Universidade do Minho

Departamento de Informática

BraGuia

(2.º Semestre/2023-2024)

Tópicos de Desenvolvimento de Software

pg47153 Diogo Paulo Lopes de Vasconcelos
pg53944 João Pedro Moreira Brito
pg52690 José Pedro Gomes Ferreira

Contents

1 Introdução **3**

2 Detalhes de implementação **4**
2.1 Estrutura do projeto 4
2.1.1 Gradle Scripts 4
2.1.2 Manifest 4
2.1.3 Resources 4
2.1.4 java.com.example.braguia 5
2.1.5 Arquitetura do projeto 6
2.2 Soluções de implementação 7
2.2.1 Integridade dos dados 7
2.2.2 Modelo da base de dados 7
2.2.3 Pedidos à API 8
2.2.4 Integração de um mapa do Google Maps SDK e Geofences 9
2.2.5 Navegação por um roteiro 9
2.2.6 Menu hambúrguer 9
2.3 Bibliotecas/dependências utilizadas 10
2.3.1 AndroidX Appcompat + Core + Lifecycle 10
2.3.2 Google Maps SDK + Location 10
2.3.3 JUnit + Mockito 10
2.3.4 Material Design 10
2.3.5 Media3 10
2.3.6 OkHTTP + Retrofit 10
2.3.7 Picasso 10
2.3.8 Room 10
2.4 Padrões de software utilizados 10
2.4.1 Padrões creacionais 10
2.4.2 Padrões estruturais 11
2.4.3 Padrões comportamentais 11

3 Mapa de navegação **12**

4 Funcionalidades **13**
4.1 Funcionalidades sem autenticação obrigatória 13
4.2 Funcionalidades para utilizadores normais 13
4.3 Funcionalidades exclusivas a utilizadores premium 14

5 Discussão de resultados **15**
5.1 Trabalho realizado 15
5.2 Limitações 15
5.3 Funcionalidades extra 15

6 Gestão de projeto **17**
6.1 Gestão e distribuição de trabalho 17
6.2 Eventuais metodologias de controlo de versão utilizadas 17
6.3 Reflexão sobre performance individual 17

7 Conclusão **19**

8 Anexos **21**

Chapter 1

Introdução

A cidade e distrito de Braga têm crescido em popularidade no panorama nacional e existem vários pontos de interesse ao longo deste território que potencialmente atraem pessoas numa escala potencialmente global. Dado isto, surge a proposta de realizar uma aplicação cujo nome é "BraGuia" e que serve como guia turístico para as pessoas que pretendem conhecer mais sobre a região de Braga.

Esta aplicação terá elementos de navegação geográfica, localização, reprodução de conteúdo *media* e informações sobre roteiros e pontos de interesse relativos. A fonte de informação é proveniente de uma API desenvolvida para o projeto.

A aplicação será desenvolvida com recurso a tecnologias nativas *Android*, utilizando o Android Studio como ambiente de desenvolvimento. Serão referidos com mais detalhe pormenores de implementação, funcionalidades, estrutura do projeto e a sua gestão.

Chapter 2

Detalhes de implementação

2.1 Estrutura do projeto

A estrutura geral do projeto seguida pelo grupo foi a recomendada pelo professor Rui Rua, através do padrão de arquitetura *MVVM (Model-View-ViewModel)*. Através da biblioteca de persistência Room, esta estabelece a comunicação entre os modelos e a base de dados com *SQLite* e *DAOs (Data Access Objects)*. Os dados para povoar a base de dados são extraídos através de uma API externa fornecida pelo professor.

No que toca à estrutura das diretorias, esta foi seguindo inicialmente o modelo base do *Android Studio*, sendo feita posteriormente uma separação entre modelos, repositórios, *ViewModels*, elementos da interface gráfica, e outras classes que não se encaixam em nenhum dos contextos referidos previamente.

2.1.1 Gradle Scripts

Esta secção contém as bibliotecas e dependências necessárias para realizar este projeto, com as suas versões associadas.

2.1.2 Manifest

Esta diretoria contém o *AndroidManifest.xml*, ficheiro que contém componentes *Android* como: atividades, serviços, recetores, e permissões possíveis de ser adicionadas à aplicação, bem como a chave API do Google Maps.

