Relatório de Projeto

Telmo Miguel Amaral Salvado

Engenharia Informática

fev | 2023

GUARDA
POLI
TÉCNICO

POLI TÉCNICO GUARDA

Escola Superior de Tecnologia e Gestão

RENT A HOUSE

PROJETO APLICADO
PARA OBTENÇÃO DO GRAU DE LICENCIADO(A) EM ENGENHARIA INFORMÁTICA

Telmo Salvado
Fevereiro / 2023

Escola Superior de Tecnologia e Gestão

RENT A HOUSE

PROJETO APLICADO
PARA OBTENÇÃO DO GRAU DE LICENCIADO(A) EM ENGENHARIA INFORMÁTICA

Professor(a) Orientador(a): Noel Lopes

Telmo Salvado

Fevereiro / 2023

Agradecimentos

Primeiramente, queria agradecer ao meu professor orientador, Professor Noel Lopes, por toda a disponibilidade demonstrada ao longo do projeto.

Agradeço também ao diretor de curso, professor José Fonseca, por toda a disponibilidade e por todo o esclarecimento de dúvidas, não só ao longo do curso, como também ao longo do projeto.

Agradeço também a todos os outros professores que transmitiram o seu conhecimento de forma a conseguir atingir este objetivo também.

Por fim, um agradecimento especial à minha família por todo o apoio dado durante estes anos.

Ficha de Identificação

Aluno

Nome: Telmo Miguel Amaral Salvado

Número: 1700712

Licenciatura: Engenharia Informática

Estabelecimento de Ensino

Instituto Politécnico da Guarda (IPG)

Escola Superior de Tecnologia e Gestão (ESTG)

Docente Orientador

Nome: Noel Lopes

Grau Académico: Doutoramento

Resumo

O presente documento descreve o processo de desenvolvimento do projeto, realizado no âmbito da unidade curricular Projeto Informática, integrado na Licenciatura em Engenharia Informática, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

Com o constante crescimento do turismo e com o aumento pela procura de alojamento para passar férias ou fim de semana, e com o crescimento do chamado "alojamento local", por vezes fica difícil encontrar de uma forma simples e rápida um alojamento.

O presente projeto consiste no desenvolvimento de uma aplicação móvel, em plataforma Android, cujo principal objetivo é criar anúncios e reservas a alojamentos, apresentado também pontos de interesse presentes nessa cidade.

No desenvolvimento do projeto foi utilizado a metodologia de desenvolvimento ágil SCRUM. A aplicação foi desenvolvida em *Android Studio IDE* utilizando a linguagem Java para programação. Foi o usado o *Firebase* como o recurso para a gestão de base de dados e autenticação, e por fim, *Google Maps* para identificar a localização dos alojamentos, bem como dos locais de interesse.

Palavras-Chaves: Aplicação móvel, Turismo, *Android*, *Java*, *Firebase*, Alojamento, Pontos de interesse

Abstract

This paper describes the development process of this project, which was carried out in the context of the course unit Informatics Project, part of the bachelor's degree in Computer Engineering, of the School of Technology and Management of Politécnico da Guarda.

With ever growing tourism and increase in demand for vacation and weekend lodging, coupled with the growth of the so-called "local accommodations", finding a place to stay quickly and easily can sometimes become a challenge.

The aim of this project is to develop a mobile app, on an Android platform, where the main objective is to create listings and bookings of housing, as well as showcasing points of interest in the city.

The SCRUM agile development methodology was used to develop this project. The app was developed in Android Studio IDE using the Java language for coding. Firebase was used as a resource for database management and authentication, and lastly, Google Maps was used to pinpoint the locations of the accommodations, as well as the points of interest.

Keywords: Mobile App, Tourism, Android, Java, Firebase, Accommodation, Points of interest

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

O presente relatório descreve o projeto desenvolvido pelo aluno Telmo Salvado, no âmbito da Unidade Curricular Projeto de Informática, integrado na Licenciatura em Engenharia Informática.

O projeto consiste no desenvolvimento de uma aplicação móvel, na plataforma *Android*, com a funcionalidade de criação de reservas e anúncios de quartos ou apartamentos locais, com o objetivo de ajudar utilizadores a encontrarem ou a oferecerem alojamento.

A aplicação foi intitulada de "Rent a House".

1.1 Motivação

Em Portugal, a pandemia causou grandes quebras a nível de turismo, sendo que muitas unidades acabaram por ter de fechar portas. O impacto neste setor foi imediato, devido às medidas que obrigavam ao confinamento e ao distanciamento social [7]. Em 2022, o setor do alojamento turístico registou 3,0 milhões de hóspedes. Face a julho de 2019, registaram-se aumentos de 6,3%. [14]

O Alojamento Local (AL) tem vindo a ter um impacto positivo na economia. Este veio diversificar a oferta turística em Portugal, criou novos segmentos e novos públicos, não só nas grandes cidades, mas também no interior do país.[15]

Neste sentido, face a este aumento do número de turistas em Portugal e ao constante crescimento do Alojamento Local, resolveu-se desenvolver uma aplicação que permitisse criar anúncios de alojamentos e também fazer reservas dos mesmos. A aplicação é útil para qualquer pessoa que queira fazer registar a seu anúncio ou apenas procurar alojamento para as suas férias.

1.2 Objetivos

Este projeto tem como objetivo a implementação de uma aplicação móvel que possibilite o registo de oferta de alojamentos, por parte dos arrendatários, e a criação de reservas, por parte do cliente. A aplicação deverá cumprir os seguintes objetivos:

1. Sistema de autenticação de utilizadores;
2. Guardar e gerir a informação relativa a reservas e anúncios;
3. Pesquisa e filtro de dados;
4. Fazer uso do Google Maps API para apresentar a localização dos apartamentos.

1.3 Estrutura do Relatório

O presente relatório é constituído por cinco capítulos. No primeiro capítulo é feita uma introdução que descreve a motivação e os objetivos do projeto. No segundo capítulo apresenta-se uma pesquisa e análise de aplicações semelhantes, já existentes no mercado, com o objetivo de identificar requisitos e possíveis diferenças, relativamente à aplicação a desenvolver. No terceiro capítulo, temos a descrição da metodologia aplicada ao longo do projeto e seguidamente apresenta-se todo o processo de desenvolvimento do sistema, a engenharia de software do projeto detalhando requisitos e apresentando a modulação dos dados do sistema. No quarto capitulo, temos uma divisão em duas partes. A primeira aborda as ferramentas e as tecnologias utilizadas para o desenvolvimento do projeto. Na segunda parte, temos presentes algumas interfaces da aplicação e o seu funcionamento, bem como os testes que foram realizados ao longo do projeto. E, no quinto e o último capitulo, temos presente a conclusão sobre a realização, aprendizagem e dificuldades do projeto.

2 Estado da Arte

Uma vez definidos os objetivos do projeto, foi necessário realizar uma pesquisa e uma análise sobre possíveis aplicações que já existem, no mercado informático, com funcionalidades idênticas ao projeto que propomos implementar. A análise de outras aplicações tem como objetivo adquirir possíveis novas informações que visam ajudar no desenvolvimento do projeto.

2.1 Apresentação de Aplicações Existentes no Mercado

Após algumas pesquisas na Internet foram identificadas diversas aplicações com características idênticas aos requisitos definidos para o projeto. As aplicações são de livre acesso, pelo que foi possível recolher algumas informações sobre as suas funcionalidades e que nos permitiram compreender melhor como estas funcionam. Destas foram selecionadas duas, Booking.com e Airbnb, sendo elas aplicações de reserva de alojamento. Foram também selecionadas duas aplicações móveis de reserva de apartamentos, sendo elas a Trulia Rentals e a Mitula Imóveis.

2.1.1 Booking.com

A Booking.com, foi fundada em 1996, em Amesterdão, e cresceu de uma pequena start-up holandesa para uma empresa de viagens digitais de renome mundial. A grande missão é "facilitar a todas as pessoas a possibilidade de explorar o mundo".[4] Esta está disponível em 43 idiomas e oferece diversos tipos de alojamento, desde casas, apartamentos e outro tipo de alojamentos únicos.

Esta conta também com um serviço de marcação de voos, sugestão de atrações baseando-se no local a visitar e serviços de aluguer de carros. O utilizador pode registar a sua propriedade que pretende alugar, mas também pode fazer reservas. É possível fazer pesquisas por locais, data de entrada, número de pessoas, entre outras.

A Figura 1 mostra a interface da aplicação.

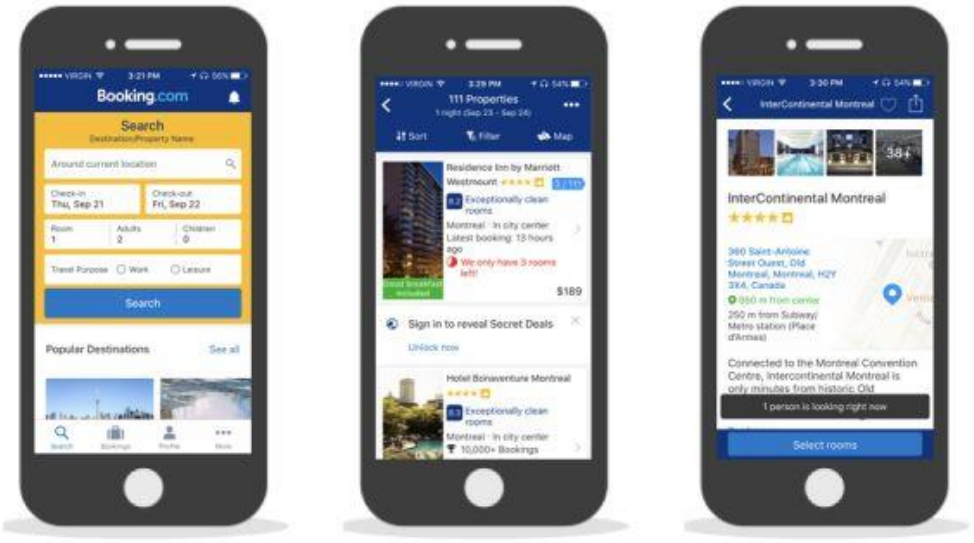

Figura 1- Interface da aplicação do Booking

2.1.2 Airbnb

A Airbnb [5] nasceu em 2008 quando dois designers que tinham um espaço, na sua habitação, hospedaram três viajantes que estava à procura de alojamento. Hoje, milhões de anfitriões e viajantes escolhem criar uma conta gratuita na Airbnb para poder anunciar o seu espaço ou para reservar alojamentos.

A plataforma da Airbnb funciona como uma rede social. Esta, conecta pessoas interessadas em encontrar um local para dormir e anfitriões com alojamento disponível. É necessário criar perfil para podermos utilizar esta aplicação.

Conforme fazemos reservas ou recebemos pessoas, vamos recebendo avaliações. Tanto hóspedes como anfitriões podem ser avaliados, isto é, para o hóspede uma boa avaliação facilita a que o próximo anfitrião o aceite. Já para o anfitrião, as avaliações ajudam na divulgação.

Em suma, a Airbnb apresenta as seguintes funcionalidades:

- Registo de utilizadores como anunciante ou hospede;
- Criação de anúncios por parte do anunciante;
- Criação de reservas por parte do hospede;
- Sistema de avaliação;
- Localização no mapa dos alojamentos;
- Sistema de pesquisa.

A Figura 2 representa a interface da Airbnb.

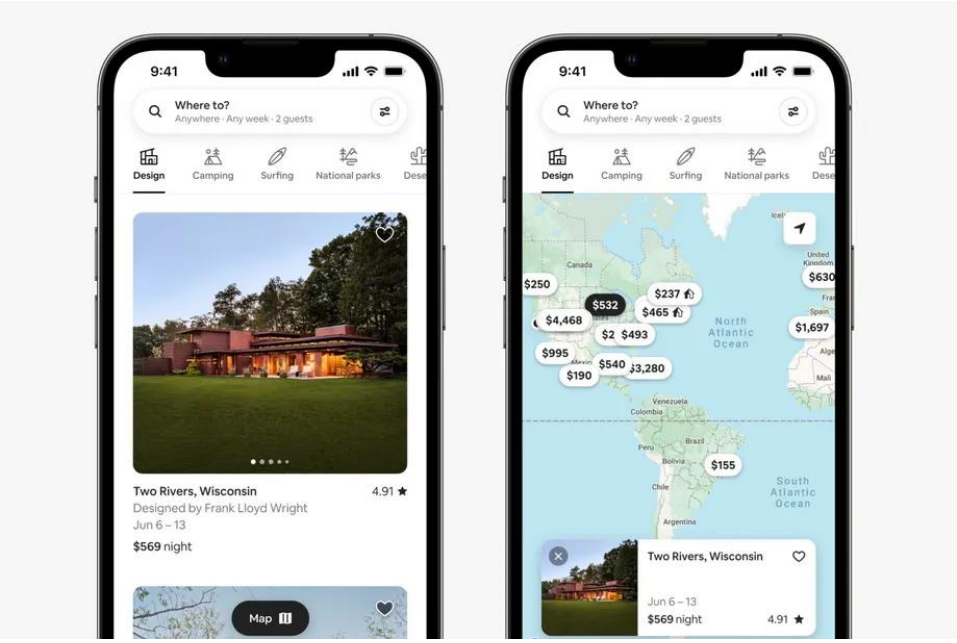

Figura 2 - Interface da aplicação da Airbnb

2.1.3 Trulia Rentals

A Trulia Rentals [13] é uma aplicação imobiliária para encontrar apartamentos, casas ou quartos individuais para alugar. Além de nos apresentar os factos típicos do imóvel, esta aplicação dá-nos uma ideia de como é realmente viver no bairro em que a mesma está presente.

Isto é possível através do "*What Locals Say*", em que são feitos inquéritos aos moradores com perguntas que podem ir desde se as pessoas se sentem seguras a andar à noite sozinhas até se as pessoas decoram a casa em épocas festivas. Com esta aplicação é possível termos uma sensação genuína de como é viver no bairro.

Abrindo esta aplicação somos deparados com um mapa, presente na Figura 3. Há medida que fazemos zoom, aparecem os pontos que marcam a localização dos apartamentos.

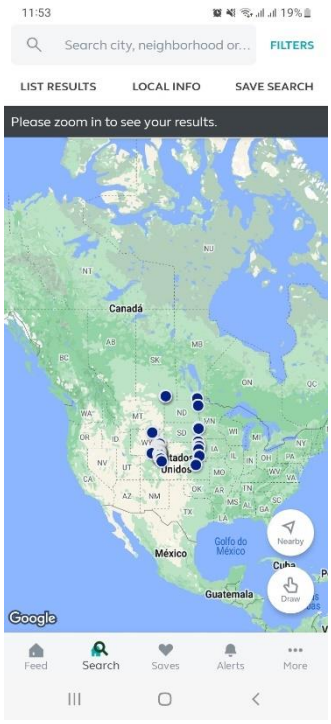

Figura 3 - Mapa com a localização de apartamentos

Clicando num dos pontos, abre nos um pequeno interface com alguns detalhes do apartamento, Figura 4.

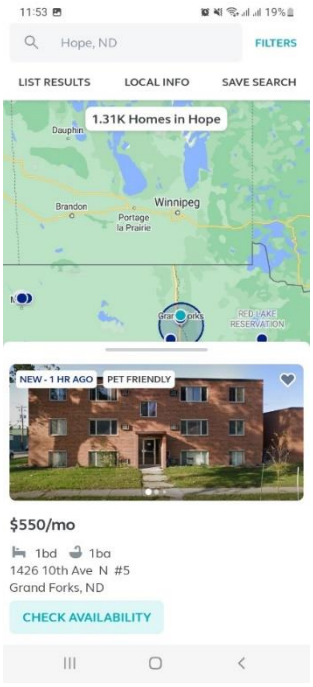

Figura 4 - Detalhes do apartamento

Se carregarmos na foto, conseguimos obter uma informação mais detalhada do apartamento, como representa a Figura 5. Além dos detalhes, também possui um mapa com a localização do apartamento além de alguns detalhes mais específicos, como há quanto tempo a casa está presente no mercado, o estilo do apartamento e informações do agente que a está a alugar.

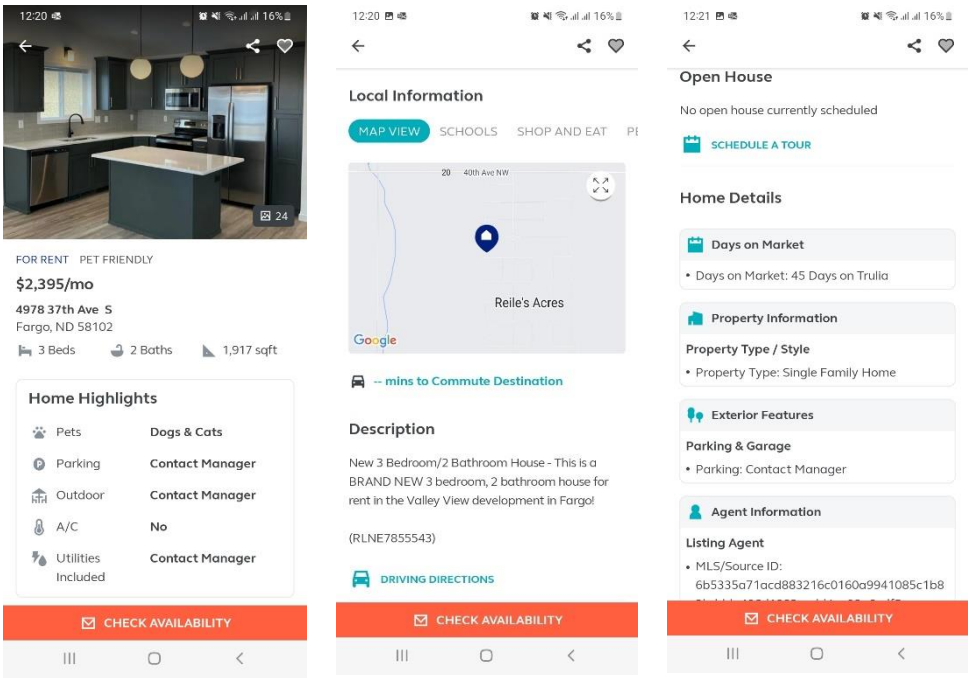

Figura 5 - Detalhes de um anúncio

2.1.4 Mitula Imóveis

A aplicação Mitula Imóveis [12] é uma aplicação que está disponível em mais de 40 países e em 15 idiomas. Esta aplicação classifica os imóveis com o objetivo de economizar tempo na pesquisa. Podemos encontrar todo o tipo de imóveis para compra ou aluguer.

