

P.PORTO

ESCOLA  
SUPERIOR  
DE TECNOLOGIA  
E GESTÃO

# Trabalho Prático de Computação Móvel e Ubíqua

Licenciatura em Engenharia Informática

2021/2022



Grupo 6

8200586, Bruno Ferreira

8200595, Gonçalo Oliveira

## Conteúdo

Identificação e caracterização do projeto	4
Ferramentas e tecnologias utilizadas	5
Opções estratégicas	6
Sistema de caching com Firebase e Room	6
Estruturação da base de dados Room	7
Entidade Districts	7
Entidade vehicles	8
Entidade Routes	9
Entidade SavedInterests	10
Entidade GasStations	11
Armazenamento em Firebase	12
Firebase CloudStorage	12
Estrutura de pastas	13
Layouts e respetivas funcionalidades	17
Login	17
Registo	18
Mapa	19
Pontos de Interesse	20
Postos de combustível	21
More	24
Tema	25
Preferência de navegação	26
Veículos	27
Rotas	29
Detalhes da viagem	30
Conexão à internet	31
Serviços	32
Testes à UI	32
API's utilizadas	33
Google Places API	33

Google Routes API	33
Google Geocoding API	33
Preço dos combustíveis online API	33
Conteúdos lecionados utilizados	34
Gitlab	34
Conclusão	35

## Identificação e caracterização do projeto

O trabalho apresentado foi desenvolvido no âmbito da disciplina de Computação Móvel e Ubíqua, onde foi proposto o desenvolvimento de uma aplicação baseada em um dos três temas apresentados. O tema escolhido para este trabalho foi o assistente de viagem, uma aplicação capaz de acompanhar e satisfazer as necessidades de um utilizador durante as suas viagens pessoais.

A abordagem seguida para este projeto foi de desenvolver funcionalidades como:

- Gestão de veículos pessoais do utilizador, com possibilidade de escolher que veículo vai ser utilizado.
- Monitorizar distâncias percorridas e localidades visitadas
- Visualizar postos de abastecimentos mais próximos/baratos
- Visualizar pontos de interesse mais próximos com vários filtros disponíveis
- Visualizar rotas frequentes
- Calcular rotas entre pontos
- Visualizar direções ao percorrer uma rota

## Ferramentas e tecnologias utilizadas

### Android

O Android é um sistema operativo móvel desenvolvido pela Google baseado no núcleo Linux e é utilizado em dispositivos como smartphones, tablets, televisões, smartwatches entre outros.



### Android Studio

O Android Studio é um ambiente de desenvolvimento integrado (IDE) oficial para desenvolver aplicações Android. Inclui ferramentas para desenvolvimento, depuração, teste e implementação de aplicações. Tem integração com outras ferramentas do Google, como o Firebase, e suporta diversas linguagens de programação, como Java e Kotlin.



### Jetpack Compose

O Jetpack Compose é uma ferramenta inovadora para construir aplicações em Android nativo, é uma framework moderna e reativa que permite criar interfaces de forma declarativa, ou seja, é possível descrever o estado desejado da interface utilizando código, em vez de utilizar ficheiros XML. Isto oferece uma forma mais simples e flexível de criar UI's e permite aos developers experimentar rapidamente diferentes layouts e estilos.



### Firebase

O Firebase é uma plataforma desenvolvida que fornece uma variedade de ferramentas e serviços para desenvolver aplicações mobile e web, tais como armazenamento na cloud, autenticação, base de dados em tempo real e alojamento. É uma ferramenta que permite aos programadores adicionar facilmente serviços de backend às suas aplicações sem a necessidade de gerir servidores ou infraestruturas.



### Room

O Room é uma biblioteca de persistência de dados que faz parte dos componentes de arquitetura do Android que foi desenhado para facilitar a criação de aplicações com necessidade de armazenar e recuperar dados de uma base de dados relacional. Este fornece uma camada de abstração acima do SQLite, tornando mais fácil e seguro trabalhar com bases de dados em aplicações Android.

## Opções estratégicas

### Sistema de caching com Firebase e Room

Tal como mencionado no enunciado deveria ser dado uso a base de dados em Firebase e Room. Por isso decidimos que seria benéfico para o funcionamento da aplicação, a implementação de um sistema de caching tirando proveito das duas.

Isto é, alguma da informação disponibilizada na aplicação não deveria ser inacessível mesmo quando o utilizador se encontra sem rede, por isso todos os dados que são inseridos pelos utilizadores são armazenados na base de dados firebase e consequentemente sincronizados com a base dados Room, pelo que mesmo que o utilizador se encontre sem rede poderá aceder aos dados da última sincronização, uma vez que estes se encontram armazenados internamente.

Sempre que o utilizador acede a uma screen que tire proveito desta funcionalidade de caching, os dados da cloud serão sincronizados com os internos e apenas serão sincronizados novamente caso algum documento seja alterado na cloud ou caso o utilizador entre novamente na screen.

Além da vantagem de permitir que o utilizador possa visualizar informação sem rede, o facto de aceder diretamente à informação da base de dados interna ao invés de aceder aos que estão na cloud, proporciona uma maior rapidez e eficácia.

## Estruturação da base de dados Room

### Entidade Districts

Estas tabelas têm como objetivo principal armazenar todos os distritos e municípios de Portugal, sendo estes dados fornecidos pela API **Preço dos combustíveis online API**.

Ao armazenar esta informação numa base de dados Room, permite que o acesso à mesma seja mais rápido, poupando requests à API e permitindo disponibilizar esta informação mesmo que o dispositivo não possua rede.

```
@Entity(tableName = "districts")
data class DistrictResult (
    @PrimaryKey
    val Descritivo: String,
    @ColumnInfo(name = "Id")
    val Id: String,
)
```

A  
tabela

```
@Entity(tableName = "municipios")
data class MunicipiosResult (
    @PrimaryKey
    val Descritivo: String,
    @ColumnInfo(name = "IdDistrito")
    val IdDistrito: Int,
)
```

## Entidade vehicles

Esta tabela serve apenas para armazenar os veículos pessoais do utilizador.

```
@Entity(tableName = "userVehicles", primaryKeys = ["licensePlate", "userEmail"])
data class Vehicle(
    @ColumnInfo(name = "licensePlate")
    val licencePlate: String,
    @ColumnInfo(name = "userEmail")
    val userEmail : String,
    @ColumnInfo(name = "brand")
    val brand: String,
    @ColumnInfo(name = "model")
    val model: String,
    @ColumnInfo(name = "name")
    val name: String,
    @ColumnInfo(name="kms")
    val kms: Double,
    @ColumnInfo(name="imageUrl")
    val imageUrl: String
)
```