2.1.3 Resources

Nesta diretoria, encontram-se todos os recursos utilizados na aplicação:

drawables

Contém todos os ícones e elementos passíveis de serem desenhados na interface do utilizador.

layout

Contém todos os *layouts* das atividades e elementos (ex.: *RecyclerViews* desenhadas no ecrã.

menu

Contém o *layout* do menu hamburguer onde é possível aceder a várias funcionalidades da aplicação.

navigation

Contém o gráfico de navegação de alguns elementos da aplicação.

values

Contém constantes de, por exemplo, cores e *strings* associadas ao projeto.

2.1.4 java.com.example.braguia

Esta diretoria contém os ficheiros *Java* associados a cada um dos elementos do *layout*, ou seja, atividades, bem como outros ficheiros associados a persistência de dados e outras funções.

data

Contém todos os ficheiros associados aos modelos de dados e sua representação, os *DAOs* que permitem acesso à base de dados, e conversores de tipos utilizado para fazer a ligação entre classes através de chaves estrangeiras, bem como a base de dados em si.

repository

Contém todos os repositórios associados a 5 tipos de dados: Informação da aplicação; conteúdo *media*, pontos de interesse, roteiros, e informação do utilizador.

ui

Contém todas as atividades da aplicação. Dentro desta pasta, existe, ainda, todos os adaptadores e *ViewHolders* necessários para a criação de *RecyclerViews* da aplicação.

utils

Contém todos os ficheiros que percorrem tarefas variadas, como, por exemplo, monitorizar a localização do utilizador em primeiro plano, ou escutar se o utilizador está perto ou não de alguma *geofence* associada a um ponto de interesse.

viewmodel

Contém todos os *ViewModels* associados a cada repositório necessário para demonstrar a informação a ser visualizada.

test

Contém alguns testes unitários acerca das classes associadas aos modelos da aplicação.

2.1.5 Arquitetura do projeto

O projeto segue uma arquitetura semelhante a este formato:

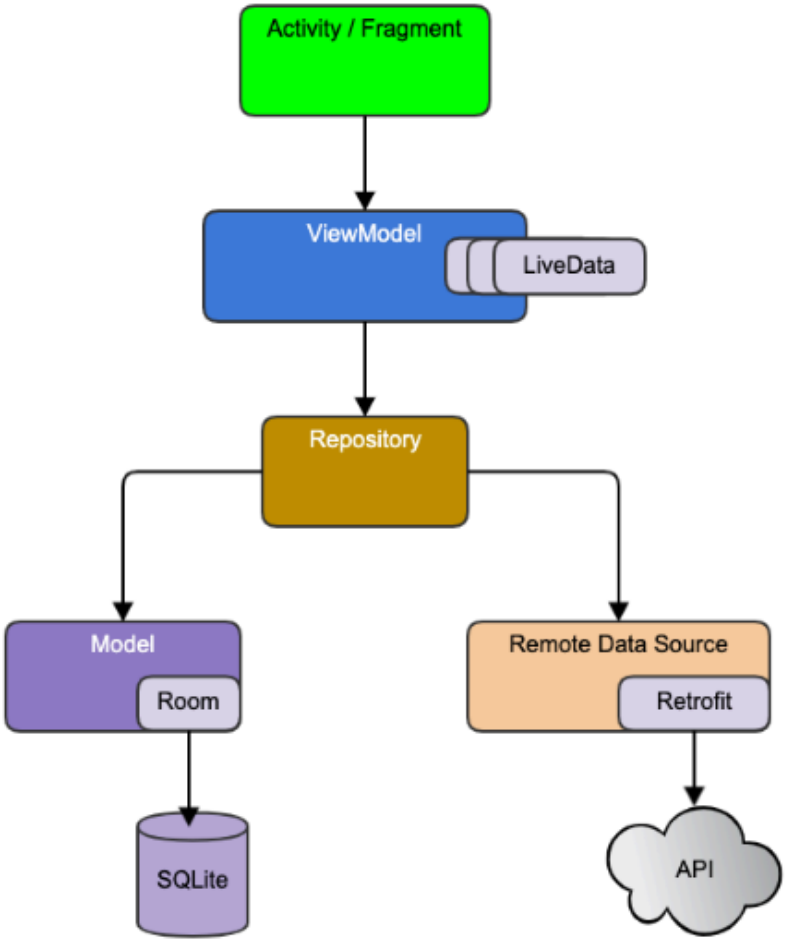

Figure 2.1: Arquitetura geral da aplicação

2.2 Soluções de implementação

Nesta secção iremos abordar algumas soluções de implementação que o grupo decidiu por optar no desenvolvimento deste projeto, com o objetivo de satisfazer as funcionalidades propostas.

2.2.1 Integridade dos dados

Para preservar a integridade dos dados do modelo, o grupo optou pelo uso da RoomDatabase, através dos repositórios e das suas respectivas DAOs, verifica-se se há dados armazenados na base de dados local. Caso não haja, a classe *utils/ApiService* acede à API para recuperar os dados e repopular a base de dado. Este processo assegura que os dados estão sempre atualizados e disponíveis para uso, mantendo a consistência entre a base de dados remota e local. Este processo está demonstrado no seguinte exemplo:

```
1     public AppRepository(Application application){
2         RoomDb database = RoomDb.getInstance(application);
3         appDao = database.appDao();
4         allApp = new MediatorLiveData<>();
5         allApp.addSource(
6                 appDao.getApp(), localApp -> {
7                     // TODO: ADD cache validation logic
8                     if (localApp != null && !localApp.isEmpty()) {
9                         allApp.setValue(localApp);
10                    } else {
11                        ApiService.getInstance(application).
                              fetchAppInfo(application);
12                    }
13                }
14        );
15    }
```

Para além disto, o grupo utiliza a biblioteca sharedPreferences para armazenar informações mais pequenas e que vão sendo acedidas com menos frequência, como o estado da sessão de um possível utilizador, cookies e o tipo do utilizador.

2.2.2 Modelo da base de dados

Com a estrutura mencionada acima, resultou a seguinte estrutura da base de dados. Devido à extensão do diagrama de classes, a construção da base de dados foi realizada seguindo o propósito de cada atividade.