- Esta aplicação tem como principais características:
- Entrar em contacto diretamente com os anunciantes;
- Criar uma conta gratuita para que possamos gravar os nossos imóveis favoritos;
- Descartar anúncios de forma a reduzir a nossa pesquisa, sendo assim mais fácil encontrarmos o que procuramos;
- Receber notificações quando aparecem imóveis com as características semelhantes ao imóvel que procuramos.
- Ver num mapa os imóveis que nos mais interessam.

Abrindo a aplicação, Figura 6 , temos logo a presença de um filtro por localidade, estado em que imóvel está (alugar ou venda) e tipo de imóvel.

Figura 6 - Ecrã incial da aplicação Mitupa

Carregando no botão “Ver resultados”, vamos obter uma lista de resultados relacionada com os filtros que aplicámos, Figura 7.

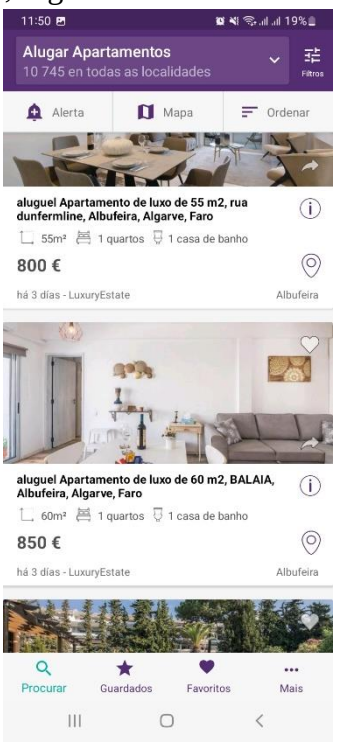

Figura 7 - Reusltados da pesquisa

Carregando num anúncio, a aplicação reencaminha-nos para o site do anunciante, Figura 8, pelo que podemos perceber que a aplicação não permite a criação de anúncios e reservas dentro da própria aplicação.

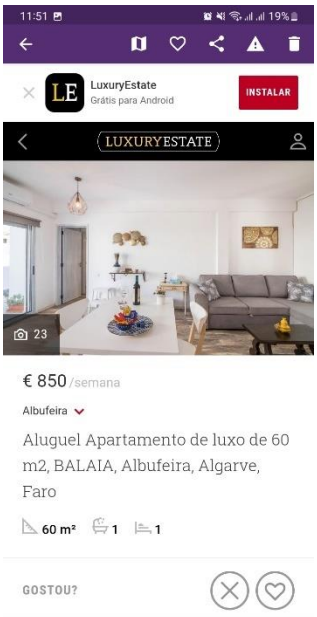

Figura 8 - Site do anuciante dentro da aplicação

2.2 Análise das aplicações apresentadas

As aplicações apresentadas, pareceram-nos bastante interessantes e em concordância com os objetivos do nosso projeto. A analise destas aplicações permitiu-nos identificar novas ideias para o desenvolvimento do projeto.

A Booking.com apresenta uma grande facilidade na pesquisa por alojamento, assim como na reserva do mesmo. Apresenta também um conjunto de funcionalidades relacionadas com o tema, mas que não iremos abordar neste projeto. Apresenta também uma funcionalidade de sugestão de locais a visitar, de acordo com o sítio para onde estamos a ir.

A Airbnb, mostrou-se ser uma aplicação muito interessante, a nível de publicação de anúncios de reserva ou de procurar por reserva. Apresenta várias semelhanças ao nosso projeto, desde logo a parte de anúncio e de reserva.

A Trulia Rentals, é uma aplicação que desde do principio ajuda o cliente a procurar anúncios na localização que ele pretende. Para isso, através do uso de mapas, o utilizador consegue ver quando imóveis existem na zona para onde ele quer ir. Nos detalhes dos anúncios, a aplicação também apresenta uma informação bastante detalhada e útil para o utilizador, de maneira que o mesmo perceba como será o ambiente onde irá estar.

A Mitula Imóveis, apresenta ser uma aplicação com uma grande diversidade de anúncios e com os filtros de pesquisa que ajudam muito o cliente a procurar por aquilo que deseja. Como podemos verificar, os anúncios não se criados diretamente na aplicação, vindo estes de outros sites pertencentes aos anunciantes. As reservas também são efetuadas dentro dos sites dos anunciantes, pelo que podemos dizer que esta aplicação apenas ajuda na encontra do imóvel para alugar.

A Tabela 1, apresenta um quadro comparativo entre a nossa aplicação e as que foram estudadas.

Tabela 1 - Quadro Comparativo

	Permite criação de anúncios	Permite acesso aos detalhes dos anúncios	Permite efetuar reservas	Permite ver no mapa onde fica localizado	Permite acesso a pontos de interesse antes da reserva
Booking.com	X	X	X	X	X
Airbnb	X	X	X	X	-
Mitula Imóveis	-	X	-	-	-
Trulia Rentals	X	X	X	X	X
Rent a House	X	X	X	X	X

Tendo em conta as aplicações anteriores, podemos referir que o projeto terá algumas semelhanças, no que diz respeito de anunciar alojamento e de reservar o mesmo. Contudo, e dadas as características especificas, a aplicação a desenvolver contará com novos requisitos que permitirão ao utilizador obter novas funcionalidades de modo a atingir o objetivo inicial.

3 Metodologia de Desenvolvimento e Análise de Requisitos

O presente capítulo tem como objetivo apresentar a metodologia de desenvolvimento utilizada ao longo da realização deste projeto.

3.1 Metodologia de Desenvolvimento

A metodologia de software tem um papel importante na estruturação, planeamento e controlo do processo de desenvolvimento de software. Por outras palavras, trata-se de utilizar um conjunto coerente e coordenado de métodos para atingir um objetivo, de modo que seja evitada a subjetividade na execução do trabalho. Através de um processo dinâmico e interativo para o desenvolvimento estruturado de projetos com visto à qualidade do produto final.

A metodologia a ser usada neste projeto é a metodologia ágil. A metodologia ágil tem as seguintes caraterísticas [8]:

- A prioridade é satisfazer o cliente através do fornecimento continuo e antecipado de software útil;
- Aceitar mudanças de requisitos, mesmo que estas tardem o desenvolvimento. Processos ágeis aproveitam as mudanças para a vantagem competitiva;
- Fornecer software operacional frequentemente;
- O método mais eficiente e eficaz de passar informação para dentro da equipa de desenvolvimento é a comunicação cara-a-cara;
- Processos ágeis promovem desenvolvimento sustentável.

Para a realização deste projeto e dentre todas as metodologias ágeis optou-se por utilizar a metodologia *Scrum*.

3.2 Metodologia de Desenvolvimento: *Scrum*

O *Scrum* é uma *framework* de coordenação de projetos, da organização ao desenvolvimento ágil de produtos complexos e adaptativos com o mais alto valor possível, através de várias técnicas. Hoje em dia é utilizado em mais de 60% dos processos ágeis em todo o mundo. Posto isto, os projetos são progressivamente desenvolvidos e melhorados de forma iterativa e incremental. O *Scrum* é divido em três fases:

- A fase inicial é um esboço da fase de planeamento, onde se estabelece os objetivos gerais para o projeto e desenha-se a arquitetura de software;
- Isto é seguido de uma série de ciclos sprint, onde cada ciclo faz um acréscimo do sistema;
- A fase de fecho do projeto encerra o projeto, completa a documentação necessária como os quadros de ajuda ao sistema e manuais de utilizador e avalia as lições aprendidas no projeto.

O funcionamento desta *framework* está descrito na Figura 9.

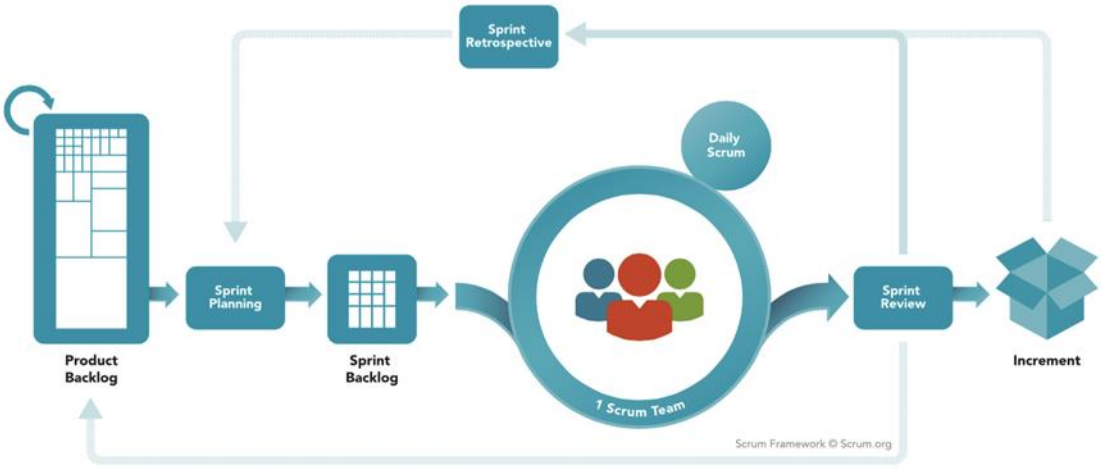

Figura 9 - Fundamento do Scrum. Fonte [10]

No início de cada *Sprint*, é realizada uma reunião de planeamento em que serão definidas o conjunto de funcionalidade do *Product Backlog* que irão ser implementadas durante o próximo Sprint, estas tarefas fazem assim, parte do *Sprint Backlog*. A equipa de desenvolvimento, juntamente com o Scrum *Master* que definem quais as tarefas a ser desenvolvidas e entregues. Na Figura 10 temos representado um exemplo do quadro com o planeamento do Sprint Backlog.

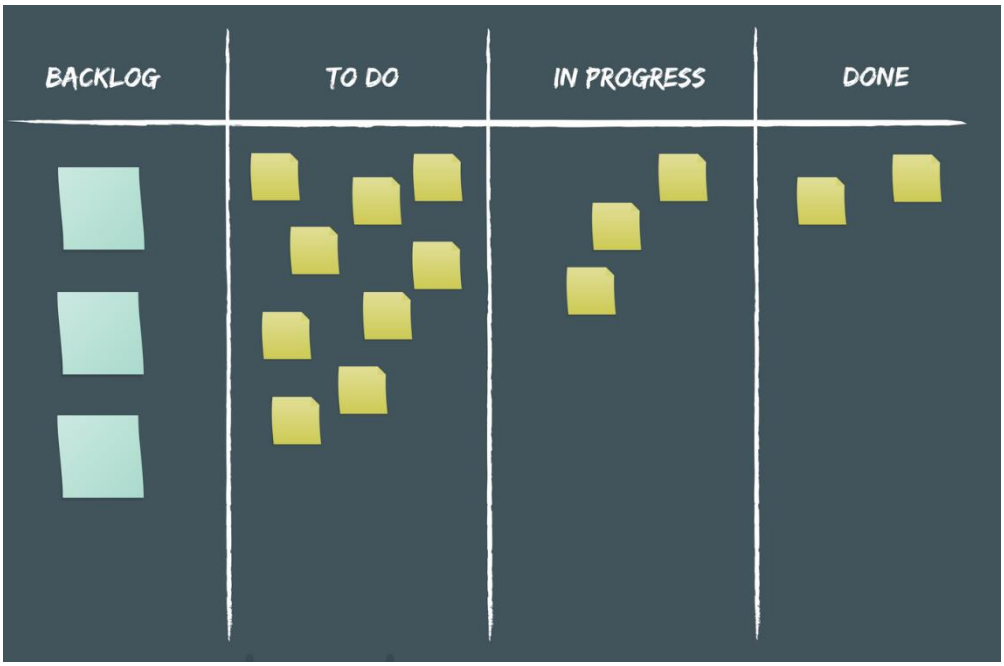

Figura 10 - Exemplo de quadro de planeamento do Sprint. Fonte [11]

A *Daily Scrum* é uma reunião com a duração máxima de 15 min que ocorre no início de cada dia, em que cada membro da equipa informa sobre o que foi realizado no dia anterior e o plano para o próprio dia. São também identificadas as eventuais falhas que estejam a bloquear o andamento do processo de desenvolvimento. A *Sprint Review Meeting* é uma reunião que é realizada no final de um ciclo de Sprint, em que a equipa de desenvolvimento apresenta o que foi implementado durante esse ciclo.

De acordo com a metodologia, e para o desenvolvimento da aplicação, seguiram-se as seguintes etapas (*sprints*) [7]:

1- Após a definição dos requisitos do projeto, obtidos através do estudo de outras aplicações idênticas já existentes no mercado e através da interação com o cliente, neste caso o orientador académico, foram definidos um conjunto de funcionalidades para implementar no projeto (*Product Backlog*);
2- Para cada etapa de desenvolvimento foram selecionados vários requisitos do *Product Backlog*, para assim se proceder ao planeamento de um *sprint*, que normalmente duraram entre 2 e 4 semanas. É nesta fase que determina o que fazer;

3- Ao longo de cada sprint foram realizadas reuniões de curta duração (via Zoom), envolvendo os membros da equipa do projeto, que visavam relatar o que foi feito e o que se pretendia fazer até à próxima reunião. Desta forma era possível transmitir o estado atual do trabalho, o planeamento do trabalho futuro e expor ou resolver as situações críticas ou dúvidas;
4- No fim de cada *sprint* foi realizada uma reunião para avaliar os objetivos cumpridos, tentando perceber possíveis falhas e assim melhorar nos seguintes sprints;
5- Cada *sprint* contará como um novo desenvolvimento do projeto. O processo, anteriormente, descrito realizou-se várias vezes, ao longo da realização do projeto, de modo que cada funcionalidade implementada respondesse aos objetivos propostos para esta aplicação.

3.3 Analise de requisitos do Sistema

Os requisitos de um sistema são a informação que descrevem as funcionalidades da aplicação. Com base nos objetivos propostos no projeto foi realizada uma análise e seleção que definem os requisitos do sistema a desenvolver.

Sendo assim, utilizaram-se práticas da metodologia descrita anteriormente, para o estudo detalhado sobre o que se deveria implementar, através da definição de requisitos. Estes requisitos serão apresentados em diagramas e símbolos gráficos de modelação na linguagem UML.

Pretende-se que na versão final os requisitos implementados cumpram os objetivos propostos.

3.3.1 Requisitos do Sistema

Tendo em vista a realização de uma aplicação que cumpra com os objetivos definidos, criar anúncios e permitir a reserva dos mesmos, definiram-se os seguintes requisitos funcionais:

- Registar utilizadores;

- Recuperar password;
- Realizar Login na aplicação;

Funcionalidades para o utilizador:

- Consultar perfil;
- Editar dados do perfil;
- Criar Anúncio;
- Consultar Anúncio;
- Eliminar Anúncio;
- Criar Reserva;
- Ver Reserva;
- Eliminar Reserva;

Funcionalidades para o Administrador:

- Adicionar métodos de pagamento;
- Editar métodos de pagamento;
- Eliminar métodos de pagamento;
- Adicionar pontos de interesse a cidades;
- Editar pontos de interesse;
- Eliminar pontos de interesse;
- Operações CRUD em todas as tabelas;

3.3.2 Atores e respetivos casos de uso

A Tabela 2, define os atores e intervenientes na aplicação, bem com os seus casos de uso a desenvolver.

Tabela 2 - Atores e seus casos de uso

Ator	Caso de uso
Utilizador	- Adicionar, editar, consultar e eliminar os seus anúncios; - Adicionar, editar, consultar e cancelar as suas reservas; - Consultar pontos de interesse; - Consultar e editar o seu perfil;
Administrador	- Operação CRUD para todas as tabelas

3.3.3 Descrição de casos de uso e Diagramas de Sequência

Os diagramas de sequência têm como objetivo mostrar as trocas de mensagens entre os objetos, quando ocorre uma operação, mostrar o fluxo de dados e as condições necessárias na realização das operações.

A. Criar Anúncio

Este caso de uso tem como objetivo criar um anúncio, onde podemos verificar a interação entre várias interfaces e uma sequência de eventos que permite ver o desenvolvimento quando o utilizador interage com o sistema.

A Tabela 3 demonstra, de uma forma detalhada, a descrição do caso de uso “Criar anúncio”.

Tabela 3 - Caso de uso "Criar Anúncio"

Nome	Criar Anúncio
Descrição	Este caso de uso serve para quando o utilizador quer criar um anúncio;
Tamanho	M
Pré-condição	Login Válido
Caminho Principal	1. O ator escolhe “Criar Anúncio”; 2. A aplicação mostra o formulário para criar um novo anúncio. 3. O ator insere os dados; 4. A aplicação grava os dados.
Caminhos Alternativos	3. a) O anúncio não foi criado porque o utilizador não preencheu todos os campos; 4. a) O anúncio não foi guardado porque não existe ligação com a internet.
Pós-condição	Não aplicável.
Suplementos ou adornos	Verificar se a aplicação deixa adicionar anúncios sem ter os campos obrigatórios preenchidos.

