplicação móvel para criar alertas para ocorrências na via pública aos agentes responsáveis da Câmara Municipal de Sintra.

Para a criação de uma ocorrência, o utilizador precisa de escolher o local e qual o tipo de ocorrência. As ocorrências podem ser do tipo "Higiene Pública", "Iluminação Pública" ou "Sinalização", entre outras. Para escolher o local, o utilizador tem à sua disposição um mapa onde pode identificar o local da ocorrência, tanto pela sua localização ou por uma

barra de pesquisa onde pode introduzir o nome do local. Pode, opcionalmente, adicionar uma pequena descrição e uma foto para ajudar a descrever o sucedido. Desde a criação até à sua resolução, a aplicação disponibiliza informação sobre o estado da ocorrência com o objetivo de o utilizador ficar a par da sua resolução.

2.3.5. Na Minha Rua LX

Na Minha Rua LX [**2**] é uma aplicação móvel e *web* que visa reportar ocorrências na via pública que necessitem intervenção da Câmara Municipal de Lisboa ou das Juntas de Freguesia deste município.

Para criar uma ocorrência, o utilizador precisa de escolher o local e o tipo de ocorrência, como, por exemplo, questões de saneamento, segurança pública e ruído ou iluminação pública, entre outros. O utilizador deve também adicionar uma descrição do problema em questão. Após a sua criação, o utilizador recebe uma mensagem no correio eletrónico, informando que a ocorrência foi criada, e fica também listado nos seus eventos criados.

As ocorrências criadas pelos utilizadores têm quatro estados a que podem estar associados:

Análise - A ocorrência que estiver neste estado ainda está à espera da confirmação por parte dos responsáveis para verificar se é realmente necessária a sua resolução.

Registado para resolução - As ocorrências já foram confirmadas e estão à espera da resolução por parte da entidade responsável.

Execução - Este tipo de ocorrências já se encontra em resolução.

Resolvida - Estas ocorrências já se encontram resolvidas.

O utilizador pode também visualizar ocorrências criadas por outros utilizadores e seguir o seu desenvolvimento, como se pode observar pela seguinte imagem[5] da figura 5. Quando uma ocorrência que é de interesse ao utilizador é resolvida, o utilizador recebe uma mensagem no seu correio eletrónico com essa informação.

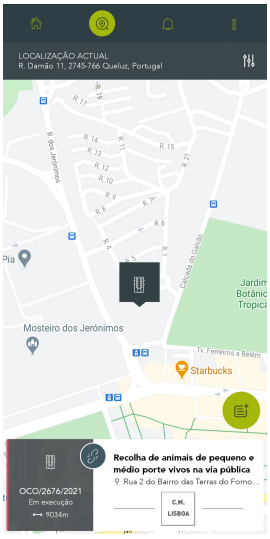

FIGURA 5. NaMinhaRuaLX - Mapa com uma ocorrência

[5]Captura de ecrã retirada da aplicação Na Minha Rua Lx

CAPíTULO 3

Desenvolvimento

Esta secção detalha como foi feita a aplicação móvel e como foram abordados os vários problemas que esta dissertação pretende resolver.

Vai ser também explicada e justificada a escolha das tecnologias utilizadas.

3.1. Base de Dados

A base de dados escolhida foi a solução orientada a documentos MongoDB. Apesar de se ter verificado na revisão de leitura que esta é a base de dados que fornece menos tipos de dados espaciais e menos funções, as que fornece são suficientes para o objetivo da aplicação.

Como foi referido na secção anterior, esta base de dados oferece um serviço de grande disponibilidade sendo esse um fator importante a ter em consideração tendo em conta que o objetivo desta dissertação é a construção duma aplicação que permita alojar um grande número de utilizadores.

Esta base de dados oferece também escalabilidade horizontal, conforme discutido na próxima secção, é uma funcionalidade importante para aplicações de larga escala, enquadrando-se assim no objetivo desta aplicação.

3.1.1. Escalabilidade

A escalabilidade define-se pelo "número de pedidos que uma aplicação consegue suportar eficientemente" [**41**]. Quando tal deixa de ser possível, está-se provavelmente numa situação em que o sistema chegou ao seu limite de processamento ou de memória, sendo assim necessário aumentar os recursos de computação do mesmo para conseguir funcionar eficientemente.

3.1.1.1. *Escalabilidade Horizontal vs Escalabilidade Vertical* Estas duas alternativas permitem aumentar a escalabilidade do sistema, porém as diferenças estão na maneira em como são adicionados estes recursos.

Para aumentar os recursos utilizando escalabilidade vertical, é necessário melhorar as características do servidor que aloja a base de dados. Estas melhorias podem ser feitas ao adicionar mais memória RAM, por exemplo. Em contrapartida, para aumentar os recursos utilizando escalabilidade horizontal, são adicionados mais servidores para alojar a mesma base de dados. Podem observar-se estas diferenças na figura 1[1] [**41**].

[1] `https://www.section.io/assets/images/blog/featured-images/horizontal-vs-vertical-scaling-diagram.png`

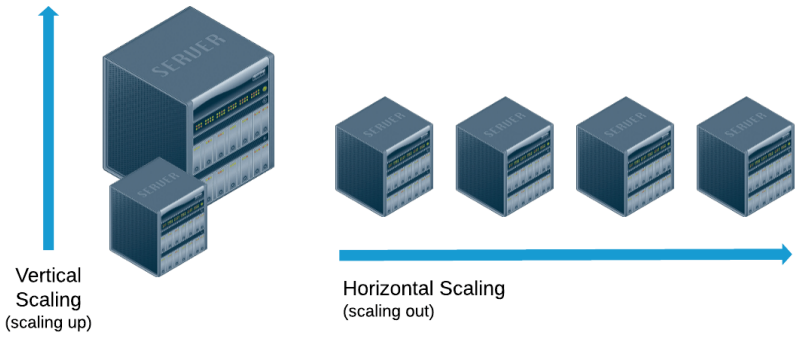

FIGURA 1. Escalabilidade Horizontal vs Vertical

Ambas as opções têm as suas vantagens e desvantagens, tendo de se fazer uma escolha para verificar qual a melhor opção para cada caso.

3.1.1.2. *Comparação dos benefícios de cada opção* Nesta secção vão ser comparadas, através de uma tabela, as diferenças entre escalabilidade horizontal e vertical.

	Escalabilidade Horizontal	Escalabilidade Vertical
Bases de Dados	Dados são divididos por vários servidores	Dados são todos armazenados no mesmo servidor
Indisponibilidade	< indisponibilidade	> indisponibilidade
Performance	Maior performance para grandes recursos	Mais em conta para um número mais pequeno de recursos
Distribuição geográfica	Pode reduzir a latência	Pode ter maior latência
Custo	> Custo	< Custo

TABELA 1. Escalabilidade Horizontal vs Vertical

A tabela 1 apresenta várias características diferentes destas duas abordagens à escalabilidade.

Como já demonstrado na secção 3.1.1.1, as bases de dados com escalabilidade horizontal apresentam um conjunto de máquinas entre as quais os dados são divididos, enquanto que na escalabilidade vertical os dados são todos armazenados na mesma máquina.

Em relação ao tempo de indisponibilidade da base de dados, as bases de dados que são administradas com vista a aumentar a escalabilidade horizontal oferecem redundância dos dados, visto que podem existir servidores com a mesma informação replicada. Assim, se um dos servidores deixar de funcionar, é sempre possível ir buscar a informação a outro servidor, promovendo desta forma um menor tempo de indisponibilidade. Isto não é possível em bases de dados geridas com vista em escalabilidade vertical, visto que a informação está toda contida no mesmo servidor.

Relativamente à performance, a escalabilidade horizontal oferece uma quantidade ilimitada no crescimento do número de utilizadores, enquanto que a utilização da escalabilidade vertical está dependente do poder do servidor em questão. Isto significa que para aplicações de larga escala, com um grande número de utilizadores, poderá haver um ponto onde será necessário optar por distribuir informação por vários servidores, dado que pode não ser viável a utilização de apenas um servidor.

Para uma aplicação que cubra uma vasta região geográfica, pode ser aconselhável que seja utilizado escalabilidade horizontal para reduzir a latência.

Por fim, em termos de custo, uma base de dados com escalabilidade vertical geralmente exige custos menores, dado que se recorre a uma única máquina. [**41**].

3.1.2. Contexto da Escolha da Base de Dados

O objetivo desta dissertação é fazer uma aplicação móvel para o utilizador criar alertas para ocorrências no espaço público. Isto faz com que seja necessário criar uma aplicação que esteja preparada para receber um grande número de utilizadores, sendo assim importante uma base de dados que ofereça alternativas para lidar com este tipo de situações, pretendendo oferecer escalabilidade horizontal.

Por esta razão e por interesse nas soluções deste tipo, foi escolhido utilizar MongoDB Atlas. Esta é uma base de dados na nuvem, ou seja, "um serviço de base de dados construído e acedido através de uma plataforma cloud" [**42**]. Estas soluções trazem a vantagem de não ser necessário comprar *hardware* dedicado para alojar a base de dados.

3.1.2.1. *MongoDB Atlas* O MongoDB Atlas é um serviço que oferece várias funcionalidades, tais como monitorização do sistema, disponibilizando ao utilizador um conjunto de métricas importantes sobre o estado dos servidores, com o objetivo de ser fácil verificar, se existe algum tipo de problema na base de dados. Oferece também um serviço com grande disponibilidade, existindo servidores que possuem dados redundantes de modo a, no evento de acontecer algum imprevisto ou algum tipo de manutenção, poder haver sempre uma alternativa para obter os mesmos [**43**].

Sendo assim, foi criada uma conta com um *cluster* de pacote "M0" grátis, que oferece um armazenamento de 512 MB (Megabyte) suficiente para o desenvolvimento do protótipo.

3.1.3. Ligação de uma aplicação móvel ao MongoDB Atlas

Para ligar uma aplicação móvel à MongoDB Atlas existem várias alternativas distintas, tendo sido estudadas duas em particular: utilizar o serviço que a MongoDB oferece (MongoDB Realm) ou criar um servidor utilizando NodeJS que faça a ligação entre a aplicação móvel e a base de dados.

Foi explorada também a opção de ligar diretamente a aplicação Android à base de dados, todavia esta opção mostrou-se pouco fiável visto que foram encontrados vários problemas com a ligação da aplicação à base de dados.

Assim, optou-se pela criação de um servidor em NodeJS que recebe os pedidos da aplicação, estando este conectado à base de dados. Não foi escolhida a utilização do serviço MongoDB Realm por este ser pago, apesar de poder ser feita uma utilização grátis se estiver dentro dos limites máximos de utilização que oferecem. Existiu também interesse em explorar a tecnologia NodeJS e comprovar que existem alternativas ao serviço pago da MongoDB.

O servidor foi desenvolvido e testado localmente, tendo sido utilizados os serviços da Azure para tornar o servidor acessível a todos os utilizadores. A Azure é "uma plataforma de computação em nuvem que permite aceder e gerir serviços de nuvem fornecidos pela Microsoft" [**44**]. Estes serviços em geral são pagos, existindo, contudo, opções gratuitas com o intuito de permitir o teste dos seus serviços. No caso desta dissertação, foi criada uma conta estudante que pode ser renovada anualmente, e que oferece um crédito de 100 dólares anuais [**45**].

Foi assim criado um serviço para alojar o servidor NodeJS e poder ser acedido publicamente.

Em relação ao servidor, foram utilizados vários recursos para tornar possível a comunicação entre a base de dados e a aplicação móvel, que se descrevem de seguida.

3.1.3.1. ***Express:*** "Estrutura flexível NodeJS que fornece um conjunto robusto de funcionalidades para desenvolver aplicações *web* e móveis" [**46**]. Com o Express torna-se possível que o servidor comunique com a aplicação móvel através de pedidos HTTP (*Hypertext Transfer Protocol*).

3.1.3.2. ***Mongoose:*** Biblioteca para NodeJS e MongoDB com o intuito de "gerir relações entre dados e fornecer validação de esquemas" [**47**]. Utilizando esta biblioteca fica mais fácil acrescentar lógica entre os documentos através da criação de modelos.

Visa assim assegurar ordem entre os documentos criados, visto que uma base de dados orientada a documentos não obriga a que tenham uma estrutura igual entre si, sendo fácil ficar desorganizado se não for imposto um modelo [**48**].

3.1.3.3. ***Retrofit*** O Retrofit é "um cliente de tipo REST (*Representational State Transfer*) para Android, Java e Kotlin desenvolvido pela Square. A biblioteca fornece uma estrutura poderosa para autenticar e interagir com APIs (*Application Programming Interface*) e enviar pedidos de rede com OkHttp." [**49**].

Tendo em conta a aplicação móvel que foi feita para a dissertação, o Retrofit é utilizado para fazer pedidos HTTP através da aplicação móvel para o servidor NodeJS, fazendo assim vários pedidos GET e POST quando é necessária alguma informação da base de dados ou alterar parâmetros.

