

UMinho

**Mestrado em Engenharia Informática**  
**Tópicos de Desenvolvimento de Software**  
**(2022/2023)**

TRABALHO PRÁTICO: PARTE 1

Luís Silva (pg50564)  
Simão Cunha (a93262)  
Gonçalo Pereira (a93168)

Braga, 21 de maio de 2023

# Conteúdo

|       |  |    |
|-------|--|----|
| 1     | Introdução . . . . .   | 2  |
| 2     | Detalhes de implementação . . . . .                          | 2  |
| 2.1   | Estrutura do projeto . . . . .                               | 2  |
| 2.2   | Soluções de implementação . . . . .                          | 3  |
| 2.3   | Bibliotecas/dependências utilizadas . . . . .                | 6  |
| 2.4   | Padrões de software utilizados . . . . .                     | 7  |
| 2.4.1 | Creational Pattern: Singleton . . . . .                      | 7  |
| 2.4.2 | Architectural pattern: Model-View-ViewModel (MVVM) . . . . . | 8  |
| 2.4.3 | Structural Patterns: Adapter . . . . .                       | 8  |
| 3     | Mapa de navegação de GUI . . . . .                           | 8  |
| 4     | Funcionalidades . . . . .                                    | 9  |
| 5     | Discussão de resultados . . . . .                            | 25 |
| 5.1   | Trabalho realizado . . . . .                                 | 25 |
| 5.2   | Limitações . . . . .   | 25 |
| 5.3   | Testes . . . . .   | 25 |
| 5.3.1 | Software Testing . . . . .                                   | 25 |
| 5.3.2 | Software Analysis . . . . .                                  | 27 |
| 5.4   | Funcionalidades extra . . . . .                              | 28 |
| 6     | Gestão de projeto . . . . .                                  | 29 |
| 6.1   | Gestão e Distribuição de trabalho . . . . .                  | 29 |
| 7     | Conclusão . . . . .  | 30 |
| 1     | Anexos . . . . .   | 31 |

# 1 Introdução

O presente relatório surge no âmbito da primeira parte do trabalho prático da UC de Tópicos de Desenvolvimento de Software. Neste trabalho foi-nos proposto o desenvolvimento de um guia turístico, sob a forma de uma aplicação móvel híbrida. Deste modo, o projeto foi desenvolvido com recurso ao conhecimento adquirido ao longo do semestre relativo a desenvolvimento nativo e multi-plataforma, implicando o uso de Java e a tecnologia nativa Android.

A aplicação, denominada “BraGuia”, é uma aplicação para orientação turística que oferece roteiros turísticos aos seus utilizadores, tendo a capacidade de substituir um verdadeiro guia turístico, oferecendo funcionalidades de localização e navegação geográfica, reprodução de conteúdo multimédia acerca de pontos de interesse, etc. De forma a obter o conteúdo para mostrar ao utilizador, a aplicação deve consumir informação de um backend desenvolvido pela equipa docente - obtido por <https://c5a2-193-137-92-29.eu.ngrok.io/>.

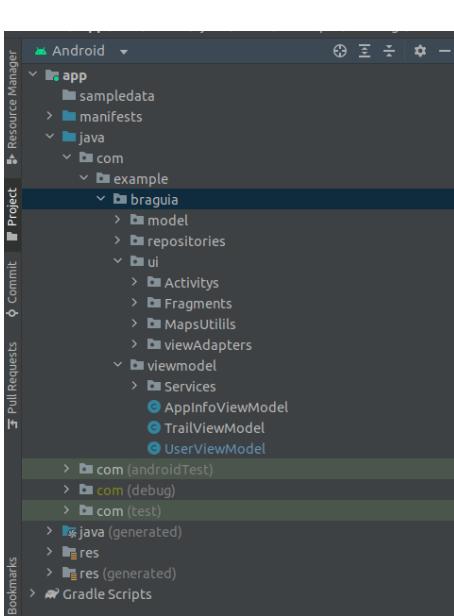
## 2 Detalhes de implementação

### 2.1 Estrutura do projeto

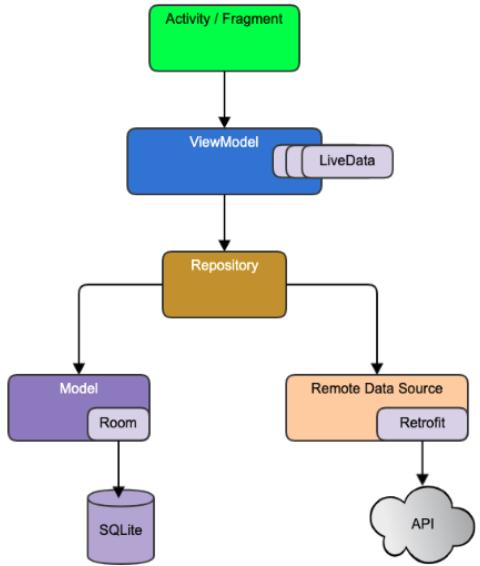
A arquitetura de software desempenha um papel fundamental no desenvolvimento de aplicações Android robustas, escaláveis e fáceis de manter. A abordagem arquitetural recomendada pelo professor é o *Model-view-viewmodel* (MVVM), que oferece uma separação clara entre a lógica de negócios e a interface do utilizador. Nesta seção, explicaremos como integramos essa arquitetura no nosso projeto.

Na Figura 1, são apresentadas duas imagens: uma representa a estrutura de ficheiros do projeto, enquanto a outra ilustra a arquitetura MVVM. Com o intuito de destacar a integração da arquitetura ao projeto, optamos por fornecer explicações sobre a estrutura dos ficheiros e a função de cada secção.

Na pasta *model/*, temos a RoomDatabase (GuideDatabase), onde todos os modelos de dados serão armazenados. A pasta *repositories/* contém os repositórios responsáveis por fazer pedidos à API e armazenar/consultar dados da base de dados. Na pasta *viewmodel/*, temos vários *viewModels* que possuem LiveData dos dados do repositório e fornecem métodos à view, facilitando a obtenção de informações do banco de dados. Os serviços também estão incluídos nesta pasta, sendo que comunicam diretamente com os repositórios e possuem comportamentos semelhantes a *viewModels*. Na pasta *ui/*, estão localizados todos os fragmentos e atividades da aplicação, além de métodos suplementares para a criação de views.



(a) Estrutura dos ficheiros



(b) Arquitetura MVVM

**Figura 1:** Estrutura do projeto

## 2.2 Soluções de implementação

Nesta secção iremos abordar as soluções que o grupo achou mais interessantes/complexas para satisfazer funcionalidades propostas.

- **Persistência de dados:** De forma a preservar os dados do modelo de dados, o grupo utilizou a roomDatabase para guardar os dados de forma persistente. Os repositórios verificam através de DAO's se há dados inseridos na base de dados e caso não estejam, este vai buscá-los à API de forma a repopular a base de dados. Este comportamento está demonstrado na seguinte código:

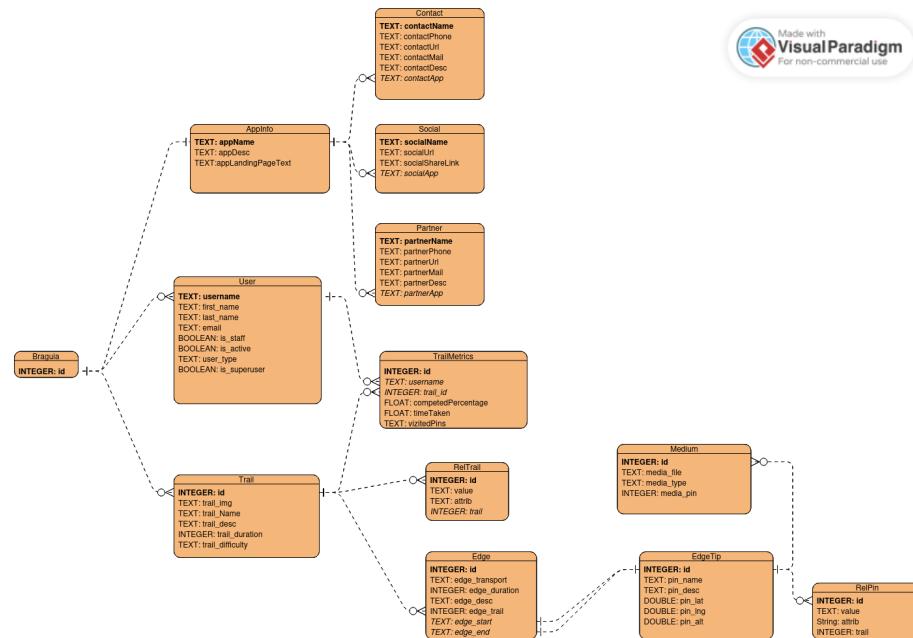
```

public AppInfoRepository(Application application){
    database = GuideDatabase.getInstance(application);
    appInfoDAO = database.appInfoDAO();
    appInfo = new MediatorLiveData<>();
    appInfo.addSource(
        appInfoDAO.getAppInfo(), localAppInfo -> {
            if (localAppInfo != null) {
                appInfo.setValue(localAppInfo);
            } else {
                try {
                    makeRequest();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }
    );
}

```

Para além disto, o grupo utiliza sharedPreferences para armazenar informações mais pequenas e que vão ser acedidas com menos frequência como a variável que verifica se o utilizador corre a aplicação em *dark mode*.