A Figura 11 seguinte apresenta o respetivo diagrama de sequência, onde mostra o cenário principal do caso de uso, isto é, quando o utilizador cria um novo anúncio.

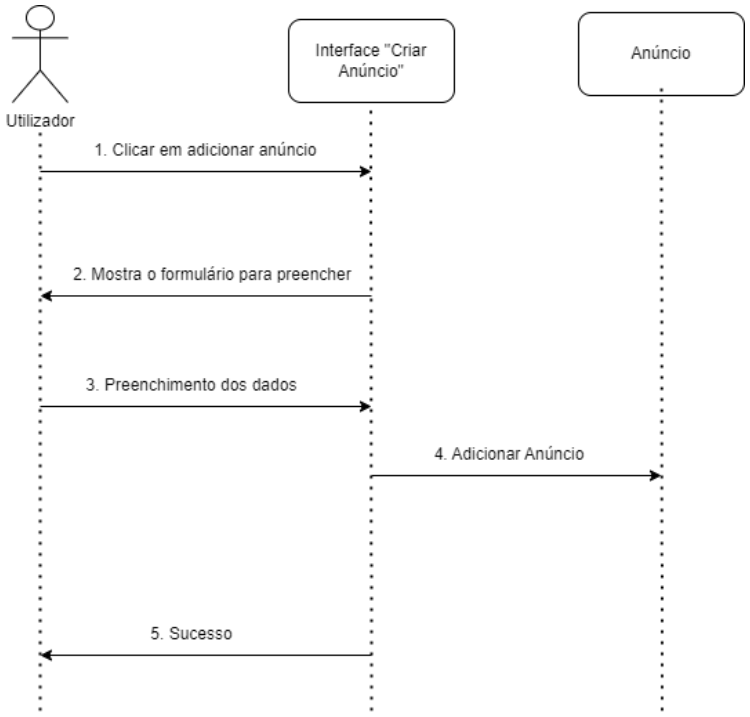

Figura 11 - Diagrama de Sequência "Adicionar Anúncio"

B. Criar Reserva

Este caso de uso tem como objetivo criar uma reserva, onde podemos verificar a interação entre várias interfaces e uma sequência de eventos que permite ver o desenvolvimento quando o utilizador interage com o sistema.

A Tabela 4 demonstra, de uma forma detalhada, a descrição do caso de uso “Criar Reserva”.

Tabela 4 - Caso de uso "Criar Reserva"

Nome	Criar Reserva
Descrição	Este caso de uso serve para quando o utilizador quer criar uma reserva;
Tamanho	M
Pré-condição	Login Válido
Caminho Principal	1. O ator escolhe um anúncio e carrega em "Ver Detalhes"; 2. Nos detalhes o anúncio, o ator seleciona "Criar Reserva"; 3. A aplicação mostra o formulário para criar uma nova reserva; 4. O ator insere os dados; 5. A aplicação grava os dados.
Caminhos Alternativos	4. a) A reserva não foi criada porque o utilizador não preencheu todos os campos; 5. a) A reserva não foi guardada porque não existe ligação com a internet.
Pós-condição	Não aplicável.
Suplementos ou adornos	Verificar se a aplicação deixa adicionar anúncios sem ter os campos obrigatórios preenchidos.

A Figura 12 seguinte apresenta o respetivo diagrama de sequência, onde mostra o cenário principal do caso de uso, isto é, quando o utilizador cria uma nova reserva.

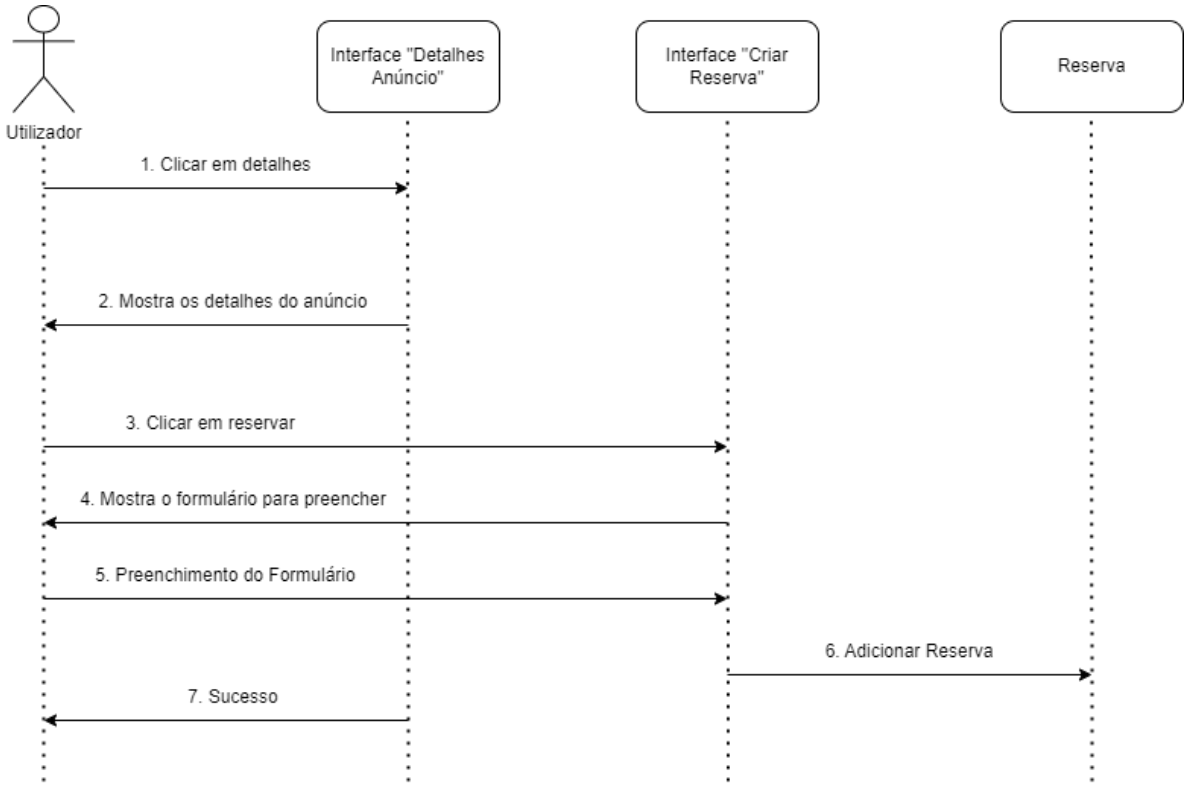

Figura 12 - Diagrama de Sequência "Adiconar Reserva"

3.4 Diagrama de Classes e dicionário de dados

3.4.1 Diagrama de Classes

Neste subcapítulo é apresentado o diagrama de classes da aplicação. Este diagrama mostra as diferentes classes que se relacionam entre si. Cada classe é constituída pelo nome da respetiva da classe, os atributos e as operações previstas. Apresenta-se de seguida uma breve reflexão sobre as classes definidas como mais importante no sistema, criadas de acordo com os requisitos do mesmo.

A classe “Anúncios” é uma das classes mais importante de entre todas, uma vez que este regista os dados relativos aos anúncios. Esta classe, juntamento com as restantes, formam o sistema, possibilitando assim um sistema, complexo, capaz de guardar e processar informação útil para os seus utilizadores.

A classe “Reserva”, “Pontos de Interesse”, “Utilizador” e “Cidades” também são importantes para aplicação, uma vez que estas guardam os dados sobre as reservas do

utilizador, os pontos de interesse de uma determinada cidade, os dados pessoais do utilizador e todas as cidades existentes, respetivamente.

A Figura 13 representa o diagrama de classes da aplicação.

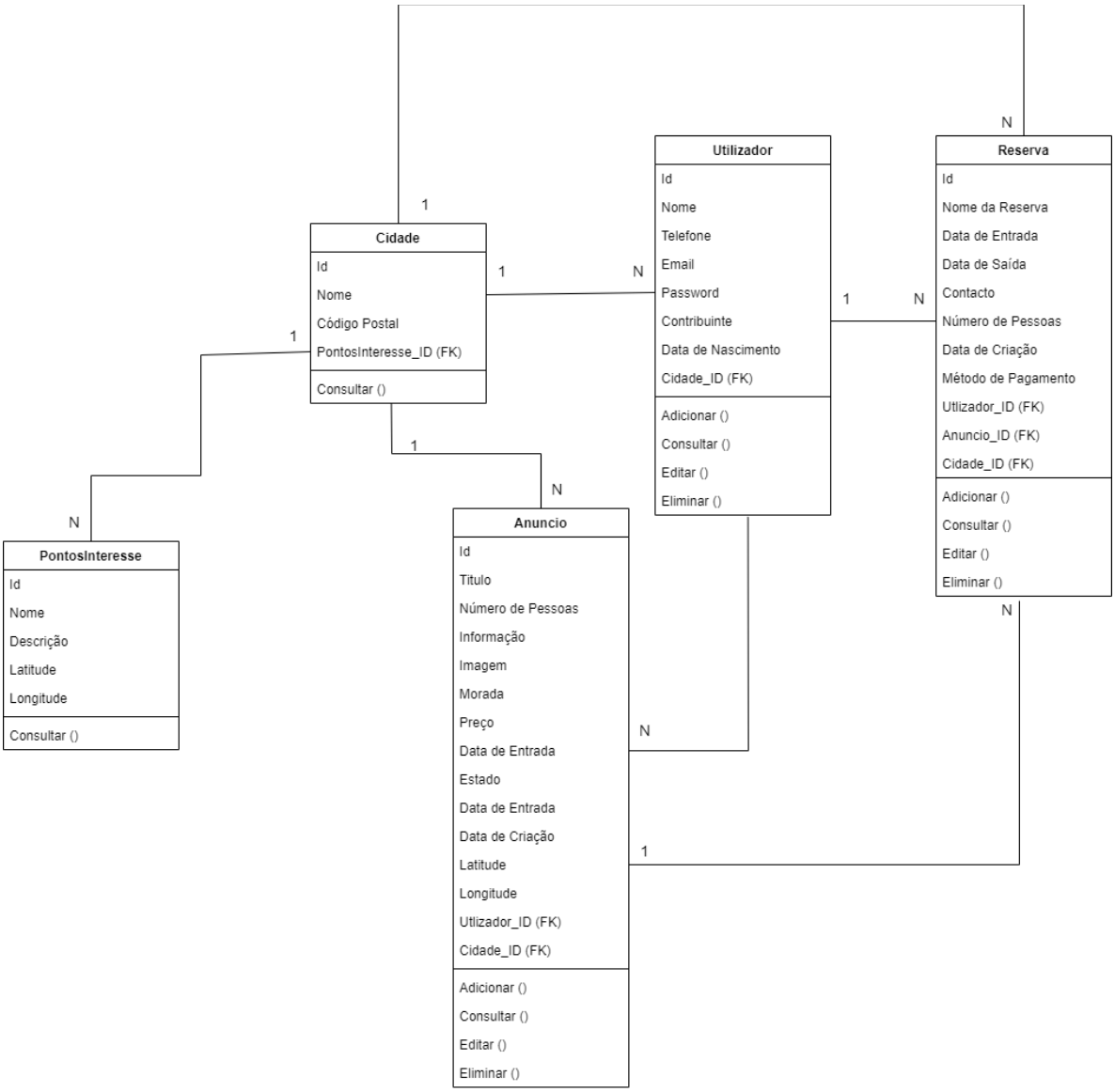

Figura 13 - Diagrama de Classes

3.4.2 Dicionário de dados

Nesta secção descrevem-se os campos de cada classe, apresentando o tipo de dados, o tamanho, uma breve descrição, o formato e as restrições associadas a cada atributo da classe. O dicionário de dados corresponde aos campos/atributos da base de dados.

A. Classe "Anúncio"

Tabela 5 - Dicionário de dados "Anúncio"

Anúncio					
Nome do campo	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
Anuncio_id	Number	50	Chave primária da tabela	Até 50 dígitos	Maior que 0. Obrigatório. Não alterável. Único.
Título	Varchar2	250	Título do anúncio	Até 250 carateres	Obrigatório. Alterável
Número de pessoas	Number	2	Número de pessoas que o apartamento suporta	Até 2 digitos	Obrigatório. Alterável.
Informação	Varchar2	500	Informação extra sobre o anúncio	Até 500 carateres	Opcional.
Morada	Varchar2	150	Morada do apartamento	Até 150 carateres	Obrigatório. Alterável.
Preço	Float	10,2	Preço do Anúncio	Até 10 digitos com 2 casas decimais	Obrigatório. Alterável.
Data de Entrada	Date	19	Data de entrada	dd/MM/yyyy	dd – [1,31]. MM – [1,12]. yyyy – [>= atual]. Obrigatório. Alterável.
Data de Criação	Date	19	Data de criação do anúncio	dd/MM/yyyy	dd – [1,31]. MM – [1,12]. yyyy – [>= atual]. Obrigatório. Não Alterável.
Latitude	nvarchar2	250	Latitude da localização	Até 250 carateres	Obrigatório. Não Alterável.
Longitude	nvarchar2	250	Longitude da localização	Até 250 carateres	Obrigatório. Não Alterável.

Utilizador_id	Int	25	Chave estrangeira que faz referência á tabela “Utilizadores”	Até 25 dígitos	Obrigatório. Não Alterável.
Cidade_id	Int	25	Chave estrangeira que faz referência á tabela “Cidades”	Até 25 dígitos	Obrigatório. Não Alterável.

B. Classe “Reserva”

Tabela 6 - Dicionário de dados "Reserva"

Reserva					
Nome do campo	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
Id	Number	50	Chave primária da tabela	Até 50 dígitos	Maior que 0. Obrigatório. Não alterável. Único.
Nome da Reserva	Varchar2	250	Nome da Reserva	Até 250 carateres	Obrigatório. Alterável
Data de Entrada	Date	19	Data de entrada	dd/MM/yyyy	dd – [1,31]. MM – [1,12]. yyyy – [>= atual]. Obrigatório. Não Alterável.
Data de Saída	Date	19	Data de Saída	dd/MM/yyyy	dd – [1,31]. MM – [1,12]. yyyy – [>= atual]. Obrigatório. Não Alterável.
Contacto	Number	9	Número de contacto do utilizador	Até 9 dígitos	Obrigatório. Alterável.
Número de Pessoas	Number	2	Número de pessoas	Até 2 dígitos	Obrigatório. Alterável.
Data de Criação	Date	19	Data de Criação da reserva	dd/MM/yyyy	dd – [1,31]. MM – [1,12]. yyyy – [>= atual]. Obrigatório. Não Alterável.
Utilizador_id	Number	50	Chave estrangeira que faz referência á tabela “Utilizadores”	Até 50 dígitos	Obrigatório. Não Alterável.
Anuncio_id	Number	50	Chave estrangeira que	Até 50 dígitos	Obrigatório. Não Alterável.

			faz referência á tabela "Anúncios"		
Cidade_id	Number	50	Chave estrangeira que faz referência á tabela "Cidades"	Até 50 dígitos	Obrigatório. Não Alterável.

C. Classe "Utilizador"

Tabela 7 - Dicionário de Dados "Utilizador"

Utilizador					
Nome do campo	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
Utilizador_id	Number	50	Chave primária da tabela	Até 50 dígitos	Maior que 0. Obrigatório. Não alterável. Único.
Nome	Varchar2	250	Nome do utilizador	Até 250 carateres	Obrigatório. Alterável
Telefone	Number	9	Número de contacto	Até 9 dígitos	Obrigatório. Alterável
Email	Varchar2	250	Email do utilizador	Até 250 carateres	Obrigatório. Alterável
Password	Nvarchar2	12	Password do utilizador	Até 12 carateres	Obrigatório. Alterável.
Contribuinte	Number	9	Número de Contribuinte	Até 9 dígitos	Obrigatório. Alterável.
Data de Nascimento	Date	19	Data de nascimento do utilizador	dd/MM/yyyy	dd – [1,31]. MM – [1,12]. yyyy – [>= atual]. Obrigatório. Alterável.
Cidade_id	Number	50	Chave estrangeira que faz referência á tabela "Cidades"	Até 50 dígitos	Obrigatório. Não Alterável.

D. Classe "Cidade"

Tabela 8 - Dicionário de dados "Cidade"

Cidade					
Nome do campo	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**

Cidade_id	Number	50	Chave primária da tabela	Até 50 dígitos	Maior que 0. Obrigatório. Não alterável. Único.
Nome	Varchar2	250	Nome da Cidade	Até 250 carateres	Obrigatório. Não Alterável
Código Postal	Nvarchar2	8	Código Postal da cidade	Até 8 carateres	Obrigatório. Não Alterável
Pontos_Interesse_Id	Int	25	Chave estrangeira que faz referência á tabela "Pontos Interesse"	Até 25 dígitos	Obrigatório. Não Alterável

E. Classe "Pontos Interesse"

Tabela 9 - Dicionário de dados "Pontos de interesse"

Pontos Interesse					
Nome do campo	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
Pontos_Interesse_Id	Int	25	Chave primária da tabela	Até 25 dígitos	Maior que 0. Obrigatório. Não alterável. Único.
Nome	Varchar2	250	Nome do Ponto de interesse	Até 250 carateres	Obrigatório. Não Alterável
Descrição	Varchar2	250	Breve descrição sobre o Ponto de interesse	Até 250 carateres	Obrigatório. Não Alterável
Latitude	nvarchar2	250	Latitude da Ponto de interesse	Até 250 carateres	Obrigatório. Não Alterável.
Longitude	nvarchar2	250	Longitude da Ponto de interesse	Até 250 carateres	Obrigatório. Não Alterável.

4 Desenvolvimento do projeto

O presente capítulo é subdividido em 2 pontos principais. O primeiro descreve as tecnologias utilizadas durante o desenvolvimento do projeto, o segundo retrata o processo de desenvolvimento da aplicação e os testes realizados.