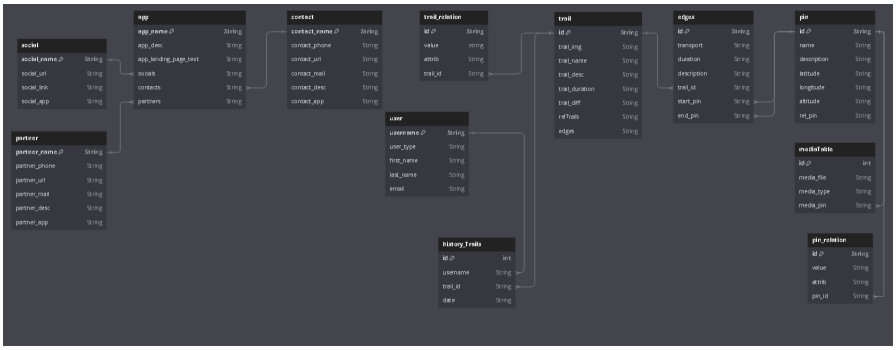

Figure 2.2: Modelo concetual da base de dados

2.2.3 Pedidos à API

Sempre que se deseja realizar um pedido à API, este é sempre feito utilizando o Retrofit de forma a realizar pedidos HTTP de forma assíncrona à thread principal com a utilização do comando enqueue, que permite ao utilizador continuar a interagir com a aplicação enquanto o pedido executa. Além disso, utilizamos um cliente da biblioteca OkHTTP para garantir consistência e estabilidade na realização dos pedidos HTTP. Após a conclusão do pedido, podemos reagir de acordo com a resposta por meio de callbacks. De forma a tornar mais intuitivo o código e de certa forma melhorar a estrutura do projeto, o grupo optou por realizar todos os pedidos à API numa única classe, denominada ApiService. Em seguida, encontra-se um exemplo de um destes pedidos à API que se encontra na ApiService:

```
1     private static OkHttpClient client () {
2         return new OkHttpClient.Builder()
3                 .connectTimeout(10000, TimeUnit.SECONDS)
4                 .readTimeout(10000, TimeUnit.SECONDS)
5                 .addInterceptor(new HttpLoggingInterceptor().setLevel
                      (HttpLoggingInterceptor.Level.BODY))
6                 .build();
7     }
8
9     private static Retrofit retrofit () {
10        return new Retrofit.Builder()
11                .baseUrl(BASE_URL)
12                .client(client())
13                .addConverterFactory(GsonConverterFactory.create())
14                .build();
15    }
16
17        public void fetchAppInfo (Application app) {
18        appRepository = AppRepository.getInstance(app);
19
20        Api api = retrofit().create(Api.class);
21        Call<List<App>> call = api.getApp();
22
23        call.enqueue(new retrofit2.Callback<List<App>>() {
24            @Override
25            public void onResponse(Call<List<App>> call, Response<
                  List<App>> response) {
26                if(response.isSuccessful()) {
27                    Gson gson = new Gson();
28                    String responseGson = gson.toJson(response.body()
                          );
29                    Log.d("Siu", responseGson);
30                    appRepository.insert(response.body());
31                }
32                else{
33                    Log.e("main", "onFailure: " + response.errorBody
                          ());
34                }
35            }
36
37            @Override
38            public void onFailure(Call<List<App>> call, Throwable t)
                  {
39                Log.e("main", "onFailure: " + t.getMessage());
40            }
```

```
41          });
42      }
```

2.2.4 Integração de um mapa do Google Maps SDK e Geofences

Na aplicação desenvolvida pelo grupo, decidiu-se disponibilizar um mapa interativo, utilizando a API do Google Maps para fornecer este mapa, contudo, quando é inicializado um roteiro, o grupo optou por outra abordagem que será discutida à frente. Além disso, um serviço de localização é ativado para monitorizar a localização do utilizador em primeiro plano, quando este mapa é iniciado. Por fim, o utilizador, mesmo não tendo o serviço em primeiro plano, pode receber notificações quando passa perto de pontos de interesse, através de objetos *Geofence* criados para o efeito do requisito funcional, A notificação, quando emitida, redireciona o utilizador para a atividade relativa ao ponto de interesse associado à *Geofence*.

2.2.5 Navegação por um roteiro

Quando um utilizador inicia a navegação de um roteiro, ele é redirecionado para aplicação do Google Maps, exibindo os pontos do trajeto. O utilizador tem a capacidade de interromper este roteiro a qualquer momento, e, quando dentro do roteiro, tem auxílio visual e de voz caso esteja a conduzir.

2.2.6 Menu hambúrguer

De forma a manter a *MainActivity* intacta durante a navegação pela aplicação, o grupo decidiu usar um menu hambúrguer. Esta abordagem é recomendada pela Google e facilita bastante a leitura do código. De seguida está o exemplo da atribuição de atividades a botões do menu hambúrguer:

```
1       public boolean onNavigationItemSelected(@NonNull MenuItem
            menuItem) {
2           if (menuItem.getItemId() == R.id.history) {
3               if (sessionManager.getSharedPreferences().getString("
                    isLoggedIn", "false").equals("true")) {
4                   Intent intent = new Intent(MainActivity.this,
                        HistoryActivity.class);
5                   startActivity(intent);
6               }
7               else {
8                   Toast.makeText(this, "You are not logged in!", Toast.
                        LENGTH_SHORT).show();
9               }
10          }
11          else if (menuItem.getItemId() == R.id.location_settings) {
12              Intent intent = new Intent(
                    ACTION_LOCATION_SOURCE_SETTINGS);
13              startActivity(intent);
14          }
15          else if (menuItem.getItemId() == R.id.
                location_service_settings) {
16              Intent intent = new Intent(MainActivity.this,
                    LocationServiceActivity.class);
17              startActivity(intent);
18          }
19
20           ...
```

```
21 }
```

2.3 Bibliotecas/dependências utilizadas

2.3.1 AndroidX Appcompat + Core + Lifecycle

Oferece implementações compatíveis com voltar atrás de funcionalidade relacionadas com a interface do utilizador e gestão dos componentes de ciclo de vida da aplicação.