## Entidade Routes

Esta é a entidade responsável por guardar as informações sobre as rotas predefinidas do utilizador.

```
@Entity(tableName = "routes", primaryKeys = ["userEmail", "endPointLat", "endPointLon"])
data class Route (
    @ColumnInfo(name = "userEmail")
    val userEmail : String,
    @ColumnInfo(name = "endPoint")
    val endPoint: String,
    @ColumnInfo(name = "endPointLat")
    val endPointLat: Double,
    @ColumnInfo(name = "endPointLon")
    val endPointLon: Double,
)
```

## Entidade SavedInterests

Na aplicação o utilizador poderá guardar pontos de interesse para que possa num futuro próximo deslocar-se até lá, por isso esta entidade será a responsável por armazenar esses mesmos locais guardados pelo utilizador.

```
@Entity(tableName = "savedinterests")
data class PointOfInterestResultSimplifiedRoom(
    @PrimaryKey
    val place_id: String,
    val business_status: String,
    val lat: String,
    val lng: String,
    val name: String,
    val opened_now: Boolean?,
    val photoURL: String?,
    val rating: Double,
    val user_ratings_total: Int,
    val user_email: String,
)
```

## Entidade GasStations

GasStations permite armazenar todos os postos de combustível de portugal, tendo em conta que tal como a informação dos distritos e municípios, a informação correspondente a cada posto de combustível é retornada pela API **Preço dos combustíveis online API**.

```
@Entity(tableName = "gasStations", primaryKeys = ["id", "combustivel"])
data class GasStationResult(
    @ColumnInfo(name = "id")
    val Id: String,

    @ColumnInfo(name = "nome")
    val Nome: String,

    @ColumnInfo(name = "tipoPosto")
    val TipoPosto: String,

    @ColumnInfo(name = "municipio")
    val Municipio: String,
    @ColumnInfo(name = "preco")
    val Preco: String,

    @ColumnInfo(name = "marca")
    val Marca: String,

    @ColumnInfo(name = "combustivel")
    val Combustivel: String,

    @ColumnInfo(name = "distrito")
    val Distrito: String,

    @ColumnInfo(name = "morada")
    val Morada: String,

    @ColumnInfo(name = "localidade")
```

## Armazenamento em Firebase

Sentimos a necessidade de armazenar algumas das estruturas de dados apresentadas anteriormente numa base de dados Firebase para que os dados não fossem perdidos de dispositivo para dispositivo. Tendo isto em conta, as estruturas apresentadas anteriormente também armazenadas em Room são:

- Pontos de interesse guardados pelo utilizador
- Rotas predefinidas
- Veículos

## Firebase CloudStorage

Para o armazenamento de ficheiros multimédia decidimos utilizar também o firebase, utilizando a storage para armazenar as imagens selecionadas e enviadas pelo utilizador.

Storage

Files    Rules    Usage

gs://travelassistant-3506e.appspot.com > images

Name	Tamanho	Tipo	Última modificação
image:25	195.71 KB	image/jpeg	19 de jan. de 2023
image:28	195.67 KB	image/jpeg	21 de jan. de 2023

**Fazer upload do arquivo**

## Estrutura de pastas

Para que o projeto seja desenvolvido com o nível de qualidade desejado é estritamente necessário que este seja organizado e estruturado, por isso decidimos organizar as pastas do projeto da seguinte maneira:

```
▽ com.example.travelassistant
  > database
  > models
  > network.api
  > preferences
  > repository
  > sensors
  > ui
  > utils
  > viewmodels
    ↴ MainActivity.kt
    ↴ TravelAssistant.kt
```

Na pasta **database** contém a configuração da base de dados e as respectivas interfaces

```
▽ database
  ▽ dao
    ↴ DistrictsDao
    ↴ FuelTypeDao
    ↴ GasStationsDao
    ↴ MunicipiosDao
    ↴ RoutesDao
    ↴ SavedInterestsDao
    ↴ VehiclesDao
    ↴ TravelAssistantDatabase
```

A pasta **models**, tal como o nome indica possui todos os modelos existentes na aplicação

```
▽  models
  > Districts
  > gasStations
  > googledirections
  > googlegeocoding
  > googleplaces
  > location
  > Municipios
  > navigation
    Route
    User
    Vehicle
    VisitedLocation
```

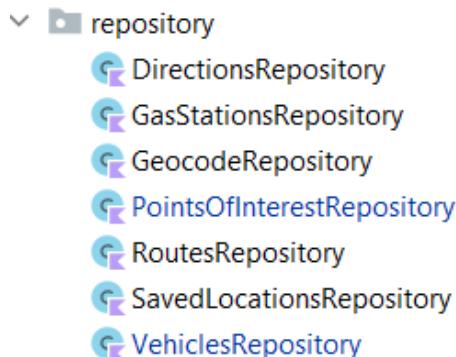
Na pasta **network** contém as interfaces para as diferentes API's e o RetrofitHelper, classe essencial para realizar requests às API's.

```
▽  network.api
  GasStationAPI
  GoogleGeocodeAPI
  GooglePlacesAPI
  GoogleRoutesAPI
  RetrofitHelper
```

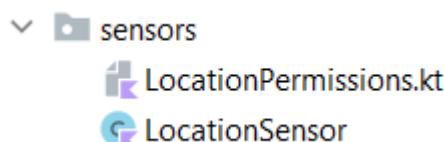
A pasta **preferences** é dedicada às preferências da aplicação.

```
▽  preferences
  UserPreferences
```

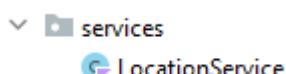
Na pasta **repository** são armazenados todos os repositórios do projeto.



A pasta **sensors** contém a configuração dos sensores utilizados pela aplicação, nomeadamente de localização.

