



**INSTITUTO POLITÉCNICO DE LISBOA
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**FuelWise - Informação sobre postos de
abastecimento e gestão de veículos**

Bruno Costa n.º 45836, e-mail: a45836@alunos.isel.pt

Orientador: Professor Nuno Leite

Relatório Final realizado no âmbito de Projeto e Seminário
Licenciatura em Engenharia de Informática e de
Computadores

Julho de 2023

Resumo

Atualmente, tem-se verificado um aumento significativo no preço dos bens essenciais, nos combustíveis e na energia. Estes aumentos fazem com que as pessoas procurem os preços mais baixos de forma a conseguirem poupar. No caso dos combustíveis, tem-se observado uma enorme variação dos preços derivado a diversos fatores, e na hora de abastecer as viaturas, os condutores procuram saber qual o posto de combustível perto de si que tem o preço mais vantajoso. O presente projeto tem como principal finalidade desenvolver uma aplicação móvel na plataforma *Android* que permita a consulta de preços de combustível nos vários postos de abastecimento localizados em Portugal. De forma a tornar a aplicação mais completa a mesma tem mais funcionalidades, tais como o registo de despesas (por exemplo: abastecimentos, *Imposto Único de Circulação* (IUC), inspeções, seguros, entre outros) e a possibilidade de criar lembretes. No mercado foram observadas algumas aplicações com as características mencionadas anteriormente, porém, esta aplicação diferencia-se das restantes, pois gera notificações no dispositivo sobre as alterações dos preços de combustíveis nos postos de abastecimento selecionados pelo utilizador como favoritos. Esta funcionalidade permite informar o utilizador da aplicação sobre alterações de preços dos combustíveis de um posto com interesse, sem ter de o fazer de forma proativa, ou seja, recorrendo a uma pesquisa na aplicação sobre o posto de combustível em questão.

Palavras-Chave: Posto de abastecimento, Combustível, Despesa, Lembrete, Abastecimento, Aplicação, *Android*.

Lista de Acrónimos

API Application Programming Interface

BD Base de Dados

CEME Comercializador de Eletricidade para a Mobilidade Elétrica

DAO Data Access Object

DGEG Direção-Geral de Energia e Geologia

DPC Detentor de Ponto de Carregamento

EAFO European Alternative Fuels Observatory

ENSE Entidade Nacional para o Setor Energético

GPL Gás de Petróleo Liquefeito

HTTP Hypertext Transfer Protocol

IEC Imposto Especial sobre o Consumo de Energia Elétrica

ISP Imposto sobre Produtos Petrolíferos

IUC Imposto Único de Circulação

IVA Imposto sobre Valor Acrescentado

JSON JavaScript Object Notation

MVVM Model View ViewModel

OPC Operador de Ponto de Carregamento

OPEP Organização dos Países Exportadores de Petróleo

REST Representational State Transfer

UI User Interface

URL Uniform Resource Locator

WWW World Wide Web

Conteúdo

Resumo	iii
Lista de Acrónimos	v
Lista de Figuras	ix
1 Introdução	1
2 Formulação do Problema	9
2.1 Estado de Arte	10
2.2 Requisitos funcionais	11
3 Abordagem	13
3.1 Tecnologias	14
3.2 Arquitetura	15
3.2.1 Análise	15
3.2.2 Fluxos de comunicação	18
3.3 Navegação no <i>Android</i>	19
3.4 Arquitetura de classes	23
4 Implementação	25
4.1 Ferramentas utilizadas	26
4.2 Modelo de Base de Dados	26
4.3 Permissões	27
4.4 Serviço e trabalho persistente	30
4.5 Inicialização da aplicação	31
4.6 Criação de utilizador	31

4.7	Ecrã inicial	31
4.8	Lista de opções	32
4.9	Criação de um veículo	33
4.10	Consulta e edição de um veículo	33
4.11	Lista de veículos	34
4.12	Criação de um abastecimento	34
4.13	Criação de uma despesa	35
4.14	Criação de um lembrete	36
4.15	Postos de abastecimento favoritos	37
4.16	Mapa com os postos de abastecimento	38
4.17	Manual de utilizador da aplicação	39
4.18	Testes unitários	40
5	Conclusão e Trabalho Futuro	47
5.1	Conclusão	48
5.2	Trabalho futuro	49
A	Modelo Entidade-Associação	51
B	Modelo Relacional	53
	Referências	55

Listas de Figuras

1.1	Evolução do preço do barril de petróleo desde 1946 até 2023. Fonte Macrotrends (2023).	3
1.2	Custo médio de carregamento doméstico de veículos elétricos na União Europeia. Fonte ACP (2023).	6
3.1	Diagrama da arquitetura da aplicação. Este encontra-se organizado em quatro fluxos de comunicação.	17
3.2	Fluxo de dados com múltiplas <i>Activities</i> ou uma única <i>Activity</i> . Fonte Academy (2023).	20
3.3	Navegação com <i>Compose</i> . Fonte Navigation (2023).	21
3.4	Ciclo de vida de uma <i>Activity</i> e <i>ViewModel</i> respetivo. Fonte Google (2023l).	22
3.5	Diagrama da arquitetura Model View ViewModel (<i>MVVM</i>). Fonte Coder (2023).	22
4.1	Diagrama da arquitetura da biblioteca <i>Room</i> . Fonte Google (2023j). . . .	26
4.2	Diagrama do fluxo sobre a solicitação de permissão para acesso a um recurso no <i>Android</i> . Fonte Google (2023h).	28
4.3	Solicitação de permissão para envio de notificações pela aplicação <i>FuelWise</i>	29
4.4	Tipos de trabalho persistente. Fonte Google (2023k).	31
4.5	Criação de utilizador.	32
4.6	Ecrã inicial.	33
4.7	Lista de opções.	34
4.8	Criação de veículo.	35
4.9	Consulta e edição de um veículo.	36
4.10	Lista de veículos.	37
4.11	Abastecimento de um veículo.	38

4.12 Registo de uma despesa.	39
4.13 Registo e notificação de um lembrete.	40
4.14 Assinalar um posto de abastecimento como favorito no mapa e listar os postos favoritos.	41
4.15 Ativação do serviço e apresentação das notificações sobre postos de abastecimento marcados como favoritos.	42
4.16 Consulta de postos de abastecimento para a gasolina especial 98.	43
4.17 Serviços, preços de combustível e navegação para um dado posto de abastecimento.	44
4.18 Informação sobre um posto de carregamento elétrico.	45
A.1 Modelo Entidade-Associação	52
B.1 Modelo Relacional	54

Capítulo 1

Introdução

A energia é um componente primordial numa sociedade moderna. Segundo Roger A. Hinrichs (2003), o desenvolvimento económico e os altos padrões de vida são processos complexos que compartilham um denominador comum, a disponibilidade de um fornecimento adequado e fiável de energia.

Os combustíveis fósseis são, por definição, o grupo de recursos naturais disponíveis na natureza usados para a produção de energia por meio da sua queima e com origem na decomposição de material orgânico ao longo do tempo. Os três principais tipos de combustíveis fósseis são: o petróleo, o gás natural e o carvão mineral. Embora existam esforços políticos em todo o mundo para diminuir a dependência em relação ao petróleo, este é ainda o combustível mais utilizado. Além de ser um recurso não renovável, o petróleo apresenta também como desvantagem a emissão de grandes quantidades de poluentes para a atmosfera durante a sua queima. A principal utilização do petróleo é a sua conversão em gasolina e gasóleo para veículos. São também produzidos com base no petróleo: o *Gás de Petróleo Liquefeito* (GPL), a nafta (um derivado utilizado na indústria petroquímica), a querosene de avião, o plástico, alguns tipos de solvente e outros produtos.

A indústria petrolífera cresceu no século XIX, impulsionada pela procura de querosene para as lâmpadas de iluminação, porém o verdadeiro impulsor foi a indústria automóvel no início do século XX. Nas vésperas da Primeira Guerra Mundial, havia no mundo mais de um milhão de automóveis, no ano de 1922 já havia 18 milhões e no ano de 1929 esse número ascendia a 32 milhões. Ao terminar a Primeira Guerra Mundial deu-se uma retoma do preço do barril, porém esse aumento durou pouco tempo, tendo sido registado em 1931 o seu valor mínimo histórico. Nos anos 1950 e 1960, a indústria petrolífera cresceu continuadamente e esteve dominada por sete grandes companhias petrolíferas ocidentais que mantiveram o controlo do mercado até à década de 80.

Durante o ano de 1960 os governos do Iraque, Irão, Kuwait, Arábia Saudita e Venezuela realizaram a Conferência de Bagdade que levou à criação da *Organização dos Países Exportadores de Petróleo* (OPEP). A influência deste poderoso cartel seria muito notada nos anos seguintes. A figura 1.1 mostra a evolução do preço do barril do petróleo entre o período de 1946 até à atualidade.

No início do século XXI, vários acontecimentos fizeram subir o preço do barril: os atentados de 11 de setembro de 2001, a segunda Guerra do Golfo (2003), a crise do Médio Oriente e a enorme bolha financeira nos últimos anos do século XX. O preço do barril alcançou o valor máximo na primavera de 2008, quando chegou a rondar os 140 dólares por barril.

De acordo com Gonçalves (2023), no preço dos combustíveis estão incorporados vá-

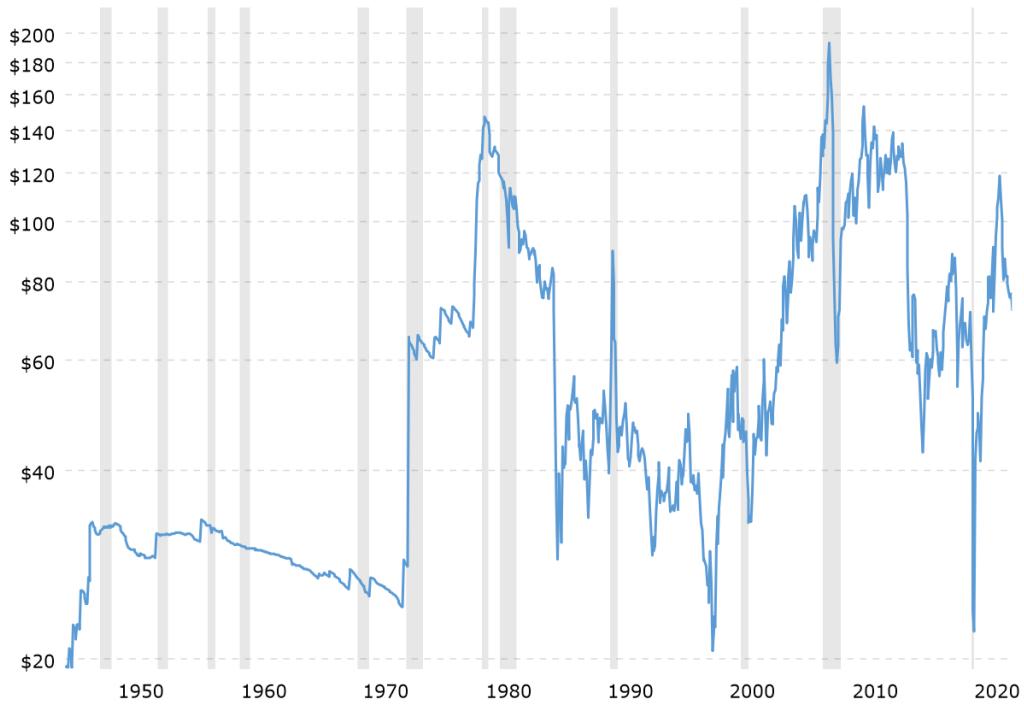


Figura 1.1: Evolução do preço do barril de petróleo desde 1946 até 2023. Fonte Macro-trends (2023).

rios custos, sendo os principais elementos que o influenciam:

- Cotação internacional: um elemento importante na formação dos preços dos combustíveis é a cotação da matéria-prima. No caso de Portugal, o preço dos combustíveis não é influenciado diretamente pelo preço do petróleo, mas sim pela cotação dos seus derivados, nomeadamente o gasóleo e a gasolina. No entanto, é de referir que os preços do petróleo e dos seus derivados acabam por estar fortemente relacionados. O câmbio euro/dólar tem também impacto na formação dos preços dos combustíveis. Isto porque o preço dos derivados de petróleo nos mercados internacionais está fixado em dólares. Assim, se por exemplo, o euro valorizar face ao dólar, estes produtos, quando expressos em euros, ficam mais baratos. No ano de 2022, o euro tem desvalorizado face ao dólar, o que, a somar ao aumento do preço dos derivados de petróleo, criou uma pressão para o aumento do preço dos combustíveis;
- Custos de transporte: os produtos petrolíferos necessitam de ser transportados para o mercado nacional. E esses custos de transporte também são passados para o consumidor no preço final. Além disso, é necessário manter reservas estratégicas de petróleo e derivados, que são importantes em situações de emergência, nomeada-

mente em caso de falhas no fornecimento regular. Em Portugal, a gestão destas reservas estratégicas está a cargo da *Entidade Nacional para o Setor Energético* (ENSE). De notar que, ao nível das reservas estratégicas, existem custos com a sua gestão e armazenamento, que devem ser também refletidos no preço final;

- Incorporação de biocombustíveis: os biocombustíveis são combustíveis feitos a partir de produtos agrícolas ou vegetais, sendo assim uma forma de energia alternativa aos combustíveis fósseis. O Etanol, bioetanol, biodiesel e biogás são alguns exemplos de biocombustíveis. Devido ao seu impacto em termos de redução de emissões, as autoridades definem a obrigatoriedade de incorporação de uma determinada percentagem de biocombustíveis no gasóleo e na gasolina. Atualmente, é de 11% ainda que os biocombustíveis apresentem algumas vantagens para o meio ambiente, também têm desvantagens, como o facto de levarem à desflorestação;
- *Imposto sobre Produtos Petrolíferos* (ISP): é outro dos custos que o consumidor suporta quando vai a um posto de abastecimento. As taxas de ISP são revistas regularmente pelo Governo;
- *Imposto sobre Valor Acrescentado* (IVA): na compra de gasóleo e gasolina também é cobrado IVA aos consumidores, à taxa atual de 23%. No entanto, atualmente existe um mecanismo de redução das taxas do ISP para, na prática, a taxa de IVA aplicada aos produtos petrolíferos ser de 13% e não 23%;
- Margem de comercialização: é um custo suportado pelos consumidores, e corresponde à margem de lucro das petrolíferas pela venda do combustível. Tem também como objetivo suportar os custos de distribuição dos combustíveis. É normal existirem diferenças no preço dos combustíveis entre postos de abastecimento. Deve-se essencialmente às diferentes margens de comercialização que estão a ser praticadas e com custos relacionados com a distribuição dos combustíveis.