Um exemplo desta troca de informações é o pedido de login do utilizador, em que a aplicação faz um pedido GET, fornecendo ao servidor o email e senha do utilizador que se está a tentar autenticar. O servidor está sempre à espera destes pedidos. Ao receber este pedido, o servidor faz uma busca na base de dados e verifica se existe um utilizador com esse mesmo endereço de email, caso exista, verifica se a senha corresponde à guardada na base de dados. Se isto tudo se verificar, o servidor envia de volta um código 200 e um objeto JSON com as informações do utilizador. Se ocorrer um erro, é enviado um código 404 ao utilizador, aparecendo uma mensagem de erro na aplicação móvel.

3.2. Lógica da Aplicação

A aplicação está baseada no diagrama apresentado na figura 2, em que existem duas classes principais, a classe Utilizador e a classe Evento.

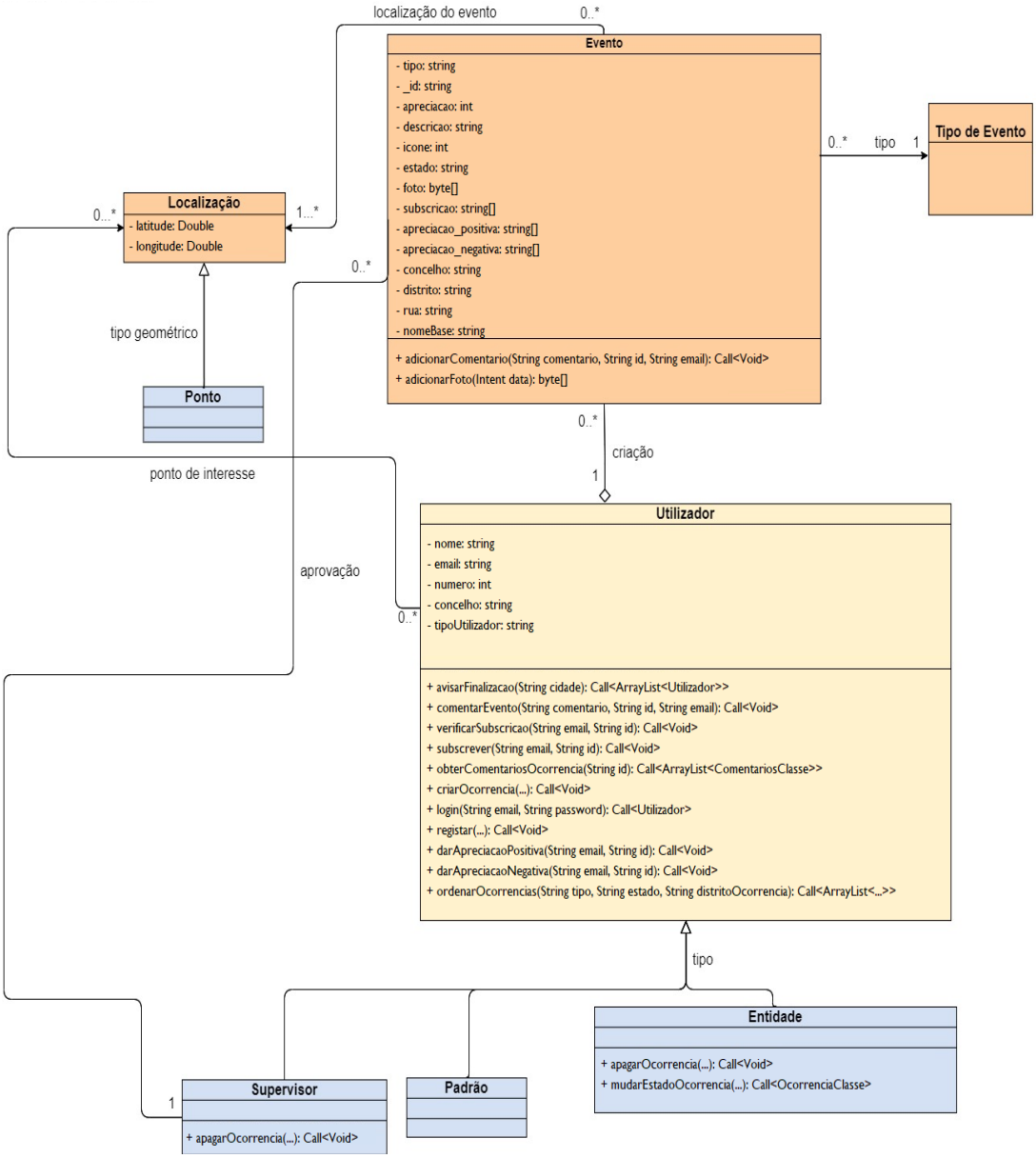

FIGURA 2. Diagrama de Classes da Aplicação

A classe Utilizador é uma superclasse que tem três subclasses: Supervisor, Padrão e Entidade, dado que existem três categorias de utilizadores que podem utilizar a aplicação com privilégios diferentes, como foi referido na secção 1.2.2. Estes três tipos de utilizador herdam os atributos e os métodos da superclasse Utilizador.

A classe Utilizador tem cinco atributos: o nome, o email, o id, o concelho e, por fim, o tipo de utilizador.

Estes valores são todos inseridos pelo utilizador, à exceção do id que é gerado automaticamente pelo MongoDB.

Esta classe tem um conjunto de métodos utilizados pelos três tipos de utilizadores, como, por exemplo, o login, o registo e o pedido de criar ocorrência. Os métodos que são apenas possíveis utilizar através de um utilizador especial estão incluídos na sua respetiva classe, como, por exemplo, a opção de apagar ocorrências criadas por outros utilizadores.

Em relação à classe Evento, esta possui um conjunto de atributos mais extenso.
O Evento contém:

(1) id - Identificador dos eventos criado automaticamente pelo MongoDB.
(2) tipo - Tipo de ocorrência (Grafitis, Candeeiro Danificado...).
(3) nomeBase - Categoria da ocorrência (Iluminação Pública, Higiene Urbana...).
(4) apreciação - Número de apreciações que o evento tem.
(5) descrição - Caracterização textual que o utilizador fez sobre a ocorrência.
(6) ícone - Ícone predefinido que está ligado a cada ocorrência.
(7) estado - Estado em que se encontra a ocorrência (Registado, Em execução...).
(8) foto - Foto que o utilizador pode associar à ocorrência.
(9) subscrição - Vetor com os utilizadores que subscreveram a ocorrência.
(10) apreciacao_positiva - Vetor com os utilizadores que deram uma apreciação positiva à ocorrência.
(11) apreciacao_negativa - Vetor com os utilizadores que deram uma apreciação negativa à ocorrência.
(12) concelho - Concelho da ocorrência.
(13) distrito - Distrito que se situa a ocorrência.
(14) rua - Nome da rua em que a ocorrência se situa.

Pode-se ver na figura 3 a continuação do diagrama, que mostra detalhadamente os tipos de eventos que existem.

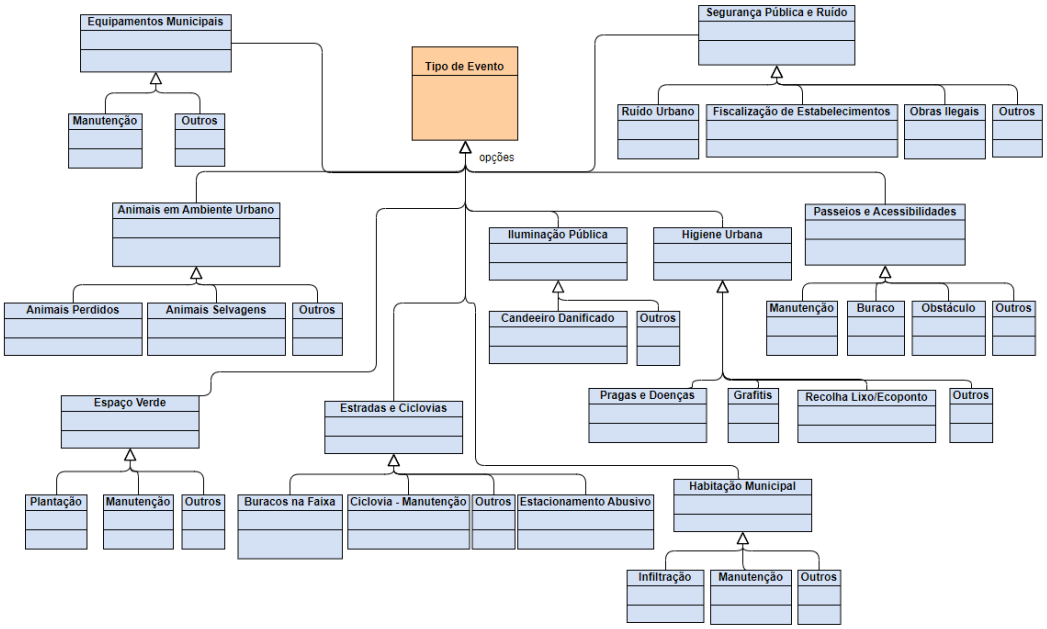

FIGURA 3. Diagrama Tipos de Evento

Consegue-se assim verificar as ocorrências que se podem criar, em que o "nomeBase" corresponde ao nome das categorias (como, por exemplo, Higiene Urbana) e o "tipo" corresponde às subclasses (tal como Grafitis).

3.3. Servidor NodeJS

Nesta secção vai ser ilustrada a utilização do servidor NodeJS. Vão ser também descritas as configurações do servidor, tanto para a sua ligação à base de dados MongoDB, como na utilização dos módulos "Mongoose" e "Express".

3.3.1. Modelos para o Mongoose

Como foi referido anteriormente, para utilizar o Mongoose é necessário impor um modelo às coleções na base de dados, tendo sido criadas na base de dados três coleções diferentes: a coleção de utilizadores, de eventos, e dos concelhos de Portugal.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const utilizadorSchema = new Schema( {
    nome: {
        type: String,
        required:false
    },
    email: {
        type: String,
        required: true
    },
    password: {
        type: String,
        required: true
    },
    tipoUtilizador: {
        type: String,
        required: true
    },
    concelho: {
        type: String,
        required: true
    }
}, { timestamps: true})

const Utilizador = mongoose.model('utilizadore', utilizadorSchema);
module.exports = Utilizador;
```

FIGURA 4. Modelo da coleção Utilizador

Na figura 4, pode visualizar-se o modelo dos documentos da informação dos utilizadores. São especificadas as informações necessárias para criar um utilizador, gerando assim uma coleção organizada de utilizadores.

Após serem definidos os atributos necessários, é preciso converter o esquema "utilizadorSchema" num modelo, sendo dado, ao primeiro argumento, o nome da coleção no singular e, como segundo argumento, o esquema para converter. Por fim, o modelo é exportado para poder ser utilizado. Os modelos do evento e dos concelhos de Portugal seguem o mesmo procedimento deste modelo.

3.3.2. Módulos e Ligação à Base de Dados

Como foi referido anteriormente, para a produção da aplicação foram utilizados alguns módulos para facilitar a criação da aplicação. Como se pode verificar pela figura 5, para adicionar estes módulos é necessário criar uma constante e utilizar a função "require" para

ler o ficheiro *JavaScript* em questão, e guardar nessa constante, para se poder utilizar as funcionalidades do mesmo.

```
const express = require('express')
const app = express()
const mongoose = require('mongoose');
const Utilizador = require('./models/utilizador');
const Evento = require('./models/evento');
const Concelho = require('./models/concelhos');

const hostname = process.env.HOST;
const port = process.env.PORT;

app.use(express.json({limit: '5mb'}));
const { MongoClient } = require("mongodb");
const uri =

mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true })
    .then((result) => app.listen(port, hostname) && console.log("Ligado no porto 3000..."))
    .catch((err) => console.log(err));
```

FIGURA 5. Módulos e Conexão à Base de Dados

Foi possível deste modo utilizar os módulos "Express" e "Mongoose", assim como os modelos "Utilizador", "Evento" e "Concelho" criados. Em relação à ligação da base de dados, foi utilizado um servidor Azure para alojar este servidor, sendo necessário guardar o *hostname* e *port* do servidor. Da parte dos servidores da base de dados MongoDB, é necessário utilizar a *string* de conexão do utilizador que pode ser obtido na conta pessoal do utilizador. Com isto, a aplicação liga-se à base de dados e fica à espera dos pedidos no servidor da Azure.

3.4. Pedidos ao Servidor

Nesta secção vão ser mostrados pedidos de informação relevantes do utilizador da aplicação, ao servidor NodeJS e à base de dados MongoDB.

Como descrito anteriormente, o utilizador faz pedidos a um servidor que vai buscar informação à base de dados e a devolve ao utilizador. Existem 24 pedidos diferentes que o utilizador pode fazer ao servidor, e parte destes vai ser detalhada de seguida.

3.4.1. Registo e Login do Utilizador

O utilizador começa por enviar a sua informação de registo, sendo esta guardada na constante "newUser". Depois é feita uma pesquisa na coleção da base de dados de utilizadores para verificar se existe algum utilizador com o endereço fornecido e, se já existir, é enviada uma mensagem de erro 406. Se não existir, é guardado na base de dados, sendo enviada uma mensagem 201 ao utilizador. Pode-se observar o código correspondente na figura 6.

Estas mensagens são respostas que o servidor envia ao cliente, a fim de informar se o pedido HTTP foi bem-sucedido. Existem cinco tipos de resposta HTTP diferentes e, neste caso, as mensagens do tipo 200 a 299 são respostas de sucesso e de 400 a 499 são respostas de erro.

Por conseguinte, a mensagem 201 representa uma mensagem de sucesso na criação do utilizador e a mensagem 406 representa uma mensagem erro, visto que já existe um utilizador com o mesmo endereço de email [**50**].