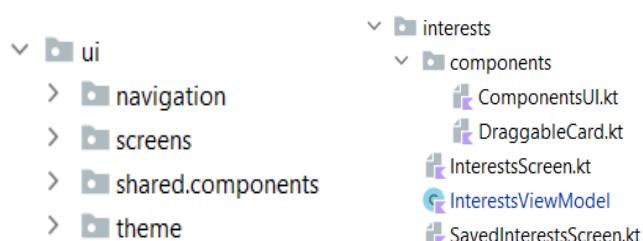


A pasta **services** contém os serviços utilizados pela aplicação, nomeadamente um serviço em foreground de localização

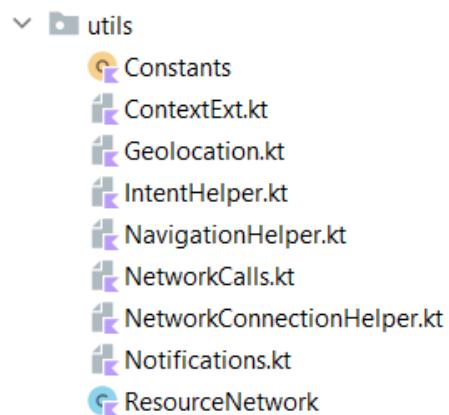


A pasta **ui** encontra-se subdividida em quatro subpastas, sendo elas:

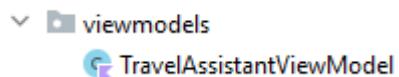
- **navigation**: Package dedicado às navegações entre as diferentes screens da aplicação
- **screens**: Package que contém todas as screens, em que cada uma possui um package dedicado com ficheiro principal para o design da mesma, um ficheiro para componentes da UI utilizados na screen e o respetivo viewModel
- **sharedComponents**: Package dedicado para componentes partilhados da aplicação, ou seja, componentes que são utilizados em várias screens.
- **theme**: Package para a customização do tema da aplicação



A pasta **utils** contém ficheiros que podem ser úteis para vários componentes



A pasta **viewmodels** contém viewmodels que devem ser utilizados/partilhados por vários componentes

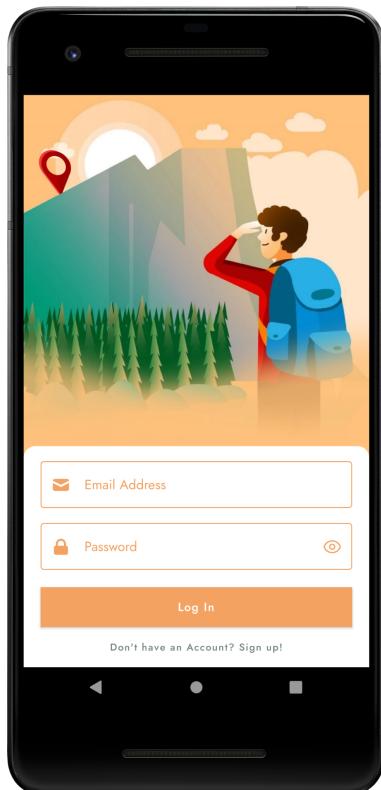


## Layouts e respetivas funcionalidades

### Login

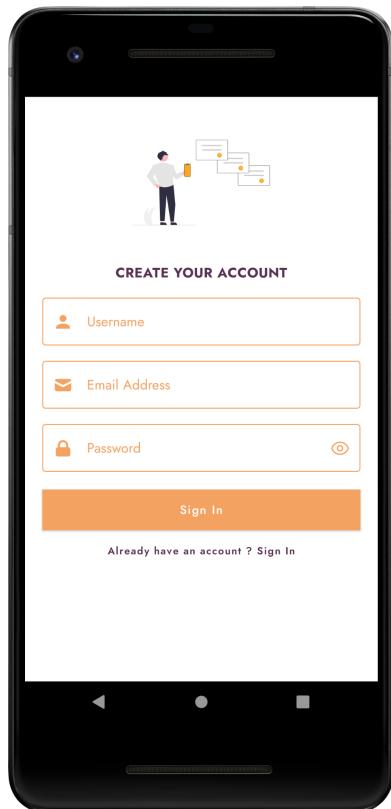
A screen de Login é a primeira screen com que o utilizador tem algum tipo de interação, onde pode inserir as suas credenciais e começar a utilizar as funcionalidades fornecidas pela aplicação.

Caso o utilizador não possua ainda uma conta registada na aplicação poderá clicar na opção **"Don't have an Account? Sign up!"** que permitirá ao utilizador ser redirecionado para a screen de registo.



## Registo

Tal como mencionado anteriormente, esta screen tem como objetivo permitir ao utilizador criar uma conta para posteriormente poder aceder às respectivas funcionalidades fornecidas pela aplicação.



Como se pode observar, o utilizador para criar uma conta necessita apenas de mencionar o seu nome de utilizador, email e password para a sua conta.

Caso o utilizador deseje navegar de volta para a screen de Login sem criar uma conta pode clicar na opção de "**Already have an account? Sign In!**".

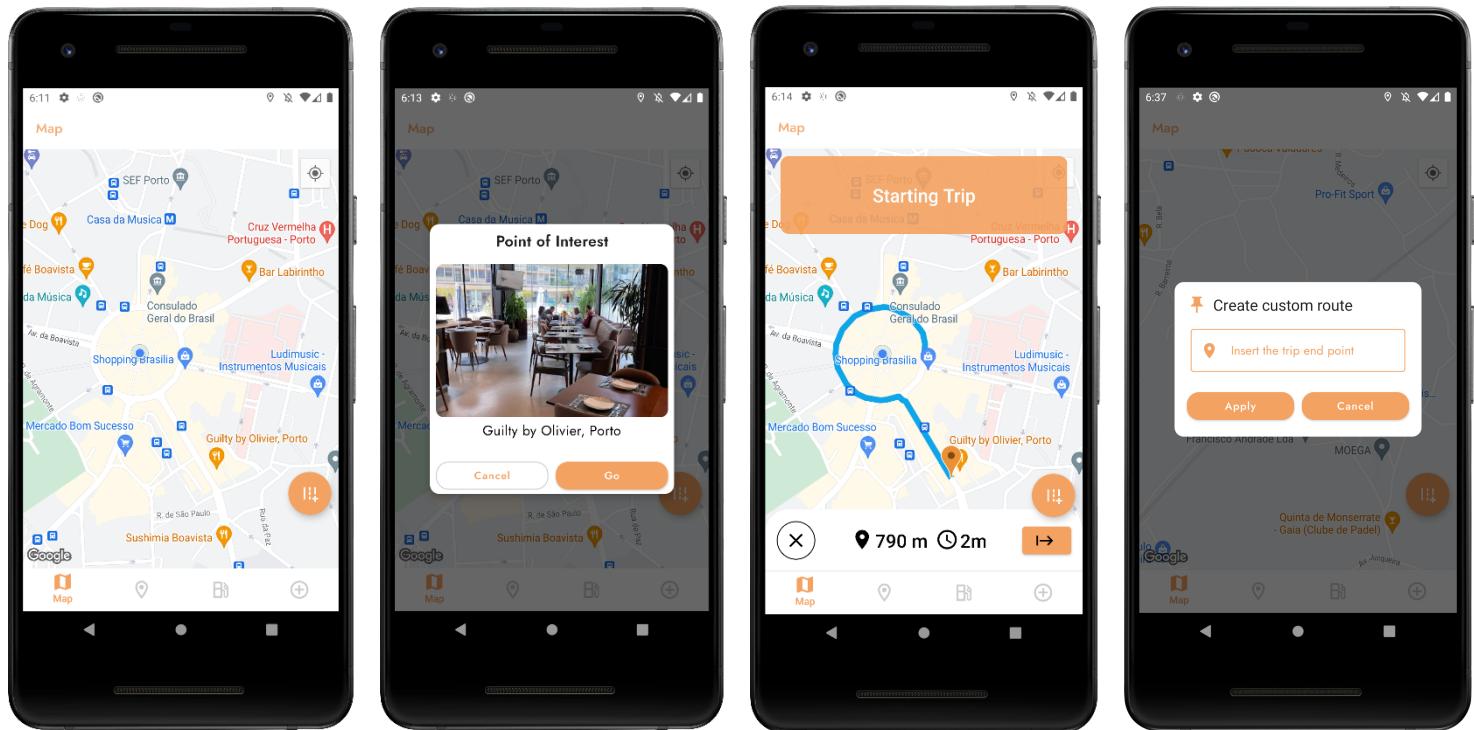
**Nota:** Caso o login/registo seja realizado o seu estado é guardado durante as próximas vezes que o utilizador utilize a app. Para reverter este comportamento é necessário realizar o logout acedendo às settings.