No que respeita aos carros elétricos, o preço do carregamento do veículo apresenta as seguintes variáveis:

- Tipo de carregamento: a potência selecionada afeta diretamente o custo. Por exemplo, os postos de carregamento rápido (com uma potência até 50 kWh) aplicam, regra geral, preços mais caros do que os de carregamento normal (com potência igual ou inferior a 22 kWh);

- Local de carregamento: o preço do kWh é determinado pela tarifa de cada posto (no caso da rede pública de carregamento) ou pelas condições contratuais de fornecimento de energia (nos carregamentos domésticos);
- Taxas e impostos: neste âmbito, é necessário equacionar o valor taxado para a energia e as taxas de cada região. Além disso, englobam-se aqui impostos de consumo, como o *Imposto Especial sobre o Consumo de Energia Elétrica* (IEC) e o IVA;

O *European Alternative Fuels Observatory* (EAFO), da Comissão Europeia, refere que existem dois princípios gerais aplicáveis aos países europeus, relativamente ao custo de carregar um carro elétrico:

- o carregamento com potência normal, em casa, é a opção mais económica;
- o carregamento de elevada potência (rápido) é a alternativa mais cara, no mercado atual.

É apresentado na figura 1.2 o custo médio do carregamento doméstico de um veículo elétrico nos vários países europeus.

A rede Mobi.E (2023) integra:

- todos os postos de carregamento de acesso público e de acesso privativo instalados por qualquer um dos *Operador de Ponto de Carregamento* (OPC);
- os postos de carregamento de acesso privado ligados à rede por opção do *Detentor de Ponto de Carregamento* (DPC).

Segundo a rede *Movi.E* é possível carregar o veículo em qualquer posto de carregamento da Rede (Continente, Açores e Madeira), independentemente do respetivo OPC ou DPC, desde que exista um contrato ativo com qualquer *Comercializador de Eletricidade para a Mobilidade Elétrica* (CEME). Os CEME cobram os seus serviços com uma margem em função de uma, ou de uma combinação, da seguinte estrutura tarifária:

- custo por energia consumida – €/kWh;
- custo por unidade de tempo – €/min;
- custo por sessão de carregamento – €/carregamento.

Tendo em consideração a variação de preços dos combustíveis/energia e o facto de existirem muitos consumidores com telemóveis com sistema operativo *Android*, foi desenvolvido o presente projeto que permite ao utilizador ter acesso a uma aplicação móvel (na plataforma *Android*) com várias funcionalidades, nomeadamente:

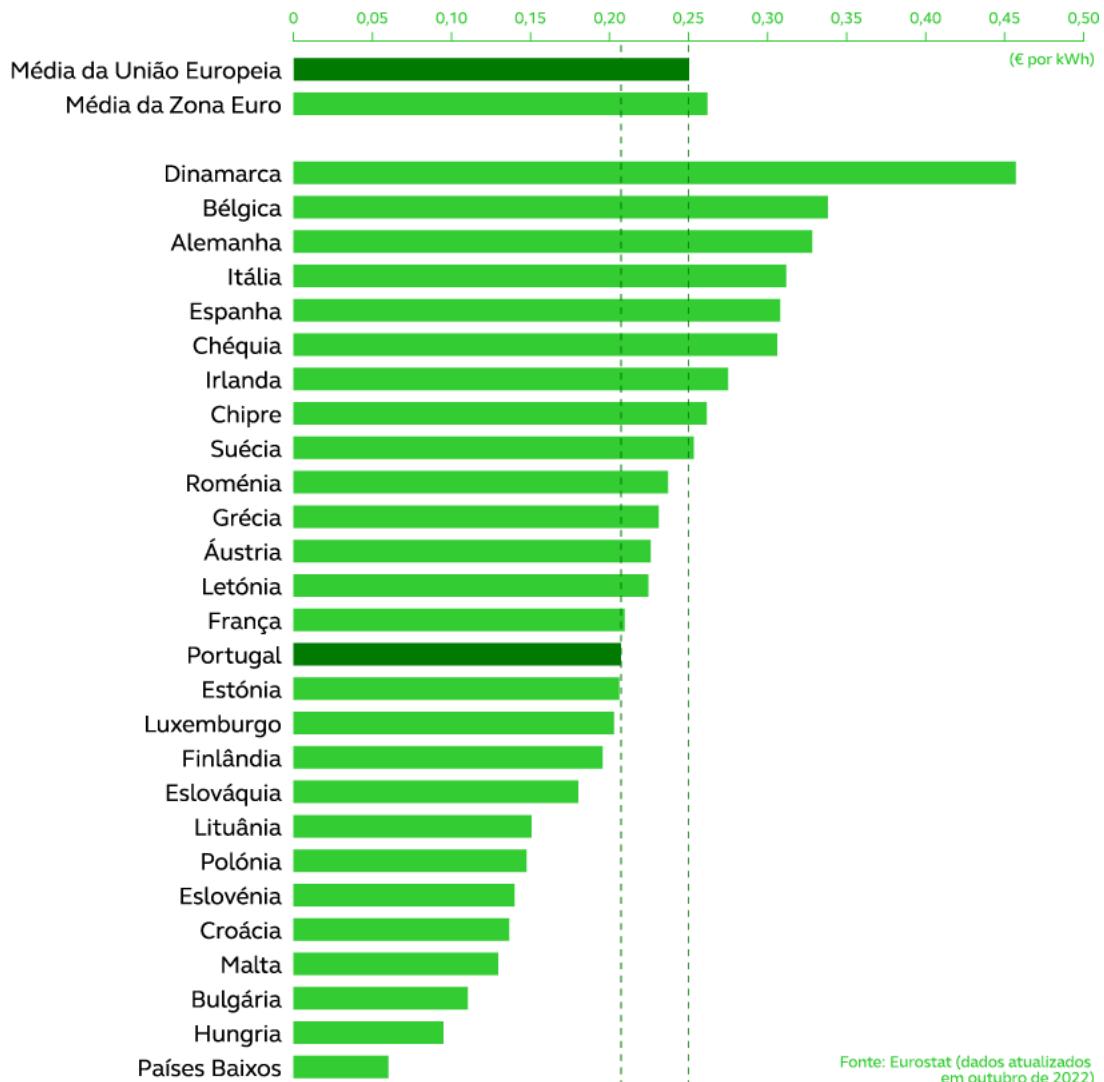


Figura 1.2: Custo médio de carregamento doméstico de veículos elétricos na União Europeia. Fonte ACP (2023).

- localização geográfica dos postos de abastecimento tendo por base a localização do dispositivo móvel;
- consultar preços dos combustíveis/energia e serviços de um dado posto;
- notificações de alteração de preços dos combustíveis em postos favoritos;
- gestão dos veículos do utilizador (criação de despesas gerais e abastecimentos);
- lembretes sobre as manutenções, inspeções, pagamento do Imposto Único de Circulação, entre outros.

O presente relatório encontra-se organizado em 5 capítulos. Nos dois primeiros é dado a conhecer o problema que este projeto pretende resolver, as aplicações existentes no mercado que partilham a mesma finalidade e os requisitos funcionais da aplicação. No terceiro capítulo são descritas as tecnologias usadas, a arquitetura da aplicação e das classes. No quarto capítulo é abordada a implementação das funcionalidades desenvolvidas. Por fim, no último capítulo descreve-se a discussão de resultados, conclusões do trabalho e são indicados pontos de futuras melhorias. Para simplificar a organização do relatório foi criada adicionalmente a seguinte adenda:

- Manual de utilização da aplicação *FuelWise* – descreve o modo de instalação e toda a navegação da aplicação.

Capítulo **2**

Formulação do Problema

Conteúdo

2.1 Estado de Arte	10
2.2 Requisitos funcionais	11

2.1 Estado de Arte

Segundo Abreu (2023), no mercado existem aplicações móveis que têm como finalidade disponibilizar preços de combustível, assim como controlar os gastos com o veículo. Constatou-se que as aplicações analisadas a seguir usam a informação disponibilizada pela Direção-Geral de Energia e Geologia (2023) sobre os preços de combustível de Portugal Continental. A *Direção-Geral de Energia e Geologia* (DGEG) obriga a que todos os postos de abastecimento de Portugal Continental enviem atualizações dos valores dos combustíveis, sempre que os mesmos sofram alterações. Esta obrigação está regulamentada pelo Decreto-Lei n.º 243/2008 de 18 de Dezembro.

O DRE (2023) refere a obrigatoriedade de indicação do preço de venda a retalho dos combustíveis de forma fácil e inequívoca nos postos de abastecimento, bem como fora dos mesmos. Através da utilização de painéis com os preços dos combustíveis permite que o preço dos combustíveis possa constituir um fator de ponderação na opção do consumidor, antes de entrar no posto de abastecimento e, deste modo, também dinamizar a concorrência. No mesmo sentido, o presente decreto-lei refere da importância da informação estar o mais acessível possível. Assim, foi desenvolvida uma página de *Internet* da DGEG de acesso livre para a consulta de preços de combustível sobre postos de abastecimento de Portugal Continental.

A aplicação VivaGas (2023) permite informar sobre os preços dos combustíveis dispersos pelos postos de abastecimento em Portugal. Está disponível nas plataformas *Android* e *iOS*. Esta aplicação mostra a informação sobre serviços, horários dos postos de abastecimento e apresenta numa lista os preços dos combustíveis ordenados de forma crescente com base no menor preço ou na menor distância face à localização atual. Tem disponível a opção de visualizar os postos de abastecimento no mapa.

Uma aplicação que se insere na mesma tipologia da anterior é a MaisGasolina (2023). Está disponível para plataforma *Android* e via *World Wide Web* (WWW). Apresenta os preços de combustível numa lista com ordenação do mais barato para mais caro ou tem a opção de ordenar com base na menor distância face à localização atual. Não disponibiliza nenhum mapa com os postos de abastecimento georreferenciados. Tem como particularidade, filtrar os postos de abastecimento por concelho e por local. Por último, esta aplicação apresenta publicidade ao longo dos vários ecrãs.

A aplicação Fuelio (2023) para além de informar sobre o preço dos combustíveis, também permite fazer uma gestão de despesas dos veículos e está disponível nas plataformas *Android* e *iOS*. Tem como funcionalidades, o registo de abastecimentos, despesas organizadas por categorias, e lembretes com diferentes periodicidades. Apresenta um mapa com a localização dos postos de combustível e respetivos preços, assim como uma linha crono-

lógica dos vários regtos. Por fim, mostra alguns gráficos estatísticos sobre as despesas e consumos.

A outra aplicação do mesmo tipo é a Drivvo (2023), que para além do suporte das funcionalidades descritas pela aplicação anterior também está desenvolvida para plataformas *Android* e *iOS* e disponibiliza outras funcionalidades numa versão paga, nomeadamente, suporte de multi-utilizadores. Adicionalmente, apresenta publicidade ao longo da aplicação. Tem planos pagos para remover publicidade, ter acesso a registo de lembretes, despesas e abastecimentos ilimitados, definição de vários veículos ativos e exportação de dados e apoio técnico, entre outras opções conforme o plano contratualizado. Uma diferença entre a aplicação *Drivvo* e as aplicações *Fuelio* e a desenvolvida neste projeto, reside no facto de ser necessário autenticação para usar as funcionalidades da aplicação.

A aplicação deste projeto FuelWise (2023), para além de garantir as principais funcionalidades das aplicações existentes no mercado adiciona uma funcionalidade que a torna distinta das aplicações referenciadas. Essa funcionalidade consiste na geração de notificações no dispositivo sobre as alterações de valores no combustível pretendido referente aos postos de abastecimento assinalados pelo utilizador como favoritos. As aplicações presentes no mercado, permitem assinalar postos como favoritos, mas não geram notificações sobre alteração de preços de combustível.

2.2 Requisitos funcionais

De modo a cumprir as funcionalidades descritas, a aplicação deste projeto suporta:

- criar veículos com a possibilidade de escolher até dois combustíveis (no caso de ser híbrido);
- consultar a lista de veículos criados e visualização de um dado veículo;
- permitir editar e apagar um veículo;
- criar lembretes;
- consultar a lista de lembretes criados e visualização de um dado lembrete;
- permitir editar e apagar um lembrete;
- criar despesas;
- consultar a lista de despesas criadas e visualização de uma dada despesa;

- permitir editar e apagar uma despesa;
- criar abastecimentos;
- consultar a lista de abastecimentos criados e visualização de um dado abastecimento;
- permitir editar e apagar um abastecimento;
- visualizar postos de abastecimento georreferenciados no mapa;
- consultar preços de combustível, serviços e horário de um dado posto de abastecimento;
- navegar para um dado posto de abastecimento através de uma aplicação competente para o efeito;
- assinalar um posto de abastecimento como favorito;
- consultar a lista de postos de abastecimento favoritos;
- obter notificações sobre alterações de preços de combustível de um dado posto assinalado como favorito.

A informação dos postos de abastecimento é obtida recorrendo à *Application Programming Interface* (API) da DGEG, disponível em Direção-Geral de Energia e Geologia (2023).

Capítulo **3**

Abordagem

Conteúdo

3.1	Tecnologias	14
3.2	Arquitetura	15
3.2.1	Análise	15
3.2.2	Fluxos de comunicação	18
3.3	Navegação no <i>Android</i>	19
3.4	Arquitetura de classes	23

3.1 Tecnologias

Com base na premissa de desenvolver uma aplicação móvel, a primeira decisão tomada foi em que sistema operativo iria ser desenvolvida. Existia a possibilidade de desenvolver a aplicação em *cross-plataform* (por exemplo: Xamarin (2023)). Através de um código único a aplicação seria executada em vários sistemas operativos móveis (*Android* e *iOS*). Por outro lado, outra solução seria desenvolver de forma nativa em *Android* ou *iOS*. Foi ponderada a decisão tendo em conta as várias soluções disponíveis e foi decidido desenvolver a aplicação de forma nativa em *Android* com base em diversos fatores. Durante a frequência do curso de licenciatura em Engenharia Informática e de Computadores, existiu a oportunidade de poder aprender a programar aplicações nativas em *Android* através da unidade curricular de Programação em Dispositivos Móveis, o que possibilitou ao autor deste projeto ter conhecimento das boas práticas na arquitetura de uma aplicação desta natureza. Para além disso, optou-se por programar numa linguagem nativa, tendo como vantagem obter um maior desempenho, em detrimento de programar em *cross-plataform*. Ter um dispositivo *Android* também contribuiu na decisão, dado que permite testar aplicação num dispositivo físico.

A escolha da linguagem *Kotlin* para desenvolver a aplicação deveu-se a dois fatores. Em 2017, a Google anunciou como linguagem oficial do *Android* o *Kotlin* em detrimento da linguagem Java. De acordo com Google (2023b), o *Kotlin* é descrito como uma linguagem de programação moderna e estaticamente tipificada e é usada por mais de 60% dos programadores *Android*. Em relação à criação de *User Interface* (UI) foi decidido desenvolver usando *Jetpack Compose*. Segundo Google (2023a) o *Jetpack Compose* é um kit de ferramentas moderno para criar UI nativas, simplificando e acelerando o desenvolvimento da UI. Este kit de ferramentas moderno de UI do *Android* foi também desenvolvido em *Kotlin*.