```
app.post('/signup', (req, res) => {

  const newUser = new Utilizador({
      nome: req.body.nome,
      email: req.body.email,
      password: req.body.password,
      tipoUtilizador: req.body.tipoUtilizador,
      concelho: req.body.concelho
  });

  Utilizador.count({email: newUser.email}, function (err, count){
      if(count>0){
          res.status(406).send();
          console.log("Enviado mensagem 406!")
      } else {
          newUser.save();
          res.status(201).send();
          console.log("Enviado mensagem 201!")
      }
  })
})
```

FIGURA 6. Código de registo do Utilizador

Por sua vez, a autenticação do utilizador é feita com o método "findOne()" a partir do qual é feita uma pesquisa na coleção dos utilizadores, para verificar se existe algum utilizador registado com o endereço eletrónico fornecido. Se já existir, é verificado se as palavras passe guardada e fornecida são iguais. Se forem, são devolvidos os dados ao utilizador e se diferirem é devolvida uma mensagem de erro.

São devolvidos os dados do utilizador quando é feita a sua autenticação, dado que estes vão ser utilizados pela aplicação posteriormente, não sendo assim necessário carregar o servidor com pedidos desnecessários.

3.4.2. Perfil do Utilizador

Para o utilizador consultar o seu perfil, é necessário fazer um pedido ao servidor enviando o seu endereço eletrónico, como se pode verificar na seguinte figura 7.

```
app.post('/getPrivateEvents', (req, res) => {
Evento.find({ $or:
    [
      {subcricao: { $elemMatch: {$eq: req.body.email} }},
      {email: req.body.email}
    ]}).select('-foto').then((eventos) => {
        res.status(200).send(eventos);
    }).catch((error) => {
        res.status(400).send(error);
    })
})
```

FIGURA 7. Código para o Perfil do Utilizador

É utilizado o método "find()", visto que o utilizador pode ter várias ocorrências no seu perfil.

Existem duas situações diferentes que resultam no aparecimento de uma ocorrência no perfil do utilizador: se a ocorrência for pessoal ou se o utilizador escolher seguir a ocorrência. Sendo assim, é feita uma pesquisa de todas as ocorrências que tenham o mesmo campo de email que o utilizador, ou todas as ocorrências que tenham o email do utilizador guardado no campo "subscrição".

A função "select()" serve para selecionar um campo específico do documento, neste caso o campo "foto". O '-' na função "select()" tem a funcionalidade de excluir os atributos referidos. Neste caso, esta opção vai fazer com que não seja enviada ao utilizador a foto da ocorrência.

Optou-se por descartar a foto, a fim de otimizar a velocidade da aplicação, dado que o campo foto tem um tamanho consideravelmente grande comparativamente ao resto dos dados. Se fosse enviada a foto com o resto do documento para todas as ocorrências, a aplicação tornar-se-ia muito mais lenta, principalmente se o utilizador subscrever muitas ocorrências. Independentemente deste problema, a foto da ocorrência só é necessária quando o utilizador pretender visualizar a mesma detalhadamente, sendo apenas nestas situações feito o pedido.

3.4.3. Filtragem de Ocorrências

Quando o utilizador quer pesquisar uma ocorrência, tem três escolhas possíveis. Pode escolher a categoria da ocorrência que quer pesquisar, o estado ou o concelho a que pertence. Para isto, foram criados 4 tipos de pedidos diferentes. O pedido feito ao servidor depende da escolha do utilizador sobre o que quer pesquisar.

Os quatro tipos de pedidos são:

(1) **/getEvents** Devolve todos os tipos e estados das ocorrências.
(2) **/getStateEvent** Devolve todas as ocorrências com o estado escolhido, como, por exemplo, todas as ocorrências que estão "Em Execução".
(3) **/getTypeEvent** Devolve todas as ocorrências com o tipo escolhido, como, por exemplo, todas as ocorrências sobre "Segurança Pública".
(4) **/getTypeStateEvent** Devolve todas as ocorrências com o tipo e estado escolhidos, como, por exemplo, todas as ocorrências sobre "Segurança Pública" que estão "Em Execução".

Em relação à sua localização, estes quatro métodos têm uma condição a verificar se o utilizador escolheu um concelho. Esta filtragem é feita no servidor, verificando se no pedido do utilizador a *string* concelho, que contém a localização, está vazia ou contém o nome de um concelho, como se pode verificar na figura 8.

```
app.post('/getStateEvent', (req, res) => {
    if(req.body.concelho == "") {
        Evento.find({estado: req.body.estado}).select('-foto').then((eventos) => {
            res.status(200).send(eventos);
        }).catch((error) => {
            res.status(400).send(error);
        })
    } else {
        Evento.find({estado: req.body.estado, localidade: req.body.concelho}).select('-foto').then((eventos) => {
            res.status(200).send(eventos);
        }).catch((error) => {
            res.status(400).send(error);
        })
    }

    })
```

FIGURA 8. Código Filtragem das Ocorrências - Servidor

O pedido feito é processado através de uma declaração "if-else", como se pode verificar na figura 9. Essa declaração tem quatro condições diferentes dependendo se o utilizador escolhe um estado ou um tipo de ocorrência. Na seguinte figura é apenas demonstrado uma destas condições, dado que as restantes seguem a mesma lógica.

```
if(!estado.isEmpty() && tipo.isEmpty()) {

    Call<ArrayList<Evento>> call = retrofitInterface.getStateEventsListGET(estado,concelhoOcorrencia);

    call.enqueue(new Callback<ArrayList<Evento>>() {
        @Override
        public void onResponse(Call<ArrayList<Evento>> call, Response<ArrayList<Evento>> response) {
            if (response.code() == 200) {
                result = response.body();
                buildRecyclerView(result);

            } else if (response.code() == 404) {
                Toast.makeText( context: Activity_Ocorrencias_Filtradas.this,  text: "Erro ao mostrar eventos",
                        Toast.LENGTH_LONG).show();
            }
        }

        @Override
        public void onFailure(Call<ArrayList<Evento>> call, Throwable t) {
            Toast.makeText( context: Activity_Ocorrencias_Filtradas.this, t.getMessage(),Toast.LENGTH_LONG).show();
        }
    });
```

FIGURA 9. Código Filtragem das Ocorrências

É criada uma estrutura para guardar a informação que é preciso enviar para o servidor. Como se pode verificar pela declaração "if", é verificado se o campo "estado" e "tipo" têm informação, sendo que, neste pedido, só o campo "estado" é que contém informação.

Sendo assim, a estrutura é preenchida com informação sobre o estado da ocorrência e da localidade, sendo esta de seguida enviada ao servidor. Por sua vez, o servidor devolve as ocorrências que correspondem aos pedidos do utilizador. Como se pode verificar na figura 9, este pedido apenas contém informação sobre o estado da ocorrência, correspondendo assim ao pedido "/getStateEvent".

3.4.4. Estatísticas de uma Localidade

Nesta parte da aplicação o utilizador escolhe uma localidade, retornando assim um novo ecrã, disponibilizando ao utilizador os tipos de ocorrências mais reportados de uma certa área.

Com este propósito, é feito um pedido ao servidor para devolver todos os eventos da localidade escolhida, sendo depois utilizado a biblioteca "MPAndroidChart" [**51**] que permite criar vários tipos de gráficos numa aplicação.

Desta forma determinam-se quantas ocorrências de cada tipo estão numa determinada localidade, passando para percentagem essa contagem e sendo criado um gráfico circular com esta informação, como se pode verificar na figura 10.

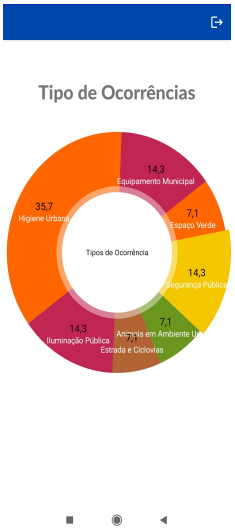

FIGURA 10. Gráfico com Informação sobre as Ocorrências

3.4.5. Remover e Alterar Ocorrências

Apenas os utilizadores especiais podem ter acesso a remover e alterar o estado das ocorrências, conseguindo apenas os utilizadores pertencentes a uma entidade fazer ambas as tarefas. Para isto, é necessário fazer uma verificação do tipo de utilizador para dar acesso a estas funcionalidades extra. É feita esta verificação em três alturas diferentes: na altura em que aparecem as ocorrências filtradas, no perfil do utilizador e na descrição detalhada das ocorrências.

Para verificar o tipo de utilizador, é feito um pedido ao servidor para fazer uma busca na base de dados para verificar e devolver o mesmo.

3.4.5.1. *Ocorrências Filtradas* Quando é feita uma verificação do tipo de utilizador e este é um utilizador com permissões especiais, é adicionado um ícone no canto inferior direito do ecrã, como se pode verificar na figura 11. Este ícone dá a opção ao utilizador de apagar as ocorrências dentro da sua área de supervisão.

Ao carregar nesse botão, o utilizador é reencaminhado para um segundo ecrã onde lhe aparecem todas as ocorrências que tem autoridade para apagar. Isto é possível dado que na autenticação do utilizador é devolvida a localidade do mesmo, sendo esta informação passada pelos vários ecrãs. Sendo assim, o utilizador fica com um ecrã com o aspeto da figura 12 após carregar no botão de apagar ocorrências da figura 11.

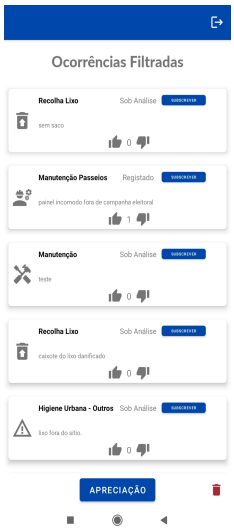

FIGURA 11. Ecrã Utilizador Especial - Ocorrências Filtradas 1

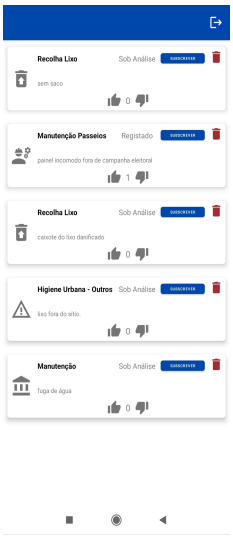

FIGURA 12. Ecrã Utilizador Especial - Ocorrências Filtradas 2

3.4.5.2. *Descrição Detalhada das Ocorrências* Este ecrã tem uma aparência diferente para os três tipos distintos de utilizadores. Para os utilizadores com permissões especiais, o ecrã é ligeiramente diferente, visto que estes utilizadores têm permissões para apagar a ocorrência ou mudar o seu estado. A funcionalidade de mudar os estados das ocorrências está guardado somente a utilizadores pertencentes a uma entidade.

Sendo assim, o ecrã tem o aspeto seguinte para um utilizador "Padrão" e para o utilizador "Entidade".

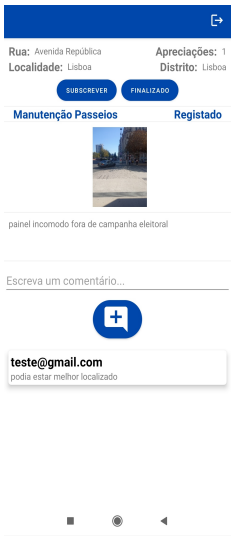

FIGURA 13. Ecrã Ocorrência Detalhada - Utilizador Padrão

FIGURA 14. Ecrã Ocorrência Detalhada - Utilizador Entidade

3.4.6. Aviso de Finalização de uma Ocorrência

No ecrã que possui a informação detalhada de uma ocorrência, o utilizador pode avisar os supervisores da localidade que a mesma se encontra resolvida ou mudar o estado para "Finalizado" se for uma ocorrência pessoal. É assim feito um pedido ao servidor que encontre todos os utilizadores do tipo "Supervisor" e que pertençam à localidade da ocorrência em questão, como se pode observar na figura 15.