4.1 Tecnologias utilizadas

Conforme referido anteriormente, este projeto foi desenvolvido usando o *Android Studio IDE* tendo-se optado pela linguagem *Java*. A escolha desta tecnologia deve-se ao facto de ter sido lecionada durante a licenciatura, nomeadamente na Unidade Curricular de Programação Avançada. Relativamente ao armazenamento de informação, usou-se *Firebase* como base de dados. Fez-se uso do *Google Maps API* para mostrar a localização das habitações para reserva. Também foi usada esta *API* para mostrar ao utilizador locais de interesse.

Em seguida, será explicado de forma sucinta e clara cada uma das tecnologias utilizadas para a realização deste projeto.

4.1.1 Android Studio IDE

O *Android Studio* [6] é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicações para *Android* e é baseado no *IntelliJ IDEA*. Além deste possuir um editor de código e ferramentas de desenvolvimento avançadas do *IntelliJ*, o *Android Studio* oferece ainda recursos que permitem aumenta a produtividade na criação de aplicações, tais como:

- Um sistema de compilação flexível baseado em *Gradle*;
- Um emulador rápido com inúmeros recursos;
- Um ambiente que permite o desenvolvimento para todos os dispositivos *Android*, independentemente da versão que estes possuem;
- Permite mudanças no código e nos recursos, durante a execução da aplicação, sem ser necessário reiniciar a aplicação;
- Possui integração com o *GitHub* para ajudar a criar recursos comuns de aplicações e importar exemplos de código;
- *Frameworks* e ferramentas de teste;
- Ferramentas de *lint* para a deteção de problemas de desempenho, usabilidade, compatibilidade com versões, entre outros;
- Compatibilidade com C++ e NDK;

- Compatibilidade integrada com o *Google Cloud Platform*, facilitando assim a integração do *Google Cloud Messaging* e do *App Engine*.

4.1.2 Java

O *Java* [1] é uma das linguagens de programação mais usadas no mundo. O código *Java* é baseado em classes e orientado a objetos, com foco em segurança, portabilidade e alta performance.

Tem também, como principais características uma sintaxe similar a C/C++, uma extensa biblioteca de rotinas e *APIs* para trabalhar com recursos de rede, e uma poderosa manutenção automática de memória.

Diferente de outras linguagens de programação, o *software* não é compilado em "código nativo" para ser executado diretamente pelo computador, mas sim um código intermediário chamado "*bytecode*", que é então interpretado e executado pela *Java Virtual Machine* (JVM).

Desta forma, um sistema ou aplicação criado em *Java* torna-se muito mais portátil, podendo ser executado em praticamente qualquer ambiente ou dispositivo no qual a JVM esteja instalada.

4.1.3 Firebase

O *Firebase* [2] é uma plataforma digital de desenvolvimento de aplicações, lançada pela *Google*. Com ele, é possível criar e expandir aplicações com simplicidade, agilidade e facilidade para *Android*, iOS e Web.

Além disso, os seus recursos permitem melhorar o rendimento e a performance das aplicações, torná-las mais seguras e oferecer uma experiência mais rica e completa ao utilizador.

Neste sentido, o *Firebase* é considerado como um *Backend as a Service* (BaaS), isto é, um modelo de serviço que oferece toda a infraestrutura voltada para o funcionamento interno do *software*, como base de dados, envio e receção de informação, armazenamento, entre outros.

Para o desenvolvimento, foram utilizados os seguintes recursos do *Firebase*:

- Realtime Database – O *Firebase Realtime Database* é uma base de dados NoSQL hospedada na *cloud*. Com ela, podemos armazenar e sincronizar dados entre os outros utilizadores em tempo real. Quando os utilizadores ficam off-line, os SDKs do *Realtime Database* usam a cache local no dispositivo para aplicar e armazenar alterações. Quando o dispositivo volta a ficar online, os dados locais são sincronizados automaticamente;
- Authetication – O principal objetivo do *Firebase Authentication* é facilitar o desenvolvimento de um sistema de autenticação seguro, além de melhorar a experiência de login e integração para os utilizadores finais. Este oferece uma solução de identidade completa, compatível não só com contas com e-mail e password, mas também com autenticação por telefone, conta Google, Twitter, Facebook, Github, entre outros.
- Cloud Storage – O *Cloud Storage* foi criado para ajudar a armazenar e a disponibilizar o conteúdo gerado pelo utilizador, como fotos e vídeos, com facilidade e rapidez.

4.1.4 Google Maps API

O *Google Maps API* [3] é uma *API* desenvolvida pela Google, que permite visualizar o mundo real por meio de mapas estáticos ou interativos, e pode ser implementa nas aplicações ou web.

Esta, é um serviço público e gratuito que qualquer utilizador pode usar nos seus sites e aplicações.

Atualmente, a *Google Maps Plataform* oferece múltiplas *API's* para diferentes serviços. Exites uma API de Mapas Estáticos ("*Maps Static API*") para incorporar de forma simples o *Google Maps*, uma API de Mapas para JavaScript ("*Maps JavaScript API*") para mapas interativos e costumáveis, uma API de Locais ("*Places API*") para aceder a detalhes de pontos de interesse e ainda uma *API* de Direções ("*Directions API*") que dá as direções para uma localização.

Para este projeto foi usada a *API* de Mapas Estáticos.

4.2 Arquitetura do Sistema

A arquitetura do sistema encontra-se representada na Figura 14.

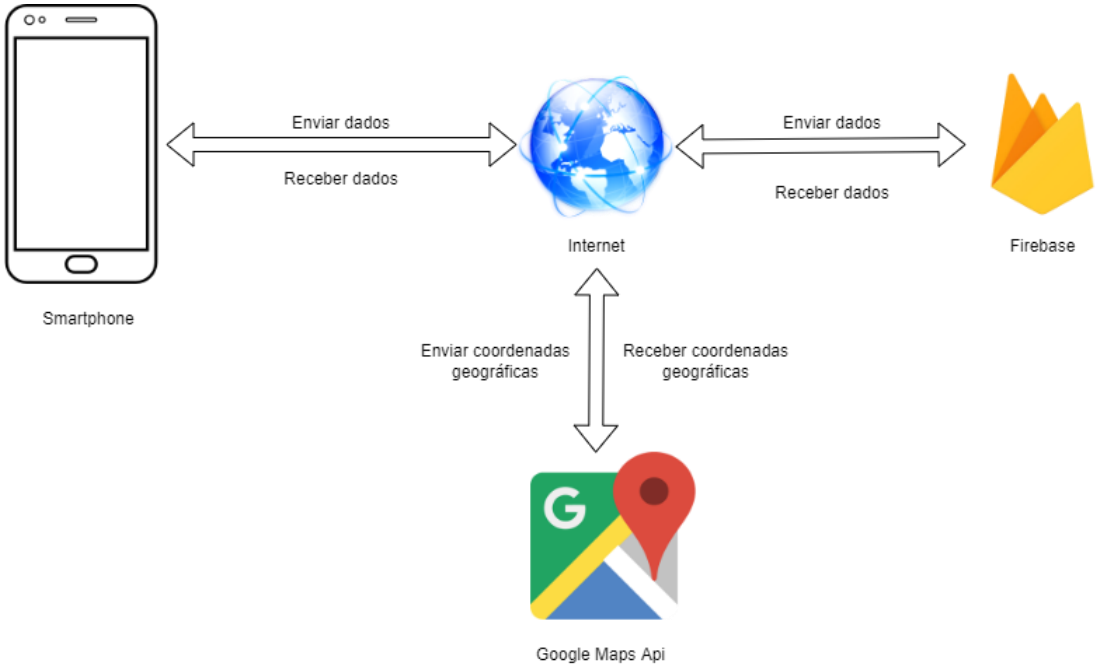

Figura 14 - Arquitetura do Sistema

O sistema inicia-se a partir do momento em que se executa a aplicação no dispositivo móvel. Todos os dados que são enviados entre o *Firebase* e o dispositivo móvel, são enviados através da internet.

Quando se cria um anúncio, o sistema obtém as coordenadas geográficas da localização do alojamento, através do *Google Maps API*. As coordenadas são guardadas na base de dados.

4.3 Desenvolvimento da Aplicação

De maneira a compreender melhor o projeto desenvolvido, neste subcapítulo, irá ser abortado as etapas principais que levaram à implementação e desenvolvimento da aplicação.

4.3.1 Implementação da aplicação

O ponto de partida do desenvolvimento desta aplicação foi a criação de um projeto no *Android Studio*. Após as primeiras configurações, procedeu-se à inserção da chave da API do google no ficheiro *AndroidManigest.xml*, para que possamos fazer uso do serviço do *Google Maps*. A Figura 15, representa como podemos ativar a API e obter a chave ao aceder ao *Google Cloud Plataform*.

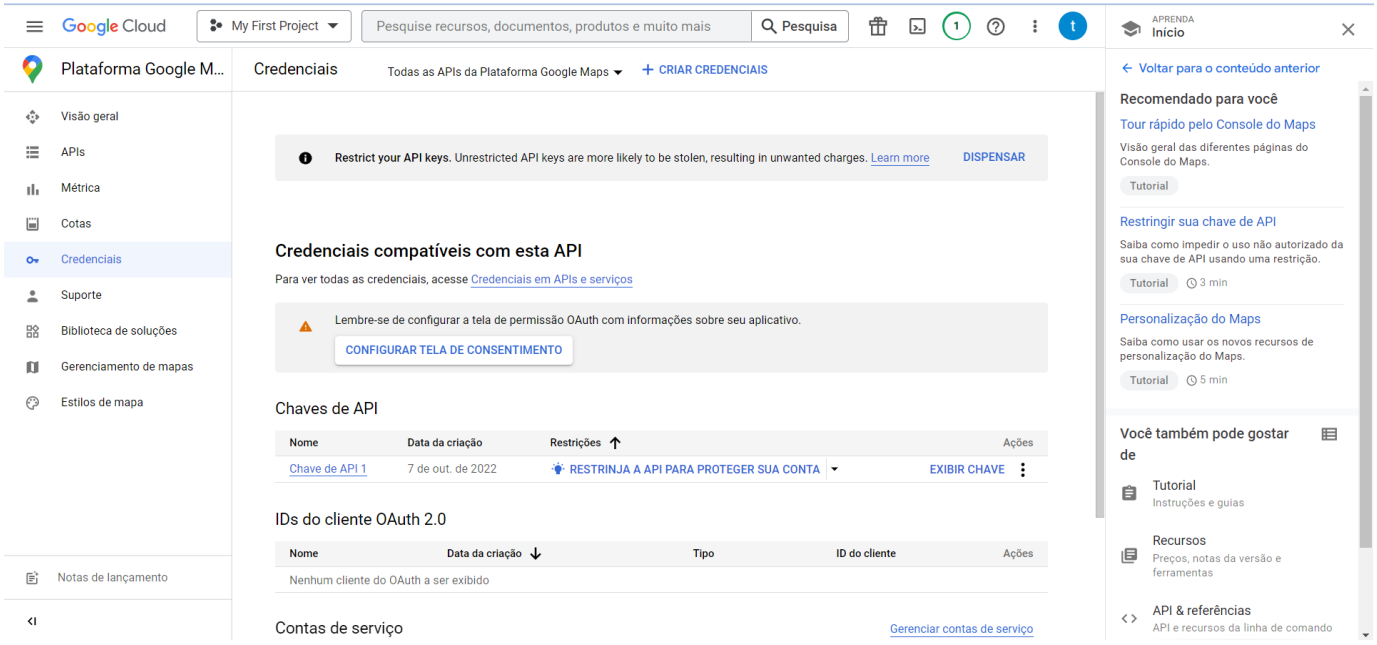

Figura 15 - Obtenção da chave no Google Cloud Plataform

O passo seguinte foi proceder-se à ligação com a base de dados. O *Android Studio* já possui uma opção para ligar automaticamente com o *Firebase*, sem que seja necessário acrescentar o código de forma manual. A representação dessa opção está presente na Figura 16.

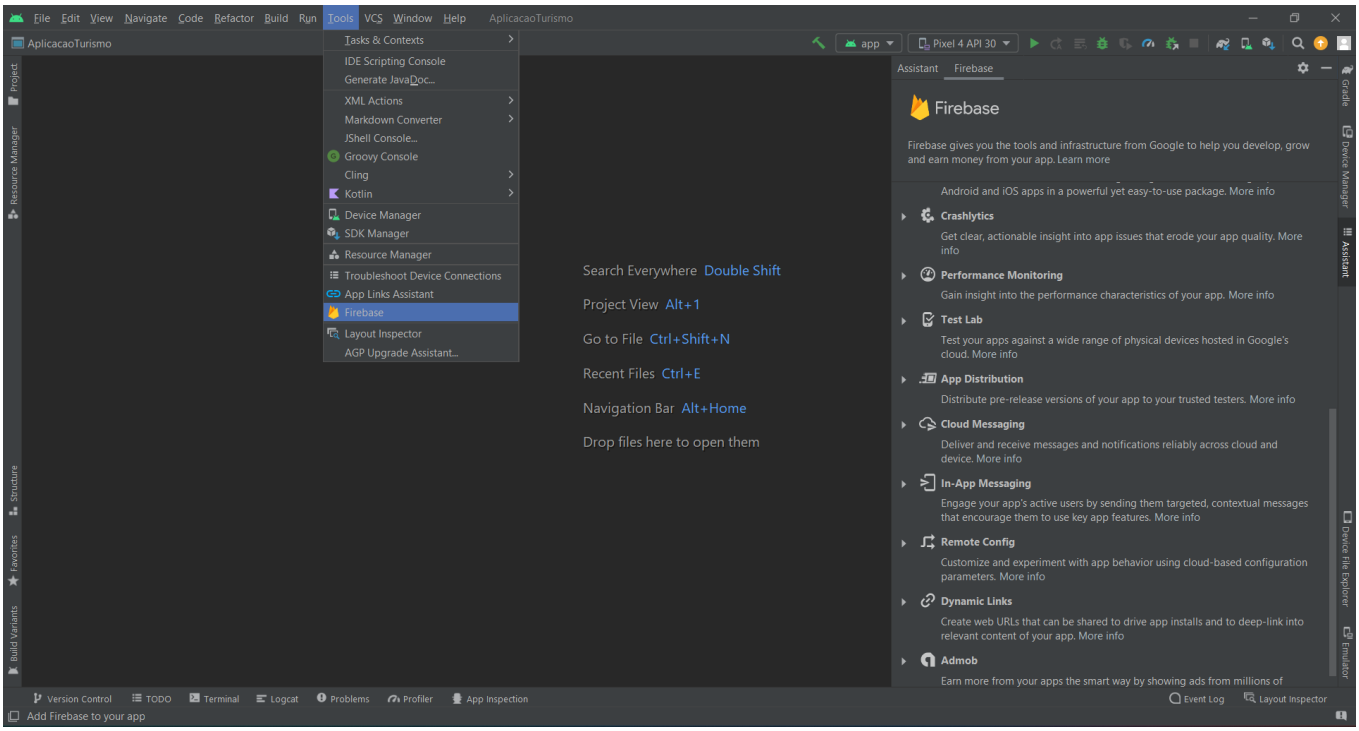

Figura 16 - Como conectar o Android Studio ao Firebase

Para o desenvolvimento deste projeto e dado os requisitos do mesmo foi necessário efetuar a configuração da autenticação dos utilizadores, usando email e palavra-passe, e a configuração de uma base dados online – *realtime database*. Estas duas funcionalidades permitem a segurança da aplicação e possibilitam o acesso à base de dados online sincronizada em tempo real para todos os utilizadores.

Após as configurações iniciais e tendo em vista a implementação dos requisitos funcionais, apresentados anteriormente, foram desenvolvidas as respetivas classes, implementando as operações CRUD, bem como as interfaces.

A Figura 17, mostra a construção do ecrã "Adicionar Anúncio", sendo este um dos ecrãs principais da aplicação.

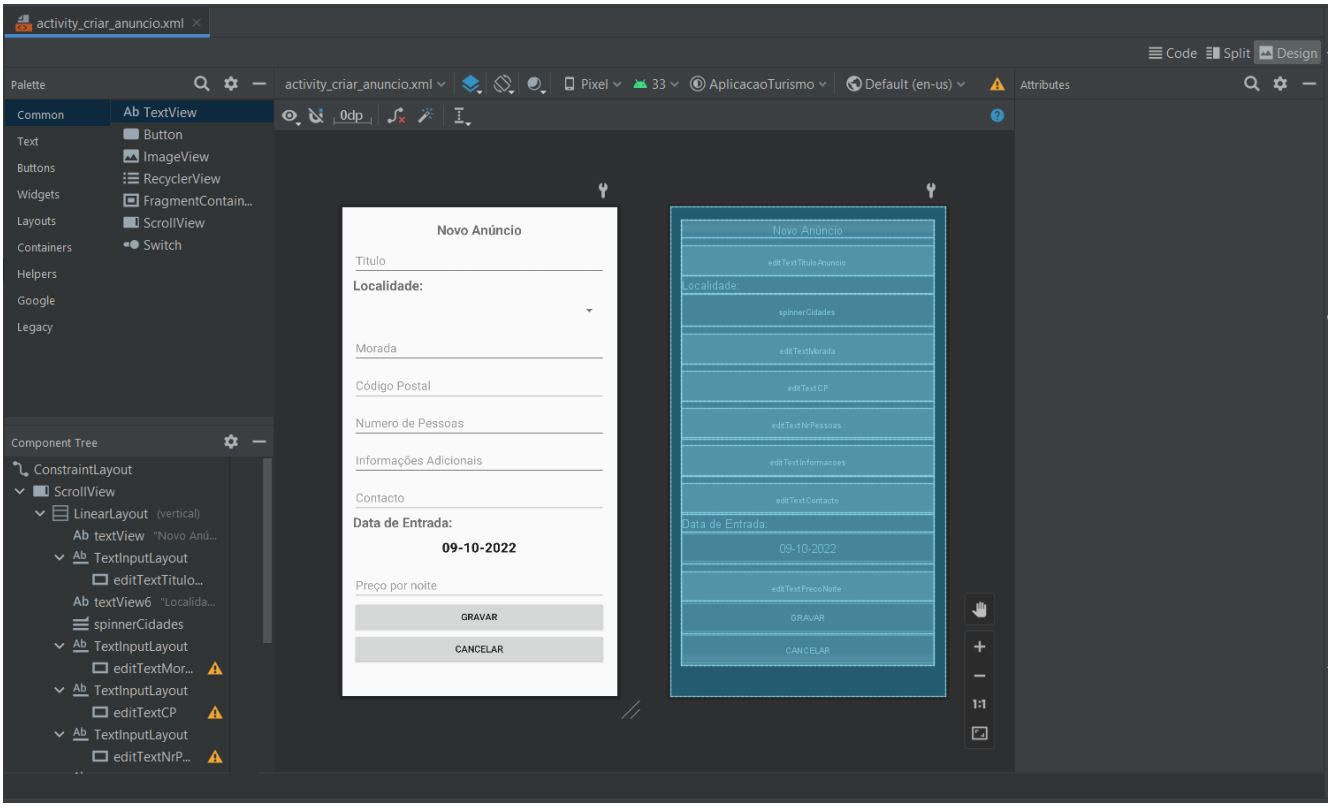

Figura 17 - Construção ecrã "Adicionar Anúncio"

A funcionalidade “Adicionar Anúncio” tem como principal objetivo adicionar na base de dados um novo anúncio. Para tal, é necessário aceder a esta interface e preencher o formulário. Cada campo, possui uma validação para evitar os erros que possam ser gerados ao preencher os campos.