## Mapa

Esta screen mostra um mapa enquadrado na localização atual do utilizador, e é utilizado para calcular rotas entre pontos de interesse.

A partir daqui o utilizador pode explorar pontos de interesse no mapa, onde ao carregar sobre eles são mostradas informações do local e é calculada a sua rota. Depois de calculada a rota, são apresentadas informações sobre a mesma como a distância, a duração e a próxima instrução a seguir.

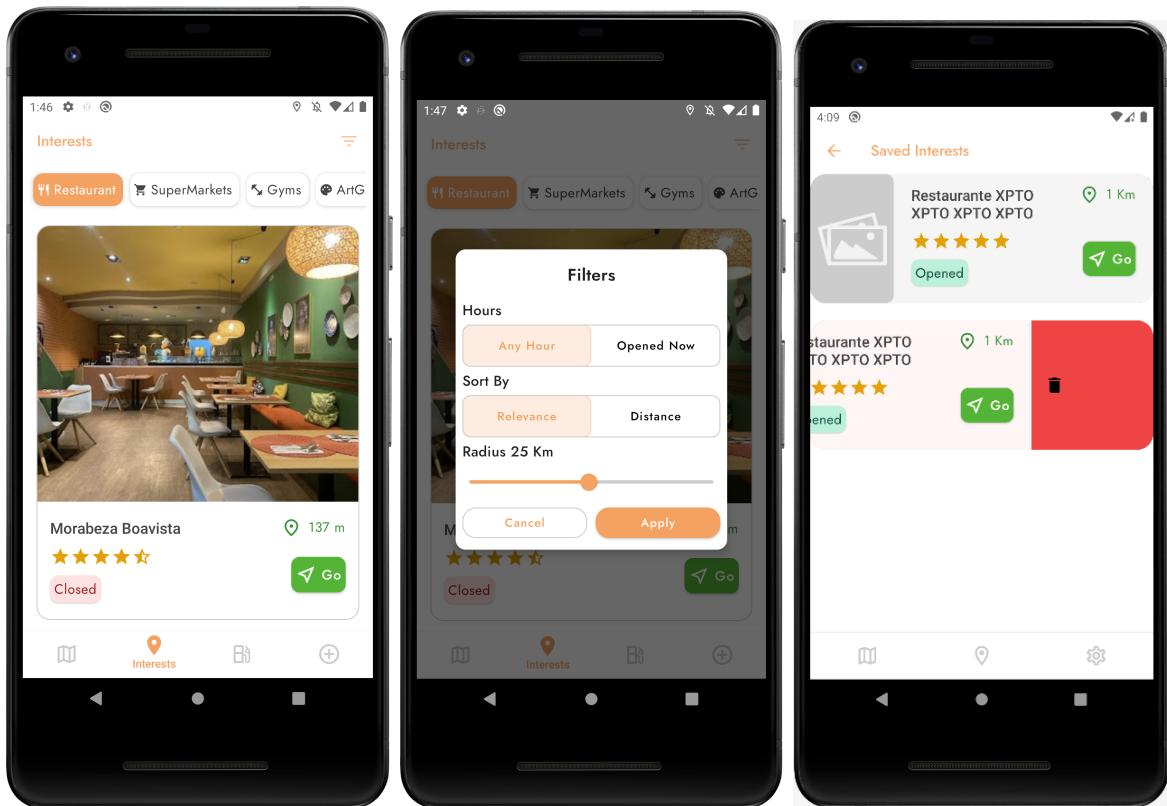
Além disso, o utilizador pode também criar rotas e escolher um ponto de origem.



## Pontos de Interesse

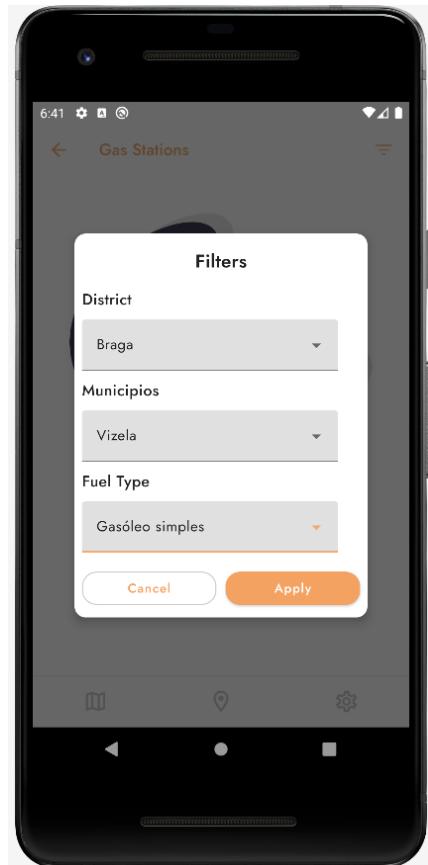
Esta screen é utilizada para visualizar pontos de interesse perto da localização do utilizador, é possível aplicar vários filtros ao seu gosto como: o tipo de estabelecimento, se o estabelecimento está aberto ou não, o raio de distância e ordenar por relevância ou distância, adaptando assim os resultados da pesquisa às necessidades do user.