Com base no tempo projetado para o desenvolvimento da aplicação optou-se por fazer o armazenamento dos dados de forma local e gestão através da biblioteca *Room*. A biblioteca de persistência Room (2023) oferece uma camada de abstração sobre o *SQLite* permitindo o acesso à *Base de Dados* (BD) e aproveitando toda a capacidade do *SQLite*. O *Room* oferece como vantagens:

- verificação de consultas SQL durante a compilação;
- anotações que minimizam o código *boilerplate* repetitivo e propenso a erros;
- migração simplificada da BD.

Esta opção de implementar uma BD local limita a sua utilização ao utilizador do dispositivo, inviabilizando os dados a serem partilhados por outros utilizadores que em determinado contexto poderiam partilhar o mesmo veículo. Neste caso os dados teriam de estar armazenados na *Cloud*, de forma a disponibilizar a informação nos dispositivos *Android* dos utilizadores que partilhem esse mesmo veículo. O armazenamento de dados localmente tem como vantagem não requerer a criação de uma conta de utilizador para aceder à aplicação. No que respeita aos pedidos *Hypertext Transfer Protocol* (HTTP) por parte da aplicação à *Application Programming Interface* (API) da *Direção-Geral de Energia e Geologia* (DGEG) foi escolhida a biblioteca *Retrofit* por ser uma biblioteca bastante usada nas aplicações desenvolvidas no sistema operativo *Android*. A biblioteca *Retrofit* é uma API desenvolvida pela Square (2023), que segue o padrão *Representational State Transfer* (REST), e fornece uma implementação simples para a transmissão de dados entre a aplicação e o servidor, fazendo uso do *JavaScript Object Notation* (JSON).

3.2 Arquitetura

3.2.1 Análise

Para cumprir os requisitos funcionais deste projeto descritos no capítulo anterior, foram analisadas duas arquiteturas para desenvolver a aplicação. A primeira corresponde ao desenvolvimento de uma aplicação *standalone*, tendo como vantagens:

- a BD está armazenada no dispositivo, de modo que os dados do utilizador ficam guardados localmente;
- melhor desempenho no acesso aos dados atualizados da BD local em detrimento de realizar repetidamente pedidos HTTP ao servidor;
- no caso do servidor não estar operacional, dado a BD ser local a informação sobre a gestão dos veículos estará sempre disponível, ficando apenas afetada a obtenção da informação sobre atualização dos preços de combustível.
- as funcionalidades relacionadas com a gestão do veículo não necessitam de recurso à Internet, possibilitando uma poupança na utilização da bateria do dispositivo.

Esta arquitetura tem como desvantagens:

- no caso de ocorrer uma mudança no *Uniform Resource Locator* (URL) referente à API da DGEG implica uma atualização da aplicação;

- todo o processamento ocorre do lado do dispositivo (por exemplo: execução de tarefas em segundo plano, guardar os dados numa BD local);
- todas as aplicações cliente comunicam diretamente com a API da DGEG, ocorrendo um maior número de pedidos a esse servidor.

A segunda arquitetura analisada corresponde à estrutura cliente-servidor. Nesta arquitetura, é desenvolvido um servidor que tem como principal função comunicar com a aplicação para a transmissão de dados. Também tem como função comunicar com a API da DGEG. Esta arquitetura tem como vantagens:

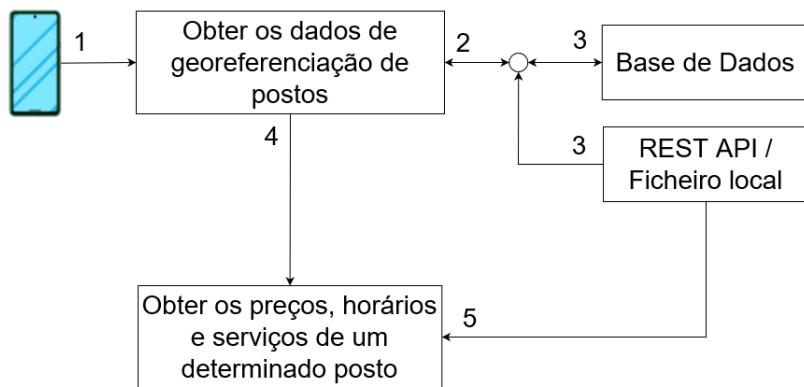
- a BD encontra-se alojada no servidor;
- todo o processamento ocorre do lado do servidor e a aplicação cliente apenas realiza pedidos HTTP;
- com o servidor a ser desenvolvido pelo programador, é possível manter uma API estável na comunicação com a aplicação cliente;
- apenas o servidor comunica com a API da DGEG permitindo minimizar o número de pedidos.

E tem como desvantagens:

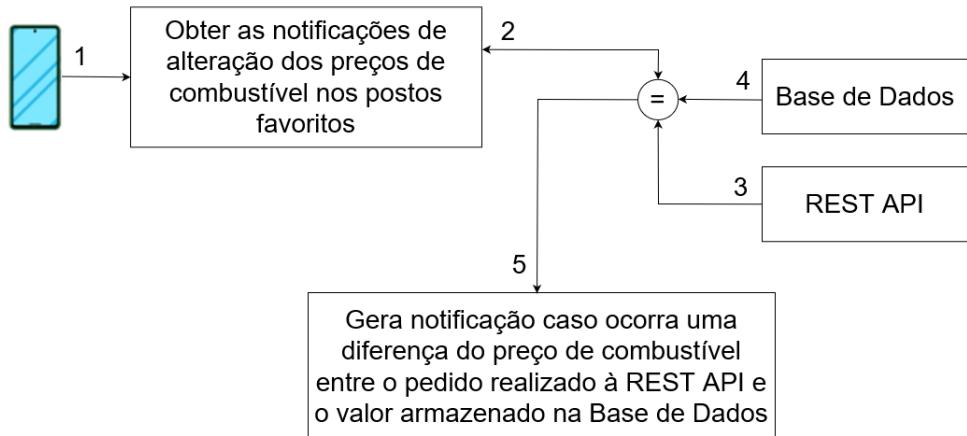
- a obtenção de todos os dados é realizado através do recurso à Internet, podendo ocorrer um maior consumo na utilização da bateria do dispositivo;
- os dados apresentados pela aplicação cliente dependem na totalidade do normal funcionamento do servidor;
- custos relacionados com o alojamento do servidor.

Com base nestas duas arquiteturas, foi decidido implementar a arquitetura de aplicação *standalone*. Esta decisão foi tomada tendo em conta existir apenas um autor neste projeto, o nível de complexidade da arquitetura cliente-servidor ser maior o que poderia implicar requerer mais tempo do que está estabelecido para este projeto.

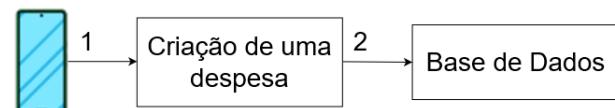
Fluxo 1



Fluxo 2



Fluxo 3



Fluxo 4

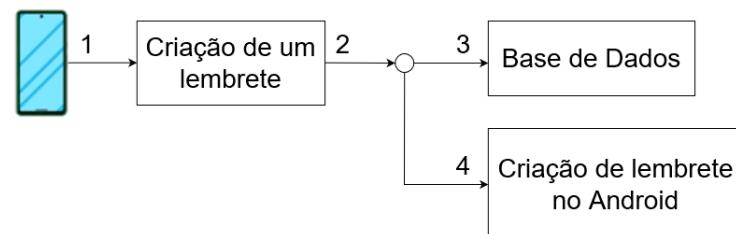


Figura 3.1: Diagrama da arquitetura da aplicação. Este encontra-se organizado em quatro fluxos de comunicação.

3.2.2 Fluxos de comunicação

Na arquitetura de aplicação *standalone* foram desenvolvidos quatro fluxos de comunicação para implementar os requisitos funcionais. O diagrama da arquitetura encontra-se ilustrado na figura 3.1.

O primeiro fluxo corresponde à obtenção de postos de abastecimento georreferenciados no mapa e na obtenção da informação de um determinado posto. Os dados sobre a georreferenciação dos postos podem ser obtidos através da BD ou através de pedidos à API da DGEG ou pela via de uma fonte de dados local (postos de carregamento para veículos elétricos – requisito opcional).

Foi tomada a decisão de, com base numa data de referência, considerar se os preços de combustível e de energia elétrica armazenados na BD estão atualizados ou não. No caso da data em análise ser diferente da data de referência, os pedidos são efetuados à API da DGEG ou à fonte de dados local dependendo do tipo de combustível. Os valores obtidos são guardados na BD. Esta implementação permite limitar os pedidos feitos aos servidores da DGEG por parte da aplicação e em melhorar o desempenho no carregamento dos postos de abastecimento no mapa. Destaca-se a opção de incluir os postos de carregamento para veículos elétricos, dado que não estava projetado essa funcionalidade inicialmente na proposta de projeto apresentada. Com o intuito de valorizar o projeto foi feita uma pesquisa na obtenção de fontes de dados fidedignas. A informação da georreferenciação dos postos de carregamento foi obtida do *site* UVE (2023) e a informação das tarifas praticadas em cada posto foi obtida através do *site* Mobi.E (2023). Sem acesso a uma API gratuita para obter informação sobre os postos de carregamento e os preços praticados, a informação obtida dos sites referenciados anteriormente foi inserida em dois ficheiros, a informação dos postos de abastecimentos e as tarifas de energia cobradas em cada posto, respetivamente.

Após a seleção de um posto é realizado um novo pedido à API da DGEG ou à fonte de dados local (no caso de ser um posto de carregamento) para apresentar informação sobre a marca do posto, o nome, a morada, os preços de combustível, assim como, os serviços e o horário disponível.

No segundo fluxo é representada a tarefa em segundo plano que tem como objetivo realizar os pedidos à API da DGEG sobre um determinado combustível dos postos assinalados pelo utilizador como favoritos. Os preços de combustível obtidos dos vários pedidos realizados são comparados com os valores guardados na BD. No caso de ocorrerem diferenças nos preços são geradas notificações sobre essas alterações. Na versão atual do projeto, o mecanismo de notificação de alteração do preço para o caso dos postos de carregamento elétrico não se encontra implementado devido à ausência de uma API

gratuita com a informação dos preços atualizados.

O terceiro fluxo relaciona-se com a criação de uma despesa na BD local.

Por fim, o quarto fluxo demonstra a criação de um lembrete na BD local e no sistema operativo *Android*.

3.3 Navegação no *Android*

Segundo Google (2023d), a classe *Activity* é um componente crucial de uma aplicação *Android*. Distinto dos paradigmas de programação em que as aplicações são lançadas com a função *main*, a plataforma *Android* inicia o código com uma instância *Activity* invocando métodos de *callback* que correspondem a fases específicas do ciclo de vida da *Activity*. Com o aumento de complexidade da aplicação, o número de atividades pode também aumentar. No caso de múltiplas *Activities*, cada uma delas é usada para definir a UI e os comportamentos dentro do contexto do próprio ecrã na interação do utilizador. O conceito de *Single Activity* é utilizado para simplificar a arquitetura da aplicação, permitindo que todos os ecrãs estejam dentro de uma única atividade, com a navegação a ser controlada por composições do *Jetpack Compose* (figura 3.2). A utilização de múltiplas *Activities* implica descrever o comportamento a adotar em relação aos vários *callbacks*, assim como, declarar as atividades no manifesto. Além disso, para realizar a passagem de uma *Activity* para outra, é necessário enviar uma solicitação para outra *Activity*. No que respeita às animações da aplicação, estas podem ficar afetadas na transição de uma *Activity* para outra, onde pode ocorrer cintilação na transição.

Neste projeto tomou-se a decisão da aplicação usar uma arquitetura de *Single Activity*. Com apenas uma *Activity* em toda arquitetura, não há necessidade de atualizar o manifesto para adicionar mais *Activities*. Também não existe necessidade de declarar o método *startActivityForResult*, dado que a navegação entre ecrãs ficará entregue ao componente de navegação, designadamente, o *NavController*. Este é a API central do componente de navegação em *Android*. Relativamente ao problema de animação na transição entre *Activities*, este fica resolvido porque a transição ocorreria entre ecrãs dentro da mesma *Activity*. Por fim, é possível passar dados entre ecrãs com recurso ao componente de navegação e assegurar que o tipo de dados passado é o esperado.

A navegação na aplicação está suportada pelo *NavController* (figura 3.3) e refere-se às interações dos utilizadores em entrar e sair dos diferentes ecrãs da aplicação.

Com base em Google (2023g), o componente de navegação está dividido em três partes principais:

- Grafo de navegação: é um recurso XML que contém toda a informação relacionada

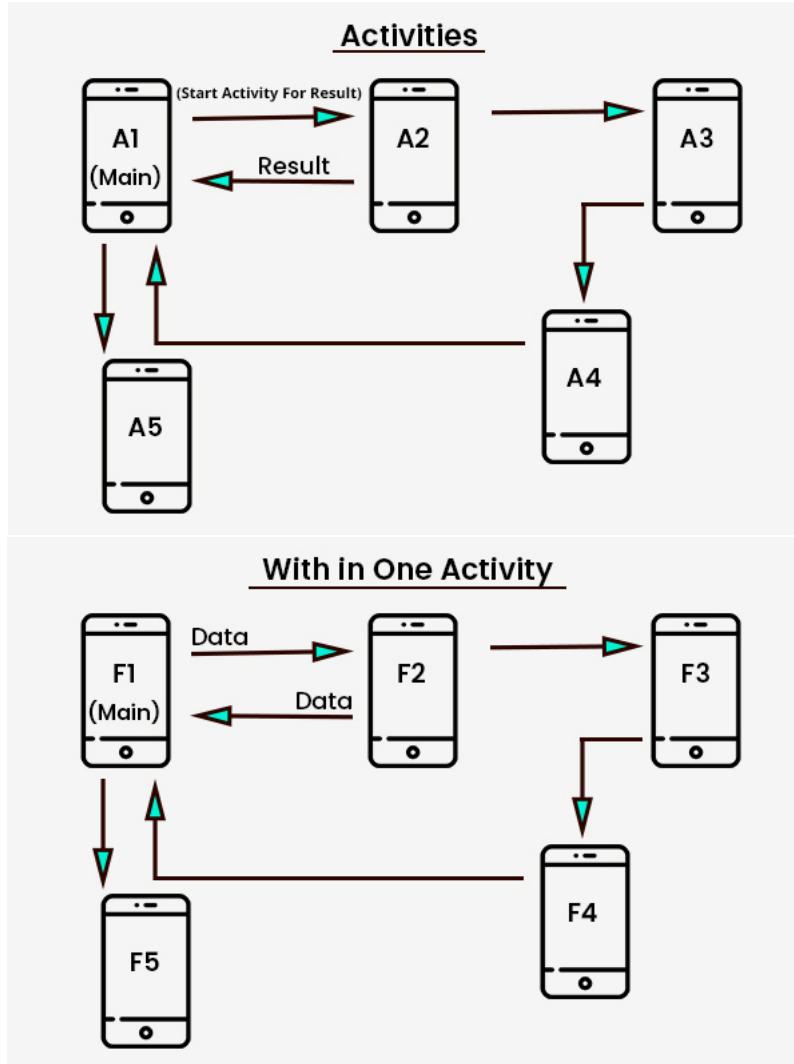


Figura 3.2: Fluxo de dados com múltiplas *Activities* ou uma única *Activity*. Fonte Academy (2023).

com navegação e inclui as rotas de destino que o utilizador pode aceder;

- *NavHost*: mostra os destinos do grafo de navegação;
- *NavController*: é um objeto que gera a navegação da aplicação a partir do *NavHost* associado.