Para além desta validação, foi criado um data *picker*, Figura 18 , que permite assim que não haja erros na formatação da data e na inserção da mesma. O código relativo a essa implementação está presente no anexo x.

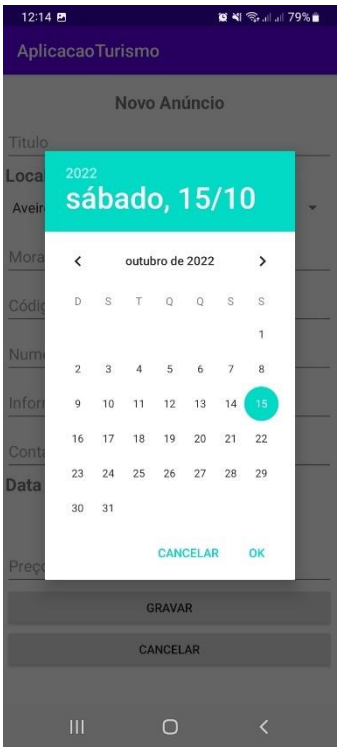

Figura 18 - Data Picker

Depois de adicionado o anúncio, e de modo a ser possível a realização das operações CRUD, são criadas as classes:

- DetailsActivity;
- AnuncioModel;
- AnuncioAdapter.

Estas classes, tem como objetivo:

- DetailsActivity – Classe responsável por apresentar todos os anúncios inseridos, mostrar os dados provenientes da base de dados e apresentar as opções de reservar, *update* e *delete*;
- AnuncioModel – Classe que contém os atributos *get* e *set*, que tem como finalidade enviar e receber os dados que provêm da base de dados;
- AnuncioAdapter – Classe responsável por definir a estrutura dos itens que aparecerem na lista.

É ainda possível, efetuar pesquisas sobre os dados inseridos. A pesquisa é feita através da Localidade de cada anúncio, em que inserindo a localidade o sistema irá devolver apenas os anúncios que pertencem a essa localização.

Por último, e no que diz respeito ao desenvolvimento da aplicação, referir que a mesma foi desenvolvida tendo em consideração os requisitos do sistema, bem como questões inerentes a acessibilidade e usabilidade de sistemas.

Neste sentido, podemos afirmar que o desenvolvimento, do projeto, foi pensado e desenvolvido com o objetivo de ser uma aplicação amigável para utilizador. O layout de toda a aplicação, relativa às opções, foi desenhado tendo em vista uma fácil utilização.

4.3.2 Interfaces da aplicação

A figura, apresenta o layout da interface relativa ao Login, onde o utilizador pode iniciar sessão na aplicação, registar-se, caso não tenha conta, e recuperar a *password* caso tenha perdido a mesma.

A. Interface do utilizador

A Figura 19, representa o formulário para efetuar o registo na aplicação.

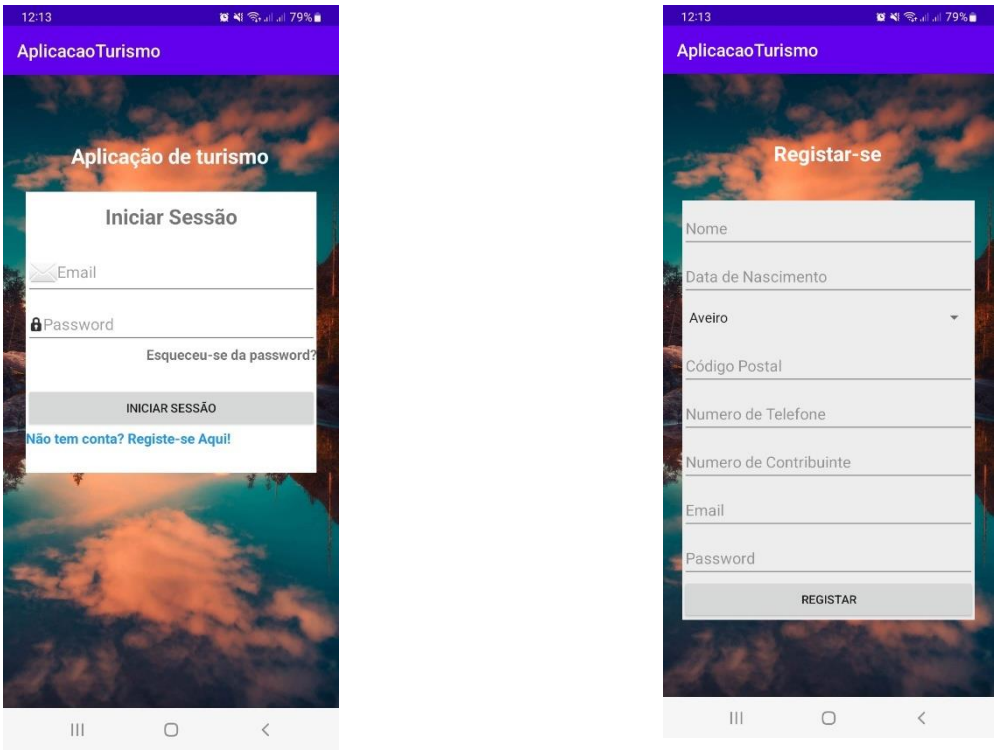

Figura 19 - Interface de Login e Interface de Registo

Após iniciar sessão, o utilizador pode aceder ao perfil usando o menu lateral. Neste menu também tem a opção de terminar sessão. O menu está representado na Figura 20.

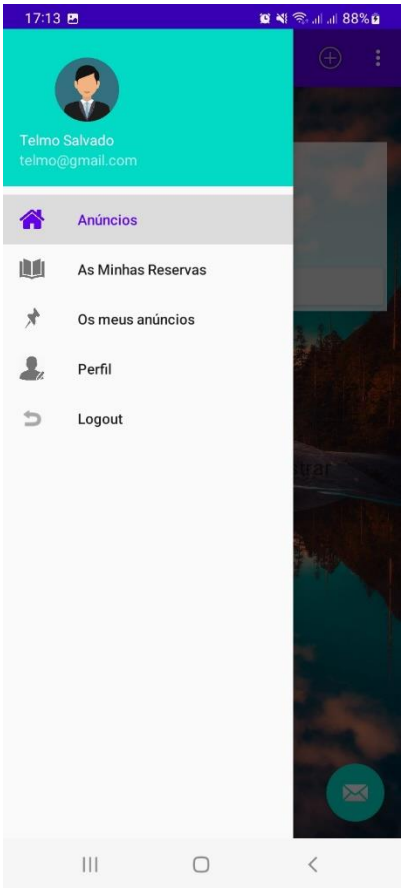

Figura 20 - Menu do Utilizador

A Figura 21 apresenta o layout onde o utilizador poder ver o seu perfil e poderá também editar os seus dados pessoais.

Figura 21 - Interface "Ver Perfil"

B. Interface com lista de anúncios

A Figura 22, apresenta o layout principal, onde se podem visualizar os anúncios disponíveis. Os anúncios apresentados são apenas os que não foram criados pelo utilizador que está com a sessão iniciada.

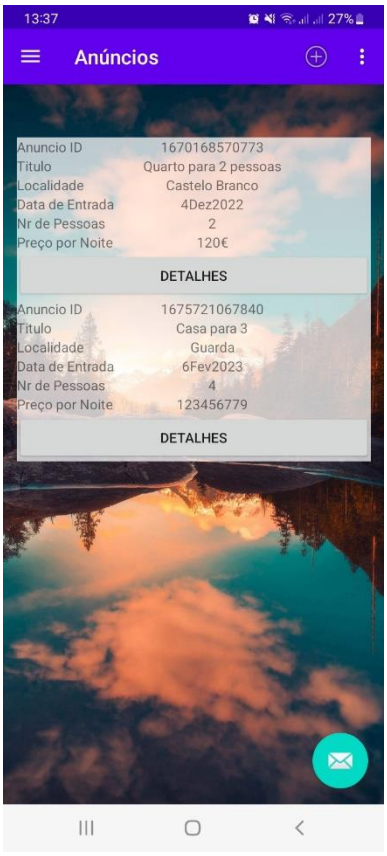

Figura 22- Lista de Anúncios

Carregando no botão dos detalhes será possível ver os detalhes do anúncio, bem como a localização do mesmo e ainda obter locais de interesse próximos dessa localização. É através desta interface que podemos criar a reserva. Essa interface está representada na Figura 23.

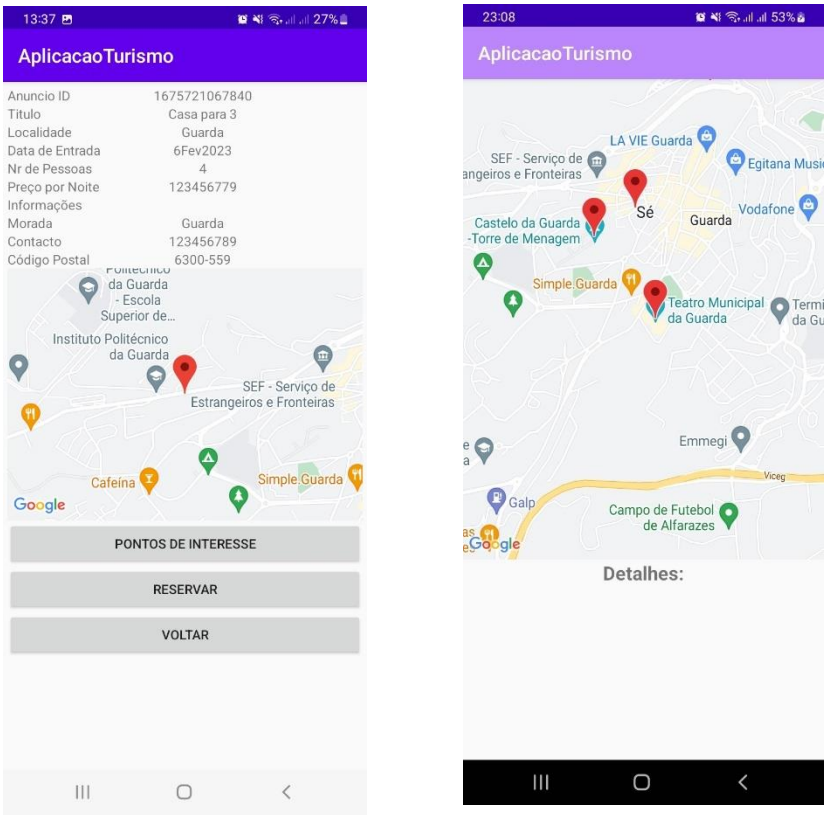

Figura 23- Interface de detalhes do Anúncio e Pontos de interesse

Carregando em cada um dos marcos, irão aparecer detalhes sobre os pontos de interesse como demonstra a Figura 24.

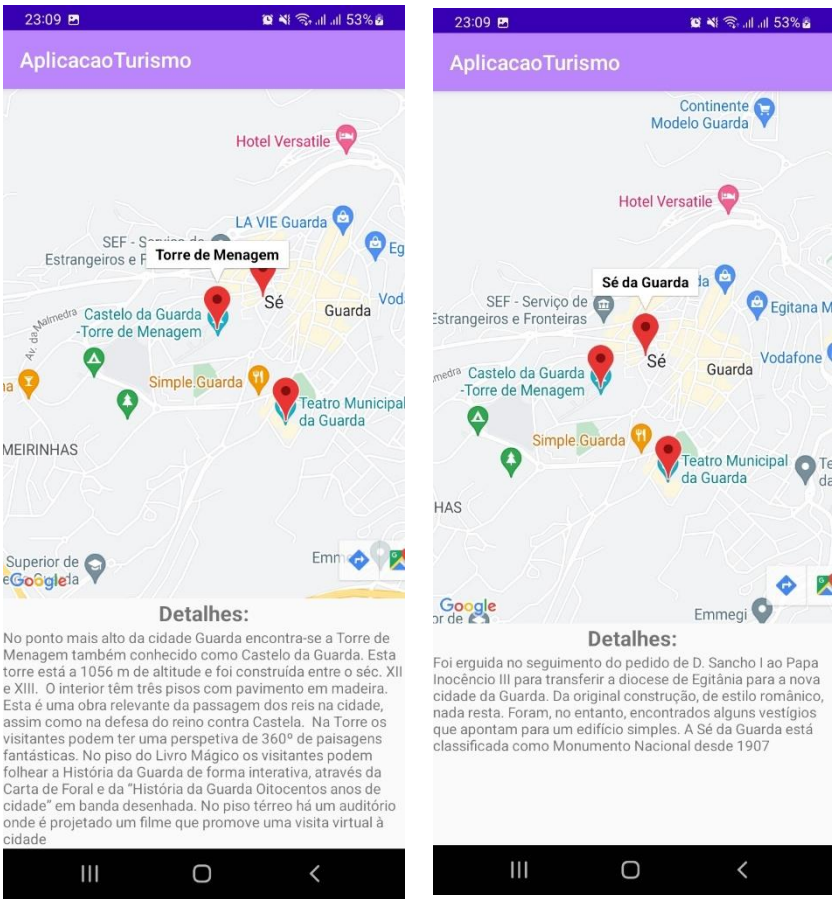

Figura 24- Exemplos de Detalhes após carregar em diferentes marcos

C. Interface de anúncio e reserva

A Figura 25 mostra o layout do formulário que é necessário preencher para quando o utilizador quer criar um novo anúncio.

Figura 25 - Formulário de Criação de Anúncio

Posteriormente, na Figura 26, o utilizador pode aceder aos seus anúncios e ver todos os detalhes. Pode fazer alteração aos mesmos, se assim o desejar, e também os poderá eliminar.

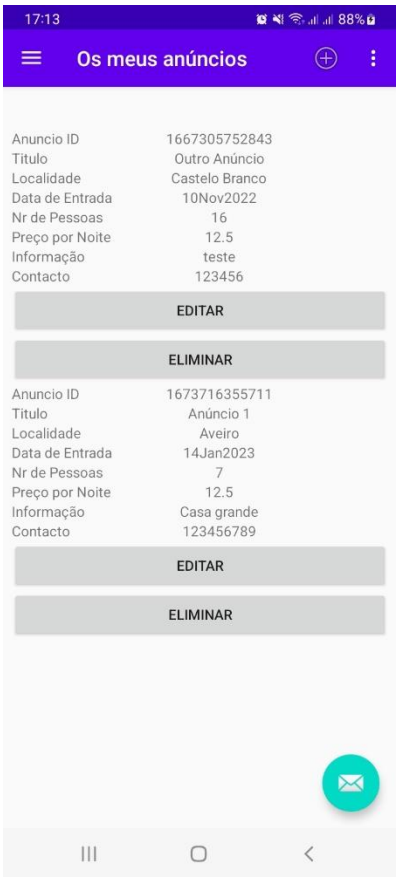

Figura 26- Interface "Os meus Anúncios"

Os anúncios ficam disponíveis na página principal para todos os utilizadores que não sejam o utilizador que os criou, isto é, apenas utilizadores diferentes do utilizador que criou o anúncio podem efetuar a reserva. Na Figura 27, temos representado o formulário que é necessário preencher quando se cria uma reserva.

Figura 27 - Formulário de Criação de uma reserva

Clicando no botão “as minhas reservas”, o utilizador consegue ver todas as suas reservas, bem como editar a mesma e até cancelar. Esta interface está representada na Figura 28.

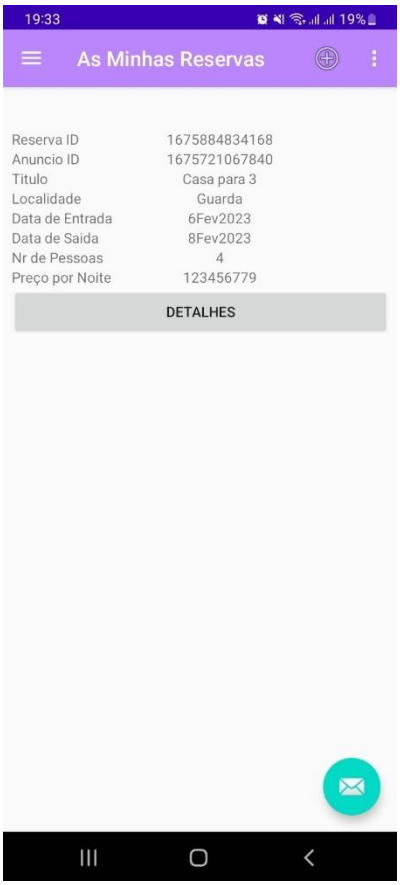

Figura 28 - Menu "As minhas Reservas"

4.4 Testes

Ao longo do desenvolvimento desta aplicação, foram realizados vários tipos de teste, de forma a garantir a qualidade do sistema. Para além dos testes realizados, verificou-se os campos da aplicação estão de acordo com as especificações apresentadas nos requisitos e se cumprem os objetivos previstos.