Além disso, é possível fazer swipe-right ou swipe-left sobre o card do estabelecimento. Caso seja realizado um swipe-right o estabelecimento é guardado numa screen dedicada onde posteriormente pode ser removido realizando um swipe-left. No caso de ser efetuado um swipe-left o estabelecimento é “descartado”

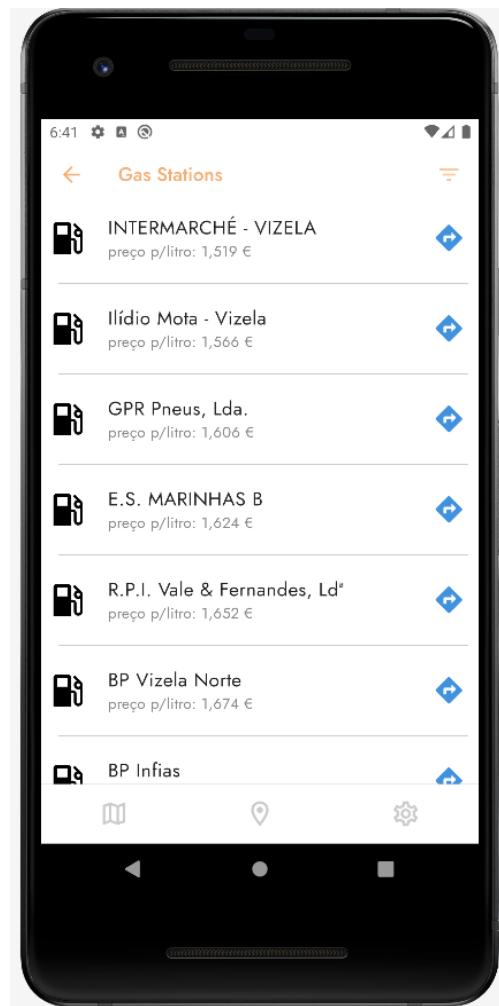


## Postos de combustível

Esta screen revela-se bastante importante para o utilizador, visto que permite realizar uma pesquisa ao preço dos combustíveis nos postos de um determinado município. Deste modo, o utilizador pode ter uma percepção de qual o posto de combustível mais rentável para aquele determinado tipo de combustível naquele município e assim tomar uma decisão mais acertada e consciente de qual escolher para abastecer o seu veículo.



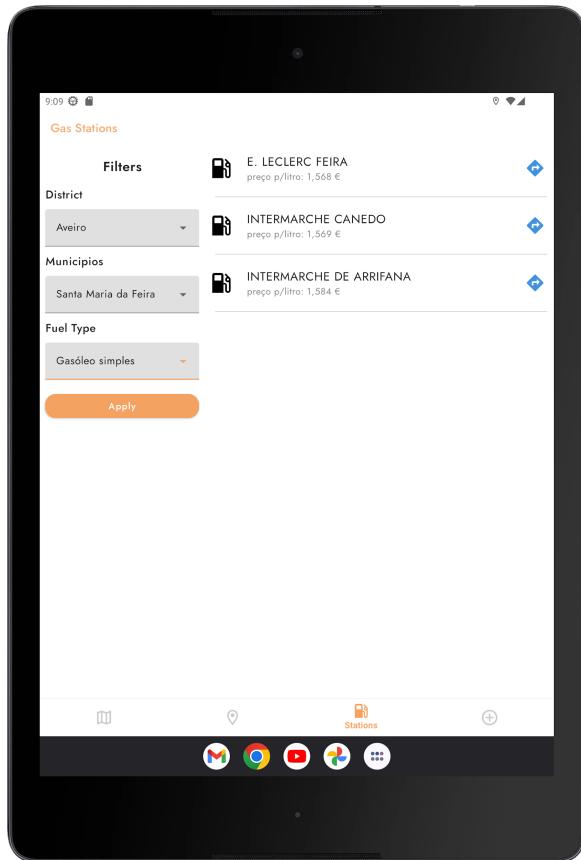
Ao aplicar os filtros, que podem ser acedidos no canto superior direito da página, caso existam postos de combustível que forneçam o tipo de combustível indicado naquele determinado município, será disponibilizada uma lista dos mesmos que, ao serem selecionados, serão mostrados alguns detalhes adicionais.



Caso o utilizador deseje obter direções para o posto de combustível em questão, basta clicar no botão de direções e será calculada a sua rota.

## Visão de tablet

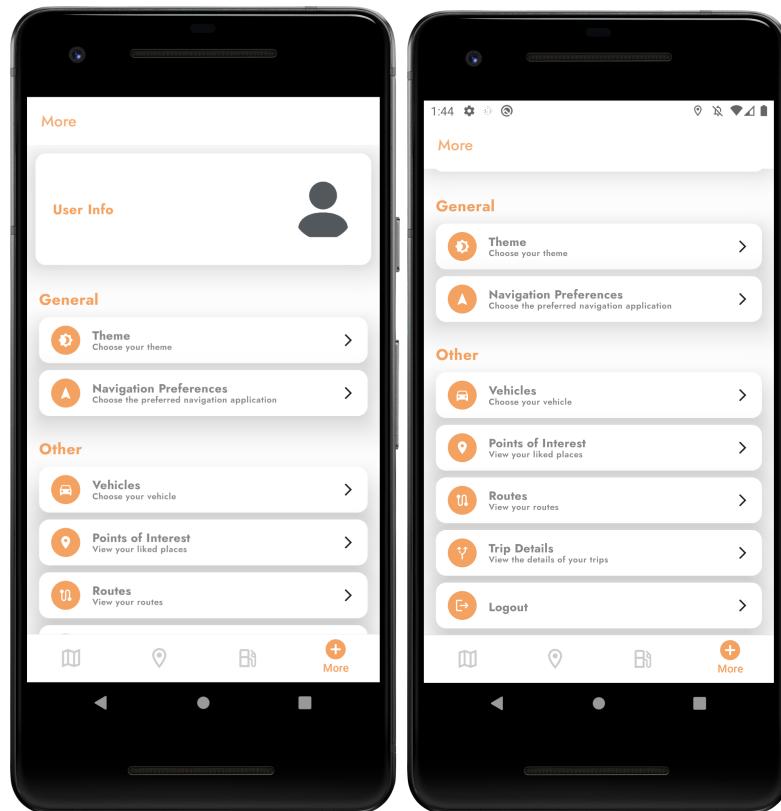
A aplicação é completamente funcional para telas com elevadas dimensões, no entanto, para esta screen a estrutura foi adaptada para que os filtros ficassem fixados do lado esquerdo, sendo que em telas mais pequenas os filtros aparecem num dialog.