A organização de código estruturado facilita a manutenção do software. Tendo em conta a arquitetura do *Android*, o modelo *Model View ViewModel* (MVVM) é o padrão de arquitetura de software reconhecido pela indústria e que melhor se enquadra no fluxo de informação pretendido na aplicação. O MVVM sugere separar a apresentação de dados (*Views*) da lógica de negócios da aplicação. O presente projeto está implementado segundo

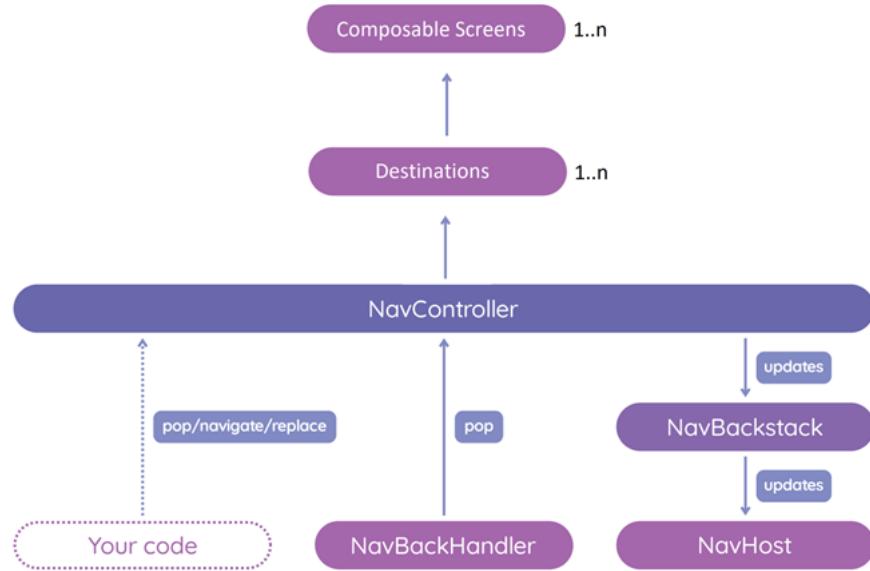


Figura 3.3: Navegação com *Compose*. Fonte Navigation (2023).

o modelo MVVM, baseado numa arquitetura de *Single Activity* em que cada ecrã desenvolvido em *Jetpack Compose* define a sua visualização, e capta as ações do utilizador enviando para o seu *ViewModel* respetivo. O *ViewModel* é responsável por manter os valores referentes à lógica de negócio Google (2023l). A principal vantagem do *ViewModel* é que ele armazena o estado em *cache* e mantém esses valores mesmo após mudanças de configuração (por exemplo, troca de idioma ou rotação de ecrã) onde a *Activity* é destruída, e é instanciada uma nova. O ciclo de vida de um *ViewModel* está diretamente vinculado ao *scope* dele. Um *ViewModel* permanece na memória até que o *ViewModelStoreOwner* desapareça. A figura 3.4 ilustra os vários estados do ciclo de vida de uma *Activity* e do *ViewModel* associado.

Através do *ViewModel* é possível comunicar com o repositório e aceder a uma fonte de dados local (no caso em concreto a *BD Room*) utilizada para armazenar os veículos, despesas, abastecimentos, lembretes e postos de abastecimento favoritos. O *ViewModel* também é usado para poder aceder a uma fonte de dados remota, neste caso, à API da DGEV para obter informação sobre os preços de combustível, horários e serviços dos postos de abastecimento (figura 3.5).

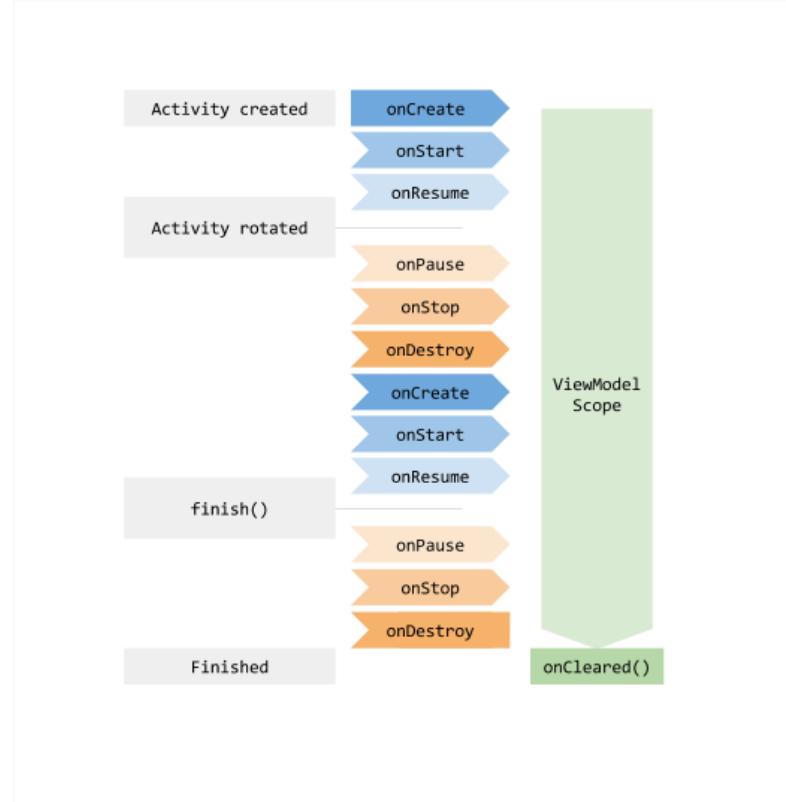


Figura 3.4: Ciclo de vida de uma *Activity* e *ViewModel* respetivo. Fonte Google (2023l).

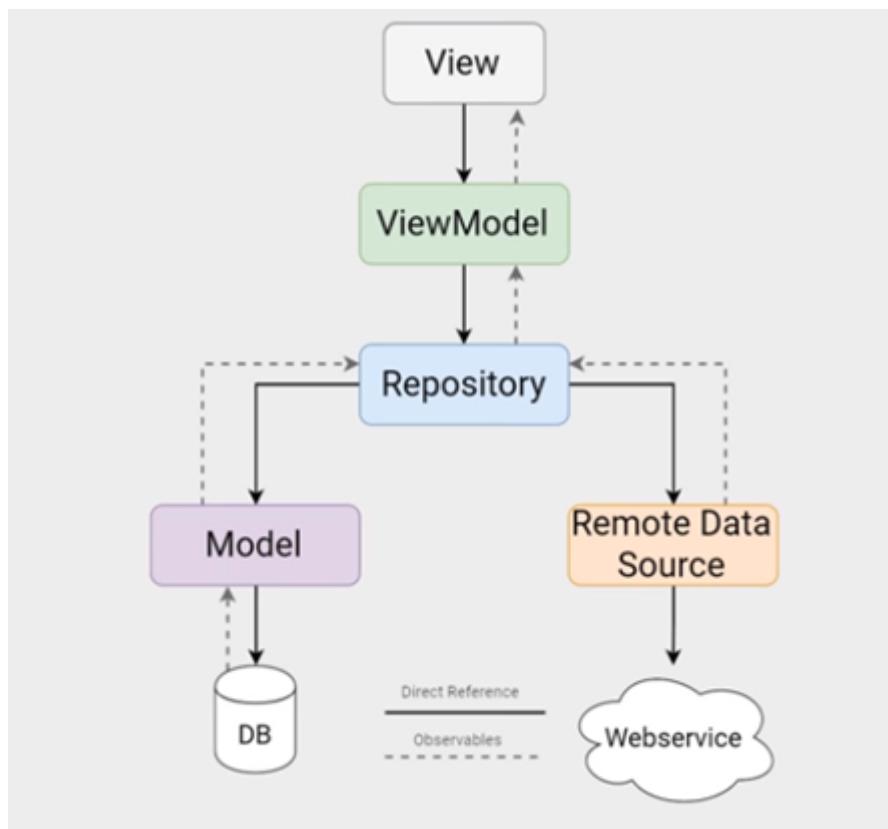


Figura 3.5: Diagrama da arquitetura *MVVM*. Fonte Coder (2023).

3.4 Arquitetura de classes

Ao longo do desenvolvimento da aplicação organizou-se as classes em *packages*. O nome de cada *package* foi atribuído tendo em conta a função das classes que contém. Esta organização visa facilitar arrumação das classes e a navegação no projeto. O *package Common* visa agregar as classes que suportam funcionalidades que vão ser usadas por outro tipo de classes. A hierarquia do *package Common* está definida do seguinte modo:

- *AlarmManager* – permite criar e apagar alarmes no sistema operativo *Android* e agendar novos alarmes no arranque do dispositivo que ainda não expiram;
- *Codification* – agrupa todos os códigos de combustível, distritos utilizados nos pedidos realizados à API da DGEG, bem como a definição das categorias usadas nas despesas e a definição da periodicidade dos lembretes;
- *Converters* – tem uma função que converte uma imagem *Bitmap* em vetor;
- *Database* – encontra-se definida a estrutura da BD e o preenchimento das tabelas no arranque da aplicação. As classes que têm como terminação *Data Access Object* (DAO) constituem classes de acesso a dados (contêm funções que são convertidas em *queries* à BD), enquanto que as classes com a terminação *Entity* descrevem a estrutura de uma tabela, as restrições de integridade e chaves primárias. As classes com a terminação *Repository* são usadas nos vários *ViewModels* para acesso às classes de acesso a dados;
- *DataClearedManager* – tem como finalidade após a desinstalação da aplicação, ou caso sejam apagados os dados da mesma poder captar esse evento, e parar o serviço em primeiro plano (caso se encontre ativo). Esse serviço suporta o trabalho persistente periódico de verificação de alteração de preços de combustível nos postos de abastecimento assinalados como favoritos;
- *ForegroundServiceManager* – encontra-se definido o serviço em primeiro plano, e no caso do dispositivo ser reiniciado ou desligado com o serviço ativo, está definido iniciar esse serviço novamente no futuro arranque do dispositivo.
- *Location* – conjunto de funções que permitem obter a localização do dispositivo;
- *Messages* – define o modo de envio de mensagens informativas ao utilizador;
- *NotificationManager* – encontra-se definido o gestor de notificações;

- *Retrofit* – biblioteca usada para realizar pedidos HTTP. Também estão definidos os vários URL para realizar os pedidos à API da DGEG;
- *Settings* – contém as definições da aplicação;
- *Validations* – validações comuns usadas em várias classes;
- *WorkManager* – criação de uma tarefa em segundo plano para consultar a API da DGEG num intervalo de tempo definido e comparar os preços de combustível obtidos nos postos favoritos com os preços respetivos guardados na BD local.

Os *packages* com terminação *List* correspondem à descrição do ecrã e respetivo *ViewModel* das listas das várias classes de domínio. Os *packages* *User*, *Expense*, *Reload*, *Reminder* e *Vehicle* contêm a descrição do ecrã e respetivo *ViewModel* na fase de criação e de edição.

O *package* *StationInfo* contém a descrição do ecrã relativo à informação de um posto de abastecimento e o seu *ViewModel*. No *package* *StationsMap* agrupa a informação sobre a visualização do *Google Maps* com os postos de abastecimento georreferenciados e o seu *ViewModel*.

O *package* *UI* está dividido entre o *package* *Components*, que agrupa os componentes comuns da UI entre vários ecrãs e o *package* *Themes*, onde se encontra definido a paleta de cores no modo claro e escuro, tipo de formas, tipo de fonte, entre outras formatações da aplicação. O *package* *Components* está dividido entre *Common*, que reúne as funções *composable*, destacando-se a função de instanciação do *Google Maps*, bem como, a definição do fluxo de solicitação de permissões de acesso a recursos do dispositivo, e por fim, à personalização de listas no formato *drop down* e caixas de texto editáveis. O *package* *Scaffold* é composto pelas funções de UI que têm como finalidade definir o *layout* da aplicação. O Google (2023f) descreve as configurações gerais do *Scaffold* que serviu de base para a construção do design da aplicação.

As classes que não estão dentro de nenhum *package* são, nomeadamente, *FuelWiseActivity*, *FuelWiseApp*, *FuelWiseApplication* e *MakeFuelWiseNav*. A *FuelWiseActivity* corresponde à instanciação da única *Activity* da aplicação, *FuelWiseApp* está relacionada com a instanciação da classe que é responsável pela manutenção de estado da aplicação e no envio das mensagens informativas vindas do *ViewModel* para o ecrã ativo no momento. A *FuelWiseApplication* é a classe responsável pela instanciação dos serviços e dependências. Por fim, *MakeFuelWiseNav* corresponde ao grafo de navegação dos ecrãs da aplicação.

Capítulo **4**

Implementação

Conteúdo

4.1	Ferramentas utilizadas	26
4.2	Modelo de Base de Dados	26
4.3	Permissões	27
4.4	Serviço e trabalho persistente	30
4.5	Inicialização da aplicação	31
4.6	Criação de utilizador	31
4.7	Ecrã inicial	31
4.8	Lista de opções	32
4.9	Criação de um veículo	33
4.10	Consulta e edição de um veículo	33
4.11	Lista de veículos	34
4.12	Criação de um abastecimento	34
4.13	Criação de uma despesa	35
4.14	Criação de um lembrete	36
4.15	Postos de abastecimento favoritos	37
4.16	Mapa com os postos de abastecimento	38
4.17	Manual de utilizador da aplicação	39
4.18	Testes unitários	40

4.1 Ferramentas utilizadas

Como ferramenta para desenvolver o projeto foi usado o *Android Studio*, por ser uma ferramenta recomendada no desenvolvimento de aplicações nativas em *Android* e por ser de acesso gratuito. Outra vantagem, deste ambiente de desenvolvimento integrado é a de permitir testar as aplicações num emulador.

4.2 Modelo de Base de Dados

Os dados introduzidos pelo utilizador são armazenados de forma persistente através da biblioteca *Room*. A arquitetura da biblioteca *Room* tem três componentes principais, designadamente:

- a classe da *Base de Dados* (BD), que contém a BD e serve de ponto de entrada para o acesso aos dados persistentes da aplicação;
- as entidades de dados, que representam tabelas da BD da aplicação;
- os objetos de acesso a dados, *Data Access Object (DAO)*, que fornecem métodos que a aplicação pode usar para consultar, atualizar, inserir e apagar os dados da BD.

A figura 4.1 mostra a relação entre os diferentes componentes da biblioteca *Room*.