A realização de testes contínuos, permite detetar possíveis erros, de forma antecipada, para que possam ser corrigidos.

4.4.1 Teste validação dos campos

Este primeiro teste tem como objetivo validar se os campos obrigatórios do formulário de criação de anúncio, estão todos preenchidos e se nenhum deles tem um valor incorreto. Isto é, sempre que o utilizador faz um erro no preenchimento do formulário ou deixa algum campo bem branco que seja obrigatório, a aplicação deverá alertar o utilizador que terá de validar esse campo, sem deixar que o mesmo prossiga.

Após a realização deste teste ao formulário, verificamos que o mesmo se encontra a funcionar como esperado, uma vez que, o objetivo é que o utilizador não possa avançar sem que tenha preenchido o formulário corretamente.

A Figura 29, representa o formulário com a mensagem de erro, quando o utilizador não preenche os campos obrigatórios.

Figura 29- Formulário "Adicionar Anúncio" com validações

O segundo teste teve como objetivo validar se a data de entrada inserida não é inferior á data atual, isto é, a data de entrada no anúncio tem de ser superior á data atual.

Na Figura 30, podemos verificar um exemplo desse teste.

Figura 30 - Teste de Validação de Datas

Já no formulário das reservas, a data de saída não pode ser inferior à data de entrada no anúncio. Para tal, a data mínima foi limitada ao dia que está marcado no anúncio como data de entrada. Vejamos como exemplo, Figura 31, para um anúncio que tem data de entrada como dia 6/02/2023, a aplicação não deixa escolher um dia que seja inferior ao mesmo.

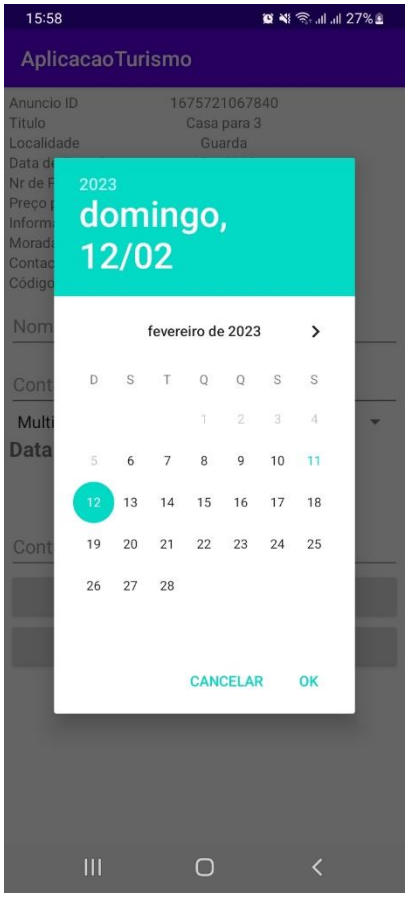

Figura 31 - Validação de data ao criar reserva

Tal como podemos verificar, ambos os testes foram realizados com sucesso.

4.4.2 Testar adicionar dados do Anúncio

Para este teste, o objetivo foi verificar se a informação inserida na aplicação era gravada com sucesso na base de dados, ou seja, no Firebase.

Na Figura 32, temos um exemplo de um formulário preenchido, pronto para ser guardado.

Figura 32 - Formulário "Novo Anúncio" preenchido

Acendendo ao menu “Meus anúncios”, Figura 33, podemos verificar que o anúncio que inserimos já está lá presente, pelo que podemos concluir que o anúncio foi adicionado com sucesso à base de dados.

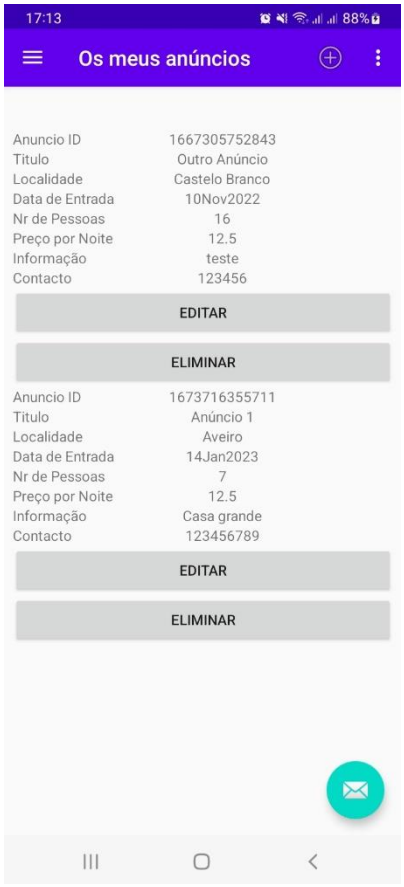

Figura 33 - Interface "Meus Anúncios"

A Figura 34, mostra os dados inseridos no Firebase. A latitude e a longitude do local são calculadas automaticamente através do código postal.

Figura 34 - Dados Inseridos no Firebase

4.4.3 Validação de filtros

Este teste tem em vista validar o filtro por localidade. Escrevendo uma localidade, a aplicação deve apenas devolver anúncios para essa localidade. A demonstração está presente na Figura 35.

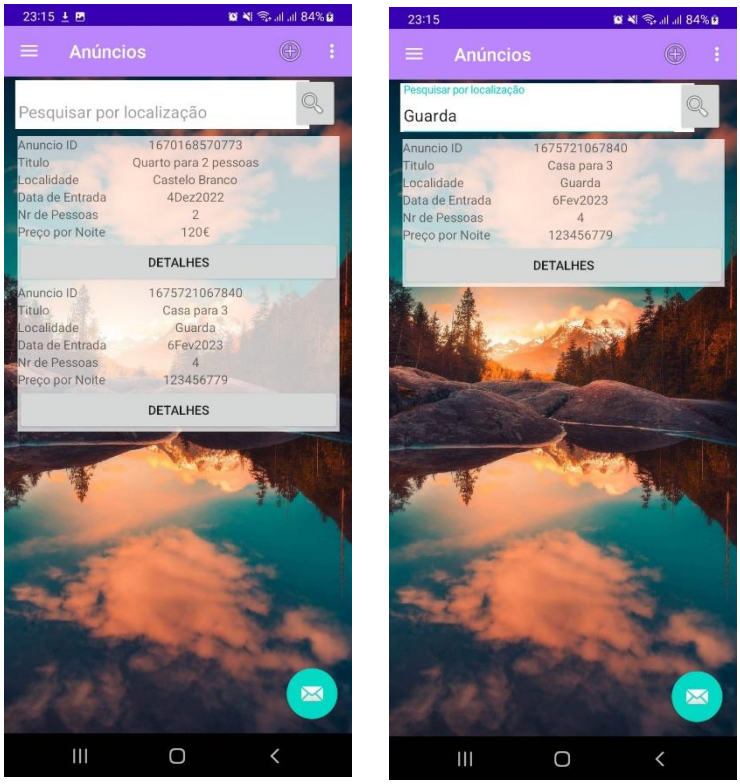

Figura 35 - Utilização do Filtro de Cidade

Já quando não existe uma localidade para o filtro selecionado, a aplicação deve demonstrar um alerta de erro, de maneira a alertar o utilizador. A demonstração deste teste, está presente na Figura 36.

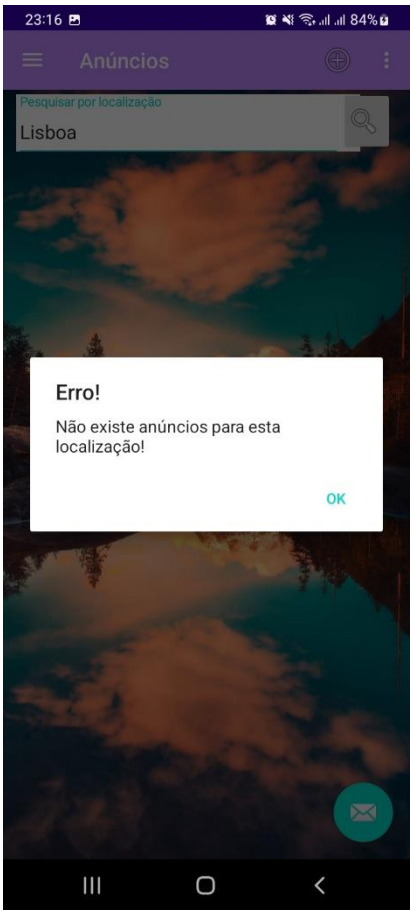

Figura 36 - Alerta de erro quando não existe dados para filtrar

Como mencionado, a realização dos testes permitiu verificar e validar a qualidade do software, garantido assim, o seu bom funcionamento.

5 Conclusão

Durante o desenvolvimento deste projeto, foi criada uma aplicação que tem como objetivo principal criar anúncios para posteriormente serem reservados por outros utilizadores.

Inicialmente, foram definidos os objetivos e os requisitos do sistema para um correto desenvolvimento da aplicação. Após isso, foram realizadas diversas pesquisas sobre aplicações idênticas, que já existem no mercado. Estas aplicações irão de encontro aos objetivos definidos inicialmente para o projeto.

Após a definição dos objetivos, requisitos do sistema e modelação procedeu-se ao desenvolvimento e à implementação da aplicação. O desenvolvimento da aplicação teve sempre por base as questões relativas à qualidade de software, sendo efetuados testes a cada módulo que foi desenvolvido.

Podemos assim referir, que a aplicação se encontra desenvolvida de acordo com os objetivos iniciais previstos e está, na nossa opinião, preparada para aceitar a criação de anúncios e de reservas.

De referir ainda, que o desenvolvimento do projeto tem por base a apresentação de um *frontend* amigável e simples, que facilite a interação do utilizador com a aplicação. Tem também um *backend* capaz de funcionar em simultâneo, permitindo assim a criação de reservas e anúncios em qualquer lugar.

Como trabalho futuro, sugere-se o desenvolvimento de um módulo que permita o envio de notificações, sempre que uma reserva é criada e também um sistema de *feedback* para que os utilizadores possam avaliar os espaços onde estiveram.

6 Bibliografia

1. **O que é o Java? [Online] https://tecnoblog.net/responde/o-que-e-java-guia-para-iniciantes/. (acedido a 1 de Novembro de 2022)**
2. Firebase. [Online] https://firebase.google.com/?hl=pt-br. (acedido a 2 de Outubro de 2022)
3. Developers, Google. Google Maps. [Online] https://developers.google.com/maps?hl=pt-br. (acedido a 1 de Novembro de 2022)
4. Booking. Sobre Booking.com. [Online] https://www.booking.com/content/about.pt-pt.html. (acedido a 10 de Outubro de 2022)
5. Airbnb. Sobre a Airbnb: o que é e como funciona. [Online] https://www.airbnb.pt/help/article/2503. (acedido a 10 de Outubro de 2022)
6. Developers, Android. Conheça o Android Studio. [Online] https://developer.android.com/studio/intro?hl=pt-br#project_structure. (acedido a 1 de Novembro de 2022)
7. Gesvalt | Consultoría, Valoración, Tasación Inmobiliaria. [Online] https://gesvalt.pt/blog/consequencias-do-covid-19-no-sector-do-turismo-em-portugal-dados-atuais-e-perspetivas. (acedido a 14 de setembro de 2022)
8. Fadel, A.C. Metodologias ágeis no contexto de desenvolvimento de software: XP, Scrum e Lean. 2010. (acedido a 1 de dezembro de 2022)
9. *SCRUM model for agile methodology.* Srivastava, Apoorva, Bhardwaj, Sukriti e Saraswat, Shipra. 2017. (acedido a 1 de dezembro de 2022)
10. *scrum.org.* [Online] https://www.scrum.org/resources/what-is-scrum. (acedido a 4 de dezembro de 2022)
11. *Userstorymap.* [Online] https://www.userstorymap.io/agile-product-management-an-introduction/. (acedido a 4 de dezembro de 2022)
12. *Google Play.* [Online] https://play.google.com/store/apps/details?id=com.mitula.homes. (acedido a 10 de fevereiro de 2023)
13. *Google Play.* [Online] https://play.google.com/store/apps/details?id=com.trulia.android.rentals. (acedido a 10 de fevereiro de 2023)
14. *Publituris.* [Online] https://www.publituris.pt/2022/08/31/alojamento-turistico-cresce-acima-de-2019-em-hospedes-e-dormidas-em-julho. (acedido a 14 de setembro de 2022)
15. *Publituris.* [Online] https://www.publituris.pt/2022/02/10/alojamento-local-veio-diversificar-oferta-turistica-em-portugal. (acedido a 14 de setembro de 2022)

7 Anexos

7.1 Código

7.1.1 Adicionar Anúncio

```
package com.example.aplicacaoturismo;

import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.content.Intent;
import android.location.Address;
import android.location.Geocoder;
import android.net.Uri;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;

public class CriarAnuncioActivity extends AppCompatActivity {
    private static final int PICK_IMAGE_REQUEST =1;
    private Button buttonGravar, buttonCancelar, buttonDataPicker;
```

```
    private EditText editTextTitulo, editTextLocalidade,
editTextMorada, editTextCPostal, editTextNrPessoas, editTextInfo,
editTextContacto, editTextDataEntrada, editTextPrecoPornoite;
    private Uri imageuri;
    private ImageView imageView;
    private DatePickerDialog datePickerDialog;
    double lat, log;
    Spinner spinner;
    FirebaseAuth firebaseAuth;
    FirebaseDatabase firebaseDatabase;
    DatabaseReference databaseReference;
    List<String> listcidades;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_criar_anuncio);
        initDataPicker();
        /*buttonChoose = findViewById(R.id.buttonChooseImage);
        buttonUpload = findViewById(R.id.buttonUploadImage);
        editText = findViewById(R.id.editTextFileName);
        imageView = findViewById(R.id.imageView3);*/
        databaseReference =
FirebaseDatabase.getInstance().getReference();
        buttonDataPicker = findViewById(R.id.button3);
        buttonDataPicker.setText(getTodayDate());
        editTextTitulo = findViewById(R.id.editTextTituloAnuncio);
        spinner = findViewById(R.id.spinnerCidades);
        editTextMorada = findViewById(R.id.editTextMorada);
        editTextCPostal = findViewById(R.id.editTextCP);
        editTextContacto = findViewById(R.id.editTextContacto);
        editTextInfo = findViewById(R.id.editTextInformacoes);
        editTextNrPessoas = findViewById(R.id.editTextNrPessoas);
        editTextPrecoPornoite =
findViewById(R.id.editTextPrecoNoite);
        buttonGravar = findViewById(R.id.buttonGravar);
        buttonCancelar = findViewById(R.id.buttonCancelar);
        listcidades =  new ArrayList<>();
        firebaseAuth = FirebaseAuth.getInstance();
        firebaseDatabase = FirebaseDatabase.getInstance();

databaseReference.child("Cidades").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                for(DataSnapshot chilSnapshot:
snapshot.getChildren()){
                    String spinnerName =
chilSnapshot.child("nome").getValue(String.class);
                    listcidades.add(spinnerName);
                };
                ArrayAdapter<String> arrayAdapter = new
```

```
ArrayAdapter<>(CriarAnuncioActivity.this,
android.R.layout.simple_spinner_item, listcidades);

arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinn
er_item);
                spinner.setAdapter(arrayAdapter);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error)
{

            }
        });
        buttonCancelar.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
        Intent intent = new Intent(getBaseContext(),
MainActivity.class);
        startActivity(intent);
        }
        });


        buttonDataPicker.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                datePickerDialog.show();
            }
        });

        buttonGravar.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                saveData();
            }
        });

    }
    private String getTodayDate(){
        Calendar cal = Calendar.getInstance();
        int year = cal.get(Calendar.YEAR);
        int month = cal.get(Calendar.MONTH);
        int day= cal.get(Calendar.DAY_OF_MONTH);
        month =  month + 1;
        return makeDataString(day, month, year);
    }

    private void initDataPicker(){
```

```
        DatePickerDialog.OnDateSetListener dateSetListener = new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker datePicker, int year,
int month, int day) {
                month = month +1;
                Date currentTime =
Calendar.getInstance().getTime();
                long startDate = currentTime.getTime();
                String date = makeDataString(day, month, year);
                datePicker.setMinDate(startDate);
                buttonDataPicker.setText(date);
            }
        };
        Calendar cal = Calendar.getInstance();
        int year = cal.get(Calendar.YEAR);
        int month = cal.get(Calendar.MONTH);
        int day= cal.get(Calendar.DAY_OF_MONTH);

        datePickerDialog = new DatePickerDialog(this,
dateSetListener, year, month, day);
    }
    private String makeDataString(int day, int mount, int year){
        return day + "" + getMonthFormat(mount) + "" + year;
    }

    private String getMonthFormat(int month){
        switch (month){
            case 1: return "Jan";
            case 2: return "Fev";
            case 3: return "Mar";
            case 4: return "Abr";
            case 5: return "Mai";
            case 6: return "Jun";
            case 7: return "Jul";
            case 8: return "Ago";
            case 9: return "Set";
            case 10: return "Out";
            case 11: return "Nov";
            case 12: return "Dez";
        }
        return "";
    }


    @Override
    protected void onActivityResult(int requestCode, int
resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if(requestCode == PICK_IMAGE_REQUEST && requestCode ==
RESULT_OK && data != null && data.getData() != null){
```

```
            imageuri = data.getData();
            Log.d(TAG,"image upload");
            Picasso.with(this).load(imageuri).into(imageView);
        }

    }

    private void openFileChooser(){
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(intent, PICK_IMAGE_REQUEST);
    }


    private void saveData(){
        String mtitulo =
editTextTitulo.getText().toString().trim();
        String mlocalidade = spinner.getSelectedItem().toString();
        String minformacao =
editTextInfo.getText().toString().trim();
        String mdatadeentrada =
buttonDataPicker.getText().toString();
        String mpreco =
editTextPrecoPornoite.getText().toString().trim();
        String mnrpessoas =
editTextNrPessoas.getText().toString().trim();
        String mCpostal =
editTextCPostal.getText().toString().trim();
        String mMorada =
editTextMorada.getText().toString().trim();
        String mContacto =
editTextContacto.getText().toString().trim();

        if(TextUtils.isEmpty(mtitulo)){
            editTextTitulo.setError("Por favor preencher o
campo!");
            editTextTitulo.requestFocus();
            return;
        }

        if(TextUtils.isEmpty(mContacto) || mContacto.length() !=
9){
            editTextContacto.setError("Informação Inválida");
            editTextContacto.requestFocus();
            return;
        }

        if(TextUtils.isEmpty(mMorada)){
            editTextMorada.setError("Por favor preencher o
campo!");
            editTextMorada.requestFocus();
```

```
            return;
        }
        if(TextUtils.isEmpty(mCpostal)){
            editTextCPostal.setError("Informação Inválida");
            editTextCPostal.requestFocus();
            return;
        }
        if(TextUtils.isEmpty(mnrpessoas)){
            editTextNrPessoas.setError("Por favor preencher o
campo!");
            editTextNrPessoas.requestFocus();
            return;
        }


        Geocoder geocoder = new Geocoder(getBaseContext());
        List<Address> addressList;
        try {
            addressList =
geocoder.getFromLocationName(mCpostal,1);
            if(addressList.size() > 0){
                log = addressList.get(0).getLongitude();
                lat = addressList.get(0).getLatitude();
            }else{
                log = 0;
                lat = 0;
            }


            Log.d(TAG, "geoLocate: found a location: " +
addressList.toString());

        }catch (Exception e){
            log = 0;
            lat = 0;
            Log.d(TAG, "error " + e);
        }


        String user =
FirebaseAuth.getInstance().getCurrentUser().getUid();
        long randomID = System.currentTimeMillis();
        DatabaseReference reference =
firebaseDatabase.getReference("anuncio").child(user).child(""+rand
omID);

        HashMap<String, String> anuncios = new HashMap<>();
        anuncios.put("AnuncioID", ""+randomID);
        anuncios.put("Titulo", mtitulo);
        anuncios.put("Localidade", mlocalidade);
        anuncios.put("Morada", mMorada);
        anuncios.put("CPostal", mCpostal);
```

```
            anuncios.put("Informacao", minformacao);
            anuncios.put("DatadeEntrada", mdatadeentrada);
            anuncios.put("Preco", mpreco);
            anuncios.put("NrPessoas", mnrpessoas);
            anuncios.put("Contacto", mContacto);
            anuncios.put("Estado", "Disponivel");
            anuncios.put("Criadoem", "" + getTodayDate());
            anuncios.put("lat", "" +lat);
            anuncios.put("log", "" +log);
            reference.setValue(anuncios);

            Intent intent = new Intent(this, MainActivity.class);
            startActivity(intent);
        }
}
```