## More

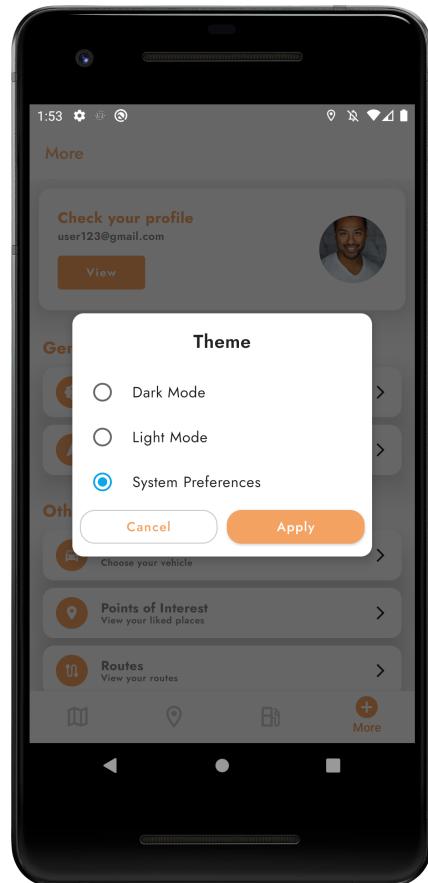
Nesta screen o utilizador tem acesso a uma grande parte das funcionalidades da aplicação, sendo elas:

- Tema
- Preferências de navegação
- Veículos
- Pontos de interesse
- Rotas
- Detalhes da viagem
- Logout



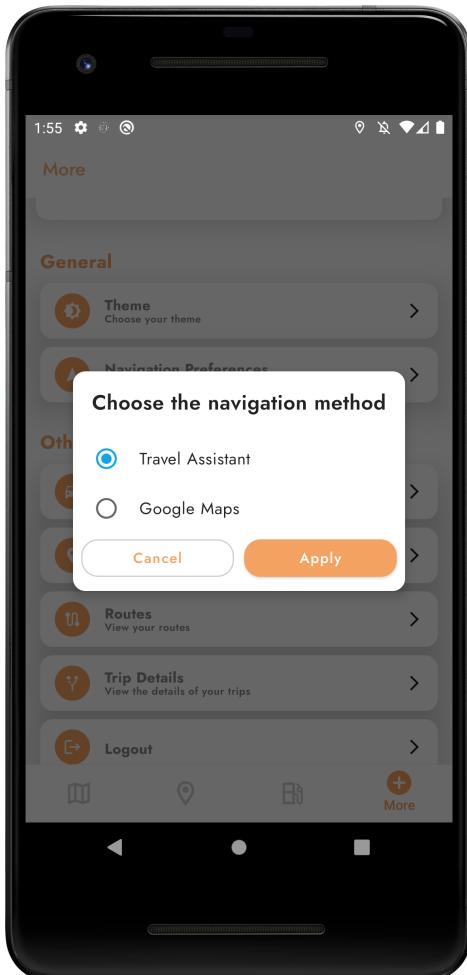
## Tema

Esta funcionalidade serve unicamente para o utilizador escolher o tema da aplicação que pode variar entre Light Mode, Dark Mode e as preferências do sistema que vai depender qual o tema definido no sistema do dispositivo usado. Além disso, o tema escolhido pelo utilizador é guardado durante as próximas utilizações da aplicação.



## Preferência de navegação

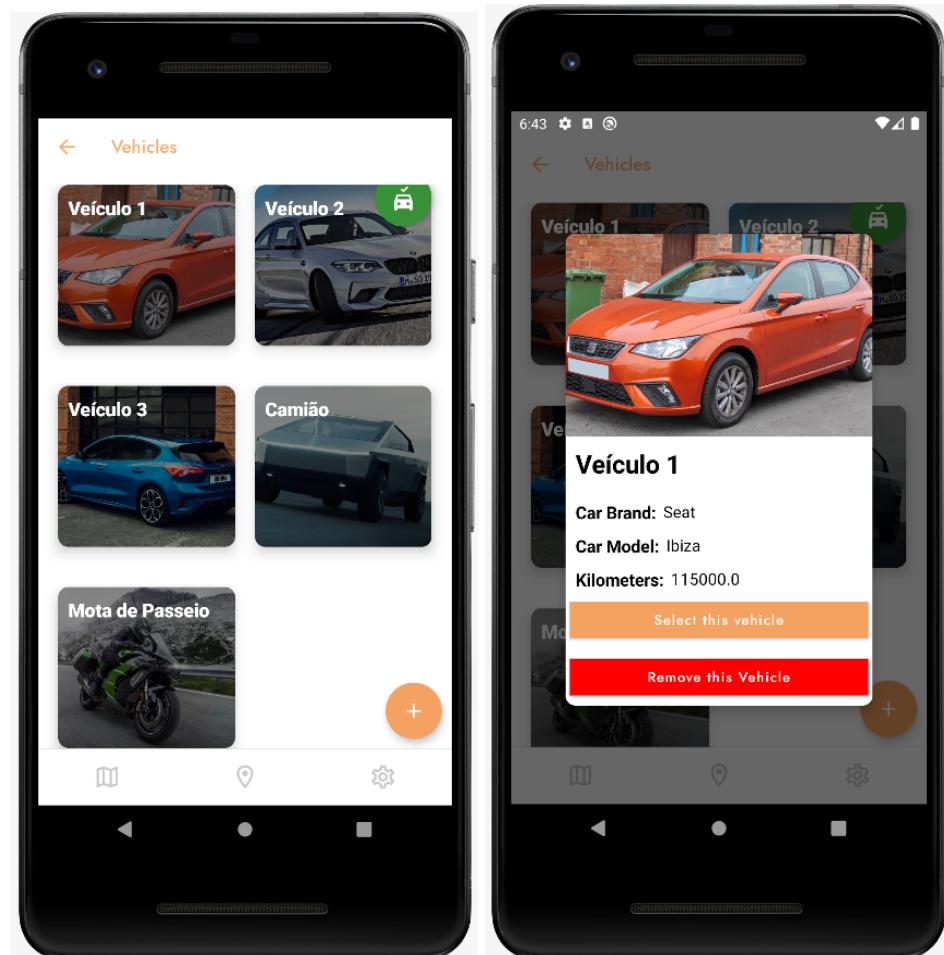
Estas funcionalidade serve para escolher o método de navegação da aplicação, caso seja escolhido a opção “TravelAssistant” a navegação é feita diretamente na aplicação, caso contrário é feita no Google Maps