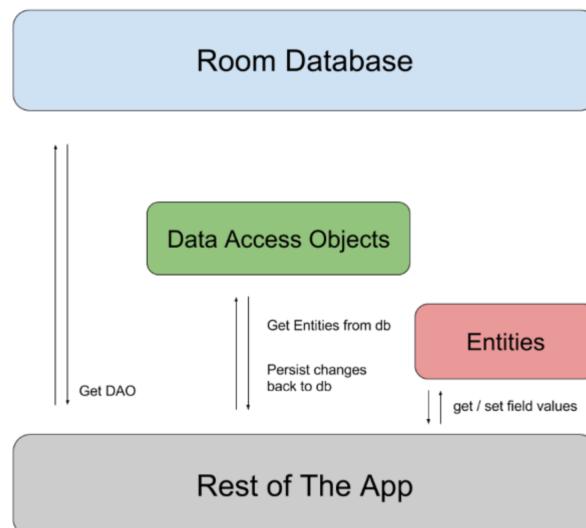


Figura 4.1: Diagrama da arquitetura da biblioteca *Room*. Fonte Google (2023j).

Começou-se por modelar a BD para suportar os requisitos funcionais da aplicação, tendo sido criados o modelo Entidade-Associação e o modelo Relacional. Estes encontram-se descritos nos apêndices A e B, respetivamente.

4.3 Permissões

As permissões da aplicação são criadas com base nos critérios de segurança do sistema operativo *Android* e ajudam a controlar a privacidade do utilizador (Google (2023h)). Os critérios são:

- Controlo: o utilizador tem controlo sobre os dados que compartilha com a aplicação;
- Transparência: o utilizador determina os dados que a aplicação utiliza e qual o motivo de acesso a esses mesmos dados;
- Minimização de dados: uma aplicação acede e usa apenas os dados necessários para uma tarefa ou ação específica que o utilizador invoca.

Segundo Google (2023i) cada aplicação *Android* é executada numa *sandbox* com acesso limitado. Se a aplicação precisar de usar recursos ou informações fora da própria *sandbox*, é necessário declarar uma permissão de execução e configurar uma solicitação de permissão para fornecer esse acesso. Essa etapa faz parte de um fluxo para usar permissões (figura 4.2).

Na solicitação de permissões existem princípios básicos que são necessários ter em conta, nomeadamente:

- solicitar uma permissão no contexto, ou seja, quando o utilizador interage com o recurso que requer a permissão;
- não bloquear o utilizador, deve ser dada a opção de cancelar o fluxo, explicando ao utilizador as permissões necessárias para dar acesso ao recurso associado ao fluxo;
- se o utilizador negar ou revogar uma permissão necessária para um recurso, deve-se aplicar a desativação dessa funcionalidade da aplicação.

Neste projeto foram criadas as solicitações de permissão com base nas práticas recomendadas e princípios mencionados anteriormente. Assim, caso o utilizador aceite as solicitações poderá aceder à funcionalidade. Contudo, no caso de negar as solicitações

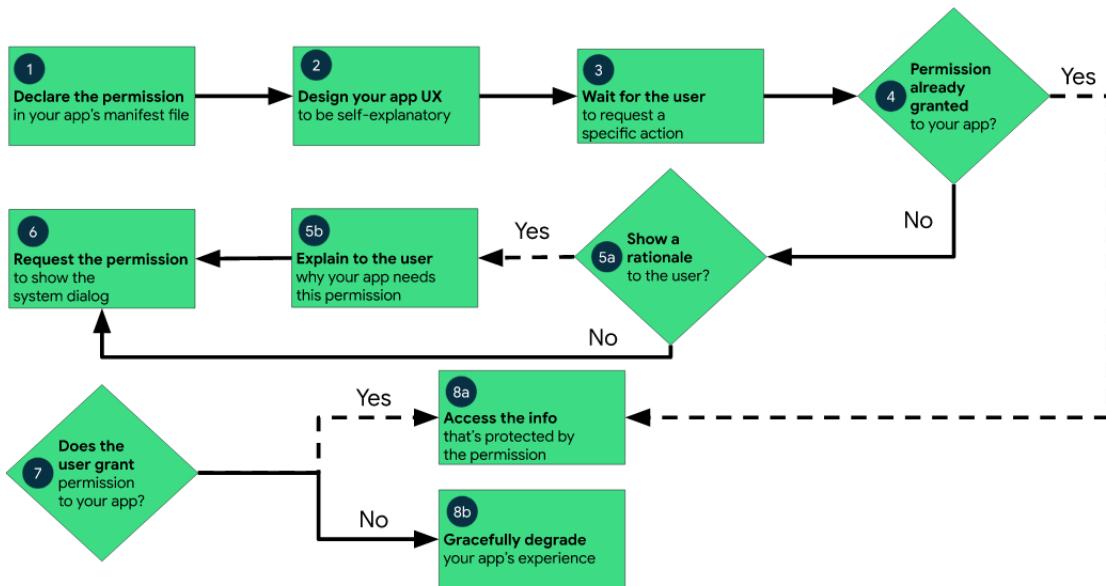


Figura 4.2: Diagrama do fluxo sobre a solicitação de permissão para acesso a um recurso no *Android*. Fonte Google (2023h).

de permissão, será negado o acesso à funcionalidade em questão apresentando uma informação no ecrã a explicar o motivo pelo qual a funcionalidade não está disponível. Na situação de autorizar o acesso aos recursos requeridos e a dada altura essa autorização for revogada pelo utilizador, a funcionalidade também deixará de estar disponível. As autorizações que o utilizador necessita de dar permissão são declaradas no manifesto da aplicação e são as seguintes:

- *Access Fine Location* – acesso à localização exata do dispositivo;
- *Access Coarse Location* – acesso à localização aproximada do dispositivo;
- *Post Notifications* – acesso no envio de notificações.

Existem outros acessos a recursos que o próprio sistema operativo assegura sem requerer a permissão do utilizador:

- *Foreground Service* – acesso a criar um serviço de primeiro plano;
- *Internet* – acesso à Internet do dispositivo;
- *Access Network State* – acesso ao estado da rede;
- *Access Wi-Fi State* – acesso ao estado do Wi-Fi;

- *Schedule Exact Alarm* – acesso a executar um alarme numa hora exata;
- *Receive Boot Completed* – acesso à informação de quando o dispositivo foi reiniciado.

A figura 4.3 ilustra um exemplo de acesso a um recurso, neste caso, uma solicitação de permissão para envio de notificações pela aplicação *FuelWise*.



Figura 4.3: Solicitação de permissão para envio de notificações pela aplicação *FuelWise*.

4.4 Serviço e trabalho persistente

Nesta aplicação foi implementado um serviço em primeiro plano, que tem como principal objetivo manter a execução de uma tarefa em segundo plano, sem requerer que a aplicação se encontre em execução. Através da integração destas duas soluções, foi desenvolvida a funcionalidade diferenciadora desta aplicação. As notificações são geradas quando ocorrem alterações de preços de combustível nos postos assinalados como favoritos. Essa análise é realizada após comparação dos dados alojados na BD local e no pedido realizado à *Application Programming Interface* (API) da *Direção-Geral de Energia e Geologia* (DGEG). Segundo Google (2023c), os serviços de primeiro plano executam operações perceptíveis ao utilizador. Os serviços em primeiro plano devem de mostrar uma notificação na barra de estado para informar os utilizadores que uma determinada aplicação está executar um serviço que consome recursos do sistema. Em relação ao trabalho persistente, a Google (2023k), desenvolveu uma API, denominada de *WorkManager*, para realizar o processamento em segundo plano. O *WorkManager* processa três tipos de trabalho persistente (figura 4.4):

- Imediato: tarefas que precisam de começar imediatamente e terminar em breve. Podem ser priorizados;
- De longa duração: tarefas que podem ser executadas por mais tempo, potencialmente mais de 10 minutos;
- Adiável: tarefas programadas que começam posteriormente e podem ser executadas periodicamente.

O *WorkManager* permite agendar trabalho para ser executado uma única vez ou repetidamente usando períodos de agendamento flexíveis (iguais ou superiores a 15 minutos). O trabalho também pode receber uma *tag* e um nome, o que possibilita agendar trabalhos exclusivos e substituíveis, bem como, proceder ao seu cancelamento. O trabalho agendado é armazenado numa BD *SQLite* gerida internamente, e o *WorkManager* encarrega-se de garantir que esse trabalho persists e seja agendado novamente durante as reinicializações do dispositivo.

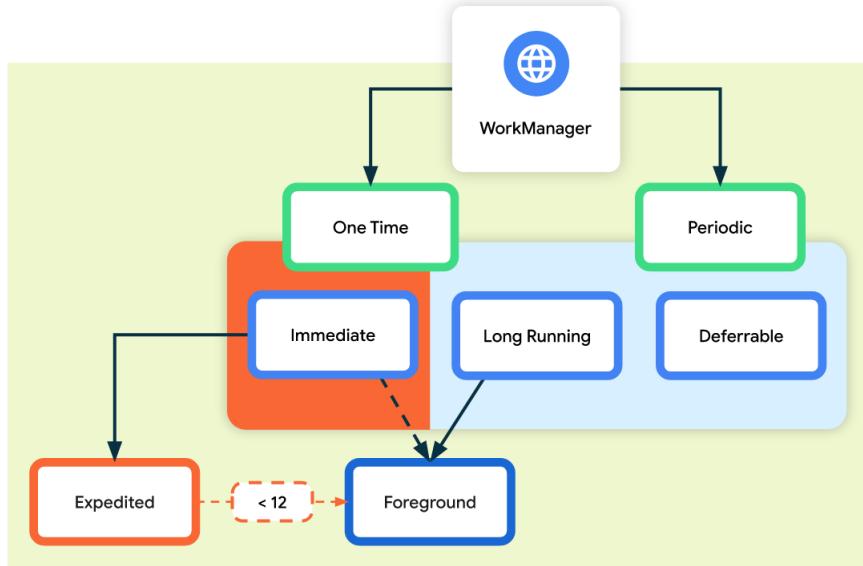


Figura 4.4: Tipos de trabalho persistente. Fonte Google (2023k).

4.5 Inicialização da aplicação

No arranque da aplicação é executada uma única vez o preenchimento das tabelas que servem de suporte às funcionalidades da aplicação. Por exemplo, as categorias definidas nas despesas, a periodicidade dos lembretes, os códigos dos combustíveis e as definições da aplicação. Outra inicialização que ocorre no arranque da aplicação é a inicialização do serviço de notificações.

4.6 Criação de utilizador

O registo do utilizador não carece de autenticação (figura 4.5), apenas são recolhidos dados para posterior apresentação no ecrã inicial da aplicação. Os campos de preenchimento do ecrã de criação de utilizador são de carácter obrigatório e no campo do email é validado o seu formato.

4.7 Ecrã inicial

O ecrã inicial dá as boas-vindas ao utilizador usando o nome registado na criação do utilizador (figura 4.6). Existem dois campos que listam os veículos criados e os combustíveis do respetivo veículo. Esta seleção permite definir o veículo ativo e o combustível ativo.

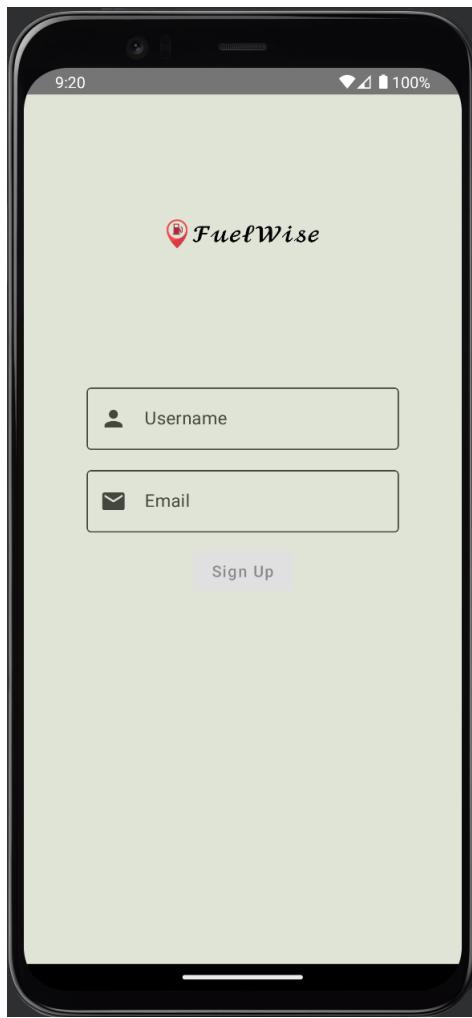


Figura 4.5: Criação de utilizador.

Assim, ao navegar na aplicação as despesas, lembretes e abastecimentos apresentados são relativos ao veículo selecionado no ecrã inicial. Para alterar para outro veículo pode ser feito no ecrã inicial ou nas listas dos abastecimentos, despesas ou lembretes. Para além disso, a consulta dos postos de abastecimento no mapa são apresentados com base no combustível selecionado no ecrã inicial.

4.8 Lista de opções

No ecrã das opções (figura 4.7) estão registados os *links* para a lista de veículos, despesas, lembretes e abastecimentos. O acesso às definições da aplicação é também feito através do ecrã de opções.

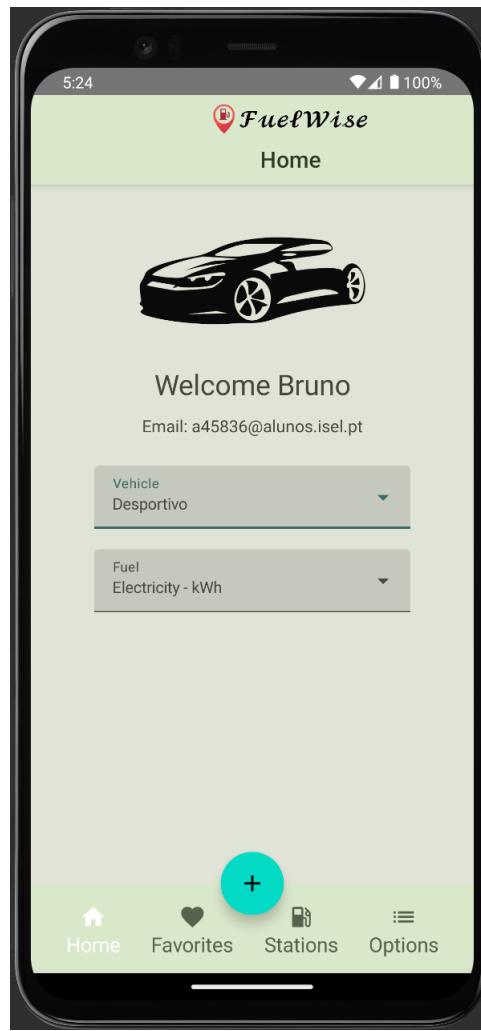


Figura 4.6: Ecrã inicial.

4.9 Criação de um veículo

Na criação de veículos todos os campos são de preenchimento obrigatório. É de salientar que é possível definir até dois combustíveis por veículo ativando a opção *híbrido* (figura 4.8).

4.10 Consulta e edição de um veículo

Pode-se selecionar um dado veículo da lista de veículos. O utilizador é direcionado para um ecrã que apresenta todos os dados relativos ao veículo (figura 4.9) e tem a possibilidade de editar alguns campos do veículo. Na fase de edição é possível, a todo o momento, cancelar o modo de edição e voltar ao modo de consulta.