7.1.2 Mostrar Anúncios

```
import static android.content.ContentValues.TAG;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.aplicacaoturismo.AnuncioAdapter;
import com.example.aplicacaoturismo.AnuncioModel;
import com.example.aplicacaoturismo.R;
import
com.example.aplicacaoturismo.databinding.FragmentHomeBinding;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class  HomeFragment extends Fragment {
```

```
    private FragmentHomeBinding binding;

    TextView textViewname, textViewemail;
    String userId;
    FirebaseAuth firebaseAuth;
    DatabaseReference databaseReference;
    Button details;
    AnuncioAdapter anuncioAdapter;
    ArrayList<AnuncioModel> list;
    RecyclerView recyclerView;
    FirebaseDatabase db = FirebaseDatabase.getInstance();
    TextView textView2;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {
        HomeViewModel homeViewModel =
                new ViewModelProvider(this, new
ViewModelProvider.NewInstanceFactory()).get(HomeViewModel.class);

        binding = FragmentHomeBinding.inflate(inflater, container,
false);
        View root = binding.getRoot();

        userId =
firebaseAuth.getInstance().getCurrentUser().getUid();
        databaseReference =
FirebaseDatabase.getInstance().getReference();

        final TextView textView = binding.textHome;
        homeViewModel.getText().observe(getViewLifecycleOwner(),
textView::setText);
        recyclerView =
root.findViewById(R.id.recyclerViewAnuncio);
        details = root.findViewById(R.id.buttonDetalhes);
        databaseReference = db.getReference("anuncio");
        recyclerView.setLayoutManager(new
LinearLayoutManager(getContext()));
        textView2 = root.findViewById(R.id.text_home);
        list = new ArrayList<>();
        anuncioAdapter = new AnuncioAdapter(getContext(), list);
        recyclerView.setAdapter(anuncioAdapter);
        databaseReference.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {

                for(DataSnapshot dataSnapshot :
snapshot.getChildren()){
                    if(dataSnapshot.getKey().equals(userId)){
                                textView2.setText("Sem anuncios
```

```
para mostrar");
                    }else{
                        for(DataSnapshot dataSnapshot2 :
dataSnapshot.getChildren()){
                            textView2.setText("");
                            AnuncioModel anuncioModel =
dataSnapshot2.getValue(AnuncioModel.class);
                            list.add(anuncioModel);
                        }
                    }

                }
                anuncioAdapter.notifyDataSetChanged();

            }


            @Override
            public void onCancelled(@NonNull DatabaseError error)
{
            }

        });

        return root;
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}
```

7.1.3 AnuncioModel

```
public class AnuncioModel {

        private String AnuncioID;
        private String Titulo;

    public String getAnuncioID() {
        return AnuncioID;
    }

    public void setAnuncioID(String anuncioID) {
        AnuncioID = anuncioID;
    }

    public String getTitulo() {
```

```
        return Titulo;
    }

    public void setTitulo(String titulo) {
        Titulo = titulo;
    }

    public String getLocalidade() {
        return Localidade;
    }

    public void setLocalidade(String localidade) {
        Localidade = localidade;
    }

    public String getMorada() {
        return Morada;
    }

    public void setMorada(String morada) {
        Morada = morada;
    }

    public String getCPostal() {
        return CPostal;
    }

    public void setCPostal(String CPostal) {
        this.CPostal = CPostal;
    }

    public String getInformacao() {
        return Informacao;
    }

    public void setInformacao(String informacao) {
        Informacao = informacao;
    }

    public String getDatadeEntrada() {
        return DatadeEntrada;
    }

    public void setDatadeEntrada(String datadeEntrada) {
        DatadeEntrada = datadeEntrada;
    }

    public String getPreco() {
        return Preco;
    }

    public void setPreco(String preco) {
```

```
        Preco = preco;
    }

    public String getNrPessoas() {
        return NrPessoas;
    }

    public void setNrPessoas(String nrPessoas) {
        NrPessoas = nrPessoas;
    }

    public String getContacto() {
        return Contacto;
    }

    public void setContacto(String contacto) {
        Contacto = contacto;
    }

    public String getEstado() {
        return Estado;
    }

    public void setEstado(String estado) {
        Estado = estado;
    }

    public String getCriadoem() {
        return Criadoem;
    }

    public void setCriadoem(String criadoem) {
        Criadoem = criadoem;
    }

    public String getLat() {
        return lat;
    }

    public void setLat(String lat) {
        this.lat = lat;
    }

    public String getLog() {
        return log;
    }

    public void setLog(String log) {
        this.log = log;
    }

    private String Localidade;
```

```
        private String Morada;
        private String CPostal;
        private String Informacao;
        private String DatadeEntrada;
        private String Preco;
        private String NrPessoas;
        private String Contacto;
        private String Estado;
        private String Criadoem;
        private String lat;
        private String log;
}
```

7.1.4 Anúncio Adpater

```
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.gms.maps.MapView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.storage.StorageReference;

import java.util.ArrayList;

public class AnuncioAdapter extends
RecyclerView.Adapter<AnuncioAdapter.MyViewHolder> {

    Context context;
    ArrayList<AnuncioModel> list;
    StorageReference firebaseStorage;
    public AnuncioAdapter(Context context,ArrayList<AnuncioModel>
list ){
        this.context = context;
        this.list = list;
    }

    @NonNull
    @Override
    public AnuncioAdapter.MyViewHolder onCreateViewHolder(@NonNull
```

```
ViewGroup parent, int viewType) {
        View v =
LayoutInflater.from(context).inflate(R.layout.anuncioitem,parent,f
alse);
        AnuncioAdapter.MyViewHolder myViewHolder = new
AnuncioAdapter.MyViewHolder(v);


        return myViewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull
AnuncioAdapter.MyViewHolder holder, int position) {

        AnuncioAdapter.MyViewHolder myViewHolder=
(AnuncioAdapter.MyViewHolder)holder;

        AnuncioModel anuncio = list.get(position);
        holder.Titulo.setText(anuncio.getTitulo());
        holder.Localidade.setText(anuncio.getLocalidade());
        holder.NrPessoas.setText(anuncio.getNrPessoas());
        holder.Preco.setText(anuncio.getPreco());
        holder.DataDeEntrada.setText(anuncio.getDatadeEntrada());
        holder.ID.setText(anuncio.getAnuncioID());


        holder.detalhes.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(holder.ID.getContext(),
DetailsActivity.class);
                intent.putExtra("id",
list.get(position).getAnuncioID());
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

                holder.ID.getContext().startActivity(intent);
            }
        });
    }

    @Override
    public int getItemCount() {
        return list.size();
    }

    public  static  class MyViewHolder extends
RecyclerView.ViewHolder{


        TextView Titulo, Localidade, NrPessoas, Preco,
```

```
DataDeEntrada, ID;
        Button detalhes;
        public MyViewHolder(@NonNull View itemView) {
            super(itemView);

            Titulo =itemView.findViewById(R.id.textView14);
            Localidade
=itemView.findViewById(R.id.textViewLocalidade);
            NrPessoas
=itemView.findViewById(R.id.textViewNrPessoas);
            Preco =itemView.findViewById(R.id.textViewPreco);
            DataDeEntrada
=itemView.findViewById(R.id.textViewData);
            detalhes = itemView.findViewById(R.id.buttonDetalhes);
            ID = itemView.findViewById(R.id.textViewAnuncioID);

        }
    }

}
```

7.1.5 Detalhes do Anúncio

```
import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.model.LatLng;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
```

```
;

import java.util.ArrayList;

public class DetailsActivity extends AppCompatActivity implements
OnMapReadyCallback {


    TextView edittextViewAnuncioID, textViewTitulo2,
textViewLocalidadeD, textViewDataD,
textViewNrPessoasD,textViewPrecoD,textViewInfo,textViewMorada,text
ViewContacto,textViewCodigoPostal;
    MapView mapView;
    Button buttonReservar, buttonCanelar, buttonPontosdeInteresse;
    String lat, log;
    FirebaseAuth firebaseAuth;
    DatabaseReference databaseReference;
    FirebaseDatabase db = FirebaseDatabase.getInstance();
    LatLng coordenadas;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_details);
        String id = getIntent().getStringExtra("id").toString();
        Log.d(TAG,"aaaaaaaaaaaaaaaaaa"+ id);
        edittextViewAnuncioID =
findViewById(R.id.edittextViewAnuncioID);
        textViewTitulo2 = findViewById(R.id.textViewTitulo2);
        textViewLocalidadeD =
findViewById(R.id.textViewLocalidadeD);
        textViewDataD = findViewById(R.id.textViewDataD);
        textViewNrPessoasD =
findViewById(R.id.textViewNrPessoasD);
        textViewPrecoD = findViewById(R.id.textViewPrecoD);
        textViewInfo = findViewById(R.id.textViewInfo);
        textViewMorada = findViewById(R.id.textViewMorada);
        textViewContacto = findViewById(R.id.textViewContacto);
        textViewCodigoPostal =
findViewById(R.id.textViewCodigoPostal);
        buttonReservar = findViewById(R.id.buttonReservar);
        buttonCanelar = findViewById(R.id.buttonVoltar);
        mapView = findViewById(R.id.mapView);
        buttonPontosdeInteresse =
findViewById(R.id.buttonPontosdeInteresse);
        databaseReference = db.getReference("anuncio");

        databaseReference.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                for(DataSnapshot dataSnapshot :
```

```
snapshot.getChildren()){
                        for(DataSnapshot dataSnapshot2 :
dataSnapshot.getChildren()){
                            if(dataSnapshot2.getKey().equals(id)){
                                String tempAnuncioID
=""+dataSnapshot2.child("AnuncioID").getValue();
                                String tempCPostal
=""+dataSnapshot2.child("CPostal").getValue();
                                String tempContacto
=""+dataSnapshot2.child("Contacto").getValue();
                                String tempDatadeEntrada
=""+dataSnapshot2.child("DatadeEntrada").getValue();
                                String tempInformacao
=""+dataSnapshot2.child("Informacao").getValue();
                                String tempLocalidade
=""+dataSnapshot2.child("Localidade").getValue();
                                String tempMorada
=""+dataSnapshot2.child("Morada").getValue();
                                String tempNrPessoas
=""+dataSnapshot2.child("NrPessoas").getValue();
                                String tempPreco
=""+dataSnapshot2.child("Preco").getValue();
                                String tempTitulo
=""+dataSnapshot2.child("Titulo").getValue();
                                String templat
=""+dataSnapshot2.child("lat").getValue();
                                String templog
=""+dataSnapshot2.child("log").getValue();



edittextViewAnuncioID.setText(tempAnuncioID);

textViewTitulo2.setText(tempTitulo);

textViewLocalidadeD.setText(tempLocalidade);

textViewDataD.setText(tempDatadeEntrada);

textViewNrPessoasD.setText(tempNrPessoas);
                                textViewPrecoD.setText(tempPreco);

textViewInfo.setText(tempInformacao);

textViewMorada.setText(tempMorada);

textViewContacto.setText(tempContacto);

textViewCodigoPostal.setText(tempCPostal);
                                Log.d(TAG,""+ templat);
                                lat = templat;
```

```
                                log = templog;

                            }
                            Log.d(TAG,""+ dataSnapshot2);

                        }
                    }

                }


            @Override
            public void onCancelled(@NonNull DatabaseError error)
{

            }
        });

        mapView.getMapAsync(this);
        mapView.onCreate(savedInstanceState);
        mapView.onResume();

        buttonReservar.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(DetailsActivity.this,
CriarReserva.class);
                intent.putExtra("AnuncioID",
edittextViewAnuncioID.getText());
                intent.putExtra("tempTitulo",
textViewTitulo2.getText());
                intent.putExtra("tempLocalidade",
textViewLocalidadeD.getText());
                intent.putExtra("tempDatadeEntrada",
textViewDataD.getText());
                intent.putExtra("tempNrPessoas",
textViewNrPessoasD.getText());
                intent.putExtra("tempPreco",
textViewPrecoD.getText());
                intent.putExtra("tempInformacao",
textViewInfo.getText());
                intent.putExtra("tempMorada",
textViewMorada.getText());
                intent.putExtra("tempContacto",
textViewContacto.getText());
                intent.putExtra("tempCPostal",
textViewCodigoPostal.getText());
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            }
```

```
        });

        buttonPontosdeInteresse.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(DetailsActivity.this,
pontosdeinteresse_activity.class);
                intent.putExtra("Cidade",
textViewLocalidadeD.getText());
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            }
        });

    }

    @Override
    public void onMapReady(@NonNull GoogleMap googleMap) {
        Double lati = Double.parseDouble(lat);
        Double longe = Double.parseDouble(log);
        coordenadas = new LatLng(lati, longe);
        googleMap.addMarker(new
MarkerOptions().position(coordenadas).title("Localização do
anuncio"));

googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(coordenadas
,15));
    }
}
```

7.1.6 Pontos de Interesse

```
import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.location.LocationProvider;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
```

```
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class pontosdeinteresse_activity extends AppCompatActivity
implements OnMapReadyCallback {

    public static final int ROUND = 10;
    private GoogleMap mMap;
    Marker marker;
    private LocationProvider locationProvider;
    DatabaseReference databaseReference;
    FirebaseDatabase db = FirebaseDatabase.getInstance();
    LatLng coordenadas;
    MapView mapView;
    MarkerOptions markerOptions;
    double lat, log;
    String desc, lat1, lat2, descComplete;
    String Cidade;
    TextView markerDetails;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pontosdeinteresse);
        Cidade = getIntent().getStringExtra("Cidade").toString();

        mapView = findViewById(R.id.mapView2);
        markerDetails = findViewById(R.id.markerDetails);
        mapView.getMapAsync( this);
        mapView.onCreate(savedInstanceState);
        mapView.onResume();
    }

    @Override
    public void onMapReady(@NonNull GoogleMap googleMap1) {

        databaseReference =
db.getReference("pontosdeinteresse").child(Cidade);

        databaseReference.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                for(DataSnapshot dataSnapshot :
snapshot.getChildren()) {
```

```
                    desc
=""+dataSnapshot.child("Desc").getValue();
                    lat1 = "" +
dataSnapshot.child("lat").getValue();
                    lat2 = ""
+dataSnapshot.child("long").getValue();
                    Double lati = Double.parseDouble(lat1);
                    Double longe = Double.parseDouble(lat2);
                    LatLng latLng = new LatLng(lati, longe);
                    markerOptions = new MarkerOptions();

markerOptions.position(latLng).icon(BitmapDescriptorFactory.defaul
tMarker(BitmapDescriptorFactory.HUE_RED)).title(desc);
                    marker = googleMap1.addMarker(markerOptions);

googleMap1.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,15)
);

                }
            }


            @Override
            public void onCancelled(@NonNull DatabaseError error)
{

            }
        });
        googleMap1.setOnMarkerClickListener(new
GoogleMap.OnMarkerClickListener() {
            @Override
            public boolean onMarkerClick(Marker marker) {
                String markerName = marker.getTitle();
                databaseReference =
db.getReference("pontosdeinteresse").child(Cidade);
                databaseReference.addValueEventListener(new
ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot
snapshot) {
                        for(DataSnapshot dataSnapshot :
snapshot.getChildren()) {
                            String monumento = ""
+dataSnapshot.child("Desc").getValue();
                            Log.d(TAG, "Monumento " + monumento +
"   " + markerName);
                            if(monumento.equals(markerName)){
                                descComplete
=""+dataSnapshot.child("DescComplete").getValue();

markerDetails.setText(descComplete);
                            }
```

```
                        }
                    }


                    @Override
                    public void onCancelled(@NonNull DatabaseError
error) {

                    }
                });
                return false;
            }
        });

    }
}
```