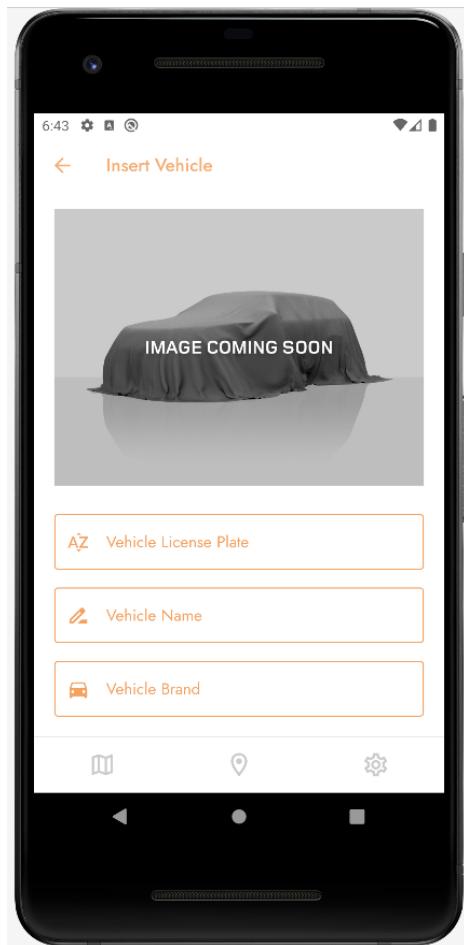
## Veículos

Esta screen serve para o utilizador gerir os seus veículos, ou seja, visualizar, inserir, remover veículos e selecionar o veículo que se encontra em uso pelo utilizador.

Inicialmente pode ser visualizada a lista de veículos e ao selecionar um determinado veículo é mostrado um Dialog com as diversas informações e as opções de selecionar o veículo para viagens ou remover o veículo. Caso o veículo já se encontre selecionado, não será mostrada a opção de selecionar e será visível um pequeno ícone de “Check” como forma de representar que aquele veículo é o que se encontra em uso.



Para inserir um novo veículo basta apenas clicar no botão que se encontra no canto inferior direito e posteriormente, após ser redirecionado para a screen de inserção de veículos, preencher os dados correspondentes.

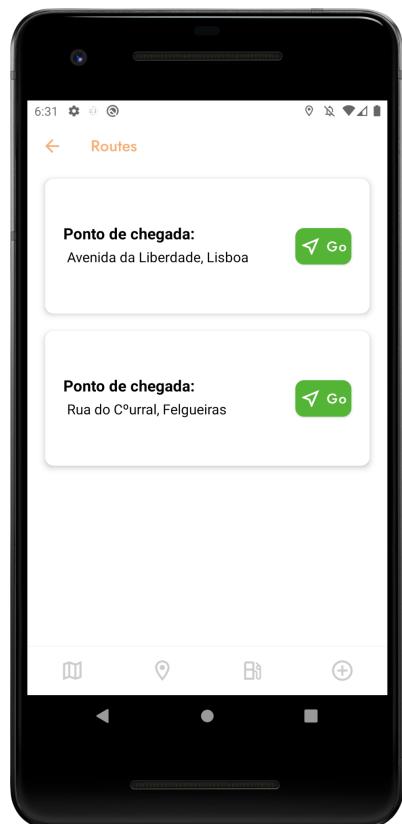


Um veículo possui a seguinte informação:

- Imagem
- Matrícula
- Nome do veículo
- Marca
- Modelo
- Quilómetros realizados com o veículo

## Rotas

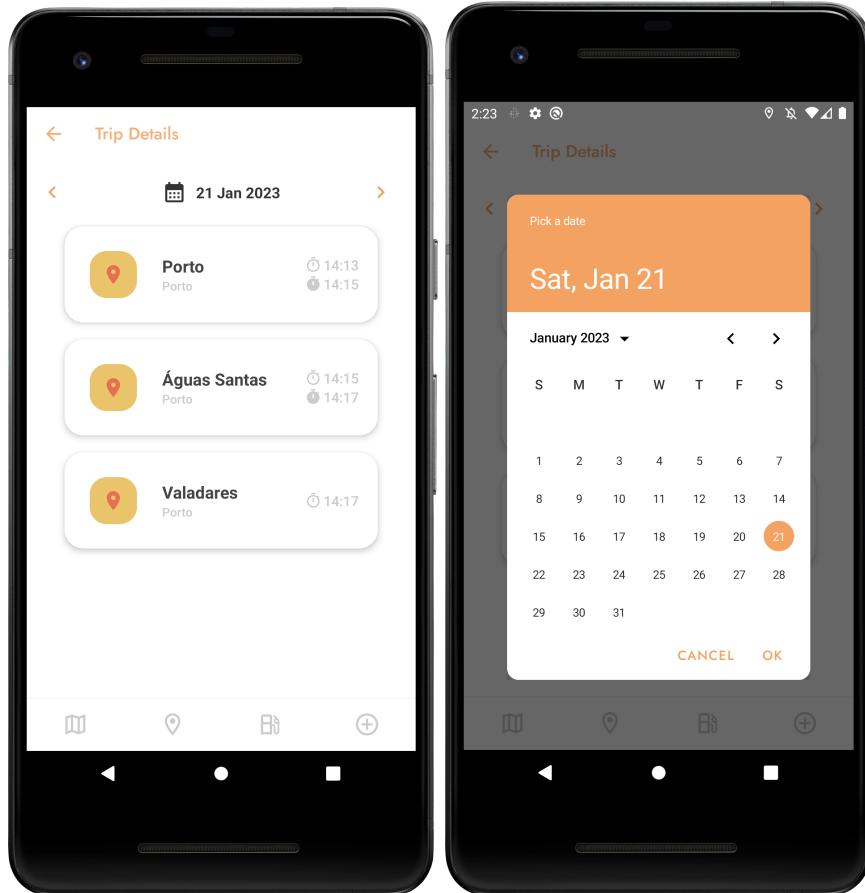
Esta screen tem como objetivo listar as rotas predefinidas criadas pelo utilizador para que quando o utilizador deseje realizar uma viagem que costuma realizar, não necessite de colocar todas as vezes o ponto de destino.