- **Modelo de base de dados:** O grupo decidiu guardar todos os dados provenientes da API fornecida de forma a possibilitar a expansão em caso de novas funcionalidades. Para além disso, foram adicionadas algumas entidades como o trailMetrics, que fornece os dados provenientes de uma navegação do roteiro. O diagrama de base de dados é o seguinte:



- **Pedidos à API:** Todos os pedidos feitos à API utilizam o Retrofit de forma a fazer pedidos HTTP de forma assíncrona à thread principal com o comando enqueue, possibilitando o utilizador de interagir com a aplicação enquanto o pedido executa. Quando o pedido acabar, é possível reagir conforme a resposta a partir de callbacks. No seguinte código é demonstrado um exemplo deste procedimento:

```
private void makeRequest() throws IOException {
    Retrofit retrofit=new Retrofit.Builder()
        .baseUrl("https://c5a2-193-137-92-29.eu.ngrok.io/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
    AppInfoAPI api = retrofit.create(AppInfoAPI.class);
    Call<List<AppInfo>> call = api.getAppInfo();
    call.enqueue(new retrofit2.Callback<List<AppInfo>>() {
        @Override
        public void onResponse(Call<List<AppInfo>> call, Response<List<AppInfo>> response) {
            if(response.isSuccessful()) {
                insert(response.body().get(0));
            }
            else{
                Log.e("main", "onFailure: "+response.errorBody());
            }
        }
        @Override
        public void onFailure(Call<List<AppInfo>> call, Throwable t) {
            Log.e("main", "onFailure: "+t.getMessage());
        }
    });
}
```

```

        }
    }
    @Override
    public void onFailure(Call<List<AppInfo>> call, Throwable t) {
        Log.e("main", "onFailure: " + t.getMessage());
        Log.e("main", "message: " + t.getCause());
    }
});
```

- Funcionamento offline:** De forma a que o utilizador entre na aplicação sem internet, há uma verificação no sharedPreferences para que o último utilizador faça login sem necessitar de fazer pedidos á API. Ao iniciar o repositorio dos trails, é criada uma instância de conteúdo para cada URL de multimédia, armazenando um arquivo com o respectivo nome. Durante o carregamento da multimédia, o sistema verifica a existência de um arquivo correspondente ao URL consultado e, se o arquivo estiver armazenado, é possível reproduzi-lo.

```

url = new URL(file_url.replace("http", "https"));
try (InputStream inputStream = url.openStream()) {
    destinationPath = file_url.replace("http://", "").replace("/", "");
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        try (FileOutputStream fos = context.openFileOutput(
destinationPath, Context.MODE_PRIVATE)) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
                fos.write(inputStream.readAllBytes());
            }
        }
    }
}
```

- Integração de um mapa para o roteiro:** Para disponibilizar um mapa interativo onde estão marcados os pontos de interesse de um roteiro, é utilizada a API do Google Maps para disponibilizar um mapa, a partir da qual fornecemos as localizações de cada ponto e configuramos algumas opções da câmara. Este procedimento está refletido no seguinte código:

```

public void loadMap(GoogleMap googleMap) {
    mMap = googleMap;

    if(edgeTips.size()>0) {
        ArrayList<LatLng> wayPointsAPI = new ArrayList<>();
        for (EdgeTip edgeTip : edgeTips) {
            wayPointsAPI.add(edgeTip.getMapsCoordinate());
            mMap.addMarker(new MarkerOptions().position(edgeTip.
getMapsCoordinate()).title(edgeTip.getPin_name()));
        }
        LatLng source = wayPointsAPI.get(0);

        if (wayPointsAPI.size() >= 2) {
            LatLng destination = wayPointsAPI.get(wayPointsAPI.size() -
1);
            wayPointsAPI.remove(0);
            wayPointsAPI.remove(wayPointsAPI.size() - 1);
        }
    }
}
```

```

        new GetPathFromLocation(getActivity(), source, destination,
wayPointsAPI, mMap, false, false, polyline -> {
            polyline.setColor(R.color.teal_200);
            mMap.addPolyline(polyline);
        }).execute();
    }
    mMap.getUiSettings().setZoomControlsEnabled(true);
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(source, 12));
}
}

```

- **Navegação por fragmentos:** De forma a manter views da main activity como a sidebar e a button bar intactas pela navegação da aplicação, o grupo decidiu usar um NavController, que substitui fragmentos que vão aparecer no meio do ecrã. Esta abordagem é recomendada pela Google e facilita bastante a leitura do código. De seguida está o exemplo da atribuição de fragmentos a botões da sidebar:

```

public void loadMap(GoogleMap googleMap) {
    menuItem -> {
        int itemId = menuItem.getItemId();
        if (itemId == R.id.profile) {
            navController.navigate(R.id.profileFragment);
        } else if (itemId == R.id.emergency_contacts) {
            navController.navigate(R.id.emergencyContactsFragment);
        } else if (itemId == R.id.socials_contacts) {
            navController.navigate(R.id.socialsListFragment);
        ...
    }
}

```

- **Navegação de um roteiro:** Quando se inicia a navegação de um roteiro o utilizador é redirecionado para a aplicação do Google Maps, no modo de navegação com todos os pontos do roteiro. Para além disso, é criado um serviço de localização para que este verifique se o utilizador se aproxima de um ponto de interesse, e caso o faça seja lançada uma notificação que o redireciona para o ponto, note que é necessário o utilizador mudar de localização de forma a acionar a verificação do ponto de passagem. Este serviço é *foreground* porque é essencial que os recursos deste não sejam libertados, e para além disso, permite ao utilizador voltar facilmente à página de navegação do roteiro. Na página de navegação do roteiro tem um botão para parar o serviço e retornar à *main activity*. Quando o serviço é parado, este accede aos repositórios e guarda a informação da viagem.

## 2.3 Bibliotecas/dependências utilizadas

De forma a acceder a recursos do dispositivo, foi necessário estabelecer algumas permissões, tais como:

- Acesso ao serviço de chamadas telefónicas;
- Acesso à internet;
- Acesso à localização (aproximada e exata);
- Acesso ao estado da ligação à internet;

- Permissão de envio de *push notifications*;
- Execução de serviços em primeiro plano.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

**Figura 2:** Permissões expressas no *AndroidManifest.xml*

Além disso, também foram necessárias utilizar algumas bibliotecas - expressas no ficheiro *build.gradle* - como por exemplo:

- **Retrofit**
  - 'com.squareup.retrofit2:retrofit:2.9.0'
  - 'com.squareup.retrofit2:converter-gson:2.9.0'
- **Picasso**
  - 'com.squareup.picasso:picasso:2.71828'
  - 'com.jakewharton.picasso:picasso2-okhttp3-downloader:1.1.0'
- **Google Maps:** 'com.google.android.gms:play-services-maps:18.1.0';
- **Compose:** "androidx.navigation:navigation-compose:\$rootProject.nav\_version"
- **Testing:**
  - JUnit ("junit:junit:\$rootProject.junitVersion")
  - Firebase ('com.google.firebaseio:firebase-analytics')
- ...