2.3.2 Google Maps SDK + Location

Biblioteca utilizada para a integração do Google Maps na aplicação, inclusive com um mapa embutido, providenciar um serviço que monitoriza a localização do utilizador em primeiro plano, bem como verificar se este se encontra perto de potenciais pontos de interesse.

2.3.3 JUnit + Mockito

Bibliotecas utilizadas para a realização de testes unitários.

2.3.4 Material Design

Biblioteca providenciada pela Google para oferecer ao utilizador uma interface mais esbelta.

2.3.5 Media3

Biblioteca para reproduzir e colocar no ecrã conteúdo *media* como áudio e vídeo, por exemplo.

2.3.6 OkHTTP + Retrofit

Permite a criação e simplificação de pedidos HTTP para APIs.

2.3.7 Picasso

Biblioteca de download e caching de imagens em Android.

2.3.8 Room

Biblioteca usada para garantir a persistência dos dados.

2.4 Padrões de software utilizados

2.4.1 Padrões creacionais

Singleton

Todos os repositórios da aplicação, a base de dados e classes como a *ApiService* ou o *GeofenceManager* apresentam o padrão de *software singleton*, em que as classes são criadas se não existirem ou oferecem uma versão já existente dessa mesma classe caso contrário.

Builder

Utilizado na criação de notificações, *geofences* e do cliente OkHTTP a partir do qual se realizam pedidos HTTP.

2.4.2 Padrões estruturais

Adapter

Utilizado para criar todos os adaptadores necessários para injeção de *RecyclerViews*.

Facade

Utilizada para os métodos da *RoomDB*.

Composite

Utilizado de forma indireta, dado que as *Views* do Android seguem esta hierarquia.

2.4.3 Padrões comportamentais

Observer

Este padrão é utilizado para obter os dados de um *ViewModel* necessários para serem mostrados na atividade ou para criação de objetos (ex.: *geofences*) necessários para realizar funcionalidades da aplicação.

Chapter 3

Mapa de navegação

Nesta secção é apresentado o mapa de navegação da aplicação. O mapa demonstra as opções de navegação por parte dos utilizadores, as respetivas atividades e rotas da aplicação.