```
app.get('/avisoFinalizadoGET', (req, res) => {
    Utilizador.find({tipoUtilizador: "Supervisor", concelho: req.query.localidade}).then((utilizadores) => {
        res.status(200).send(utilizadores);
    }).catch((error) => {
        res.status(400).send(error);
    })

})
```

FIGURA 15. Código - Aviso de Finalização

Isto vai devolver todos os supervisores da área, reencaminhado para o cliente de email do utilizador. Já irá ter um texto predefinido e endereços de email dos respetivos supervisores, como se pode visualizar pela figura 16.

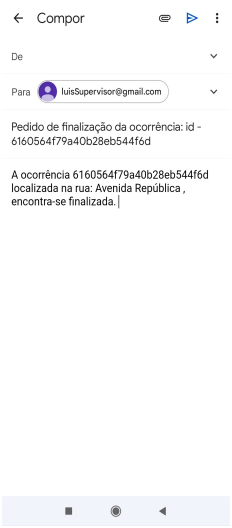

FIGURA 16. Mail de Finalização

3.4.7. Mapa da Aplicação

Foi escolhido a utilização de mapas da OpenStreetMap (OSM) para o utilizador poder navegar pela aplicação. Esta escolha foi feita porque os serviços da OSM são de utilização gratuita, ao contrário de outros fornecedores como a Google.

3.4.7.1. *Permissões* Para o utilizador poder utilizar o mapa com todas as funcionalidades tem de aceitar a permissão de localização.

O utilizador pode utilizar o mapa mesmo sem conceder esta permissão, porém irá perder a funcionalidade de ver a sua localização e de reposicionar o mapa na mesma.

Para facilitar o pedido de permissões ao utilizador e as várias opções do mesmo, foi utilizada a biblioteca "EasyPermissions" [**52**], que tem como objetivo facilitar o tratamento das mesmas.

Para tratar das permissões, primeiro é chamado o método "permissões()" em que é guardado num vetor de *strings* as permissões para pedir ao utilizador, e é verificado se as permissões já foram dadas. Se já tiverem sido dadas, é feita a obtenção da posição geográfica do utilizador. Se as permissões não tiverem sido dadas, o utilizador é informado de que não poderá utilizar a funcionalidade, e ser-lhe-á pedido, novamente, que aceite as permissões, como se pode observar na figura 17.

```
private void permissoes() {
    String[] perms = {Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION};
    if (EasyPermissions.hasPermissions( context: this, perms)) {
        localizacaoUtilizador = getLastKnownLocationteste();
    } else if(!EasyPermissions.hasPermissions( context: this,Manifest.permission.ACCESS_FINE_LOCATION)) {
        EasyPermissions.requestPermissions( host: this,  rationale: "Para ver a sua localização precisa de aceitar a permissão!",
                PERMISSAO_CODIGO, perms);
    }
}
```

FIGURA 17. Código - Permissões

Quando é negada uma permissão pelo utilizador, esse acontecimento é tratado através da função "onPermissionsDenied", como se pode verificar pela figura 18. São verificados assim os dois casos possíveis de negação, quando é negado uma permissão ou quando é negado uma permissão indeterminadamente. A diferença entre ambas rege-se apenas na mensagem que é enviada ao utilizador.

```
@Override
public void onPermissionsDenied(int requestCode, @NonNull List<String> perms) {
   if(EasyPermissions.somePermissionDenied( host: this,Manifest.permission.ACCESS_FINE_LOCATION)) {
       Toast.makeText( context: this, text: "Para ver a sua localização, tem de aceitar a permissão!",Toast.LENGTH_SHORT).show();
    }
   if(EasyPermissions.permissionPermanentlyDenied( host: this,Manifest.permission.ACCESS_FINE_LOCATION) && perms.contains(Manifest.permission.ACCESS_FINE_LOCATION)) {
       Toast.makeText( context: this, text: "Para poder ver a sua localização, por favor ative nas definições do telemóvel.",Toast.LENGTH_SHORT).show();
   }

}
```

FIGURA 18. Código - Negação da Permissão

3.4.7.2. *Marcadores* Se o utilizador escolher ver o mapa, este terá disponível um mapa cartográfico que estará posicionado na sua localização atual. Para o utilizador poder visualizar as ocorrências existentes no mapa, tem de introduzir um marcador através de um clique longo, e carregar de seguida no botão de procurar ocorrências. Isto faz com que sejam procuradas ocorrências num raio de 1000 metros do marcador que o utilizador introduziu no mapa, como se pode verificar nas figuras 19 e 20

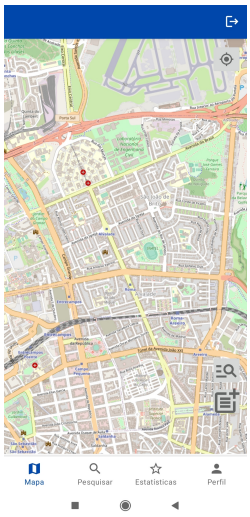

FIGURA 19. Ecrã- Sem Ocorrências

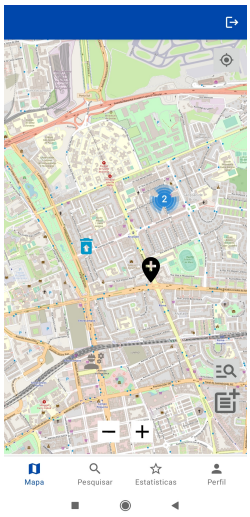

FIGURA 20. Ecrã- Com Ocorrências

Isto é possível através da utilização em conjunto dos operadores geoespaciais "centerSphere" e "geoWithin". O operador "geoWithin" tem como propósito selecionar todos os documentos que existem na coleção dentro de uma determinada área. Por sua vez, o operador "centerSphere" cria um círculo com o objetivo de fazer de área para a busca na base de dados [**53**]. Para determinar o raio do círculo, é preciso dividir a distância (em kilómetros) pelo raio da Terra (6378.1 kilómetros), dado que a função funciona com valores em radianos. Pode-se verificar o código na figura 21.

```
app.get('/getEventsArea', (req,res) => {
    latitude_num = Number(req.query.lat);
    longitude_num = Number(req.query.lon);
    raio = 6378.1;
    distancia = 1;
    Evento.find( {
        localizacao: { $geoWithin: { $centerSphere: [ [ longitude_num, latitude_num ], distancia/raio ] } }
     }, function(err, eventos) {
        res.status(200).send(eventos);
    });
})
```

FIGURA 21. Código - Obter Eventos à Volta do Utilizador

Este procedimento visa evitar faltas de desempenho que possam existir quando houver um grande número de marcadores no mapa. Isto torna-se principalmente importante numa aplicação onde se possa introduzir um elevado número de marcadores numa extensa área, como é o caso desta aplicação.

Contudo, esta solução, em conjunto com os *clusters* (que vão ser mencionados de seguida) pode não resolver o problema, uma vez que pode existir à mesma uma grande quantidade de ocorrências nessa zona. Porém, esta solução vem apaziguar problemas de desempenho no mapa, tornando-se, assim, numa aplicação mais preparada para receber uma maior quantidade de ocorrências. Ficou de ser testada esta funcionalidade pelos utilizadores, visto que foi implementada após os testes realizados.

Depois de o utilizador pesquisar por ocorrências, é feito um pedido ao servidor para devolver todas as ocorrências existentes nessa área na base de dados, transformando de

seguida essas mesmas ocorrências em marcadores para colocar no mapa, como se pode verificar na figura 22.

```
for(Evento ocorrencia : response.body()) {
    Drawable newMarker = MapaActivity.this.getResources().getDrawable(ocorrencia.getIcone(), theme: null );
    Localizacao localizacao = (Localizacao) ocorrencia.getLocalizacao();
    double[] coord = localizacao.getCoordinates();
    double latitude_ocor = coord[0];
    double longitude_ocor = coord[1];
    GeoPoint startPoint = new GeoPoint(latitude_ocor,longitude_ocor);
    Marker marcador = new Marker(map);
    marcador.setTitle(ocorrencia.getTipo());
    marcador.setSnippet(ocorrencia.getDescricao());
    marcador.setPosition(startPoint);
    marcador.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM);
    marcador.setIcon(newMarker);
    poiMarkers.add(marcador);
```

FIGURA 22. Código - Criação de Marcadores

Depois da criação dos marcadores e de preencher os marcadores com a informação da ocorrência, é adicionado cada marcador ao *cluster* de ocorrências, conforme explicado de seguida. Se o utilizador escolher reduzir o *zoom* do mapa e várias ocorrências se sobrepuserem, estas vão aparecer com um círculo azul e um número, que se designa de *cluster*, que representa o número de ocorrências que se encontram juntas. Isto é possível através da biblioteca "osmbonuspack" [**54**] que adiciona funcionalidades adicionais à biblioteca "osmdroid" [**55**]. Com ela, é possível criar estes *clusters* apenas com algumas linhas de código, como se pode verificar pela figura 23.

```
RadiusMarkerClusterer poiMarkers = new RadiusMarkerClusterer( ctx: MapaActivity.this);
Bitmap clusterIcon = BitmapFactory.decodeResource(MapaActivity.this.getResources(),R.drawable.marker_cluster);
poiMarkers.setIcon(clusterIcon);
```

FIGURA 23. Código - Cluster de Ocorrências

Ao clicar no *cluster*, o mapa faz automaticamente um "*zoom in*" até as ocorrências ficarem separadas.

A utilização de *clusters* mostra-se muito importante para uma aplicação deste tipo, visto que sem eles seria impraticável ter muitas ocorrências criadas na mesma área, dado que quando várias ocorrências se começam a sobrepor, fica inexequível a utilização do mapa.

3.4.7.3. *Guardar Estado do Mapa* Esta aplicação foi feita à volta de um mapa interativo com o objetivo não só de criar ocorrências, mas também de visualizar ocorrências criadas por outros utilizadores. Desta maneira, faz sentido ser guardado o estado do mapa sempre que é mudado de ecrã na aplicação ou sempre que é fechada a aplicação.

Para isto ser possível, foram utilizados a interface "*SharedPreferences*" e os métodos "*onStop()*" e "*onResume()*".

A interface "*SharedPreferences*" tem a funcionalidade de guardar no telemóvel do utilizador informação útil que pode ser utilizada posteriormente. Os métodos "*onStop()*" e "*onResume()*" são métodos que são executados sempre que é inicializado um novo ecrã.

Sendo assim, são guardadas as informações sobre o estado do mapa quando o utilizador muda para outro ecrã, para que quando volte a escolher visualizar o mapa, possa retornar à visualização anterior.

Pode-se assim visualizar pela figura 24, o código do método "*onStop()*", em que são guardadas as informações sobre o estado do mapa em *strings* predefinidas, que têm a função de fazer de chave.

```
@Override
protected void onStop() {
    final SharedPreferences.Editor edit = mPrefs.edit();
    edit.putString(TILE_PREF, map.getTileProvider().getTileSource().name());
    edit.putString(LATITUDE_PREF, String.valueOf(map.getMapCenter().getLatitude()));
    edit.putString(LONGITUDE_PREF, String.valueOf(map.getMapCenter().getLongitude()));
    edit.putFloat(ZOOM_PREF, (float) map.getZoomLevelDouble());
    edit.commit();

    map.onPause();
    super.onStop();
}
```

FIGURA 24. Código - Método onStop()

Quando o utilizador volta ao mapa, o método "*onResume()*" é executado, sendo assim lida a informação guardada nas chaves e abrindo o mapa no enquadramento em que o utilizador estava quando saiu a última vez que o utilizou.

```
@Override
public void onResume() {
    super.onResume();
    mLocationOverlay.enableMyLocation();
    final String tileSourceName = mPrefs.getString(TILE_PREF,
            TileSourceFactory.DEFAULT_TILE_SOURCE.name());
    try {
        final ITileSource tileSource = TileSourceFactory.getTileSource(tileSourceName);
        map.setTileSource(tileSource);
        lat_default = String.valueOf(localizacaoUtilizador.getLatitude());
        lon_default = String.valueOf(localizacaoUtilizador.getLongitude());
        Double lat_double = Double.parseDouble(mPrefs.getString(LATITUDE_PREF,lat_default));
        Double lon_double = Double.parseDouble(mPrefs.getString(LONGITUDE_PREF,lon_default));
        GeoPoint geoPoint_Guardado = new GeoPoint(lat_double,lon_double);

        mapController.setCenter(geoPoint_Guardado);
        map.getController().animateTo(geoPoint_Guardado);
        double zoom = mPrefs.getFloat(ZOOM_PREF, defValue: 19);
        mapController.zoomTo(zoom);

    } catch (final IllegalArgumentException e) {
        map.setTileSource(TileSourceFactory.DEFAULT_TILE_SOURCE);

    } catch (NullPointerException e) {
        lat_default = "38.7223";
        lon_default = "-9.1393";
    }
    map.onResume();
}
```

FIGURA 25. Código - Método onResume()

3.4.8. Criar Ocorrências

Para o utilizador criar uma ocorrência é necessário saber a localização da mesma. Para isto foi utilizado a classe "Geocoder" que nos permite fazer geocodificação inversa. A geocodificação inversa é o "processo oposto à geocodificação, e envolve a introdução das coordenadas geográficas para encontrar o endereço correspondente" [**56**]. As coordenadas

geográficas são convertidas num endereço incluindo o nome da cidade, rua e concelho. Isto é feito a partir da classe "Geocoder" como se pode observar na seguinte figura.