## 2.4 Padrões de software utilizados

### 2.4.1 Creational Pattern: Singleton

De forma a garantir que apenas há uma base de dados no sistema, foi implementado o Singleton da seguinte forma:

```
public abstract class GuideDatabase extends RoomDatabase {
    public static volatile GuideDatabase INSTANCE = null;
    public static GuideDatabase getInstance(Context context) {
        if (INSTANCE == null) {
            synchronized (GuideDatabase.class) {
```

```

        if (INSTANCE == null) {
            INSTANCE = Room.databaseBuilder(context, GuideDatabase.
class, DATABASE_NAME)
                .fallbackToDestructiveMigration()
                .addCallback(callback)
                .build();
        }
    }
    return INSTANCE;
}

```

#### 2.4.2 Architectural pattern: Model-View-ViewModel (MVVM)

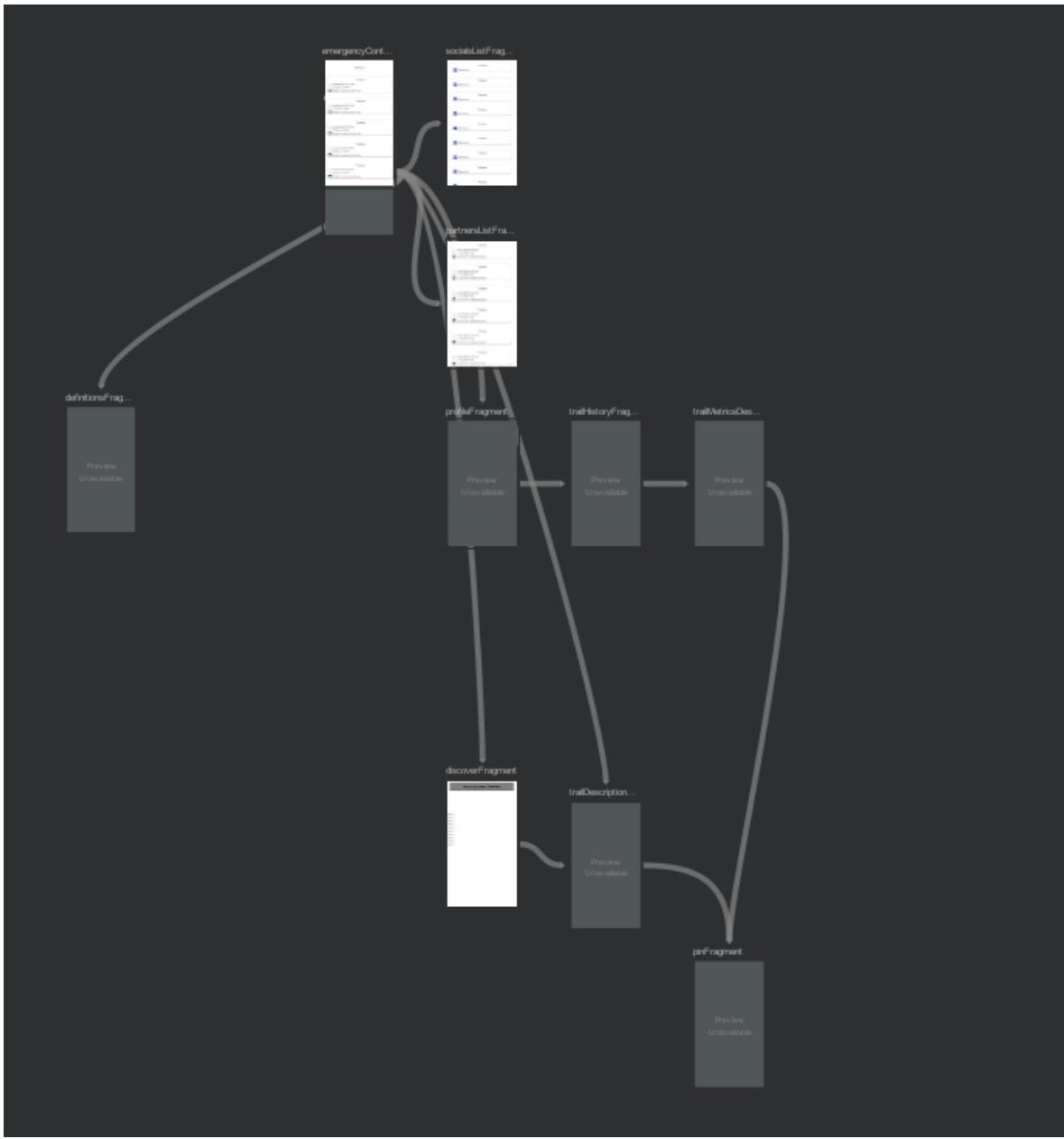
Conforme mencionado anteriormente, o projeto foi fundamentado no padrão MVVM. O grupo não seguiu o padrão de forma rígida, sendo que houve uma tendência em atribuir muita lógica à camada de visualização, permitindo que esta acedesse e utilizasse métodos das entidades contidas no modelo. Sendo assim, a camada view model não teve o seu devido uso, sendo que devia ter sido mais explorada. No entanto, o grupo considera que a camada de dados foi estruturada de forma adequada e que houve boa separação de responsabilidades entre a base de dados, o repositório e o view model. Nos anexos, está presente o nosso diagrama de classes que demonstra este comportamento (figura 29).

#### 2.4.3 Structural Patterns: Adapter

O padrão Adapter é utilizado para permitir que duas classes com interfaces incompatíveis possam trabalhar juntas. Este padrão foi principalmente utilizado para fornecer informação ao *recycler view*. Ao utilizar o padrão Adapter, como exemplo a classe ContactsRecyclerViewAdapter, é encapsulada a classe Contact e esta torna-se compatível com a interface de exibição de listas fornecida pelo RecyclerView. Isso permite que o RecyclerView exiba a lista de contatos sem precisar conhecer os detalhes internos da classe Contact.

### 3 Mapa de navegação de GUI

No navegador só estão demonstradas os diferentes fragmentos que podem ser inseridos na aplicação (na *Main Activity*). Com o uso de atividades, o GUI irá mudar ligeiramente, acrescentando as atividades de navegação ao iniciar o roteiro e também a atividade de *login*. Nos anexos estará uma representação real da GUI da aplicação gerada por um robot do testes Firebase (figura 28).



**Figura 3:** *Nav graph* do Android Studio

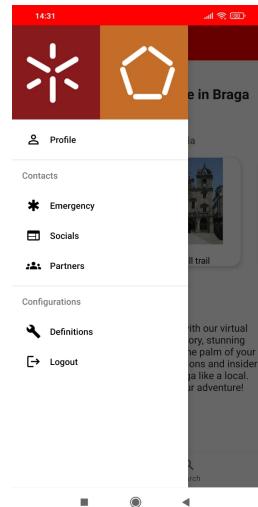
## 4 Funcionalidades

A funcionalidades contidas no sistema correspondem diretamente aos requisitos impostos para o mesmo, sendo que nesta secção o grupo optou por referir alguns dos principais requisitos.

1. A aplicação deve possuir uma página inicial onde apresenta as principais funcionalidades do guia turístico, descrição, etc.



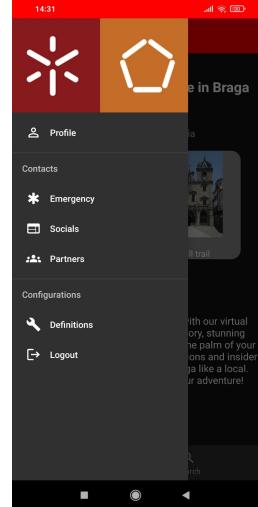
(a) Página inicial



(b) Sidebar da página inicial

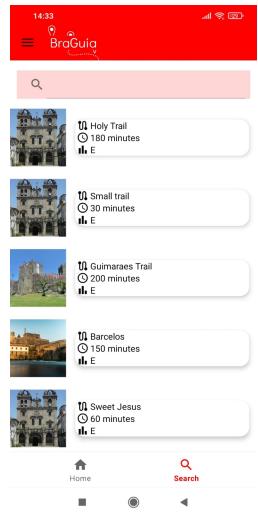


(c) Página inicial (dark mode)

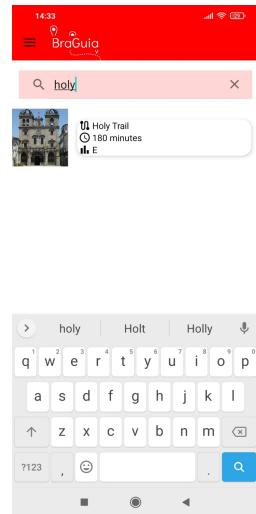


(d) Sidebar da página inicial (dark mode)

2. A aplicação deve mostrar num ecrã, de forma responsiva, uma lista de roteiros disponíveis.



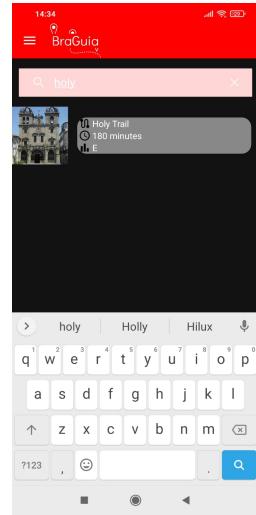
(a) Lista de roteiros



(b) Pesquisa na lista de roteiros



(c) Lista de roteiros (*dark mode*)