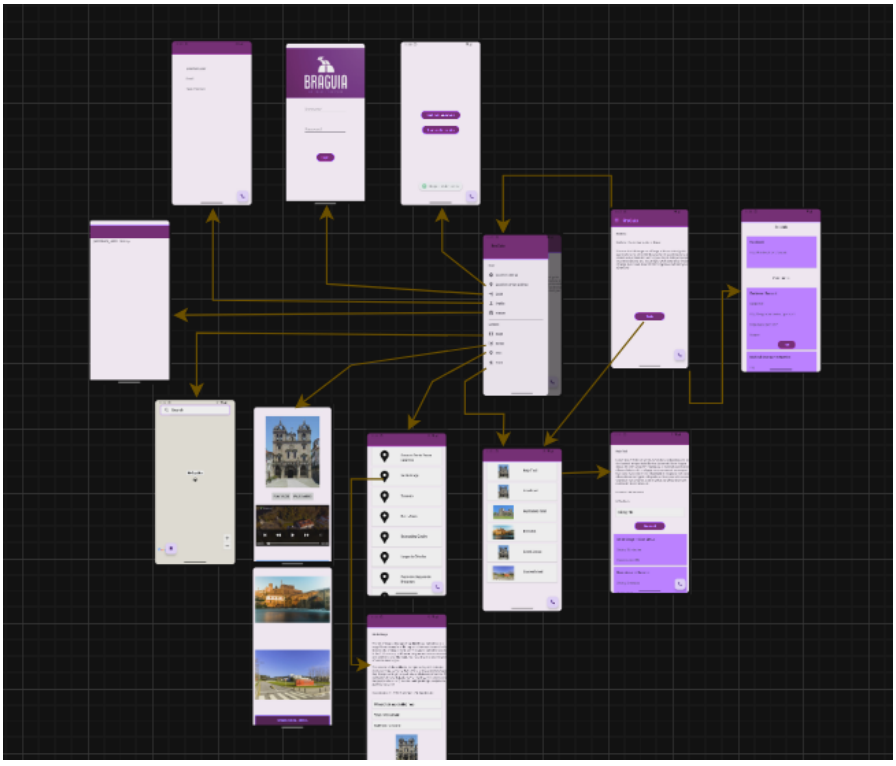

Figure 3.1: Mapa de Navegação

Chapter 4

Funcionalidades

Nesta secção são descritas as diversas funcionalidades presentes na aplicação. As mesmas poderão ser obrigatórias, mencionadas no enunciado atribuído, ou funcionalidades extra, partindo da iniciativa do grupo.

4.1 Funcionalidades sem autenticação obrigatória

- A aplicação deve possuir uma página inicial onde apresenta as principais funcionalidades do guia turístico, descrição, etc.
- A aplicação deve mostrar num ecrã, de forma responsiva, uma lista de roteiros disponíveis;
- A aplicação deve permitir efetuar autenticação;
- A aplicação deve assumir que o utilizador tem o Google Maps instalado no seu dispositivo (e notificar o utilizador que este software é necessário);
- A aplicação deve mostrar, numa única página, informação acerca de um determinado roteiro: galeria de imagens, descrição, mapa do itinerário com pontos de interesse e informações sobre a mídia disponível para os seus pontos;
- A aplicação deve possuir a capacidade de ligar, desligar e configurar os serviços de localização;
- A aplicação deve possuir a capacidade de efetuar chamadas para contactos de emergência da aplicação através de um elemento gráfico facilmente acessível na aplicação;

4.2 Funcionalidades para utilizadores normais

- A aplicação deve suportar 2 tipos de utilizadores: utilizadores standard e utilizadores premium;
- A aplicação deve possuir uma página de informações acerca do utilizador atualmente autenticado;
- A aplicação deve possuir a capacidade de emitir uma notificação quando o utilizador passa perto de um ponto de interesse;
- A notificação emitida quando o utilizador passa pelo ponto de interesse deve conter um atalho para o ecrã principal do ponto de interesse;

- A aplicação deve guardar (localmente) o histórico de roteiros e pontos de interesse visitados pelo utilizador;
- A aplicação deve ter a capacidade de apresentar e produzir 3 tipos de mídia: voz, imagem e vídeo;
- A aplicação deve possuir uma página que mostre toda a informação disponível relativa a um ponto de interesse: localização, galeria, mídia, descrição, propriedades, etc;

4.3 Funcionalidades exclusivas a utilizadores premium

- Para utilizadores premium (e apenas para estes) a aplicação deve possibilitar a capacidade de navegação, de consulta e descarregamento de mídia;
- A aplicação deve possuir a capacidade de iniciar um roteiro;
- A aplicação deve possuir a capacidade de interromper um roteiro;
- A navegação proporcionada pelo Google Maps deve poder ser feita de forma visual e com auxílio de voz, de modo a que possa ser utilizada por condutores;
- A aplicação deve possuir a capacidade de descarregar mídia do backend e aloja-la localmente, de modo a poder ser usada em contextos de conectividade reduzida;

Chapter 5

Discussão de resultados

5.1 Trabalho realizado

Em relação ao trabalho realizado, o grupo cumpriu com todas as funcionalidades propostas no projeto exceto uma, seguiu a arquitetura Android de forma rigorosa com o objetivo de generalizar o projeto e ainda utilizou o GitHub Actions para obter automaticamente a release do APK, como era descrito no enunciado.

Desta forma, o grupo considera que o trabalho realizado conseguiu cumprir com o que era pedido e encontra-se satisfeito com o trabalho apresentado, contudo considera que ainda existem elementos no projeto passíveis de melhorias em futuras abordagens.

5.2 Limitações

Nesta secção irão ser abordadas as limitações presentes no projeto desenvolvido; realça-se ainda que podem existir limitações que não foram encontradas pelo grupo durante o processo de desenvolvimento do projeto. Além disso, algumas limitações podem não ser consideradas graves ou importantes para serem incluídas no relatório final, especialmente por não afetarem significativamente a aplicabilidade do projeto.