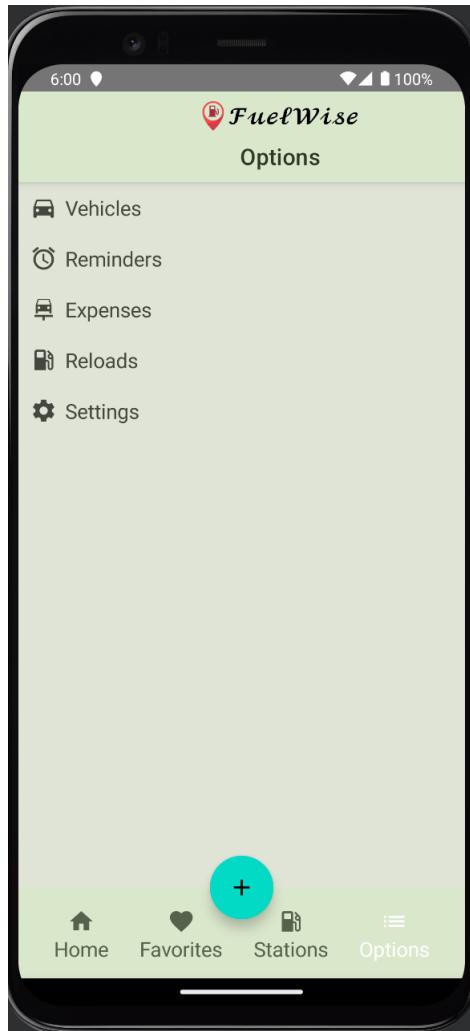


Figura 4.7: Lista de opções.

4.11 Lista de veículos

Existe um ecrã que corresponde à lista de veículos criados. Nesse ecrã aparecem alguns campos de destaque do veículo e é dada a permissão ao utilizador de apagar um dado veículo. Caso o utilizador pressione em apagar um veículo aparece uma mensagem a pedir a confirmação dessa ação antes de o apagar (figura 4.10).

4.12 Criação de um abastecimento

No registo de um abastecimento o utilizador tem de selecionar o veículo e o combustível referente a esse abastecimento (figura 4.11). Para além disso, é necessário registar a quilometragem, quantidade, preço unitário, custo e a data. Estes dados serão importantes

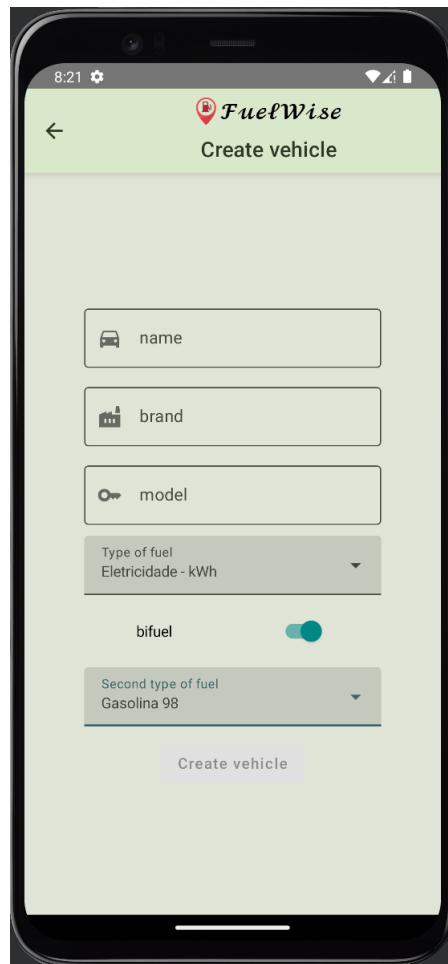


Figura 4.8: Criação de veículo.

para as novas funcionalidades da aplicação no futuro, no que respeita a estatísticas e cálculos de consumos do veículo. Estes campos são todos de preenchimento obrigatório, e são realizadas algumas validações para não permitir dados incorretos. Só são aceites dígitos, sendo para isso ativado o teclado numérico do telefone. No caso do utilizador introduzir várias vírgulas decimais apenas é aceite a primeira vírgula decimal. A data é apresentada no formato de calendário do *Android* de forma a facilitar o utilizador a inserir uma data válida. Esta solução vem facilitar o facto de não ser necessário fazer a validação do formato da data.

4.13 Criação de uma despesa

Ao registar uma despesa, o utilizador tem de indicar qual o veículo associado (figura 4.12). É necessário selecionar a categoria da despesa, descrição, custo e data. Nos campos

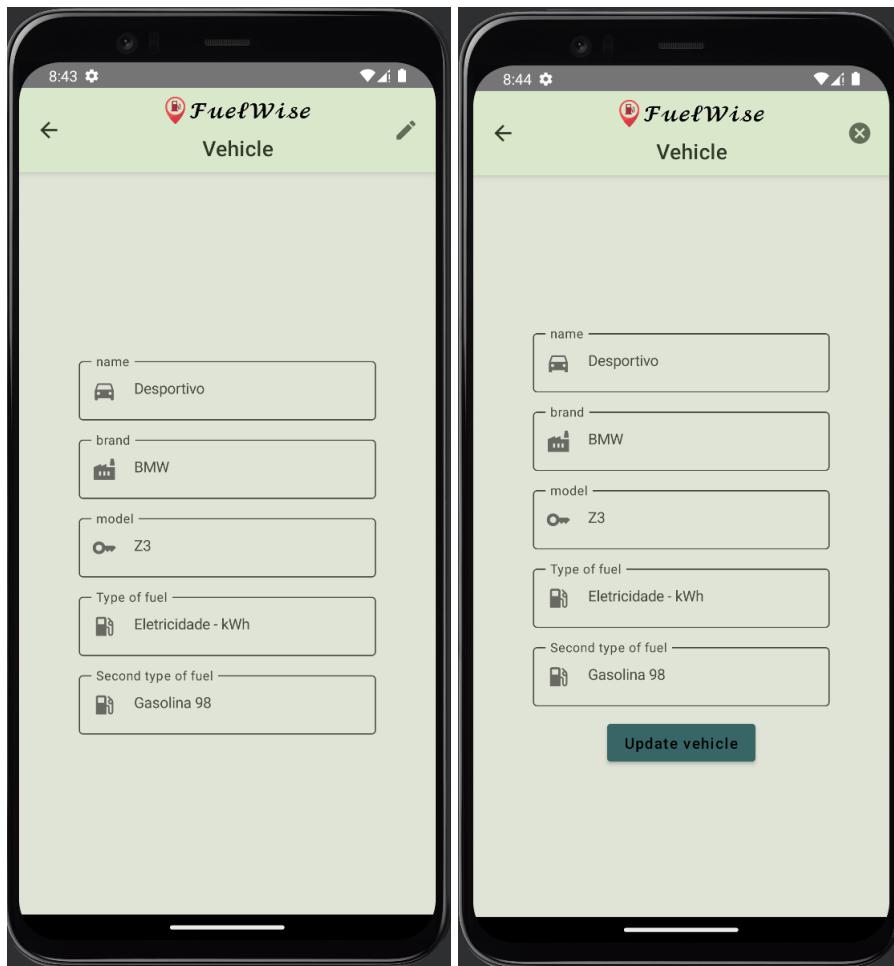


Figura 4.9: Consulta e edição de um veículo.

numéricos é apresentado o teclado numérico do *Android* para evitar erros de *input* do utilizador. A data também é apresentada no formato de calendário do *Android*, de forma a ser fácil para o utilizador inserção duma data válida.

4.14 Criação de um lembrete

Na criação de um lembrete é registado o veículo, modo de repetição (único ou anual), uma descrição, a data e hora (figura 4.13). Nesta situação, a data e a hora do lembrete não podem ser anteriores à data e hora de criação, respetivamente. Aquando do registo do lembrete na BD é também criado um alarme no *Android*, de modo a enviar uma notificação na data e hora definida.

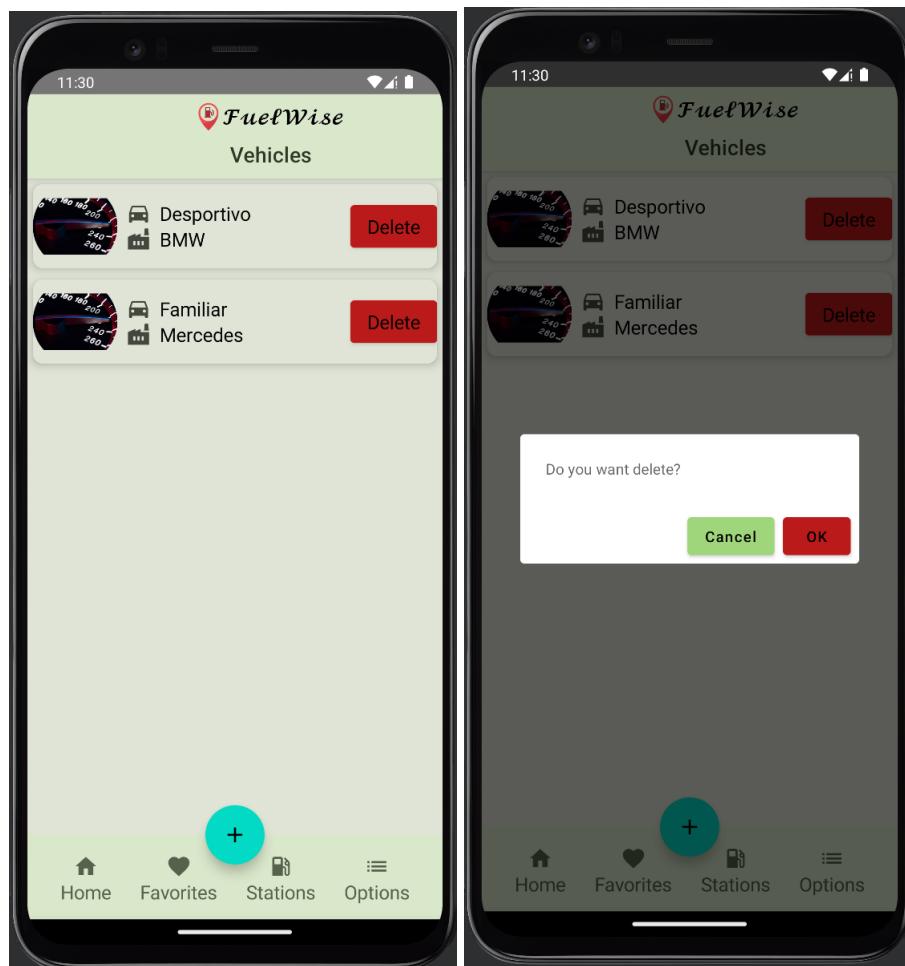


Figura 4.10: Lista de veículos.

4.15 Postos de abastecimento favoritos

Os postos de abastecimento podem ser assinalados como favoritos clicando no ícone favorito (formato em coração) conforme a figura 4.14. Para remover o posto como favorito existem duas alternativas. No próprio posto, pode-se voltar a clicar no ícone do favorito e remove a seleção. Outra alternativa é navegar para a lista de postos favoritos e clicar no botão apagar. A opção para aceder à lista dos postos de abastecimento favoritos está inserida na barra de navegação do fundo da aplicação.

Após assinalar um posto de abastecimento como favorito, caso ocorra uma alteração do preço de combustível referenciado, é enviada uma notificação para o dispositivo a informar dessa alteração (figura 4.15).

Esta funcionalidade não requer que a aplicação esteja em execução bastando apenas ativar o serviço nas definições. Deste modo, o utilizador pode manter-se informado sobre as alterações dos preços de combustível dos seus postos favoritos sem ter de consultar a

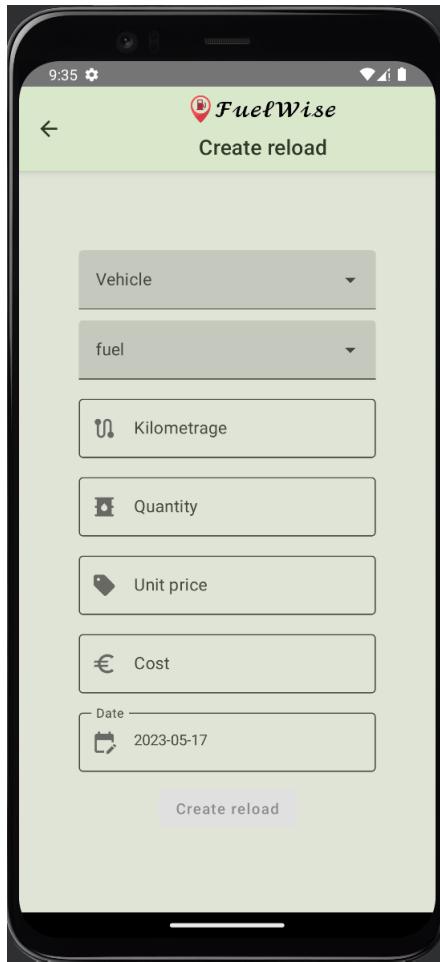


Figura 4.11: Abastecimento de um veículo.

aplicação com regularidade.

4.16 Mapa com os postos de abastecimento

A consulta de postos de abastecimento foi posicionada na barra de navegação do fundo, visto que é uma funcionalidade com bastante relevo na lógica de negócio da aplicação (figura 4.16). O mapa escolhido para aplicação foi o *Google SDK* para *Android* devido ao facto da plataforma *Android* ser desenvolvida pela *Google*, o que permite uma integração mais fácil na *User Interface (UI)* (Google (2023e)).

Com base no combustível que o utilizador assinala no ecrã inicial é consultada a BD no caso dos registo serem considerados atualizados ou é realizado um pedido à API da DGEG ou à fonte de dados local (no caso do combustível escolhido ser eletricidade). No mapa é mostrado todos os postos de Portugal para o respetivo combustível. Após uma

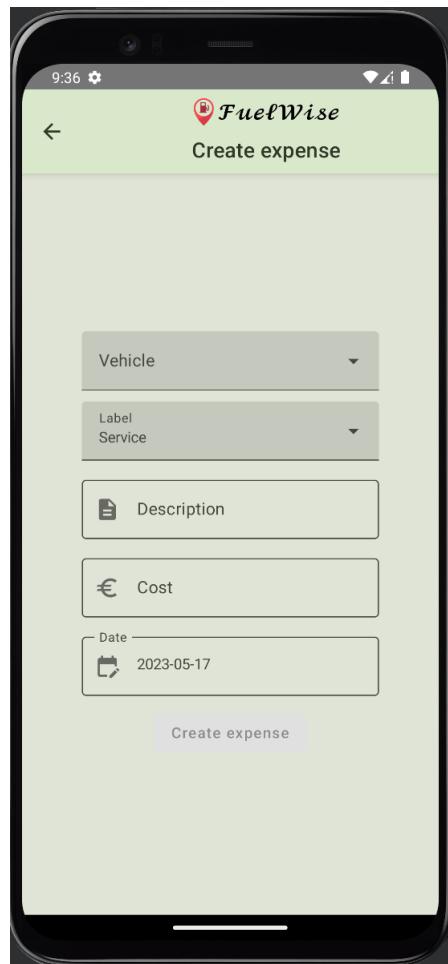


Figura 4.12: Registo de uma despesa.

primeira seleção do posto é apresentado o preço do combustível. Caso selecione o preço apresentado é realizado um novo pedido à API da DGEG para obter informações sobre esse posto, nomeadamente, a marca, o nome, a morada, os preços de combustível, assim como, os serviços e o horário disponível (figura 4.17). Existem dois ícones que têm como função permitir adicionar/remover um posto como favorito, assim como, iniciar a navegação para esse posto de abastecimento através de uma aplicação instalada no dispositivo que seja competente para o efeito.