(d) Pesquisa na lista de roteiros (*dark mode*)

3. A aplicação deve permitir efetuar autenticação.

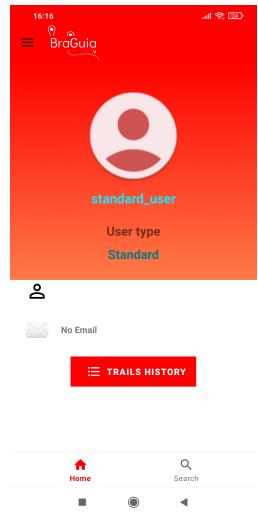


(a) Autenticação (com credenciais erradas)

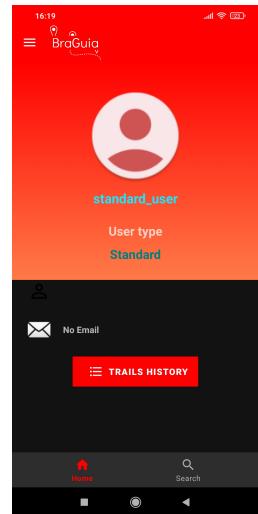


(b) Autenticação (com credenciais certas)

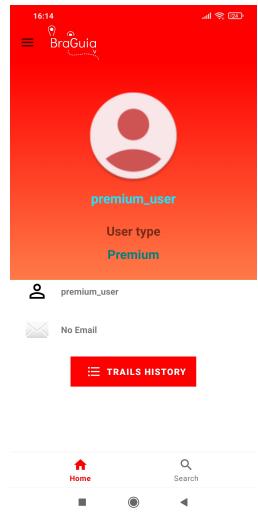
4. A aplicação deve suportar 2 tipos de utilizadores: utilizadores standard e utilizadores premium.



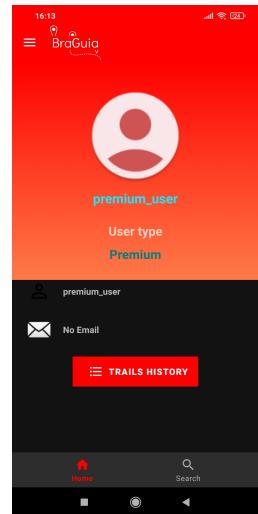
(a) Página de utilizador standard



(b) Página de utilizador standard (*dark mode*)

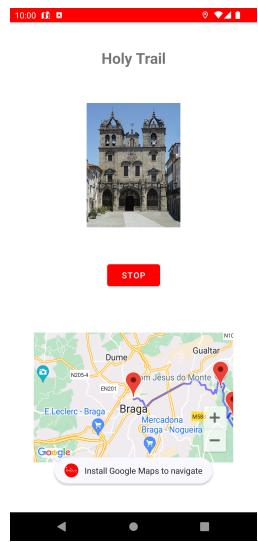


(c) Página de utilizador premium

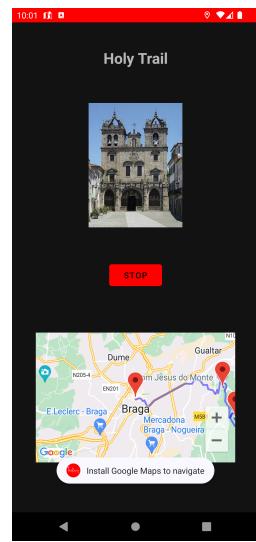


(d) Página de utilizador premium (*dark mode*)

5. 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) Aviso para instalar Google Maps

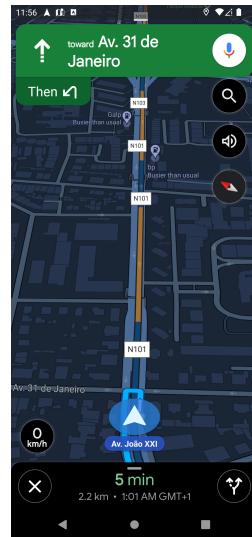


(b) Aviso para instalar Google Maps (*dark mode*)

6. Para utilizadores premium, a aplicação deve possibilitar a capacidade de navegação, de consulta e descarregamento de mídia.

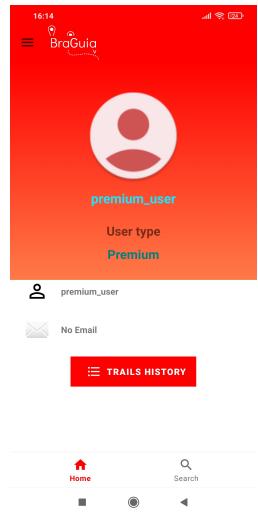
De forma a exemplificarmos este requisito, deve-se consultar as figuras 9a e 16a.

7. 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.

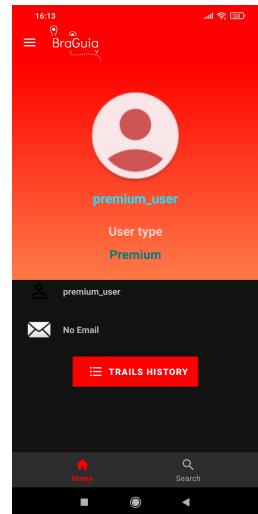


(a) Navegação Google Maps

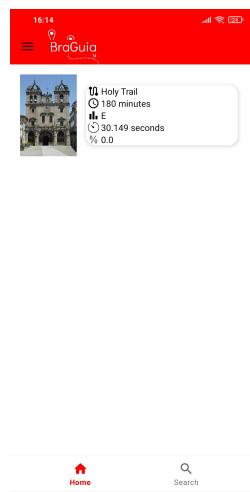
8. A aplicação deve possuir uma página de informações acerca do utilizador atualmente autenticado.



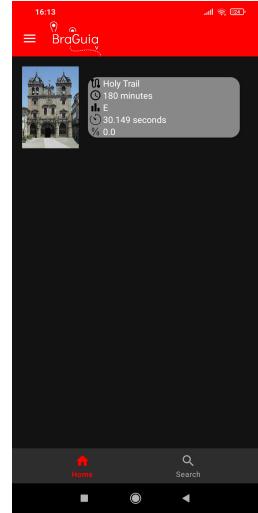
(a) Página de utilizador premium



(b) Página de utilizador premium (*dark mode*)

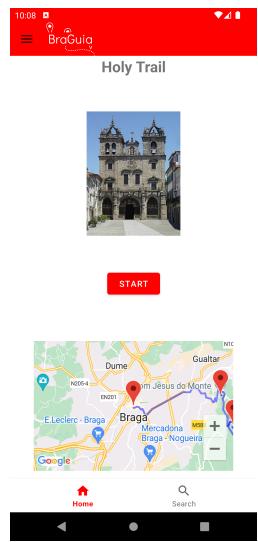


(c) Página de histórico de utilizador premium

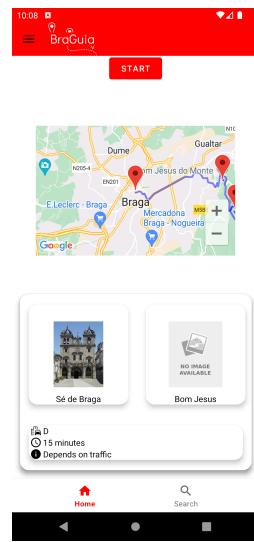


(d) Página de histórico de utilizador premium(*dark mode*)

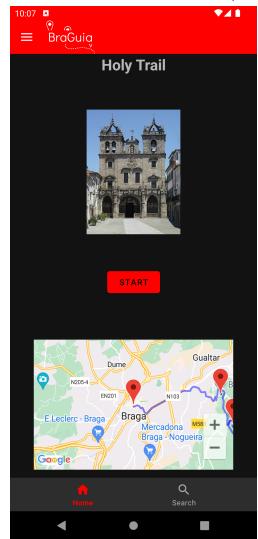
9. A aplicação deve mostrar, numa única página, informação acerca de um determinado roteiro: galeria de imagens, descrição, mapa do itinerário com pontos de interesse e informações sobre a mídia disponível para os seus pontos.