A primeira limitação que este projeto possui é o facto de a aplicação não possuir um menu com definições que o utilizador pode manipular, que era uma das funcionalidades propostas no enunciado, e que o grupo não conseguiu devido principalmente devido a incumprimento de tarefas por parte de 1 membro do grupo.

A segunda limitação do projeto é aplicação não mostrar, nas informações acerca de um determinado roteiro, informações sobre a media disponível para os seus pontos, ficando assim uma funcionalidade proposta no enunciado incompleta. O grupo tentou implementar a parte em falta da funcionalidade, contudo não teve sucesso pois estava a ter alguns problemas a aceder aos dados da base de dados e como o prazo de entrega do projeto encontrava-se próximo decidiu-se não alterar a estrutura da base de dados que poderia afetar outras funcionalidades da aplicação, ficando assim uma funcionalidade incompleta.

5.3 Funcionalidades extra

Foi decidido que o grupo iria implementar um mapa do Google Maps embutido numa das atividades da aplicação, acessível através do menu hambúrguer, com a opção "Maps". Este mapa tem um marcador para a localização atual do utilizador, que é atualizada constantemente - a cada 10 segundos - se o serviço de localização estiver ligado. Inclui,

ainda, um marcador para cada ponto de interesse da aplicação, e permite fazer pesquisa de locais, desde que existentes, no botão de pesquisa do Google Maps.

O grupo decidiu também implementar uma página na aplicação denominada Media que pode ser encontrada no menu hambúrguer, na qual o utilizador premium pode visualizar toda a media presente na aplicação e se desejar pode clicar num botão para fazer download de todo o conteúdo, o grupo optou por esta opção pois pensa que é mais fácil e prático para o utilizador fazer download de toda a media, do que ter que percorrer os pins e selecionar a media que deseja fazer download.

Chapter 6

Gestão de projeto

Nesta secção, iremos apresentar a metodologia seguida para a gestão e distribuição de tarefas ao longo de todo o desenvolvimento do trabalho, assim como uma pequena avaliação ou reflexão coletiva de todos os elementos do grupo sobre as suas performances.

6.1 Gestão e distribuição de trabalho

A distribuição inicial do trabalho ficou alocada da seguinte forma: o aluno pg47153 ficaria com a parte da aplicação mais relacionada com o Google Maps e funcionalidades associadas, o aluno pg53944 ficaria com a funcionalidade de *login* e *logout* e garantir a qualidade da sessão, cookies e tipo do utilizador associado a estas 2 tarefas. O aluno pg52690 ficou encarregue originalmente de trabalhar com a parte da persistência de dados, ou seja, RoomDb e concetualização do esquema, DAOs, repositórios e *ViewModels*.

No entanto, o último aluno referido não cumpriu com a sua parte, tendo inclusive forçado os 2 outros membros do grupo a fazer parte do trabalho que ficou associado a ele. A gestão das fases consequentes do trabalho e a distribuição das tarefas tornou-se mais custosa devido a tal.

6.2 Eventuais metodologias de controlo de versão utilizadas

Apesar de, até à data da entrega da primeira fase do trabalho, não serem referidas metodologias de controlo de versão, o grupo optou por criar uma branch associada a cada membro do grupo na qual este membro trabalhava, sendos depois aplicados *pull requests* para a *branch* principal, tratando-se de possíveis divergências entre o trabalho das diferentes branches na main.

Apesar de tudo, a abordagem mais recomendada seria ter uma branch "develop" em que recebe os pull requests das branches individuais, tratando das possíveis incongruências dentro desta mesma branch, e colocando a versão da aplicação estável nesta mesma branch "develop" para a branch principal do projeto.

6.3 Reflexão sobre performance individual

Ao longo do projeto, o aluno pg47153 foi o que efetuou a maior parte do trabalho no projeto, tendo implementado praticamente todas as funcionalidades deste projeto, incluindo embutir um mapa a partir do Maps SDK, implementou geofences para obter notificações perto dos pontos de interesse, criou a base de maior parte das Views e RecyclerViews a serem mostradas, implementou alguns repositórios e ViewModels, implementou o serviço de localização, entre outros.