```
private void geocoder(GeoPoint p) {
    Geocoder gc = new Geocoder( context: MapaActivity.this);
    if(Geocoder.isPresent()){
        List<Address> list = null;
        try {
            list = gc.getFromLocation(p.getLatitude(), p.getLongitude(), maxResults: 1);
            Address address = list.get(0);
            StringBuffer str = new StringBuffer();
            str.append("Name: " + address.getLocality() + "\n");
            localidade = address.getLocality();
            str.append("Thoroughfare name: " + address.getThoroughfare() + "\n");
            rua = address.getThoroughfare();
            str.append("Admin Area: " + address.getAdminArea() + "\n");
            cidade = address.getAdminArea();
            String strAddress = str.toString();
            Log.d( tag: "geocoder", msg: "geocoder: cidade " + cidade);
        } catch (Exception e) {
            Toast.makeText( context: MapaActivity.this, text: "Ocorreu um erro na aplicação, por favor tente mais tarde.", Toast.LENGTH_SHORT).show();
        }
    }
}
```

FIGURA 26. Código - Geocodificação Inversa

Assim conseguimos obter várias informações sobre a localização da ocorrência, sendo guardadas as informações sobre o nome da rua, da cidade e do concelho. Porém, esta informação sobre o concelho está por várias vezes errada, surgindo, ao invés do nome do concelho, o nome duma localidade. Por exemplo, ocorrências situadas no concelho de Sintra podem surgir etiquetadas como Queluz.

Sendo assim, foi desenvolvida uma base de dados com as delimitações geográficas dos concelhos de Portugal [**57**]. Estes dados estão disponíveis através do formato "shapefile", o que é um problema visto que o MongoDB não tem capacidade de processar ficheiros deste tipo, tendo sido feita a conversão do ficheiro "shapefile" para um ficheiro JSON, através de um editor online chamado "mapshaper" [**58**]. Depois de importada e tratada esta informação, os documentos ficaram com a estrutura da figura 27.


```
_id: ObjectId("614b4880107d0edd3a500f4d")
localizacao: Object
    type: "Polygon"
  > coordinates: Array
properties: Object
    ID_0: 182
    ISO: "PRT"
    NAME_0: "Portugal"
    ID_1: 1
    NAME_1: "Évora"
    ID_2: 7
    NAME_2: "Mora"
    HASC_2: "PT.EV.MR"
    CCN_2: 0
    CCA_2: "0707"
    TYPE_2: "Concelho"
    ENGTYPE_2: "Municipality"
    NL_NAME_2: ""
    VARNAME_2: ""
```


FIGURA 27. Estrutura do Documento sobre Delimitação Geográfica de concelhos

Como se pode verificar, temos na variável "NAME_2" o nome do concelho a que pertence cada área e as coordenadas do mesmo. Com isto, pode-se fazer uma busca na coleção inteira e verificar a que concelho pertence as coordenadas da ocorrência. Foi necessário adicionar uma indexação na coleção, no campo "localização", do tipo "2dsphere". Esta indexação "suporta consultas que calculam geometrias numa esfera semelhante à do planeta terra." [**22**]

Sendo assim, para verificar a que concelho pertence a ocorrência criada pelo utilizador, é feito um pedido ao servidor com as coordenadas da mesma, em que este faz uma busca na coleção onde se encontra a informação sobre as delimitações geográficas dos concelhos, e é utilizado a função "geoIntersects" para devolver o documento com a informação a que corresponde. Pode-se consultar este processo na figura 28. Esta informação é posteriormente guardada na base de dados.

```
app.post('/local', (req,res) => {
    latitude_num = Number(req.body.lat);
    longitude_num = Number(req.body.lon);
    Concelho.findOne({
        localizacao: {
            "$geoIntersects": {
                "$geometry": {
                    "type": "Point",
                    "coordinates": [longitude_num,latitude_num]
                }
            }
        }
    }, function(err, concelho) {
        res.status(200).send(concelho);
    });

})
```

FIGURA 28. Código GeoIntersects

3.4.8.1. *Mail Entidade* Para fazer a criação de uma ocorrência, é necessário enviar um email ao endereço eletrónico correto para a entidade que vai tratar da ocorrência. Para isto, foram criados "HashMaps" com os diferentes tipo de mails (mail da polícia municipal, câmara municipal, entre outros) em que a chave do "HashMap" é o nome do concelho e o conteúdo é o mail respetivo. Depois para cada tipo de ocorrência é escolhido qual o "HashMap" necessário para retirar o mail da ocorrência.

Quando o utilizador escolher criar a ocorrência, é feito um pedido ao servidor para pesquisar quais são os supervisores do concelho respetivo, obtendo assim o seu endereço de mail. De seguida, é criada a ocorrência e gravada na base de dados, sendo também pedido ao utilizador para enviar um email à entidade e aos supervisores da área, podendo este recusar, sendo criada apenas a ocorrência.

Se o utilizador aceitar, é criado um mail predefinido com a informação da ocorrência e os destinatários, como se pode verificar nas figuras 29 e 30. Para verificar esta funcionalidade, foram criados endereços fictícios para concelhos diferentes e testado se o sistema conseguia obter os endereços corretos.

```java
public void enviarMail(String[] emailDestinatario) {
    String conteudo = descricao.getText().toString();

    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.putExtra(Intent.EXTRA_EMAIL, emailDestinatario);
    intent.putExtra(Intent.EXTRA_SUBJECT, tipoOcorrencia);
    intent.putExtra(Intent.EXTRA_TEXT, conteudo);

    intent.setType("message/rfc822");
    startActivity(Intent.createChooser(intent, title: "Escolha o seu mail"));

}
```

FIGURA 29. Código - Enviar Mail

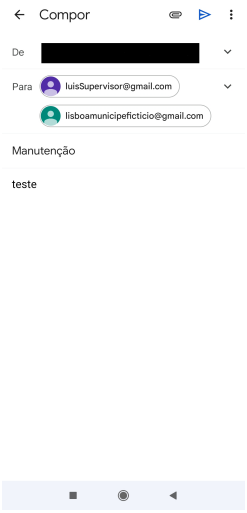

FIGURA 30. Enviar Mail

3.4.9. Dar Apreciação

Os utilizadores podem dar a sua apreciação a todas as ocorrências criadas. Para isto ser possível foram criados dois campos no documento, o campo "apreciacao_positiva" e "apreciacao_negativa". Estes campos são vetores que guardam os endereços eletrónicos dos utilizadores quando estes dão uma apreciação positiva ou negativa, respetivamente.

Porém, são precisas várias verificações para poder fazer uma apreciação, como se pode observar na figura 31.

```javascript
app.post('/verificarLike', (req, res) => {
    Evento.findById(req.body._id, function (err, evento) {
        const like_vetor = evento.get('apreciacao_positiva');
        if(like_vetor.includes(req.body.email)) {
            res.status(404).send();
        } else {
            const dislike_vetor = evento.get('apreciacao_negativa');
            if(dislike_vetor.includes(req.body.email)) {
                evento.apreciacao_negativa.pull(req.body.email);
                evento.apreciacao_positiva.push(req.body.email);
                const apreciacao = evento.get('apreciacao') + 2;
                evento.apreciacao = apreciacao;
                evento.save();
                res.status(200).send();
            } else {
                evento.apreciacao_positiva.push(req.body.email);
                const apreciacao = evento.get('apreciacao') + 1;
                evento.apreciacao = apreciacao;
                evento.save();
                res.status(200).send();
            }
        }
    });
})
```

FIGURA 31. Código - Dar Apreciação Positiva

Existem duas verificações antes de o utilizador poder dar a sua apreciação: primeiro, é necessário verificar se a apreciação já foi dada previamente, sendo que se já tiver sido

dada é enviada uma mensagem de erro ao utilizador. Isto é feito verificando se o seu endereço eletrónico está presente no campo "apreciação_positiva".

Se não tiver dado previamente, é preciso verificar se o utilizador deu anteriormente uma apreciação negativa. É preciso fazer esta verificação porque, neste caso, será necessário aumentar o valor da apreciação em dois valores em vez de apenas um e ainda retirar o endereço eletrónico do vetor "apreciação_negativa" e adicionar ao "apreciação_positiva". O pedido para fazer uma apreciação negativa segue a mesma lógica.

3.4.10. Subscrever a Ocorrência

Subscrever a ocorrências é uma funcionalidade importante da aplicação, visto que dá a oportunidade ao utilizador de seguir ocorrências que lhe interessem. Para isto ser possível, foi seguida uma lógica parecida à de dar apreciações.

Existe um campo com o nome "subscrição", que tem a finalidade de guardar os endereços eletrónicos dos utilizadores que pretendem seguir a ocorrência. É feita uma verificação a este campo sempre que o utilizador queira subscrever a uma ocorrência, como se pode observar na figura 32.