## Detalhes da viagem

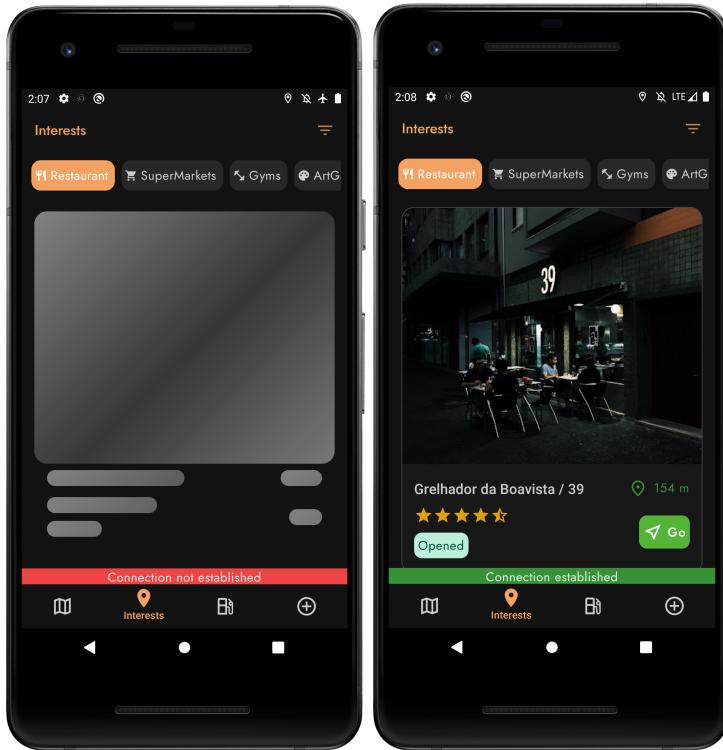
Esta screen mostra todas as localidades pela qual um utilizador passou durante um determinado dia, isto é calculado através de um serviço em foreground para realizar o tracking da localização.

No entanto, devido a falta de tempo, as informações não estão a ser armazenadas em room pelo que a parte de pesquisar pela data não está funcional.



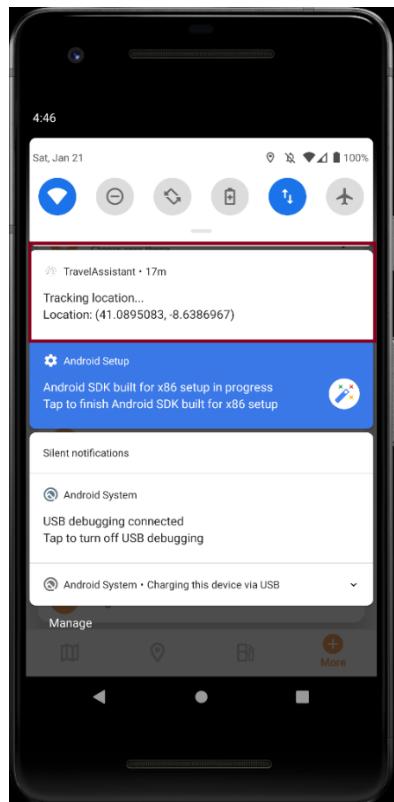
## Conexão à internet

Para gerir a conexão à internet é mostrado ao utilizador na parte de baixo da aplicação se existe conexão ou não



## Serviços

Para esta aplicação foi desenvolvido um serviço em foreground para realizar o tracking da localização. Este serviço foi realizado para acompanhar a localização do utilizador mesmo que este não esteja a utilizar a aplicação em primeiro plano, contribuindo assim para desenvolver a screen dos detalhes da viagem, onde é possível visualizar as localidades que o utilizador passou. Este serviço é ativado de 3 em 3 minutos e mostra uma notificação para o utilizador.



## Testes à UI

Para verificar o bom funcionamento do componente que mostra os pontos de interesse na screen dos interests foi realizado um teste para verificar se o card é mostrado e se ao fazer swipe é desenhado outro card com outro ponto de interesse

## API's utilizadas

### Google Places API

A Google Places API é um serviço da Google que permite aos developers aceder a informações sobre lugares, como estabelecimentos comerciais, pontos turísticos, entre outros. Esta API oferece vários recursos, como busca por localização, endereços e detalhes de estabelecimentos, fotos, comentários e avaliações.

Esta API foi utilizada no âmbito de disponibilizar pontos de interesse próximos do local onde o utilizador se encontra.

Documentação: <https://developers.google.com/maps/documentation/places/web-service>

### Google Routes API

A Google Routes API (sucessora do Google Directions) é um serviço web que permite calcular a distância e duração de uma rota entre dois ou mais locais. A API retorna o resultado em um formato JSON, que pode ser facilmente integrado com aplicações.

Documentação: <https://developers.google.com/maps/documentation/routes>

### Google Geocoding API

A Google Geocoding API é um serviço web que permite converter endereços (como "Rua do curral, Felgueiras") em coordenadas geográficas (como latitude 37.423021 e longitude -122.083739), e vice-versa. A API também pode ser usada para verificar a precisão de um endereço ou para corrigir erros comuns de escrita.

Documentação: <https://developers.google.com/maps/documentation/geocoding>

### Preço dos combustíveis online API

A API dos preços dos combustíveis é um serviço do estado português que disponibiliza os preços dos combustíveis mais atualizados nos diferentes postos existentes no país.

Esta API foi utilizada para permitir ao utilizador procurar por postos de combustível num determinado distrito e município.

Documentação: <https://precoscombustiveis.dgeg.gov.pt/>

## Conteúdos lecionados utilizados

Para este trabalho prático foram utilizados diversos conteúdos que foram lecionados na Unidade Curricular, nomeadamente:

- Room
- Firebase
- Retrofit
- Intents
- Navegação e Scaffold
- Dialogs
- Gestão de conexão à Internet
- Background Tasks (Coroutine)
- ViewModels e LiveData
- Sistema de armazenamento em cache (Chamada à API e armazenar em Room)
- Acesso a ficheiros do armazenamento interno do dispositivo (Image Picker)
- Preferências (Data Store)
- Suporte para múltiplas linguagens (É necessário mudar a linguagem do sistema do dispositivo)
- Suporte a diferentes temas (Light Mode e Dark Mode)
- Sensores (Localização)
- Integração de mapas (Google Maps)
- Cálculo de rotas e visualização das direções
- Suporte para ecrãs de diferentes dimensões
- Notificações
- Serviços
- Intents
- Testes

## Gitlab

O link para o repositório do gitlab é o seguinte:

[https://gitlab.estg.ipp.pt/8200595/cmu\\_tp\\_grupo06](https://gitlab.estg.ipp.pt/8200595/cmu_tp_grupo06)

## Conclusão

O desenvolvimento do trabalho prático permitiu adquirir competências e dominar os conceitos teóricos e práticos sobre a programação móvel em ambiente android, nomeadamente a framework jetpack compose.

Além disso, permitiu melhorar o trabalho em equipa, nomeadamente na questão de organização, utilizando o gitlab para gestão de tarefas e controlo de versões.

Considerando a aplicação desenvolvida, consideramos que a mesma está bem concebida e bastante rica em termos de funcionalidades implementadas.