O aluno pg53944 implementou com sucesso o login e logout com o uso de cookies e gestão da sessão e implementou a amostragem de conteúdo media, tanto na atividade específica, onde é posível fazer download dessa media, como nos pontos de interesse.

O aluno pg52690 implementou o histórico de trilhos visitados por parte do utilizador.

Chapter 7

Conclusão

Tratando-se de um guia turístico para uma aplicação *Android* nativa, o produto de *software* apresentado consegue satisfazer grande parte das necessidade de acordo com os requisitos apresentados.

O nosso projeto apresenta práticas e métodos de desenvolvimento que foram, por exemplo, referidos nas aulas, com padrões de design de *software* incluídos. No entanto, seria interessante abordar tópicos em que a aplicação não ficou polida, como a estética da interface e experiência do utilizador e acabar um ou outro requisito que não ficou feito/-ficou incompleto, nomeadamente a funcionalidade de incluir um menu com definições que o utilizador pode alterar.

Acreditamos que cumprimos com o nosso trabalho e que temos um protótipo bastante sólido e completo se fosse eventualmente incluído no mercado de trabalho.

Bibliography

[1] Square Dev. *A powerful image downloading and caching library for Android.* URL: https://square.github.io/picasso/.

[2] Square Dev. *A type-safe HTTP client for Android and Java.* URL: https://square.github.io/retrofit/.

[3] Square Dev. *OkHttp.* URL: https://square.github.io/okhttp/.

[4] Google for Developers. *Broadcasts overview.* URL: https://developer.android.com/guide/components/broadcasts.

[5] Google for Developers. *Create a Notification.* URL: https://developer.android.com/develop/ui/views/notifications/build-notification.

[6] Google for Developers. *Create and monitor geofences.* URL: https://developer.android.com/training/location/geofencing.

[7] Google for Developers. *Get data from the internet.* URL: https://developer.android.com/codelabs/basic-android-kotlin-compose-getting-data-internet#6.

[8] Google for Developers. *Maps SDK for Android overview.* URL: https://developers.google.com/maps/documentation/android-sdk/overview.

[9] Google for Developers. *Menus.* URL: https://developer.android.com/develop/ui/views/components/menus.

[10] Google for developers. *Accessing data using Room DAOs.* URL: https://developer.android.com/training/data-storage/room/accessing-data.

[11] Google for developers. *Media3 ExoPlayer.* URL: https://developer.android.com/guide/topics/media/exoplayer.

[12] Google for developers. *Save data in a local database using Room.* URL: https://developer.android.com/training/data-storage/room.

Chapter 8

Anexos

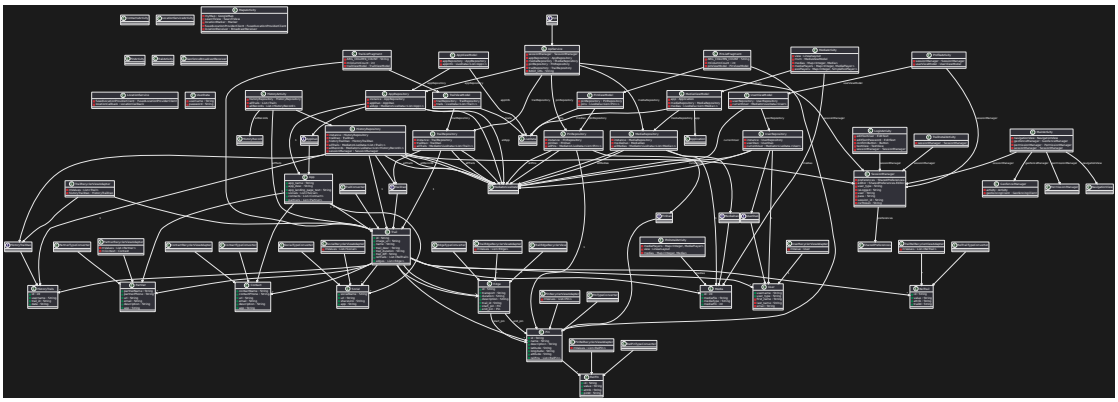

Figure 8.1: Diagrama de classes