```
app.post('/subscrever', (req, res) => {
    Evento.findById(req.body._id, function (err, evento) {
        const vetor = evento.get('subcricao');
        if(vetor.includes(req.body.email)) {
            res.status(404).send();
        } else {
            evento.subcricao.push(req.body.email);
            evento.save();
            res.status(200).send();
        }
    } );
})
```

FIGURA 32. Código - Subscrever a uma Ocorrência

Existe uma verificação extra com o intuito de verificar se o utilizador que criou a ocorrência quer subscrever ocorrências pessoais. Porém, esta verificação é feita na própria aplicação antes de ser enviado o pedido ao utilizador, de modo a fazer o menor número de pedidos ao servidor possíveis. Esta verificação é possível, visto que quando o utilizador se autentica na aplicação, o seu endereço eletrónico é guardado, podendo assim ser usado para esta verificação.

É feita uma comparação entre o endereço eletrónico do utilizador que criou a ocorrência e o utilizador que quer subscrever à ocorrência, mostrando assim uma mensagem de erro se forem idênticos.

3.5. Ecrãs da Aplicação

Nesta secção vão ser mostrados os diversos ecrãs existentes na aplicação móvel e a justificação de cada componente existente em cada um deles.

3.5.1. Utilizador Padrão

Na seguinte figura pode-se verificar um diagrama sobre parte das opções que um utilizador padrão possui.

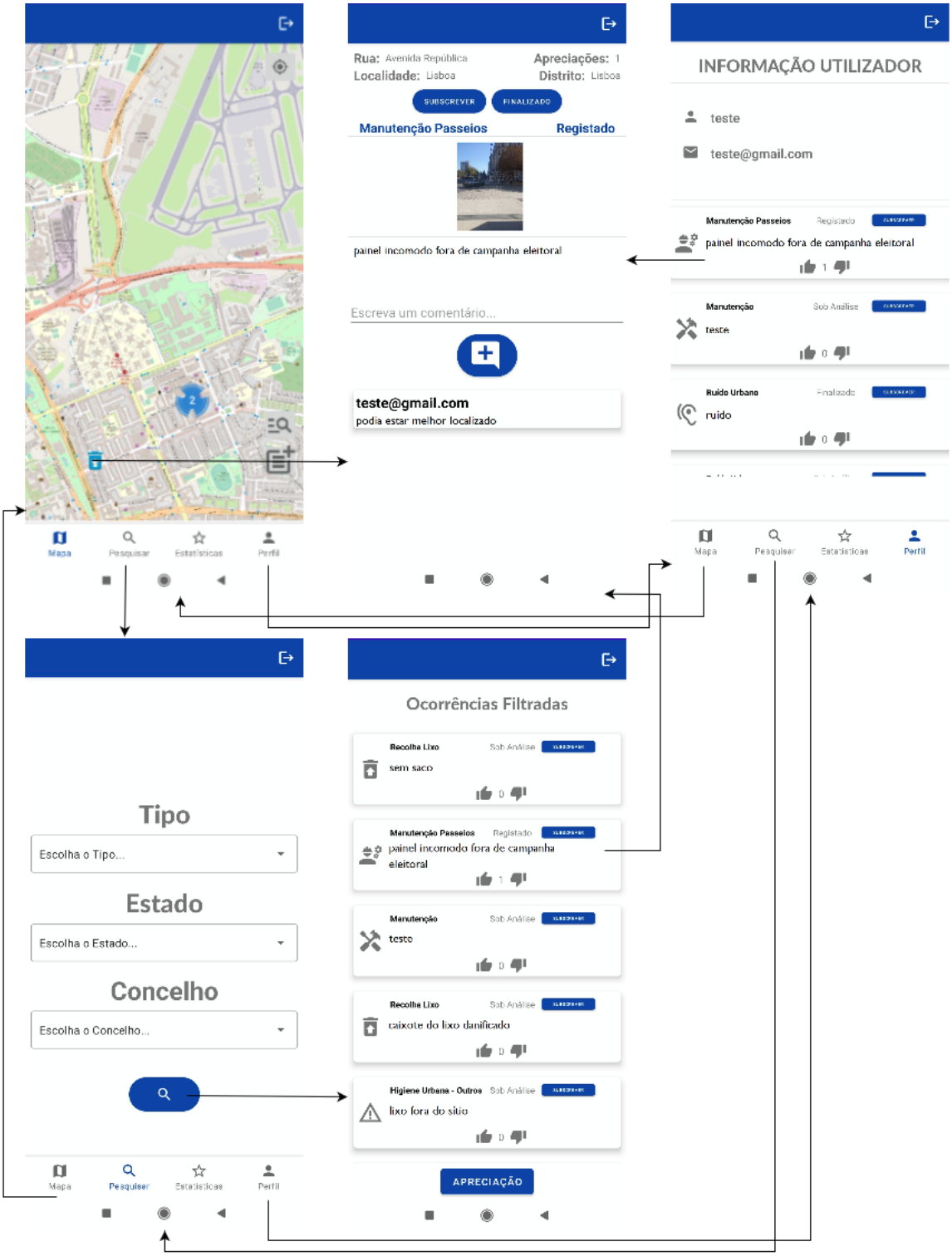

FIGURA 33. Diagrama - Ecrãs

Quando o utilizador abrir a aplicação, vai-lhe ser apresentado o ecrã inicial. São-lhe dadas três opções: fazer autenticação, registo ou entrar na aplicação sem conta. O

utilizador, ao escolher fazer o registo, tem de fornecer o nome, email, password, e o seu concelho.

Ao registar-se, o utilizador volta novamente ao ecrã inicial onde pode escolher fazer a autenticação na sua nova conta ou entrar como anónimo. Para fazer a autenticação, o utilizador terá de fornecer o seu email e a password respetiva.

Ao fazer a autenticação ou entrar como anónimo, o utilizador tem várias opções, ver o mapa, procurar por ocorrências, ver o seu perfil (se este estiver autenticado) ou, por fim, ver estatísticas sobre a cidade que tenha interesse.

Se o utilizador escolher ver o mapa, será exibido um mapa geográfico que estará posicionado na sua localização atual. O mesmo terá um conjunto de ícones a marcar as diferentes ocorrências que existem. Se o utilizador escolher reduzir o *zoom* do mapa e várias ocorrências se amontoarem, estas vão formar um *cluster*. Se o utilizador carregar no *cluster*, o mapa faz automaticamente *zoom* ao ponto das ocorrências já não ficarem juntas, como se pode verificar na figura 34.

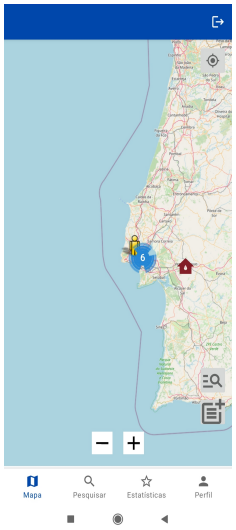

FIGURA 34. Cluster no Mapa

Quando o utilizador carrega num ícone de uma ocorrência, a aplicação abre a informação detalhada desta. Caso o criador tenha adicionado foto, esta será apresentada. Se não o tiver feito, aparecerá a imagem do ícone respetiva à ocorrência. A ocorrência também poderá ter uma descrição informativa e comentários, como se pode verificar no diagrama da figura 33.

Ao invés de visualizar detalhadamente uma ocorrência, o utilizador pode escolher criar uma nova ocorrência se estiver autenticado. Se o utilizador estiver a utilizar a aplicação como anónimo, irá aparecer uma mensagem a pedir para o utilizador se autenticar.

Ao criar uma nova ocorrência, o utilizador escolhe o tipo de que se trata, tendo a seguir que adicionar uma foto ou que descrever a situação. Ao criar a ocorrência, é perguntado ao utilizador se pretende enviar um email às entidades e supervisores competentes. Após a criação, o utilizador volta ao mapa e a ocorrência é criada e guardada na base de dados.

O utilizador pode também visitar o seu perfil, porém tem de estar também autenticado para o fazer. Se estiver, é-lhe exibido o seu nome, email e as ocorrências em que tem interesse e criadas por si. Ao carregar na ocorrência desejada, é direcionado para o ecrã com a informação detalhada sobre a ocorrência. Se o utilizador escolher procurar ocorrências, pode filtrar pelo tipo, estado, e concelho.

Ulteriormente, são mostrados os eventos respetivos, podendo o utilizador subscrever as ocorrências pelas quais tem interesse, dar a sua apreciação positiva ou negativa e filtrar pelo número de apreciações. Ao carregar na ocorrência desejada, o utilizador verá os detalhes sobre a ocorrência e os comentários associados à mesma. Também poderá aqui fazer a observação de que a ocorrência em questão já foi resolvida, enviando um email aos supervisores dessa área, que irão verificar a veracidade do pedido e mudar o estado da ocorrência para "Finalizado".

Por fim, o utilizador pode escolher ver estatísticas sobre uma cidade à escolha, aparecendo-lhe um gráfico circular com a percentagem de cada tipo de ocorrência que existe na mesma.

3.5.2. Utilizador Especial

O Supervisor e o utilizador Entidade têm funcionalidades adicionais, relativamente às do utilizador Padrão, como já foi referido nas secções anteriores. Ambos os utilizadores têm a opção de apagar ocorrências na sua zona de operação, sendo que o ecrã desses utilizadores difere ligeiramente dos utilizadores Padrão. Por fim, o utilizador Entidade tem a funcionalidade extra de mudar o estado da ocorrência quando existirem alterações no estado da mesma. Para fazer essa alteração, o utilizador precisa de ir para o ecrã em que tem a informação detalhada da ocorrência, onde terá a opção de alterar o seu estado.

CAPíTULO 4

Análise do desempenho da aplicação

4.1. Questionário

Foi preparado um questionário com o propósito de recolher o parecer de quinze utilizadores sobre a sua experiência de utilização da aplicação. Através deste questionário, foram identificados alguns pontos a melhorar na aplicação e também algumas funcionalidades extra que a aplicação poderá vir a ter.

As respostas permitiram constatar que a aplicação não funciona em alguns modelos de telemóveis com sistemas operativos mais antigos. A origem do problema está na biblioteca "Retrofit" que faz a comunicação com o servidor, visto que é sempre criada uma exceção quando a aplicação faz um pedido ao servidor. Isto seria um problema importante para resolver antes de ser lançada a aplicação ao público, uma vez que inibiria alguns possíveis utilizadores de utilizar a aplicação.

Vai ser de seguida descrita a opinião dos utilizadores sobre vários aspetos da aplicação.

4.1.1. Utilização da Aplicação

Um dos objetivos desta dissertação é fazer uma aplicação intuitiva que seja de fácil uso para os utilizadores. A opinião dos utilizadores que a testaram é apresentada na seguinte figura 1.

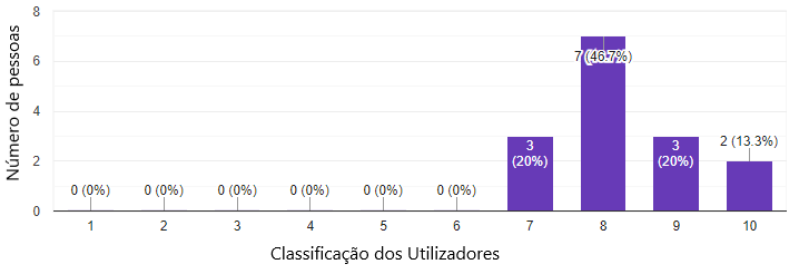

FIGURA 1. Utilização da Aplicação

Pode concluir-se que a aplicação, tendo em conta a opinião dos utilizadores que a experimentaram, é intuitiva e de fácil utilização no geral. Porém, pode ter algumas melhorias. Foi por isso pedido aos utilizadores para justificarem a sua opinião.

Uma das observações feitas pelos utilizadores indica que a criação de ocorrências não seria intuitiva, por ser feita através de um toque longo no mapa. Foi também apontado que a ocorrência podia ficar num local incorreto pela mesma razão.

Optou-se por tentar evitar este problema criando um marcador provisório quando é feito um toque longo, o que dá a possibilidade ao utilizador de o reposicionar no mapa para o local pretendido. Foi também criado um botão para confirmar a criação da ocorrência no mapa, após a criação do marcador, com o objetivo de ser clara a criação da mesma.

Foi observado também que, ao utilizar o gráfico circular, na funcionalidade das estatísticas das localidades, era preciso clicar no ecrã para o mesmo aparecer, tendo sido este problema posteriormente resolvido.

4.1.2. Aspeto Visual da Aplicação

Em relação ao aspeto visual da aplicação, foi perguntado aos utilizadores de que forma classificariam a aplicação em termos visuais, tendo sido obtido o resultado da figura 2.

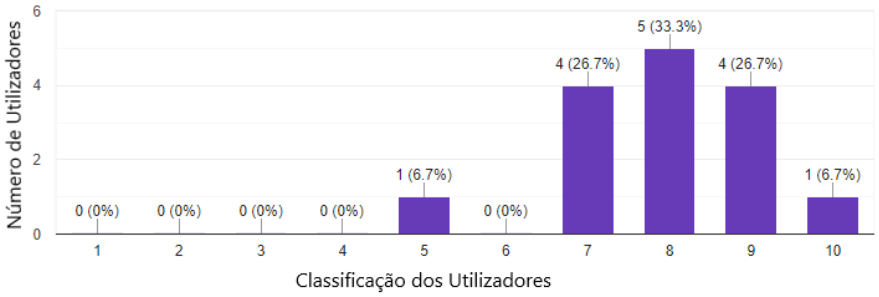

FIGURA 2. Aspeto Visual

Como se pode verificar pelo gráfico, a maioria dos utilizadores aprovou o visual da aplicação, tendo sido feitas algumas sugestões.

Foi repetidamente sugerido a introdução de um modo noturno para a aplicação, dado que uma grande quantidade de utilizadores prefere a utilização deste modo.

Foi também recomendada a adição do logo da aplicação nos vários ecrãs da aplicação e também que os botões de *dropdown* tivessem mais espaço em relação ao texto.

4.1.3. Funcionalidade Estatísticas

Foi perguntado aos utilizadores sobre a utilidade de visualizar estatísticas de localidades, tendo sido obtidas respostas conforme o gráfico da figura 3.

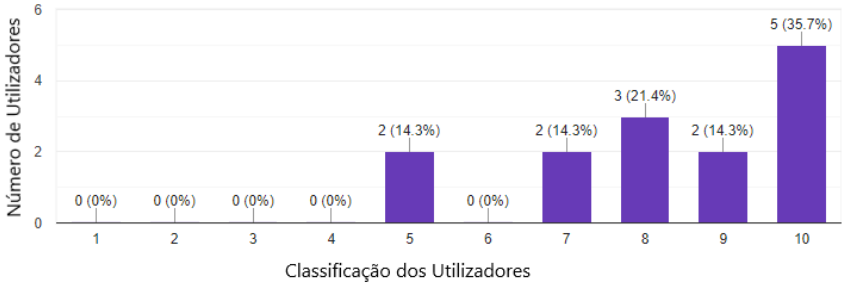

FIGURA 3. Funcionalidade Estatísticas

Para melhorar esta funcionalidade da aplicação foi sugerido que, ao clicar nas percentagens dos tipos de ocorrência, fosse mostrada ao utilizador uma lista das respetivas ocorrências. Foi também sugerido que existisse mais informação sobre estas, como o número total de ocorrências na área.

Foi também recomendada a criação de um relatório anual para todas as localidades, sobre as ocorrências e informação detalhada sobre as mesmas.

4.1.4. Rapidez e Fluidez

Foram feitas questões aos utilizadores sobre a fluidez e rapidez da aplicação, tendo sido obtidos os valores do seguinte gráfico da figura 4.

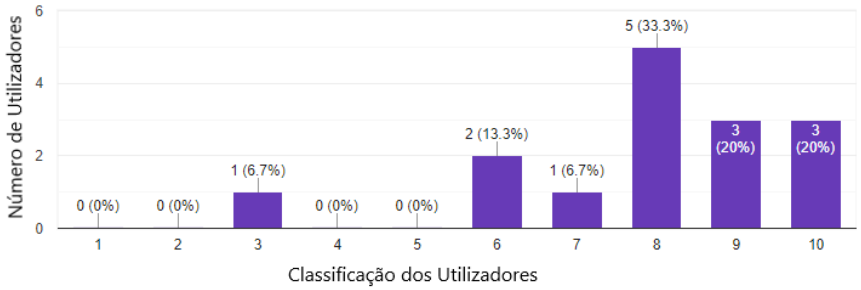

FIGURA 4. Fluidez da Aplicação

Foi pedido por alguns utilizadores uma maior rapidez na aplicação no geral. Ao experimentar a aplicação é notada alguma latência ao obter a foto da ocorrência, quando se visualiza a informação detalhada desta.

4.1.5. Ícones da Aplicação

Foi pedido aos utilizadores para darem a sua opinião sobre os ícones das ocorrências da aplicação, sendo os resultados apresentados no gráfico da figura 5.

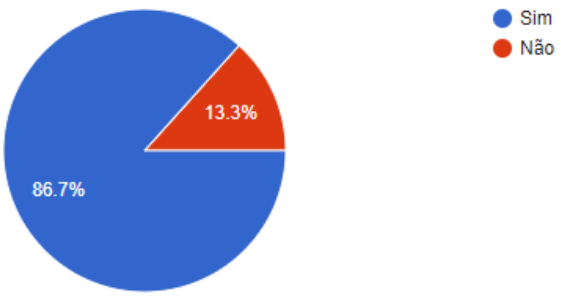

FIGURA 5. Ícones da Aplicação

Os ícones da aplicação foram bem recebidos pela maior parte dos utilizadores, como se pode verificar. Todavia, houve uma sugestão de alterar o ícone do tipo "Equipamento Municipal - Manutenção". Também foi sugerido que os ícones tivessem cores diferentes consoante o tipo de ocorrência, para ser mais fácil a identificação destes no mapa.

4.1.6. Opiniões Gerais sobre a Aplicação

Foram pedidas opiniões gerais sobre a aplicação, bem como se os utilizadores estão acostumados a utilizar aplicações móveis e se, consequentemente, usariam esta aplicação para reportar ocorrências. Todas as questões tiveram uma resposta positiva, como se pode visualizar pelas seguintes figuras.

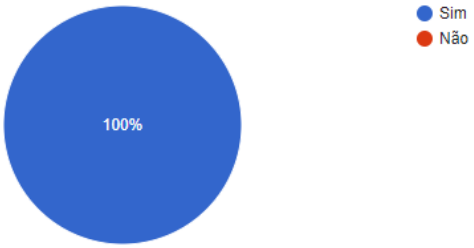

FIGURA 6. Utilização Aplicações Móveis

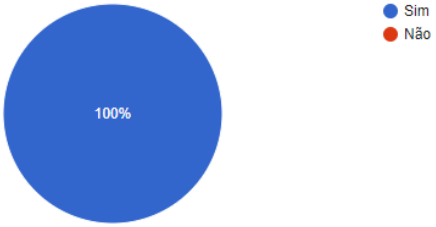