7.1.7 Criar Reserva

```
import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.content.Intent;
import android.location.Address;
import android.location.Geocoder;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;

public class CriarReserva extends AppCompatActivity {
    TextView edittextViewAnuncioID, textViewTitulo2,
textViewLocalidadeD, textViewDataD,
textViewNrPessoasD,textViewPrecoD,textViewInfo,textViewMorada,text
ViewContacto,textViewCodigoPostal;
    private Button buttonGravar, buttonCancelar, buttondatasaida;
    private EditText editTextNome, editTextContacto,
editTextContribuinte;
    private DatePickerDialog datePickerDialog;
    Spinner spinner;
    FirebaseAuth firebaseAuth;
    FirebaseDatabase firebaseDatabase;
    DatabaseReference databaseReference;
    List<String> listpagamento;
    double lat, log;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_criar_reserva);
        initDataPicker();
        String id =
getIntent().getStringExtra("AnuncioID").toString();
        String Titulo =
getIntent().getStringExtra("tempTitulo").toString();
        String Localidade =
getIntent().getStringExtra("tempLocalidade").toString();
        String DatadeEntrada =
getIntent().getStringExtra("tempDatadeEntrada").toString();
        String NrPessoas =
getIntent().getStringExtra("tempNrPessoas").toString();
        String preco =
getIntent().getStringExtra("tempPreco").toString();
        String Informacao =
getIntent().getStringExtra("tempInformacao").toString();
        String Morada =
getIntent().getStringExtra("tempMorada").toString();
        String Contacto =
getIntent().getStringExtra("tempContacto").toString();
        String CPostal =
getIntent().getStringExtra("tempCPostal").toString();
        databaseReference =
FirebaseDatabase.getInstance().getReference();
```

```
        buttondatasaida = findViewById(R.id.buttondatasaida);
        buttondatasaida.setText(getTodayDate());
        editTextNome = findViewById(R.id.editTextNomeUtilizador);
        spinner = findViewById(R.id.spinnerPagamento);
        editTextContacto = findViewById(R.id.editTextContacto);
        editTextContribuinte =
findViewById(R.id.editTextContribuinte);
        buttonGravar = findViewById(R.id.buttonGravarReserva);
        buttonCancelar = findViewById(R.id.buttonCancelar);
        listpagamento =  new ArrayList<>();
        firebaseAuth = FirebaseAuth.getInstance();
        firebaseDatabase = FirebaseDatabase.getInstance();

        edittextViewAnuncioID =
findViewById(R.id.edittextViewAnuncioID1);
        textViewTitulo2 = findViewById(R.id.textViewTitulo3);
        textViewLocalidadeD =
findViewById(R.id.textViewLocalidadeD1);
        textViewDataD = findViewById(R.id.textViewDataD1);
        textViewNrPessoasD =
findViewById(R.id.textViewNrPessoasD1);
        textViewPrecoD = findViewById(R.id.textViewPrecoD1);
        textViewInfo = findViewById(R.id.textViewInfo1);
        textViewMorada = findViewById(R.id.textViewMorada1);
        textViewContacto = findViewById(R.id.textViewContacto1);
        textViewCodigoPostal =
findViewById(R.id.textViewCodigoPostal1);

        edittextViewAnuncioID.setText(id);
        textViewTitulo2.setText(Titulo);
        textViewLocalidadeD.setText(Localidade);
        textViewDataD.setText(DatadeEntrada);
        textViewNrPessoasD.setText(NrPessoas);
        textViewPrecoD.setText(preco);
        textViewInfo.setText(Informacao);
        textViewMorada.setText(Morada);
        textViewContacto.setText(Contacto);
        textViewCodigoPostal.setText(CPostal);



databaseReference.child("Pagamento").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                for(DataSnapshot chilSnapshot:
snapshot.getChildren()){
                    String spinnerName =
chilSnapshot.child("nome").getValue(String.class);
                    listpagamento.add(spinnerName);
                };
```

```
                ArrayAdapter<String> arrayAdapter = new
ArrayAdapter<>(CriarReserva.this,
android.R.layout.simple_spinner_item, listpagamento);

arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinn
er_item);
                spinner.setAdapter(arrayAdapter);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error)
{

            }
        });
        buttondatasaida.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                datePickerDialog.show();
            }
        });

        buttonGravar.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                savedata();
            }
        });

    }
    private String getTodayDate(){
        Calendar cal = Calendar.getInstance();
        int year = cal.get(Calendar.YEAR);
        int month = cal.get(Calendar.MONTH);
        int day= cal.get(Calendar.DAY_OF_MONTH);
        month =  month + 1;
        return makeDataString(day, month, year);
    }

    private void initDataPicker(){
        DatePickerDialog.OnDateSetListener dateSetListener = new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker datePicker, int year,
int month, int day) {
                month = month +1;
                String date = makeDataString(day, month, year);
                buttondatasaida.setText(date);

       String mdatadeentrada =
```

```
getIntent().getStringExtra("tempDatadeEntrada").toString();;
        String datadeentrada = convertDataString(mdatadeentrada);

        SimpleDateFormat formatter = new
SimpleDateFormat("dd/MM/yyyy");

        Date date1 = null;
        try {
            date1 = formatter.parse(datadeentrada);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        long startDate = date1.getTime();
        datePicker.setMinDate(startDate);
            }
        };
        Calendar cal = Calendar.getInstance();
        int year = cal.get(Calendar.YEAR);
        int month = cal.get(Calendar.MONTH);
        int day= cal.get(Calendar.DAY_OF_MONTH);

        datePickerDialog = new DatePickerDialog(this,
dateSetListener, year, month, day);
    }
    private String makeDataString(int day, int mount, int year){
        return day + "" + getMonthFormat(mount) + "" + year;
    }
    private String convertDataString(String data){

        String mounth =
String.valueOf(formatDate(data.substring(1, 4)));
        String month = data.substring(1,4);

        data = data.replace(month, mounth);
        String dia = data.substring(0,1);
        String mes =  mounth;
        String ano = data.substring(4);

        String newdate = "0" + dia + "/0" + mounth + "/20" + ano;
        return newdate;
    }

    private int formatDate(String month){

        switch (month){
            case "Jan": return 1;
            case "Fev": return 2;
            case "Mar": return 3;
            case "Abr": return 4;
            case "Mai": return 5;
            case "Jun": return 6;
            case "Jul": return 7;
```

```
            case "Ago": return 8;
            case "Set": return 9;
            case "Out": return 10;
            case "Nov": return 11;
            case "Dez": return 12;
        }

        return 0;
    }

    private String getMonthFormat(int month){
        switch (month){
            case 1: return "Jan";
            case 2: return "Fev";
            case 3: return "Mar";
            case 4: return "Abr";
            case 5: return "Mai";
            case 6: return "Jun";
            case 7: return "Jul";
            case 8: return "Ago";
            case 9: return "Set";
            case 10: return "Out";
            case 11: return "Nov";
            case 12: return "Dez";
        }
        return "";
    }
    private void savedata(){
        String mId =
edittextViewAnuncioID.getText().toString().trim();
        String mtitulo =
textViewTitulo2.getText().toString().trim();
        String mLocalidade =
textViewLocalidadeD.getText().toString().trim();
        String mdatadeentrada =
textViewDataD.getText().toString();
        String mpreco =
textViewPrecoD.getText().toString().trim();
        String mnrpessoas =
textViewNrPessoasD.getText().toString().trim();
        String minformacao =
textViewInfo.getText().toString().trim();
        String mMorada =
textViewMorada.getText().toString().trim();
        String mContacto =
textViewContacto.getText().toString().trim();
        String mCpostal =
textViewCodigoPostal.getText().toString().trim();
        String mNome = editTextNome.getText().toString().trim();
        String mContactoUser =
editTextContacto.getText().toString().trim();
        String mPagamento = spinner.getSelectedItem().toString();
```

```
        String mDatadeSaida =
buttondatasaida.getText().toString();
        String mContribuinteUser =
editTextContribuinte.getText().toString().trim();


if(TextUtils.isEmpty(mNome)){
    editTextNome.setError("Por favor preencher o campo!");
    editTextNome.requestFocus();
    return;
}

if(TextUtils.isEmpty(mContactoUser) || mContacto.length() != 9){
    editTextContacto.setError("Informação Inválida");
    editTextContacto.requestFocus();
    return;
}

if(TextUtils.isEmpty(mContribuinteUser)){
    editTextContribuinte.setError("Por favor preencher o campo!");
    editTextContribuinte.requestFocus();
    return;
}


        Geocoder geocoder = new Geocoder(getBaseContext());
        List<Address> addressList;
        try {
            addressList =
geocoder.getFromLocationName(mCpostal,1);
            if(addressList.size() > 0){
                log = addressList.get(0).getLongitude();
                lat = addressList.get(0).getLatitude();
            }else{
                log = 0;
                lat = 0;
            }


            Log.d(TAG, "geoLocate: found a location: " +
addressList.toString());

        }catch (Exception e){
            log = 0;
            lat = 0;
            Log.d(TAG, "error " + e);
        }


        String user =
FirebaseAuth.getInstance().getCurrentUser().getUid();
        long randomID = System.currentTimeMillis();
```

```
        DatabaseReference reference =
firebaseDatabase.getReference("reserva").child(user).child(""+rand
omID);

        HashMap<String, String> reserva = new HashMap<>();
        reserva.put("rReservaID", ""+randomID);
        reserva.put("rAnuncioID", ""+mId);
        reserva.put("rTitulo", mtitulo);
        reserva.put("rLocalidade", mLocalidade);
        reserva.put("rMorada", mMorada);
        reserva.put("rCPostal", mCpostal);
        reserva.put("rInformacao", minformacao);
        reserva.put("rDatadeEntrada", mdatadeentrada);
        reserva.put("rPreco", mpreco);
        reserva.put("rNrPessoas", mnrpessoas);
        reserva.put("rContacto", mContacto);
        reserva.put("rNomeP", mNome);
        reserva.put("rContactoUser", mContactoUser);
        reserva.put("rPagamento", mPagamento);
        reserva.put("rDatadeSaida", mDatadeSaida);
        reserva.put("rContribuinteUser", mContribuinteUser);
        reserva.put("rCriadoem", "" + getTodayDate());
        reserva.put("rlat", "" +lat);
        reserva.put("rlog", "" +log);
        reference.setValue(reserva);

        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
}
```

7.1.8 ResevaModel

```
public class ReservaModel {



    public String getrAnuncioID() {
        return rAnuncioID;
    }

    public void setrAnuncioID(String rAnuncioID) {
        this.rAnuncioID = rAnuncioID;
    }

    public String getrTitulo() {
        return rTitulo;
    }

    public void setrTitulo(String rTitulo) {
        this.rTitulo = rTitulo;
```

```
}

public String getrLocalidade() {
    return rLocalidade;
}

public void setrLocalidade(String rLocalidade) {
    this.rLocalidade = rLocalidade;
}

public String getrMorada() {
    return rMorada;
}

public void setrMorada(String rMorada) {
    this.rMorada = rMorada;
}

public String getrCPostal() {
    return rCPostal;
}

public void setrCPostal(String rCPostal) {
    this.rCPostal = rCPostal;
}

public String getrInformacao() {
    return rInformacao;
}

public void setrInformacao(String rInformacao) {
    this.rInformacao = rInformacao;
}

public String getrDatadeEntrada() {
    return rDatadeEntrada;
}

public void setrDatadeEntrada(String rDatadeEntrada) {
    this.rDatadeEntrada = rDatadeEntrada;
}

public String getrPreco() {
    return rPreco;
}

public void setrPreco(String rPreco) {
    this.rPreco = rPreco;
}

public String getrNrPessoas() {
    return rNrPessoas;
```

```
}

public void setrNrPessoas(String rNrPessoas) {
    this.rNrPessoas = rNrPessoas;
}

public String getrContacto() {
    return rContacto;
}

public void setrContacto(String rContacto) {
    this.rContacto = rContacto;
}

public String getrNomeP() {
    return rNomeP;
}

public void setrNomeP(String rNomeP) {
    this.rNomeP = rNomeP;
}

public String getrContactoUser() {
    return rContactoUser;
}

public void setrContactoUser(String rContactoUser) {
    this.rContactoUser = rContactoUser;
}

public String getrPagamento() {
    return rPagamento;
}

public void setrPagamento(String rPagamento) {
    this.rPagamento = rPagamento;
}

public String getrDatadeSaida() {
    return rDatadeSaida;
}

public void setrDatadeSaida(String rDatadeSaida) {
    this.rDatadeSaida = rDatadeSaida;
}

public String getrContribuinteUser() {
    return rContribuinteUser;
}

public void setrContribuinteUser(String rContribuinteUser) {
    this.rContribuinteUser = rContribuinteUser;
```

```
}

public String getrCriadoem() {
    return rCriadoem;
}

public void setrCriadoem(String rCriadoem) {
    this.rCriadoem = rCriadoem;
}

public String getRlat() {
    return rlat;
}

public void setRlat(String rlat) {
    this.rlat = rlat;
}

public String getRlog() {
    return rlog;
}

public void setRlog(String rlog) {
    this.rlog = rlog;
}

public String getrReservaID() {
    return rReservaID;
}

public void setrReservaID(String rReservaID) {
    this.rReservaID = rReservaID;
}

public String rReservaID;
    public String rAnuncioID;
    public String rTitulo;
    public String rLocalidade;
    public String rMorada;
    public String rCPostal;
    public String rInformacao;
    public String rDatadeEntrada;
    public String rPreco;
    public String rNrPessoas;
    public String rContacto;
    public String rNomeP;
    public String rContactoUser;
    public String rPagamento;
    public String rDatadeSaida;
    public String rContribuinteUser;
    public String rCriadoem;
    public String rlat;
```

```
        public String rlog;
}
```

7.1.9 Reserva Adapter

```
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.storage.StorageReference;

import java.util.ArrayList;

public class ReservaAdapter extends
RecyclerView.Adapter<ReservaAdapter.MyViewHolder> {

    Context context;
    ArrayList<ReservaModel> list;
    StorageReference firebaseStorage;
    public ReservaAdapter(Context context,ArrayList<ReservaModel>
list ){
        this.context = context;
        this.list = list;
    }

    @NonNull
    @Override
    public ReservaAdapter.MyViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View v =
LayoutInflater.from(context).inflate(R.layout.reserva_item,parent,
false);
        ReservaAdapter.MyViewHolder myViewHolder = new
ReservaAdapter.MyViewHolder(v);
        return myViewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull
ReservaAdapter.MyViewHolder holder, int position) {

        ReservaAdapter.MyViewHolder myViewHolder=
(ReservaAdapter.MyViewHolder)holder;
```

```
        ReservaModel reservaModel = list.get(position);
        holder.Titulo.setText(reservaModel.getrTitulo());
        holder.Localidade.setText(reservaModel.getrLocalidade());
        holder.NrPessoas.setText(reservaModel.getrNrPessoas());
        holder.Preco.setText(reservaModel.getrPreco());

holder.DataDeEntrada.setText(reservaModel.getrDatadeEntrada());
        holder.ReservaID.setText(reservaModel.getrReservaID());
        holder.AnuncioID.setText(reservaModel.getrAnuncioID());

holder.DataDeSaida.setText(reservaModel.getrDatadeSaida());

        holder.detalhes.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new
Intent(holder.ReservaID.getContext(), Reservas.class);
                intent.putExtra("id",
list.get(position).getrReservaID());
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);


holder.ReservaID.getContext().startActivity(intent);
            }
        });
    }

    @Override
    public int getItemCount() {
        return list.size();
    }

    public  static  class MyViewHolder extends
RecyclerView.ViewHolder{


        TextView Titulo, Localidade, NrPessoas, Preco,
DataDeEntrada, ReservaID, AnuncioID, DataDeSaida;
        Button detalhes;
        public MyViewHolder(@NonNull View itemView) {
            super(itemView);

            Titulo =itemView.findViewById(R.id.textView141);
            Localidade
=itemView.findViewById(R.id.textViewLocalidade1);
            NrPessoas
=itemView.findViewById(R.id.textViewNrPessoas1);
            Preco =itemView.findViewById(R.id.textViewPreco1);
            DataDeEntrada
=itemView.findViewById(R.id.textViewData1);
```

```
            detalhes =
itemView.findViewById(R.id.buttonReservaDetalhes);
            AnuncioID =
itemView.findViewById(R.id.textViewAnuncioID1);
            DataDeSaida =
itemView.findViewById(R.id.textViewDatadeSaida);
            ReservaID =
itemView.findViewById(R.id.textViewReservaID);

        }
    }

}
```