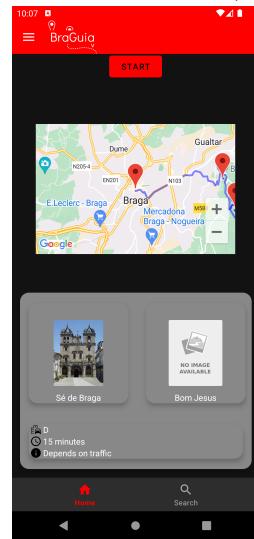
(a) Página de um roteiro (parte 1)



(b) Página de um roteiro (parte 2)



(c) Página de um roteiro (parte 1) (dark mode)



(d) Página de um roteiro (parte 2) (dark mode)

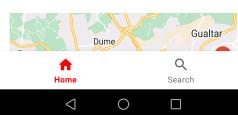
10. A aplicação deve possuir a capacidade de iniciar um roteiro.



Holy Trail



START



(a) Início de um roteiro



Holy Trail

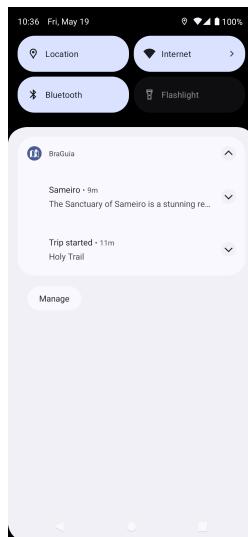


START



(b) Início de um roteiro (*dark mode*)

11. A aplicação deve possuir a capacidade de emitir uma notificação quando o utilizador passa perto de um ponto de interesse.

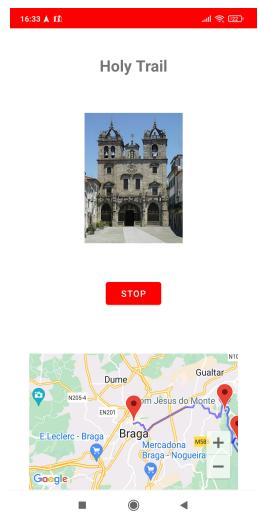


(a) Notificações com locais de passagem do roteiro

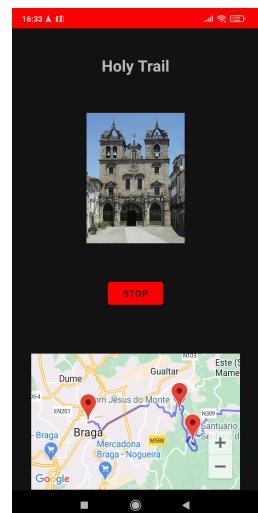
12. A notificação emitida quando o utilizador passa pelo ponto de interesse deve conter um atalho para o ecrã principal do ponto de interesse.

Ao receber a notificação tal como observado na figura 13a, um clique na mesma redireciona para a página de "Sameiro".

13. A aplicação deve possuir a capacidade de interromper um roteiro.

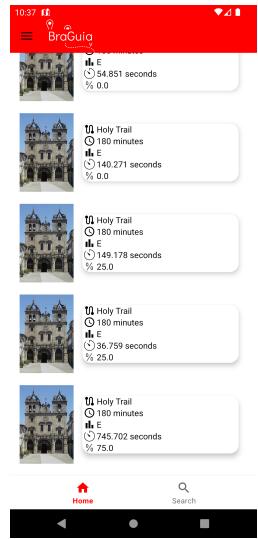


(a) Interrupção de um roteiro

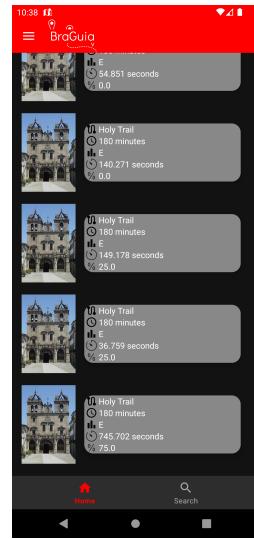


(b) Interrupção de um roteiro (*dark mode*)

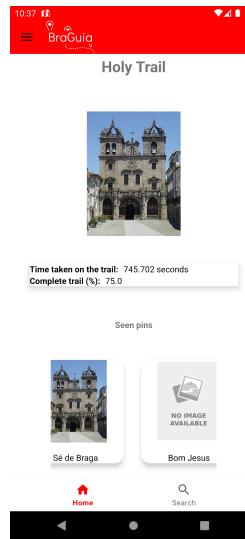
14. A aplicação deve guardar (localmente) o histórico de roteiros e pontos de interesse visitados pelo utilizador.



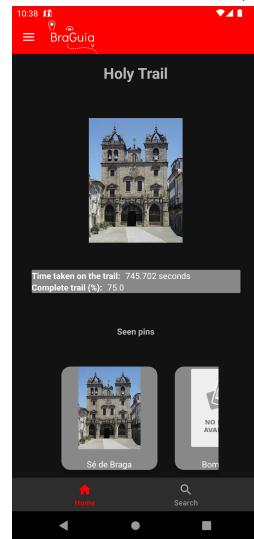
(a) Lista de roteiros visitados



(b) Lista de roteiros visitados (*dark mode*)

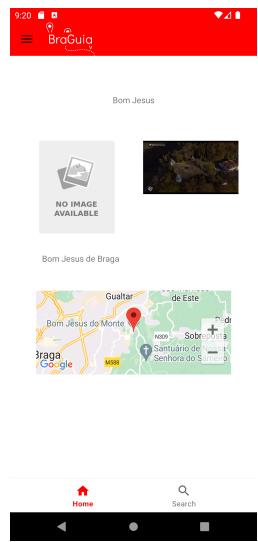


(c) Página de roteiros visitado

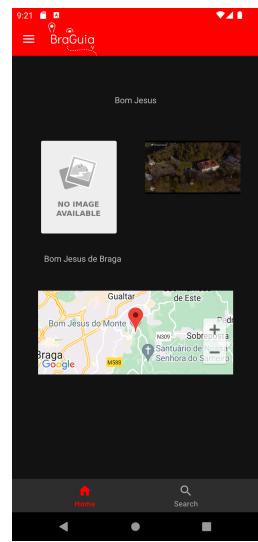


(d) Página de roteiros visitado (*dark mode*)

15. A aplicação deve possuir uma página que mostre toda a informação disponível relativa a um ponto de interesse: localização, galeria, mídia, descrição, propriedades, etc.



(a) Página de um ponto turístico

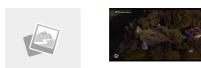


(b) Página de um ponto turístico (*dark mode*)

16. A aplicação deve ter a capacidade de apresentar e produzir 3 tipos de mídia: voz, imagem e vídeo.

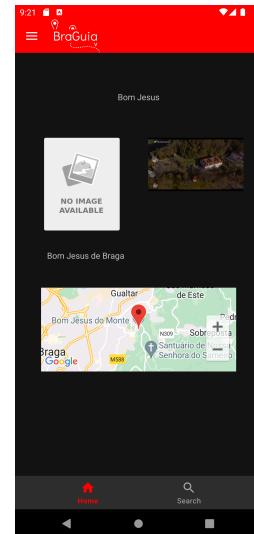


Bom Jesus



NO IMAGE  
AVAILABLE

Bom Jesus de Braga



Bom Jesus



NO IMAGE  
AVAILABLE

Bom Jesus de Braga



Home Search

(a) Produção de conteúdo multimédia de vídeo



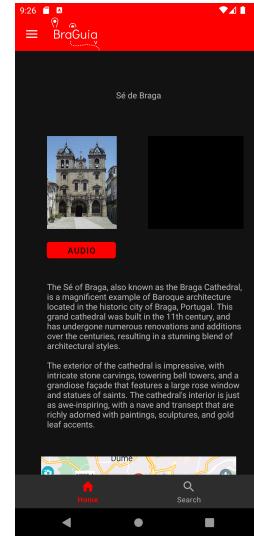
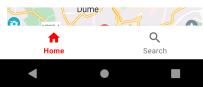
Sé de Braga



AUDIO

The Sé de Braga, also known as the Braga Cathedral, is a magnificent example of Baroque architecture located in the historic city of Braga, Portugal. This grand cathedral was built in the 11th century, and has undergone numerous renovations and additions over the centuries, resulting in a stunning blend of architectural styles.

The exterior of the cathedral is impressive, with intricate stone carvings, towering bell towers, and a grandiose facade that features a large rose window and statues of saints. The interior is just as awe-inspiring, with a nave and transept that are richly adorned with paintings, sculptures, and gold leaf accents.



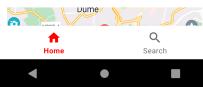
Sé de Braga



AUDIO

The Sé de Braga, also known as the Braga Cathedral, is a magnificent example of Baroque architecture located in the historic city of Braga, Portugal. This grand cathedral was built in the 11th century, and has undergone numerous renovations and additions over the centuries, resulting in a stunning blend of architectural styles.