FIGURA 7. Utilizaria a Aplicação?

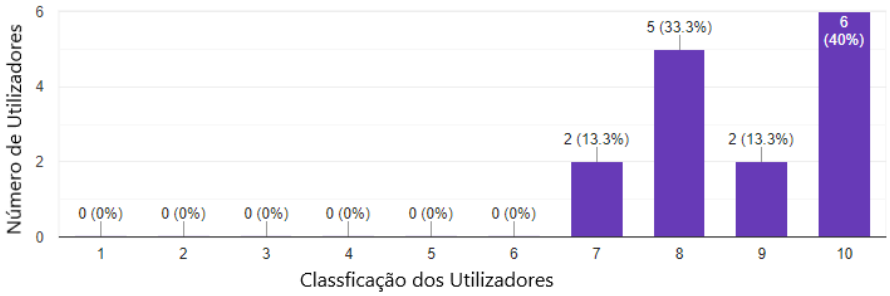

FIGURA 8. Utilidade da Aplicação

Em relação a funcionalidades extra e a alterações na aplicação, foram sugeridas pelos utilizadores as seguintes:

(1) Confirmação da conta através do endereço de email, para ser mais difícil a criação de contas falsas.
(2) Ao escolher um concelho, poder pesquisar diretamente o nome ou os concelhos estarem ordenados por distritos.
(3) Todas as ocorrências terem a sua data de criação.
(4) Na criação de uma ocorrência ter textos predefinidos para descrever ocorrências que ocorrem com mais frequência.
(5) Mais subtipos de ocorrências.
(6) Partilha de ocorrências com outros utilizadores.

4.2. Comparação entre Aplicações

Nesta secção vai ser feita uma comparação entre as aplicações "Na Minha Rua Lx", "Sintra Resolve" e a aplicação realizada nesta dissertação "OcorVia".

4.2.1. Menu Principal

Começando pelo *menu* principal das aplicações, ambas as aplicações estudadas optaram por apresentar ao utilizador o seu perfil com as ocorrências pessoais e que escolheram seguir. No caso da aplicação "Sinta Resolve" apenas as ocorrências pessoais são apresentadas.

Em relação à aplicação "OcorVia", foi escolhido que a aplicação seja centrada no mapa com o propósito de o utilizador ter a oportunidade de ter uma noção das ocorrências que existem próximas de si e ser mais rápido criar e visualizar ocorrências do seu interesse pessoal.

4.2.2. Perfil

Os perfis dos utilizadores das três aplicações são idênticos, porém o perfil mais completo é da "Na Minha Rua Lx", pois tem a funcionalidade de separar ocorrências pessoais, pendentes e as que o utilizador escolhe seguir.

Em relação à aplicação "Sintra Resolve", a aplicação não oferece a funcionalidade de seguir ocorrências de outros utilizadores, aparecendo apenas as ocorrências criadas pelo utilizador. Todavia tem a opção de criar rascunhos, que podem ser visualizados no seu perfil.

A aplicação desenvolvida nesta dissertação possui um perfil simples que mostra ao utilizador todas as suas ocorrências e as que tem interesse.

4.2.3. Mapa

Em relação aos mapas utilizados, a aplicação "Na Minha Rua Lx" e a produzida nesta dissertação têm a funcionalidade de observar ocorrências criadas por outros utilizadores através do mapa, sendo esta uma funcionalidade importante da qual a aplicação "Sintra Resolve" carece.

Tendo em conta a funcionalidade de observar ocorrências no mapa, a aplicação "Na Minha Rua Lx" dá a opção aos utilizadores de poderem filtrar a sua pesquisa de ocorrências no mapa, enquanto que a aplicação realizada nesta dissertação apenas oferece a opção de filtragem fora da utilização do mapa. Os utilizadores podem assim, por exemplo, filtrar que tipos de ocorrência pretendem visualizar diretamente no mapa.

Em relação ao desempenho do mapa, a aplicação "Na Minha Rua Lx" e a "OcorVia" só mostram ocorrências que estejam perto do local escolhido pelo utilizador, sendo que a aplicação "Na Minha Rua Lx" dá oportunidade ao utilizador de escolher um raio para visualizar as ocorrências à volta do sítio escolhido no mapa, até 500 metros, enquanto que na aplicação "OcorVia" o raio é fixo e de 1000 metros. Isto torna-se importante quando existe um número extenso de ocorrências, porque pode causar lentidão aquando da apresentação do mapa.

A aplicação "Sintra Resolve" tem a funcionalidade de os utilizadores poderem pesquisar no mapa por uma localidade, enquanto que nas outras duas aplicações é preciso procurar através da navegação do mapa.

4.2.4. Apreciação Final

Após o estudo inicial sobre as aplicações móveis e as suas limitações, conseguiu-se implementar as funcionalidades importantes que diferenciam a aplicação criada nesta dissertação com as aplicações estudadas.

Começando pela distribuição geográfica e como foi referido na introdução do trabalho, as aplicações estudadas apenas cobrem o seu concelho respetivo o que traz problemas aos utilizadores que queiram reportar ocorrências em concelhos distintos, tendo que instalar várias aplicações para poderem reportar e visualizar ocorrências ou então podem ficar sem a capacidade de reportar ocorrências desta forma.

Outra funcionalidade importante que se distingue de ambas as aplicações reside no facto destas não possuírem um espaço onde os utilizadores possam comunicar entre si. Isto torna-se um problema no sentido em que podem existir ocorrências nas quais os utilizadores tenham opiniões diferentes sobre a sua resolução, tornando-se assim numa mais-valia a existência de um sítio onde estes possam comunicar entre si na aplicação e a entidade possa ter isso em consideração.

A aplicação criada para gerir ocorrências na via pública "OcorVia" correu como esperado em relação às funcionalidades estipuladas inicialmente na proposta da dissertação. Porém, foram encontrados algumas limitações, sendo a mais relevante a incapacidade de utilização da aplicação em telemóveis mais antigos, o que incapacita alguns possíveis utilizadores de utilizar a aplicação.

Em relação a funcionalidades extra interessantes para a aplicação, seria positivo a introdução de uma filtragem de ocorrências do utilizador no próprio mapa da aplicação ao invés de apenas fora do mapa deste.

Na figura 9 pode-se visualizar uma tabela a comparar as três aplicações.

	Na Minha Rua Lx	Sintra Resolve	OcorVia
Centralizado no Mapa	×	×	✓
Reposicionar na posição do Utilizador	✓	×	✓
Comentar Ocorrências	×	×	✓
Criar ocorrências em todo o território continental	×	×	✓
Seguir Ocorrências	✓	×	✓
Separar ocorrências pessoais doutros utilizadores	✓	×	×
Criar Rascunhos	×	✓	×
Visualizar ocorrências no mapa	✓	×	✓
Filtrar ocorrências no mapa	✓	×	×
Mostrar ocorrências apenas perto do utilizador	✓	×	✓
Pesquisar no mapa uma localidade	×	✓	×
Visualizar Estatísticas sobre localidades	×	×	✓

FIGURA 9. Tabela - Comparar Aplicações

CAPíTULO 5

Conclusão

Foi criada uma aplicação móvel Android com o objetivo de tornar a gestão de ocorrências no espaço público mais fácil e simples, com vantagens relativamente a opções que já existiam previamente, e que apresentam vários problemas como a sua limitação geográfica e a sua falta de transparência para com os utilizadores sobre o estado das ocorrências.

Para resolver estas limitações, foi desenvolvida uma aplicação que cobre todo o território continental nacional através da criação de uma base de dados com as delimitações geográficas dos concelhos e com capacidade de descobrir, através das coordenadas da ocorrência, qual o município a que pertence cada ocorrência.

Para tratar do problema da falta de comunicação entre o estado e as estratégias de resolução das ocorrências, foi criado um espaço onde os utilizadores podem comunicar entre si, podendo tanto as entidades responsáveis informar os utilizadores interessados dos avanços e estratégias utilizadas para resolver as ocorrências, como os utilizadores darem a sua opinião sobre as mesmas.

Foram estudadas várias bases de dados, tendo sido escolhida a MongoDB para esta aplicação, devido à sua possibilidade de escalabilidade horizontal, que é um fator importante a considerar visto ser necessário alojar um grande número de utilizadores.

5.1. Trabalho Futuro

Ficaram alguns aspetos por melhorar na aplicação móvel, nomeadamente sugestões de utilizadores que experimentaram a aplicação, bem como problemas identificados durante a testagem da aplicação.

Começando pelo mais importante, no lançamento da aplicação ao público seria desejável que a informação de registo do utilizador fosse encriptada, com o objetivo de proteger a privacidade do utilizador.

Outro ponto importante seria a privacidade dos utilizadores aquando da criação de ocorrências com fotos anexadas. Seria uma quebra importante de privacidade se, por exemplo, ao criar uma ocorrência um utilizador fotografasse a matrícula de um carro ou a face de uma pessoa. Uma solução para este tipo de situações seria arranjar maneira de detetar matrículas ou caras de pessoas e aplicar um efeito de desfocagem. Porém, nesta fase de desenvolvimento, estes aspetos são secundários.

Por fim, em relação a funcionalidades extra, seria interessante adicionar as seguintes funcionalidades:

(1) O utilizador poder receber ocorrências no seu perfil sempre que é adicionado um evento novo no seu concelho, ao invés de apenas serem apresentadas no perfil ocorrências que o utilizador especificamente escolheu seguir ou tenha criado.
(2) Poder adicionar mais que uma foto sobre a ocorrência, e o ecrã de informação detalhada sobre a ocorrência ter uma opção para visualizar melhor as fotos da mesma.
(3) Adição de um modo noturno para os utilizadores.
(4) Tornar mais completa a funcionalidade de visualizar estatísticas sobre as localidades, mostrando informação mais detalhada sobre as mesmas.

5.2. Apreciação Final

A dissertação realizada atingiu os objetivos determinados de início, tendo sido implementada a grande maioria das funcionalidades mais importantes. Faltaram apenas algumas funcionalidades secundárias que poderão ficar para trabalho futuro.

A utilização da base de dados MongoDB decorreu sem incidentes, tendo sido fácil a aprendizagem e a utilização da mesma, assim como a criação de um servidor em NodeJS para lidar com os pedidos do utilizador e a utilização dos serviços Azure. Relativamente à aplicação móvel Android, já existiam algumas bases e experiência antes da realização da dissertação, o que facilitou a criação da aplicação.

Na versão final da aplicação móvel foram implementadas as funcionalidades mais importantes, tendo ficado algumas questões mais secundárias por implementar, como seguir zonas escolhidas pelo utilizador, seguir utilizadores, e receber notificações quando são feitas alterações às ocorrências seguidas.

Os utilizadores que experimentaram a aplicação avaliaram positivamente a mesma. Foram feitas várias sugestões para serem adicionadas novas funcionalidades à aplicação, tendo sido algumas implementadas e outras menos urgentes deixadas para trabalho futuro.

Foi alterada a forma de adicionar uma nova ocorrência, para que o utilizador tenha a certeza da localização da ocorrência criada e, também, para tornar a sua introdução mais intuitiva. Adiciona-se a isto a implementação de cores diferentes para ocorrências de tipos diferentes.

No geral os objetivos foram atingidos e as apreciações dos utilizadores foram positivas, tendo sido assim criada uma aplicação funcional e atrativa para gerir ocorrências no espaço público.

Referências Bibliográficas

[1] "Sintra Resolve." [Online]. Available: http://www.sintraresolve.pt/

[2] "naminharuaLX GOPI Gestão e Pedidos de Intervenção de Ocorrências em Lisboa." [Online]. Available: https://naminharualx.cm-lisboa.pt/

[3] I. Alexandre, *Design Science Research*, 2020.