4.17 Manual de utilizador da aplicação

Neste relatório não foram mostrados todos os ecrãs de aplicação, dado que foi decidido mostrar apenas os ecrãs que se entenderam ser mais relevantes. Deste modo foi elaborado um manual de utilizador que tem como finalidade explicar passo a passo como instalar a

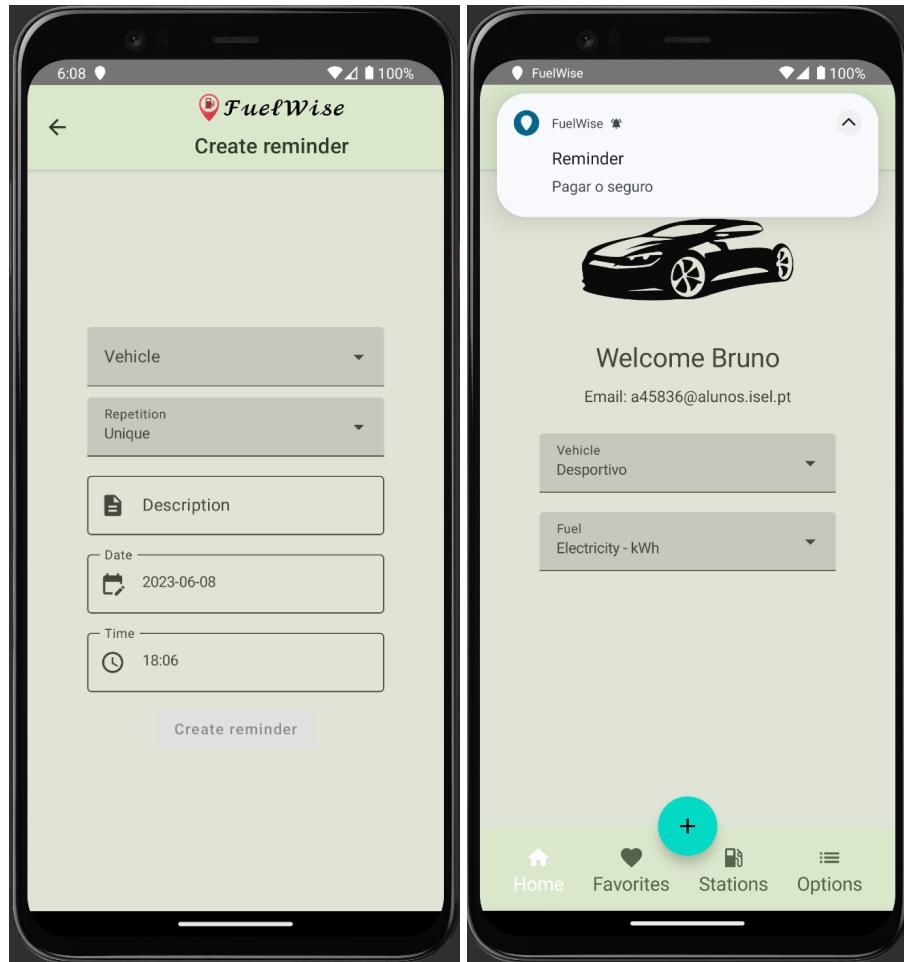


Figura 4.13: Registo e notificação de um lembrete.

aplicação no dispositivo e mostrar toda a naveabilidade da mesma.

4.18 Testes unitários

Na fase final do projeto foram realizados vários testes unitários às funções dos *ViewModels*. Os testes unitários têm como principal objetivo garantir que cada função desenvolvida funciona corretamente e produza os resultados esperados. As principais vantagens da utilização de testes unitários são:

- identificar os erros de lógica;
- facilitar a manutenção do código;
- documentar o comportamento esperado;

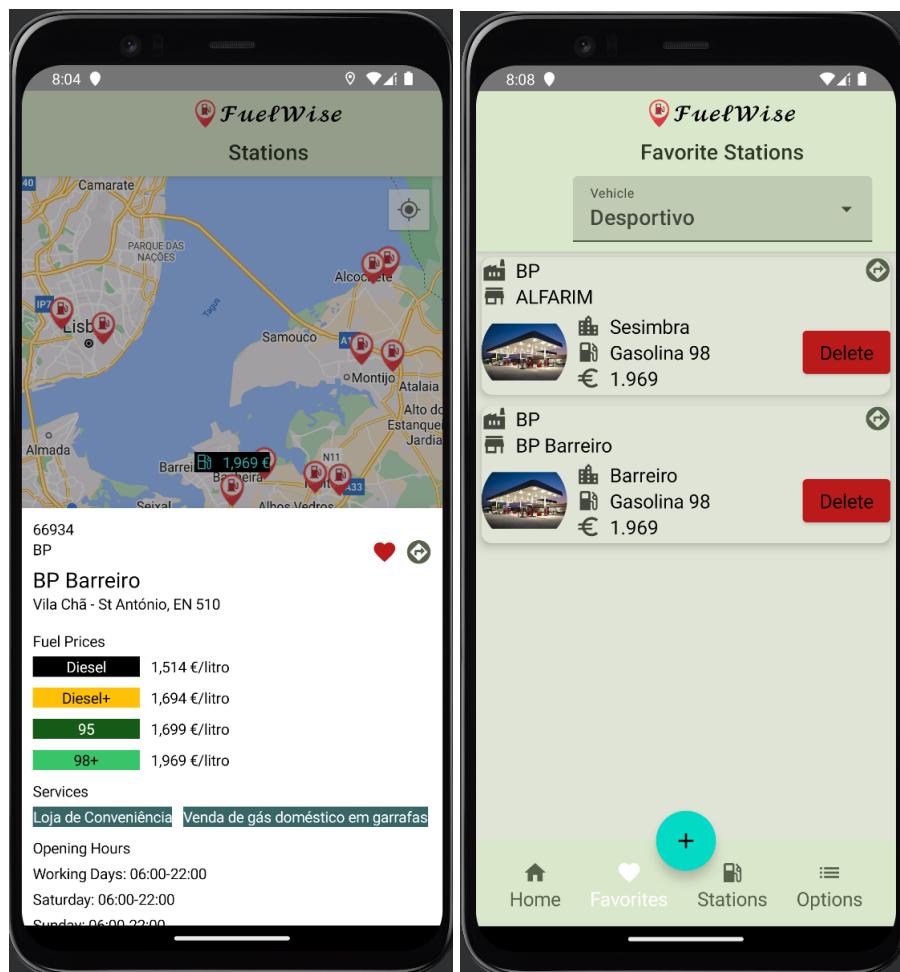


Figura 4.14: Assinalar um posto de abastecimento como favorito no mapa e listar os postos favoritos.

- melhoria da qualidade do código.

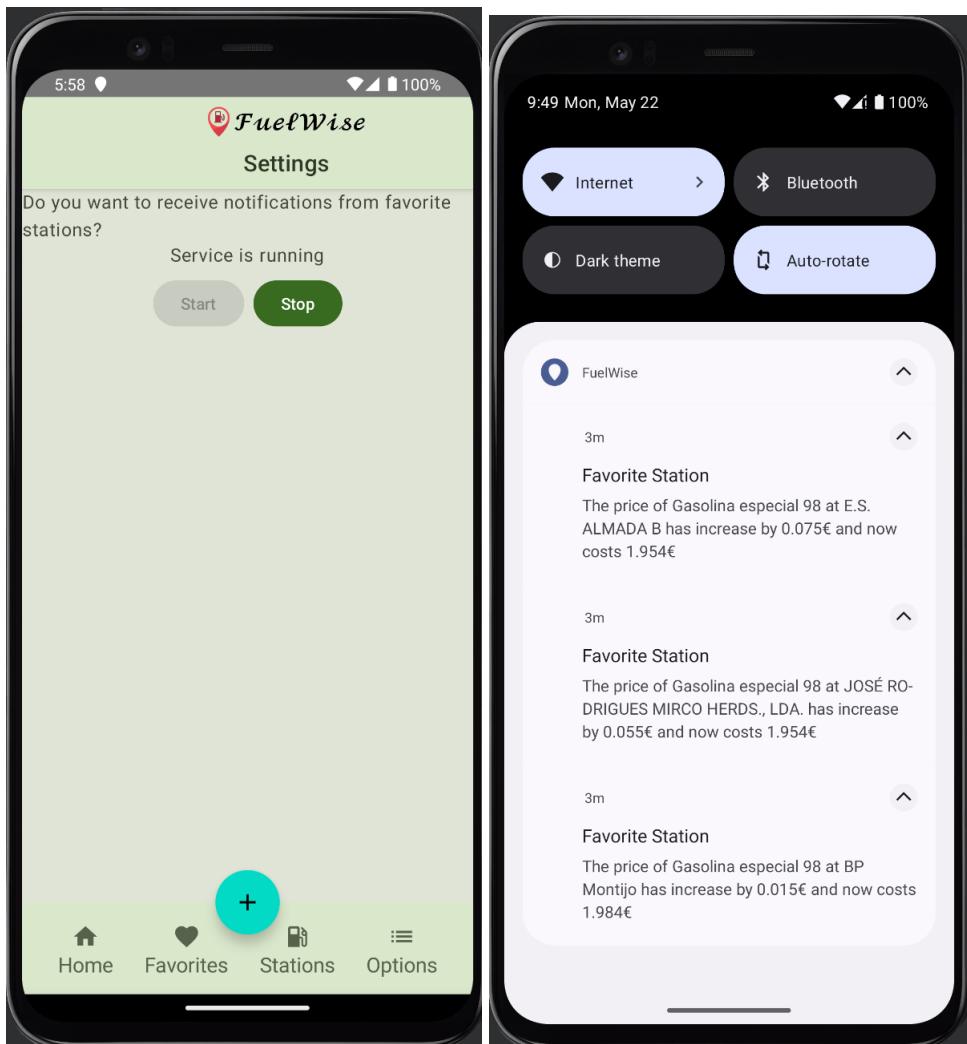


Figura 4.15: Ativação do serviço e apresentação das notificações sobre postos de abastecimento marcados como favoritos.

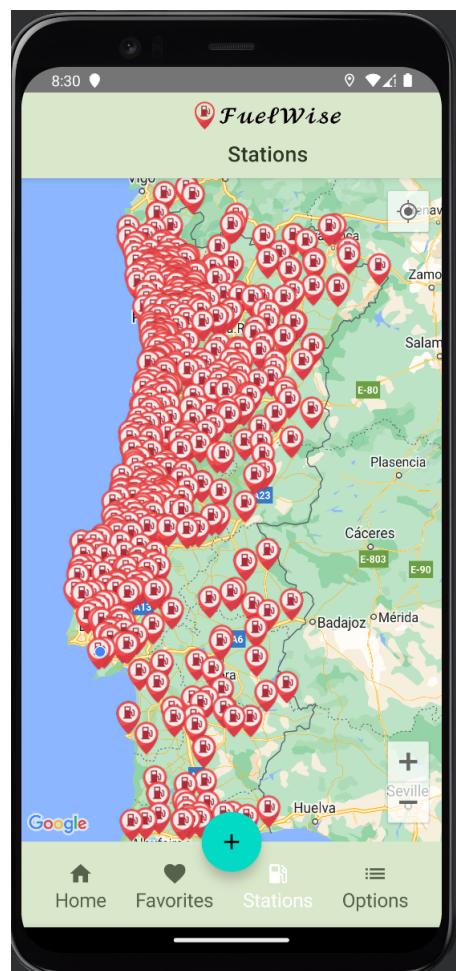


Figura 4.16: Consulta de postos de abastecimento para a gasolina especial 98.

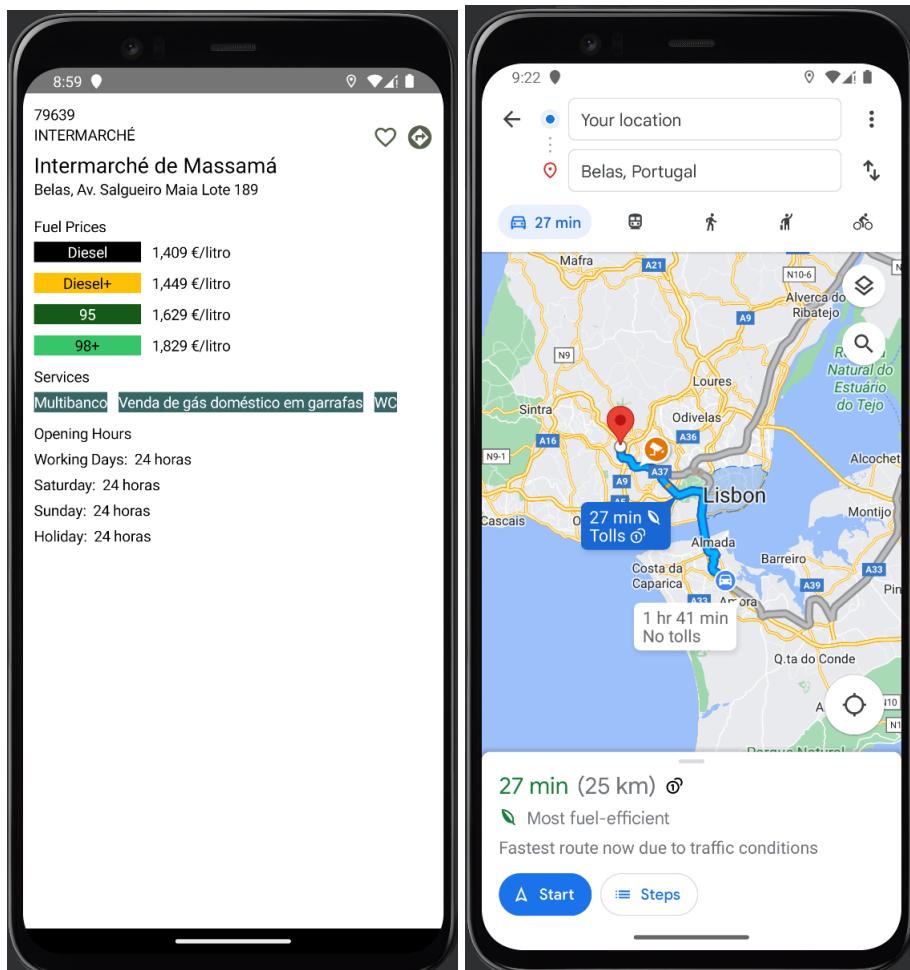


Figura 4.17: Serviços, preços de combustível e navegação para um dado posto de abastecimento.