The exterior of the cathedral is impressive, with intricate stone carvings, towering bell towers, and a grandiose facade that features a large rose window and statues of saints. The interior is just as awe-inspiring, with a nave and transept that are richly adorned with paintings, sculptures, and gold leaf accents.



(c) Produção de conteúdo multimédia de imagem e áudio

(b) Produção de conteúdo multimédia de vídeo (dark mode)



Home

Search

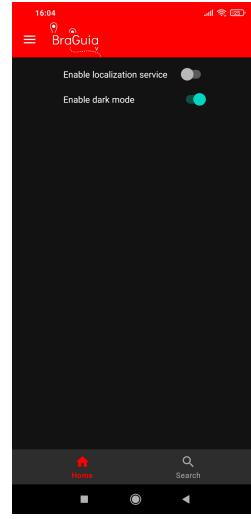


(d) Produção de conteúdo multimédia de imagem e áudio (dark mode)

17. A aplicação deve possuir um menu com definições que o utilizador pode manipular.



(a) Definições da aplicação

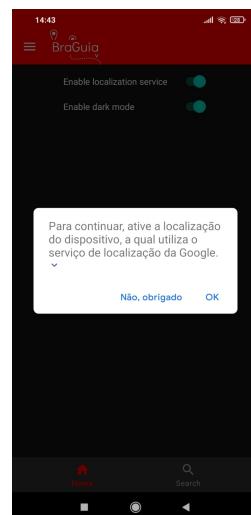


(b) Definições da aplicação (*dark mode*)

18. A aplicação deve possuir a capacidade de ligar, desligar e configurar os serviços de localização.

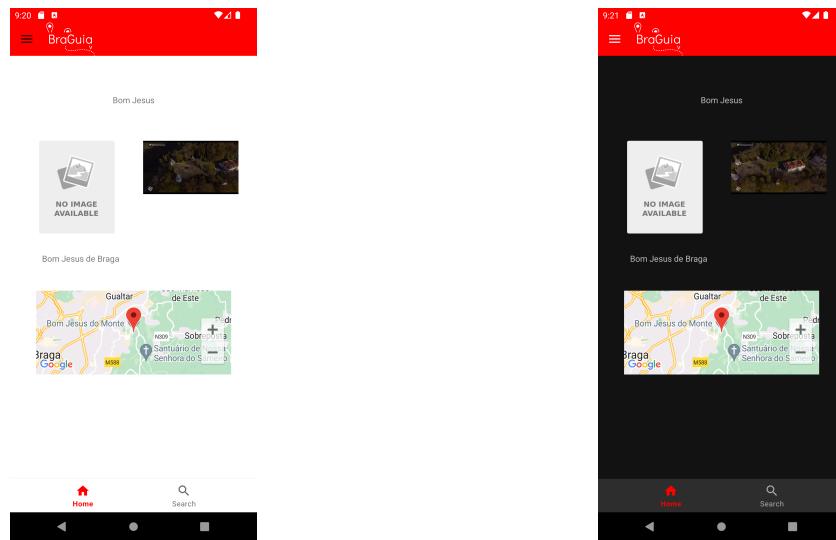


(a) Ligação da localização



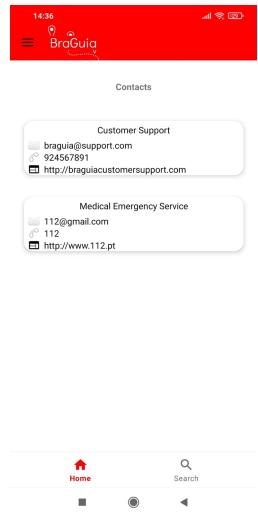
(b) Ligação da localização (*dark mode*)

19. A aplicação deve possuir a capacidade de descarregar mídia do backend e alojá-la localmente, de modo a poder ser usada em contextos de conectividade reduzida.



(a) Página de um ponto turístico a correr um vídeo  
 (b) Página de um ponto turístico a correr um vídeo  
 guardado localmente (dark mode)

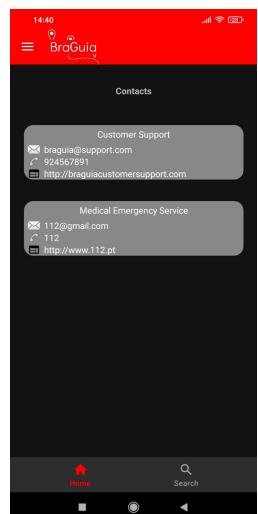
20. A aplicação deve possuir a capacidade de efetuar chamadas para contactos de emergência da aplicação através de um elemento gráfico facilmente acessível na aplicação.



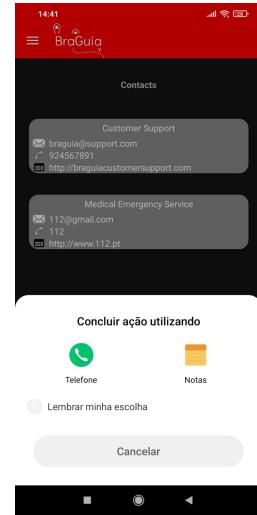
(a) Página de contactos de emergência



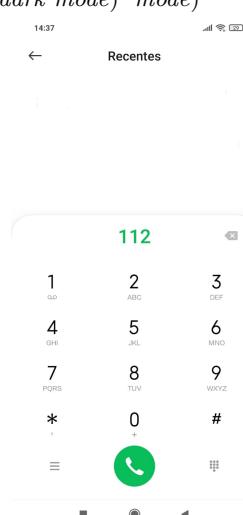
(b) Redirecionamento para efetuar a chamada



(c) Página de contactos de emergência (dark mode)



(d) Redirecionamento para efetuar a chamada (dark mode)



(e) Chamada telefónica

## 5 Discussão de resultados

### 5.1 Trabalho realizado

Considerando que todos os requisitos foram mínimamente cumpridos, o grupo está satisfeito com o trabalho. Apesar de não conseguirmos resolver todos os problemas encontrados na fase de *testing* (em vários dispositivos), o grupo conseguiu resolver a sua grande maioria, provando a grande utilidade desta etapa no processo de desenvolvimento Android. Para além disso, o grupo também considera que deveria ter seguido de forma mais rigorosa a arquitetura Android de forma a generalizar o projeto. Finalmente, gostaríamos de ter prestado mais atenção ao GitHub Actions, que, apesar de termos experimentado inúmeras vezes o seu uso, o grupo não obteve sucesso na *release* automática do APK, mais especificamente, na geração com o Gradle, mas que quereremos retificar para a segunda parte do trabalho prático.

### 5.2 Limitações

Durante a fase de *testing* (que irá ser falada na próxima secção), foram detetadas algumas *bugs* que o grupo não conseguiu corrigir, tais como:

- Em alguns dispositivos móveis, quando o utilizador ativa o *dark mode* na aplicação, esta faz um *log out* e redireciona para a página de *login*;
- Foi detetado que, em alguns dispositivos (detetados pelos dispositivos usados no Firebase), existe a dificuldade em desenhar ficheiros bitmap (i.e. ficheiros da pasta *drawable/*).

### 5.3 Testes

#### 5.3.1 Software Testing

Nesta secção, iremos descrever a etapa de *testing* à nossa aplicação. Decidimos optar por efetuar *white-box testing* com o recurso ao JUnit e *black-box testing* com o recurso do Firebase Test Lab.

Para efetuar *white-box testing*, utilizamos o JUnit para testar se as DAO's e os Repository estão a funcionar de acordo com o suposto. Os testes efetuados encontram-se na diretoria *com.example.braguia/androidTest/*.