[4] "What is GIS? | Geographic Information System Mapping Technology." [Online]. Available: https://www.esri.com/en-us/what-is-gis/overview

[5] N. G. Society, "GIS (Geographic Information System)," Jun. 2017. [Online]. Available: http://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/

[6] "How GIS works." [Online]. Available: https://www.geomatic.ma/en/arcgis/sig/how-gis-works

[7] "2. Introduction — Introduction to PostGIS." [Online]. Available: https://postgis.net/workshops/postgis-intro/introduction.html

[8] "DM-66 - Spatial Indexing | GIS&T Body of Knowledge." [Online]. Available: https://gistbok.ucgis.org/bok-topics/spatial-indexing

[9] "15. Spatial Indexing — Introduction to PostGIS." [Online]. Available: https://postgis.net/workshops/postgis-intro/indexing.html

[10] "26. 3-D — Introduction to PostGIS." [Online]. Available: https://postgis.net/workshops/postgis-intro/3d.html

[11] "Chapter 4. Using PostGIS: Data Management and Queries." [Online]. Available: https://postgis.net/docs/using_postgis_dbmanagement.html#idm2459

[12] A. Makris, K. Tserpes, G. Spiliopoulos, and D. Anagnostopoulos, "Performance Evaluation of MongoDB and PostgreSQL for spatio-temporal data," p. 9.

[13] "ST_intersects." [Online]. Available: https://postgis.net/docs/ST_Intersects.html

[14] "Chapter 5. PostGIS Reference." [Online]. Available: https://postgis.net/docs/reference.html

[15] "CARTO introduction — CartoDB Platform Documentation 1.0.0 documentation." [Online]. Available: https://cartodb.readthedocs.io/en/latest/intro.html

[16] CARTO, "How CARTO works." [Online]. Available: https://carto.com/help/getting-started/how-carto-works/

[17] "MongoDB vs MySQL." [Online]. Available: https://www.mongodb.com/compare/mongodb-mysql

[18] "What Is MongoDB?" [Online]. Available: https://www.mongodb.com/what-is-mongodb

[19] "Introduction to MongoDB — MongoDB Manual." [Online]. Available: https://docs.mongodb.com/manual/introduction

[20] A. Fowler, *NoSQL For Dummies*. John Wiley Sons, 2015.

[21] "GeoJSON Objects — MongoDB Manual." [Online]. Available: https://docs.mongodb.com/manual/reference/geojson

[22] "Geospatial Queries — MongoDB Manual." [Online]. Available: https://docs.mongodb.com/manual/geospatial-queries

[23] "About MariaDB Server." [Online]. Available: https://mariadb.org/about/

[24] "Geometry Types." [Online]. Available: https://mariadb.com/kb/en/geometry-types/

[25] "MySQL/MariaDB Spatial Support Matrix - MariaDB Knowledge Base." [Online]. Available: https://mariadb.com/kb/en/mysqlmariadb-spatial-support-matrix/

[26] “SPATIAL INDEX.” [Online]. Available: https://mariadb.com/kb/en/spatial-index/
[27] “SQLite Is Serverless.” [Online]. Available: https://www.sqlite.org/serverless.html
[28] “SpatiaLite manual.” [Online]. Available: http://www.gaia-gis.it/gaia-sins/spatialite-manual-2.3.1.html#t2
[29] “SpatiaLite SQL functions reference list.” [Online]. Available: https://www.gaia-gis.it/gaia-sins/spatialite-sql-4.3.0.html
[30] “Chapter 5. PostGIS Reference.” [Online]. Available: https://postgis.net/docs/reference.html
[31] “Geographic Functions.” [Online]. Available: https://mariadb.com/kb/en/geographic-functions/
[32] “GeoJSON Objects — MongoDB Manual.” [Online]. Available: https://docs.mongodb.com/manual/reference/geojson/
[33] “SpatiaLite SQL functions reference list.” [Online]. Available: http://www.gaia-gis.it/gaia-sins/spatialite-sql-4.2.0.html#p12
[34] “Mobile Incident Report and Management Software.” [Online]. Available: https://1stincidentreporting.com/
[35] “1st Incident Reporting - Apps on Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.emAPPetizer.first&hl=en&gl=US
[36] “Everbridge – Apps no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.everbridge.mobile.iv.recipient&hl=pt_PT&gl=US
[37] “Mobile Safety Applications – Critical Event Management.” [Online]. Available: https://www.everbridge.com/products/mobile-apps/
[38] M. Minta, M. T. August 31, and 2021, “How are Mississippi’s public universities tracking COVID cases?” Aug. 2021. [Online]. Available: https://mississippitoday.org/2021/08/31/how-are-mississippis-public-universities-tracking-covid-cases/
[39] “Everbridge App and Symptom Checker - Hostos Community College.” [Online]. Available: https://www.hostos.cuny.edu/Ready/SafeCampus/Campus-Health-Safety-Practices/Daily-Symptom-Checker-App-(Everbridge)
[40] “Real-time incident reporting.” [Online]. Available: https://safetyculture.com/spotlight/
[41] “Scaling Horizontally vs. Scaling Vertically.” [Online]. Available: https://www.section.io/blog/scaling-horizontally-vs-vertically/
[42] “What is a cloud database?” Apr. 2021. [Online]. Available: https://www.ibm.com/cloud/learn/what-is-cloud-database
[43] “MongoDB Atlas FAQ.” [Online]. Available: https://www.mongodb.com/cloud/atlas/faq
[44] “What is Azure and How Does It Work?” [Online]. Available: https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure
[45] “Azure for Students – Free Account Credit | Microsoft Azure.” [Online]. Available: https://azure.microsoft.com/en-us/free/students/
[46] “Node.js - Express Framework - Tutorialspoint.” [Online]. Available: https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm
[47] “Introduction to Mongoose for MongoDB,” Feb. 2018. [Online]. Available: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/
[48] “Do You Need Mongoose When Developing NodeJS and MongoDB Applications?” [Online]. Available: https://www.mongodb.com/developer/article/mongoose-versus-nodejs-driver/
[49] “Consuming APIs with Retrofit | CodePath Android Cliffnotes.” [Online]. Available: https://guides.codepath.com/android/consuming-apis-with-retrofit
[50] “HTTP response status codes - HTTP | MDN.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#client_error_responses

[51] P. Jahoda, "License ," Oct. 2021, original-date: 2014-04-25T14:29:47Z. [Online]. Available: https://github.com/PhilJay/MPAndroidChart
[52] "EasyPermissions," Oct. 2021, original-date: 2015-12-22T18:05:53Z. [Online]. Available: https://github.com/googlesamples/easypermissions
[53] "Geospatial Query Operators — MongoDB Manual." [Online]. Available: https://docs.mongodb.com/manual/reference/operator/query-geospatial/
[54] MKer, "About OSMBonusPack," Sep. 2021, original-date: 2015-03-26T21:16:51Z. [Online]. Available: https://github.com/MKergall/osmbonuspack
[55] "osmdroid," Sep. 2021, original-date: 2014-03-06T19:03:48Z. [Online]. Available: https://github.com/osmdroid/osmdroid
[56] "Overview | Geocoding API | Google Developers." [Online]. Available: https://developers.google.com/maps/documentation/geocoding/overview
[57] "Concelhos de Portugal - dados.gov.pt - Portal de dados abertos da Administração Pública." [Online]. Available: https://dados.gov.pt/en/datasets/concelhos-de-portugal/
[58] "mapshaper." [Online]. Available: https://mapshaper.org/

APÊNDICE A

Manual de Utilização

Nesta secção encontra-se o manual de utilização para facilitar a utilização da aplicação.

A.1. Instalação da aplicação

1 - Ao descarregar o ficheiro da figura 1, carregue no ficheiro e instale a aplicação.

FIGURA 1. Ficheiro de instalação

2 - Ao instalar podem aparecer avisos sobre a segurança da aplicação, porém ignore e proceda à instalação.

A.2. Login e Registo

1 - Ao chegar ao ecrã inicial carregue no botão de "Registo" para começar o registo da sua conta pessoal.

FIGURA 2. Ecrã Inicial

2 - Preencha com os seus dados pessoais e clique no botão registar.

3 - Clique no botão de "Login" e insira os dados para entrar na sua conta. Terá também a opção de entrar como utilizador anónimo, porém nesse modo não terá acesso a todas as funcionalidades da aplicação.

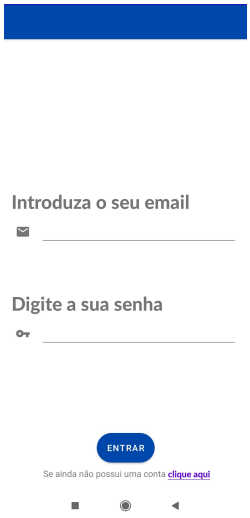

FIGURA 3. Ecrã de Login

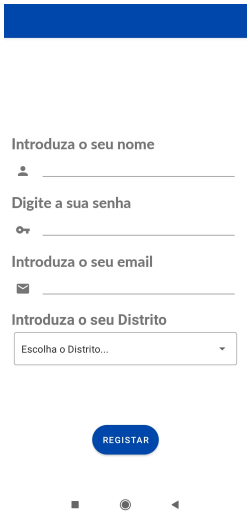

FIGURA 4. Ecrã de Registo

A.3. Criar Ocorrência

1 - Carregue no botão "Mapa"

2 - De seguida, faça um clique longo no mapa onde a ocorrência está localizada.

3 - Escolha qual o tipo de ocorrência e escreva uma descrição sobre a mesma. Pode também adicionar uma foto se desejar.

FIGURA 5. Ecrã Mapa

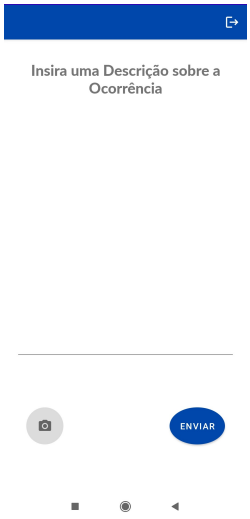

FIGURA 6. Adicionar Ocorrência

A.4. Pesquisar/Comentar/Apreciar Ocorrências

1 - Carregue no botão "Pesquisar" e insira a categoria, estado ou localização da ocorrência que está à procura e, por fim, clique no botão de pesquisa.

FIGURA 7. Ecrã Pesquisar

2 - Carregue na ocorrência em que tem interesse e veja os detalhes da mesma. Neste mesmo ecrã pode também dar a sua apreciação às ocorrências desejadas.

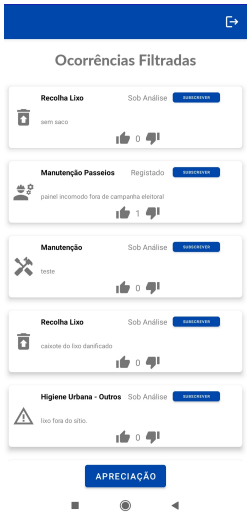

FIGURA 8. Ecrã Filtrado

FIGURA 9. Ecrã Detalhado

3 - De seguida pode adicionar o seu comentário à ocorrência, como se pode verificar na figura 9.

A.5. Subscrição de Ocorrências

1 - Pesquise a ocorrência que deseja seguir, e carregue no botão "Subscrever", como se pode verificar na figura 8 e 9.

2 - Clique no botão "Perfil" na barra de navegação para visualizar as suas ocorrências com interesse.

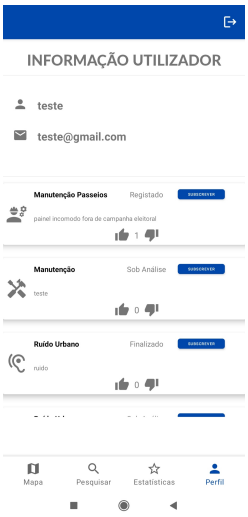

FIGURA 10. Ecrã Perfil

A.6. Ver Estatísticas de uma Localidade

1 - Clique no botão "Estatísticas" na barra de navegação.

2 - Escolha qual o concelho que pretende visualizar.

FIGURA 11. Ecrã Estatisticas1

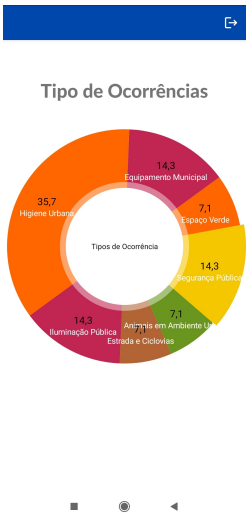

FIGURA 12. Ecrã Estatisticas2

A.7. Avisar que uma Ocorrência se encontra resolvida

1 - Pesquise a ocorrência em questão e clique na ocorrência em questão.

2 - Clique no botão "Finalizado" e envie um email aos supervisores da área.

APÊNDICE B

Questionário

Costuma utilizar aplicações móveis?

- ○ Sim
- ○ Não

Classifique de acordo se a aplicação é intuitiva e de fácil utilização *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Sugestões em como melhorar a aplicação em relação à classificação anterior

De 1 a 10, pensa que a aplicação é útil? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

De 1 a 10, quanto classifica em relação à fluidez e rapidez da aplicação? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Como avalia a funcionalidade de criar ocorrências?

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Classifique em relação ao aspeto visual da aplicação *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Melhorias no aspeto visual da aplicação

Como avalia a funcionalidade de visualizar as estatísticas de cada localidade?

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Pensa que os ícones são percetíveis e intuitivos? *

- ○ Sim
- ○ Não

Se respondeu "Não" na pergunta anterior, justifique

Usaria a aplicação para reportar ocorrências na via pública? *

- ○ Sim
- ○ Não

Se respondeu "Não" na pergunta anterior, justifique.

Que funcionalidades extra gostava que fossem implementadas?

Que mais categorias de ocorrências gostaria que fossem implementadas?