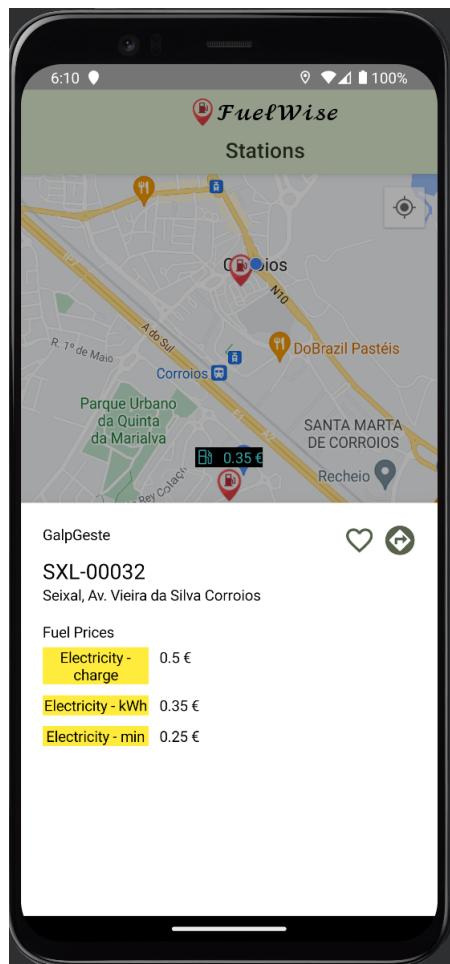


Figura 4.18: Informação sobre um posto de carregamento elétrico.

Capítulo **5**

Conclusão e Trabalho Futuro

Conteúdo

5.1	Conclusão	48
5.2	Trabalho futuro	49

5.1 Conclusão

No decorrer do desenvolvimento desta aplicação foi possível explorar novas bibliotecas e a utilização de novos elementos de *User Interface* (UI) em *Android*, com o intuito de completar os vários requisitos funcionais e integrá-los nos elementos gráficos da aplicação.

O desenvolvimento da aplicação teve em conta alguns aspectos relevantes na sua interacção com o utilizador, como por exemplo: o envio de mensagens informativas das ações do utilizador, a sua adaptação com a rotação do ecrã na vertical ou na horizontal, suportar dois idiomas (português e inglês) e, por fim, adaptar-se ao *dark mode* do dispositivo. Nesta fase de desenvolvimento da aplicação, recorreu-se ao mapa da *Google* para mostrar os postos de abastecimento. Contudo, existe uma restrição no seu acesso: é necessário uma chave de *Application Programming Interface* (API) associada a uma conta de utilizador da *Google*. Caso esta aplicação fosse lançada em produção a solução anteriormente mencionada não seria desejável. Assim, seria relevante integrar essa funcionalidade com um mapa gratuito e que não requeresse nenhuma chave de acesso associada a nenhuma conta de utilizador.

Durante o desenvolvimento desta aplicação não foi possível implementar o modo de *clustering* no mapa. Esta funcionalidade permite ao utilizador visualizar os postos que vão sendo agregados em *clusters* à medida que se faz *zoom out*, e apresenta também um contador por *cluster* indicando quantos postos estão agregados. Isto tem como vantagem uma melhor visualização dos postos de abastecimento no mapa, dado que com o *zoom out* os postos não irão aparecer sobrepostos.

Na fase final do desenvolvimento da aplicação demonstrou-se o cumprimento de todos os requisitos que foram definidos na proposta do projeto. A aplicação *FuelWise* posiciona-se como uma aplicação concorrente às demais que existem no mercado, no segmento *Android*, com o destaque na introdução das notificações sobre as alterações de preços dos combustíveis de um ou vários postos de abastecimentos com interesse para utilizador.

5.2 Trabalho futuro

As restrições temporais associadas a um projeto de aproximadamente 6 meses, implicaram um planeamento rigoroso de priorizar os requisitos funcionais em detrimento de outras funcionalidades, que foram classificadas como secundárias. Deste modo, a aplicação tem uma margem de desenvolvimento no suporte a novas funcionalidades que a tornem mais completa e distinta das aplicações concorrentes. Para implementação futura, pretende-se implementar as seguintes funcionalidades:

- mudar o mapa do *Google Maps* para uma alternativa gratuita (sem recurso chaves de API);
- encontrar uma API gratuita que dê acesso aos preços de energia atualizados relacionados com os postos de carregamento;
- implementar o *clustering* dos postos de abastecimento no mapa;
- ter a opção de importação e exportação da *Base de Dados* (BD);
- permitir digitalizar os cartões de fidelização das diversas empresas petrolíferas;
- mostrar gráficos estatísticos de despesas, abastecimentos, consumos, entre outros;
- permitir mudar a imagem do ecrã inicial;
- ao invés de escrever por extenso a marca e o modelo do veículo ter a possibilidade sugerir ao utilizador as marcas e modelos existentes na actualidade;
- realizar testes que avaliam os componentes de *Android* dos vários ecrãs.

Apêndice **A**

Modelo Entidade-Associação

Modelo Entidade-Associação:

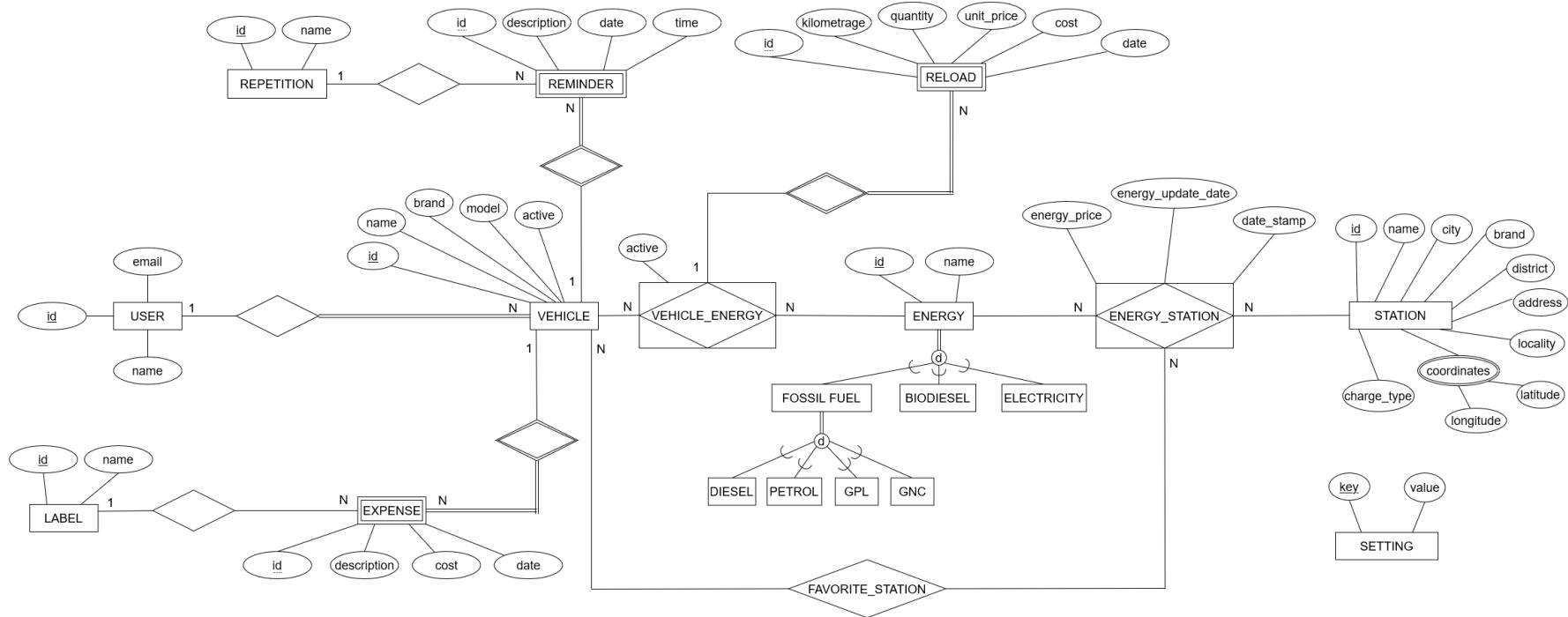


Figura A.1: Modelo Entidade-Associação

Apêndice **B**

Modelo Relacional

Modelo Relacional:

USER (id, name, email)

VEHICLE (id, name, brand, model, active, uid)

FK: {uid} de USER.id

LABEL (id, name)

EXPENSE (id, vid, description, cost, date, lid)

FK: {vid} de VEHICLE.id, {lid} de LABEL.id

RELOAD (id, vid, eid, kilometrage, quantity, unit_price, cost, date)

FK: {vid} de VEHICLE_ENERGY.vid, {eid} de VEHICLE_ENERGY.eid

REPETITION (id, name)

REMINDER (id, vid, description, date, time, rid)

FK: {vid} de VEHICLE.id, {rid} de REPETITION.id

ENERGY (id, name)

STATION (id, name, city, brand, district, address, locality, latitude, longitude, charge_type)

VEHICLE_ENERGY (vid, eid, active)

FK: {vid} de VEHICLE.id, {eid} de ENERGY.id

ENERGY_STATION (eid, sid, energy_price, energy_update_date, date_stamp)

FK: {eid} de ENERGY.id, {sid} de STATION.id

FAVORITE_STATION (vid, eid, sid)

FK: {vid} de VEHICLE.id, {eid} de ENERGY.id, {sid} de STATION.id

SETTING (key, value)

Figura B.1: Modelo Relacional

Referências

- Abreu, J. (2023). Conheça 5 apps para controlar gastos com o carro. URL: <https://www.e-konomista.pt/apps-gastos-carro/>, visitado dia 13 de maio de 2023.
- Academy, W. S. (2023). Why single activity architecture is preferred over multiple activity architecture in android app development? URL: <https://www.webskittersacademy.in/why-single-activity-architecture-android-app-development/>, visitado dia 13 de maio de 2023.
- ACP (2023). Quanto custa carregar um carro elétrico? URL: <https://www.acp.pt/electricos/carregar-carro-eletrico/quanto-custa-carregar-um-carro-eletrico>, visitado dia 13 de maio de 2023.
- Coder, R. (2023). Reso coder. URL: <https://resocoder.com/>, visitado dia 13 de maio de 2023.
- Direção-Geral de Energia e Geologia (2023). Preço dos combustíveis online - informação ao consumidor. URL: <https://precoscombustiveis.dgeg.gov.pt/>, visitado dia 13 de maio de 2023.
- DRE (2023). Decreto-lei n.º 243/2008, de 18 de dezembro. URL: <https://dre.pt/dre/detalhe/decreto-lei/243-2008-443909>, visitado dia 13 de maio de 2023.
- Drivvo (2023). How do you manage your vehicle or your fleet? URL: <https://www.drivvo.com/en>, visitado dia 13 de maio de 2023.
- Fuelio (2023). Vehicle management fuel log, costs and mileage tracking app. URL: <https://www.fuel.io/>, visitado dia 13 de maio de 2023.
- FuelWise (2023). Informação sobre postos de abastecimento e gestão de veículos. URL: <https://github.com/bfccosta/fuelwise-public>, visitado dia 13 de maio de 2023.
- Gonçalves, R. (2023). O que influencia o preço dos combustíveis? URL: <https://www.doutorfinancas.pt/energia/o-que-influencia-o-preco>

- dos-combustiveis/ , visitado dia 13 de maio de 2023.
- Google (2023a). Build better apps faster with jetpack compose. URL: <https://developer.android.com/jetpack/compose> , visitado dia 13 de maio de 2023.
- Google (2023b). Develop android apps with kotlin. URL: <https://developer.android.com/kotlin> , visitado dia 13 de maio de 2023.
- Google (2023c). Foreground services. URL: <https://developer.android.com/guide/components/foreground-services> , visitado dia 13 de maio de 2023.
- Google (2023d). Introduction to activities. URL: <https://developer.android.com/guide/components/activities/intro-activities> , visitado dia 13 de maio de 2023.
- Google (2023e). Maps sdk for android. URL: <https://developers.google.com/maps/documentation/android-sdk/map> , visitado dia 13 de maio de 2023.
- Google (2023f). Material components and layouts. URL: <https://developer.android.com/jetpack/compose/layouts/material> , visitado dia 13 de maio de 2023.
- Google (2023g). Navigating with compose. URL: <https://developer.android.com/jetpack/compose/navigation> , visitado dia 13 de maio de 2023.
- Google (2023h). Permissions on android. URL: <https://developer.android.com/guide/topics/permissions/overview#best-practices> , visitado dia 13 de maio de 2023.
- Google (2023i). Request runtime permissions. URL: <https://developer.android.com/training/permissions/requesting> , visitado dia 13 de maio de 2023.
- Google (2023j). Save data in a local database using room. URL: <https://developer.android.com/training/data-storage/room> , visitado dia 13 de maio de 2023.
- Google (2023k). Schedule tasks with workmanager. URL: <https://developer.android.com/topic/libraries/architecture/workmanager> , visitado dia 13 de maio de 2023.
- Google (2023l). Viewmodel overview. URL: <https://developer.android.com/topic/libraries/architecture/viewmodel> , visitado dia 13 de maio de 2023.
- Macrotrends (2023). Crude oil prices - 70 year historical chart. URL: <https://www.macrotrends.net/1369/crude-oil-price-history-chart> , visitado dia 13 de maio de 2023.
- MaisGasolina (2023). Maisgasolina. URL: <https://www.maisgasolina.com/> , visitado dia 13 de maio de 2023.
- Mobi.E (2023). Mobi.e - mobilidade elétrica. URL: <https://www.mobie.pt/> , visitado dia 13 de maio de 2023.

- Navigation (2023). A small and simple, yet fully fledged and customizable navigation library for jetpack compose. URL: <https://androidexample365.com/a-small-and-simple-yet-fully-fledged-and-customizable-navigation-library-for-jetpack-compose/>, visitado dia 13 de maio de 2023.
- Roger A. Hinrichs, M. K. (2003). *Energia e Meio Ambiente*.
- Room (2023). Salvar dados em um banco de dados local usando o room. URL: <https://developer.android.com/training/data-storage/room> , visitado dia 13 de maio de 2023.
- Square (2023). Retrofit: A type-safe http client for android and java. URL: <https://square.github.io/retrofit/> , visitado dia 13 de maio de 2023.
- UVE (2023). Uve- associação de utilizadores de veículos elétricos. URL: <https://www.uve.pt/page/> , visitado dia 13 de maio de 2023.
- VivaGas (2023). Encontre o combustível mais barato, mais perto. mais inteligente. URL: <http://www.appvivagas.com/> , visitado dia 13 de maio de 2023.
- Xamarin (2023). Free. cross-platform. open source.an app platform for building android and ios apps with .net and c sharp. URL: <https://dotnet.microsoft.com/en-us/apps/xamarin> , visitado dia 13 de maio de 2023.