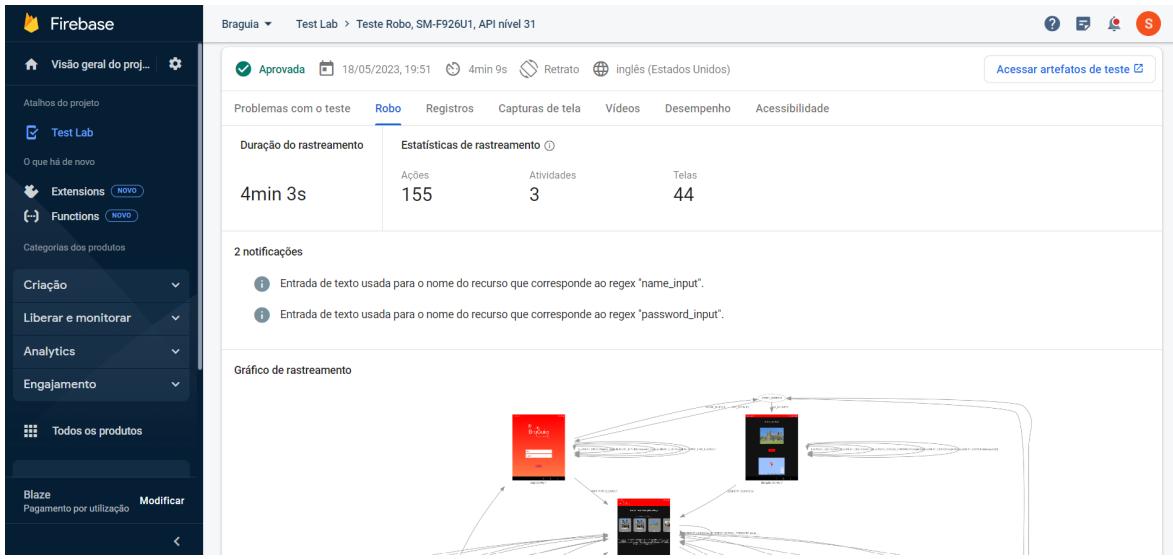
**Figura 22:** Unit testing com JUnit

Já com o intuito de efetuarmos *black-box testing*, utilizamos a ferramenta Firebase Test Lab, onde realizamos vários testes de forma a verificar se a navegação do UI está a funcionar de acordo com o suposto. Na figura abaixo, podemos observar um exemplo de testes efetuados a vários dispositivos que são disponibilizados pela ferramenta.

| Teste Robo | Data              | Falha | Estáveis | Ignorados | Inconclusivos | Total de dispositivos |
|------------|-------------------|-------|----------|-----------|---------------|-----------------------|
| Teste Robo | 18/05/2023, 19:51 | 1     | 8        | 1         | 4             | 14                    |

**Figura 23:** Lista de testes na ferramenta Firebase

Na figura seguinte, podemos observar a sequência de passos que o robot do Firebase efetuou aquando da navegação na nossa aplicação.

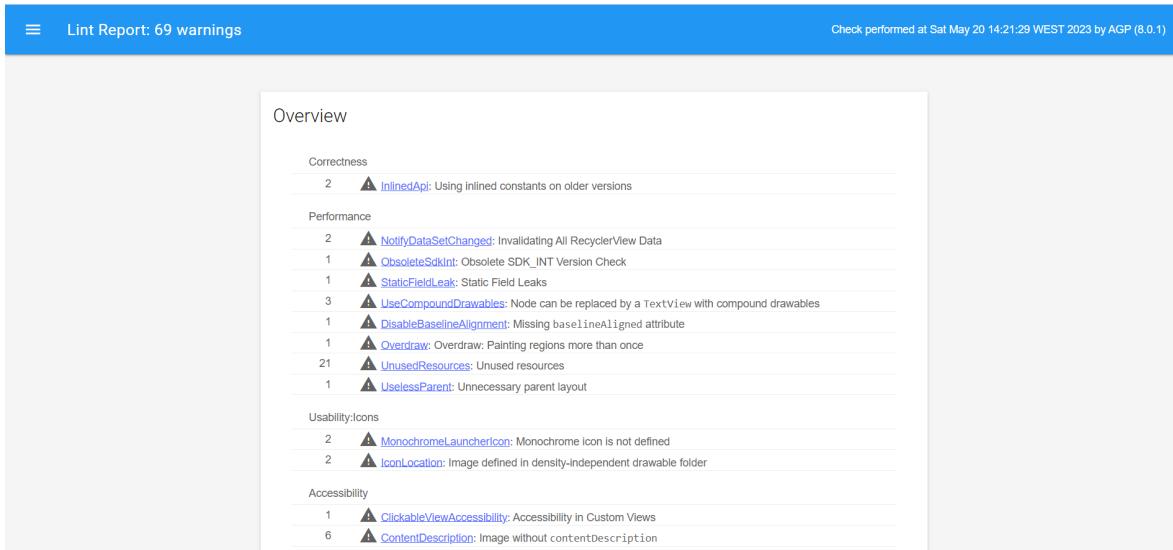


**Figura 24:** Sequência de passos de *testing* do robot da ferramenta Firebase

### 5.3.2 Software Analysis

Nesta secção, iremos descrever o nosso processo de análise de *software*. Para tal, recorremos ao *gradlew lint* de forma a podermos identificar *warnings* e eventuais erros no código da nossa aplicação - quer na lógica do programa, quer no layout dos fragmentos/atividades. Além disso, também recorremos novamente ao Firebase Test Lab para analisarmos a performance da nossa aplicação aquando dos testes efetuados pelos seus *robots*.

Tal como dito, executamos o *gradlew lint* de forma a identificarmos *warnings* e eventuais erros no código da nossa aplicação e obtivemos o seguinte input:



**Figura 25:** Execução do *gradlew lint*

De seguida, utilizamos o Firebase Test Lab para observarmos a performance da nossa apli-

cação, e, aproveitando o exemplo utilizado na figura 24, observamos o conteúdo da aba *Desempenho*:

The screenshot shows the Firebase Test Lab interface for a project named 'Braguia'. The 'Teste Robo, SM-F926U1, API nível 31' test run is selected. The 'Desempenho' (Performance) tab is active. At the top, it displays a summary: 'Aprovada' (Approved), date '18/05/2023, 19:51', duration '4min 9s', orientation 'Retrato' (Portrait), language 'inglês (Estados Unidos)' (English (United States)). Below this, there are tabs for 'Problemas com o teste' (Test problems), 'Robo' (Robot), 'Registros' (Logs), 'Capturas de tela' (Screenshots), 'Videos', 'Desempenho' (Performance), and 'Acessibilidade' (Accessibility). The 'Desempenho' tab is currently selected. It contains sections for 'Horário de inicialização do app' (App initialization time) and 'Estatísticas de elementos gráficos' (Graphic element statistics). Key data points include: Tempo para exibição inicial (Initial display time) at 596ms, VSyncs perdidas (Lost VSyncs) at 2%, Alta latência de entrada (High input latency) at 67%, Thread lenta da interface do usuário (Slow user interface thread) at 4%, Comandos de emissão lenta (Slow emission commands) at 1%, and Uploads de bitmap lentos (Slow bitmap uploads) at 0%. Below these, a section titled 'Desempenho ao longo do tempo' (Performance over time) shows a timeline of user interactions. A screenshot of the device screen shows a black background with some UI elements. To the right, a timeline shows two events: '0:04 Iniciar atividade principal' (0:04 Start main activity) and '0:07 Digitar "premium\_user" name\_input' (0:07 Type "premium\_user" name\_input). The bottom half of the screen features three line charts showing CPU usage, memory usage, and network traffic over a 4-hour period.

**Figura 26:** Visualização da aba *Desempenho* - parte 1



**Figura 27:** Visualização da aba *Desempenho* - parte 2

#### 5.4 Funcionalidades extra

Neste trabalho prático, decidimos implementar algumas funcionalidades extra que não constavam nos requisitos mínimos do enunciado:

- Tal como pode ser observado nas figuras 11a e 17a, criamos uma imagem do Google Maps que desenha o trajeto a efetuar do roteiro;

- Tal como pode ser observado nas figuras 5b, implementamos uma barra de pesquisas que nos devolve um roteiro ao escrevermos o seu nome (ou parte dele);
- Tal como observado na figura 18b, implementamos uma opção de ativar o *dark mode* na aplicação ao criarmos um *switch* nas definições.

## 6 Gestão de projeto

### 6.1 Gestão e Distribuição de trabalho

Nesta secção, iremos descrever de forma geral a gestão e distribuição de tarefas no trabalho prático - iremos colocar uma percentagem aproximada de trabalho efetuado e uma descrição geral do que cada um efetuou.

- Luís Silva (pg50564) - 50%
  - Lógica de fragmentos/atividades;
  - Serviços/notificações;
  - Integração do Google Maps;
  - Repositórios;
  - Pedidos à API;
  - Navegação;
  - Testes;
  - GitHub Actions
- Simão Cunha (a93262) - 35%
  - Layout dos fragmentos/atividades;
  - Toolbar/sidebar da *Main activity*;
  - *Recycler views* e respetivo layout e lógica;
  - Ativação/desativação do serviço de localização e *dark mode*;
  - GitHub Actions;
  - Funções auxiliares como o envio de notificações;
  - Grande contribuição para o relatório
- Gonçalo Pereira (a93168) - 15%
  - Ativação/desativação do serviço de localização e *dark mode*;
  - Utilização da aplicação em modo *offline*;
  - Tratamento do conteúdo multimédia;
  - Lógica de pesquisa de roteiros pela barra de pesquisas;
  - *Bottom navbar*

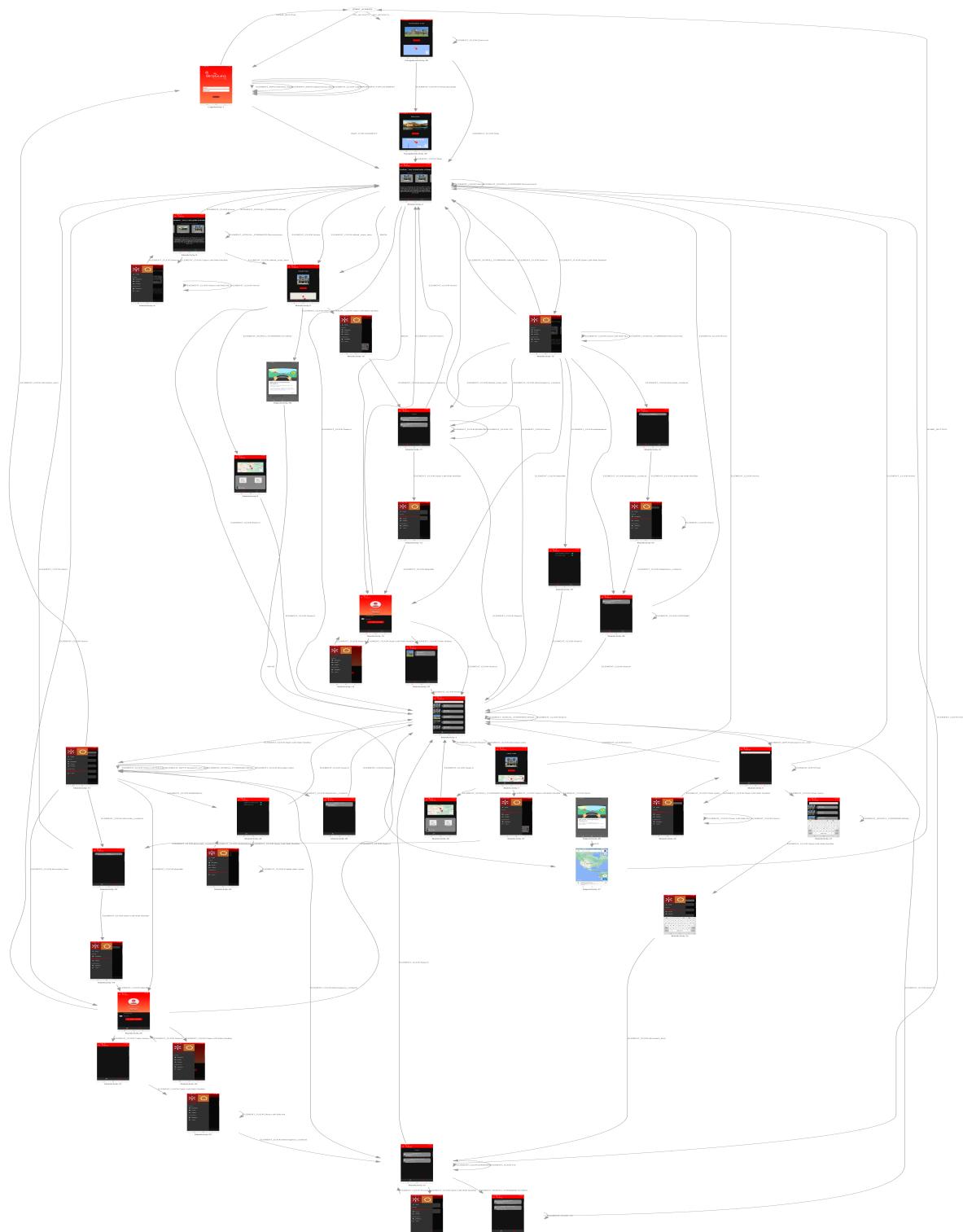
## 7 Conclusão

Neste trabalho prático sobre o desenvolvimento de um guia turístico, foram adquiridos conhecimentos fundamentais para a criação de aplicações móveis híbridas.

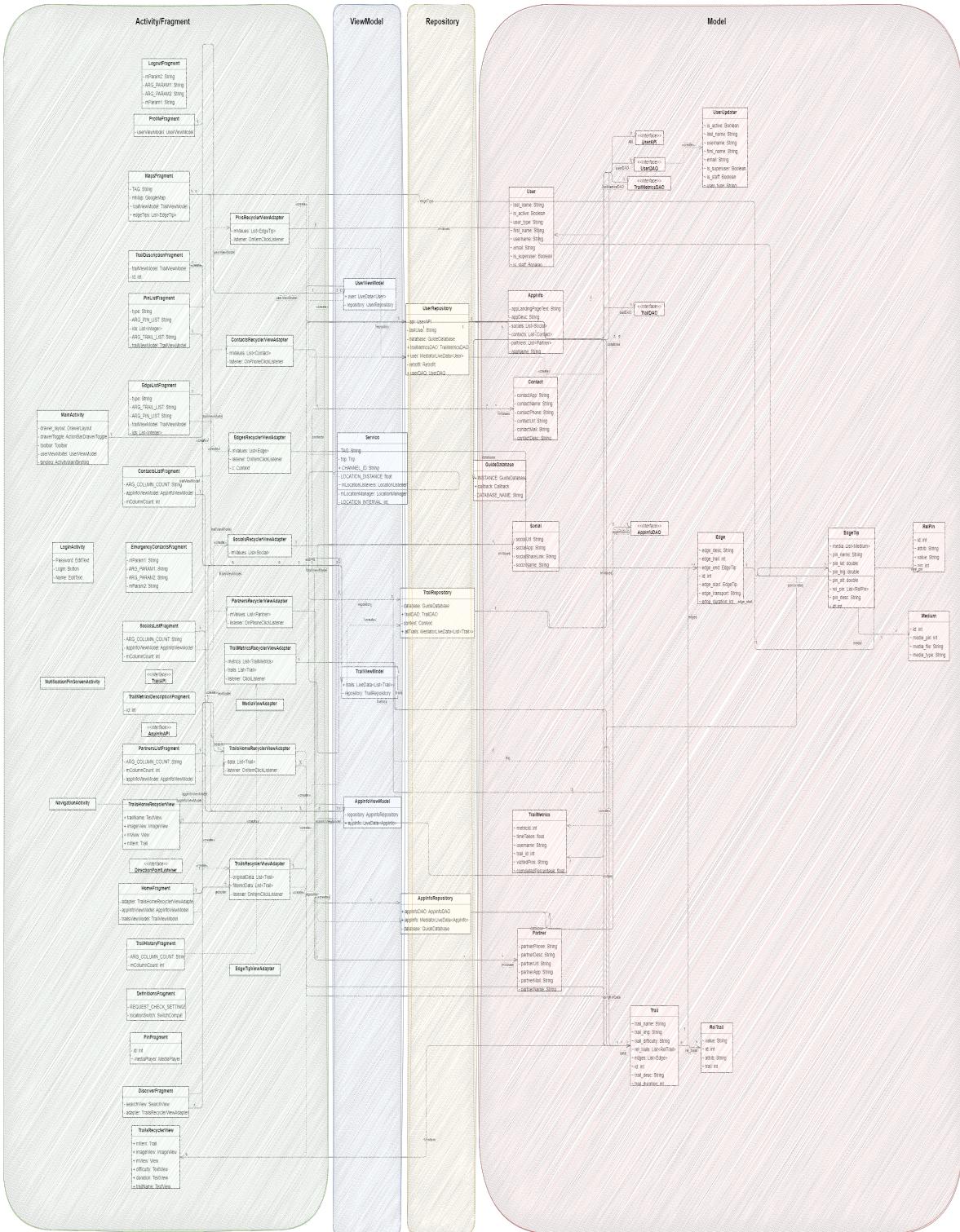
Foi-nos imposta a utilização de uma linguagem nativa do Android para elaborarmos a aplicação, à qual escolhemos a linguagem de programação Java. Além disso, também foi explorada a integração e manipulação de dados do *backend* fornecido e importânci a de utilizar e manipular bibliotecas de referência da indústria, como a Google Maps Platform.

No âmbito do desenvolvimento de um projeto de *software*, utilizamos o Git/GitHub para efetuarmos a gestão colaborativa do projeto, incluindo a automatização da *release* do APK com recurso ao GitHub Actions (embora sem sucesso), de forma a dar-mos importânci a a processos de CI/CD no futuro da nossa vida profissional.

1 Anexos



**Figura 28:** Mapa de navegação GUI



**Figura 29:** Diagrama de classes