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 assets 4

2.1.2 src 4

2.1.3 App.tsx 5

2.1.4 index.js 5

2.1.5 android 5

2.1.6 ios 5

2.1.7 Arquitetura do projeto 5

2.2 Soluções de implementação 6

2.2.1 Integridade dos dados 6

2.2.2 Pedidos à API 6

2.2.3 Funcionamento offline 6

2.2.4 Navegação entre ecrãs 6

2.2.5 Navegação por um roteiro 7

2.2.6 Pedido de localização 7

2.2.7 Notificações 7

2.3 Bibliotecas/dependências utilizadas 7

2.3.1 Axios 7

2.3.2 Moment 7

2.3.3 Notifee 7

2.3.4 React Native Async Storage 8

2.3.5 React Native Encrypted Storage 8

2.3.6 React Native Geolocation Service 8

2.3.7 React Native Maps 8

2.3.8 React Native Permissions 8

2.3.9 React Native Safe Area Context 8

2.3.10 React Native SQLite Storage 8

2.3.11 React Native Vector Icons 8

2.3.12 React Navigation 9

2.4 Padrões de software utilizados 9

2.4.1 Padrões estruturais 9

2.4.2 Padrões comportamentais 9

2.4.3 Modelo da base de dados 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.

Para esta fase, e tendo desenvolvido a aplicação em *Java* com recurso ao *Android Studio* previamente, alguns conhecimentos obtidos serão portabilizados para o desenvolvimento de uma nova aplicação. Esta irá tirar proveito da *framework* React Native, sendo desenvolvido código quase exclusivamente em *JavaScript*. 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 da organização das diretorias de acordo com o que foi dado nas aulas. A separação entre as diretorias é referida de seguida:

2.1.1 assets

Diretoria que guarda todas as imagens e ícones.

2.1.2 src

É o principal componente do projeto que contém a maior parte do código desenvolvido na aplicação.

components

Esta diretoria apresenta componentes variados utilizados ao longo da aplicação. Por exemplo, inclui um 'cartão' para cada roteiro, ou a informação relativa a cada ponto de interesse.

api

Esta diretoria inclui o controlador geral relativo à API da aplicação.

navigation

Nesta diretoria é guardado o navegador da aplicação. No nosso caso, foi implementado um *TabNavigator*.

utils

Apresenta métodos de contexto miscelâneo. Por exemplo, inclui a parte do código relativo às *geofences*, o serviço de localização e as notificações.

screens

Apresenta todos os ecrãs implementados na aplicação.

styles

Apresenta os estilos gerais do projeto.

2.1.3 App.tsx

É o componente principal que inicia a aplicação como um todo.

2.1.4 index.js

É o ponto de entrada da aplicação de acordo com os *standards* da tecnologia *React Native*.

2.1.5 android

Apresenta o código específico à plataforma *Android*.

2.1.6 ios

Apresenta o código específico ao sistema operativo *iOS*.

2.1.7 Arquitetura do projeto

O projeto segue uma arquitetura semelhante a este formato:

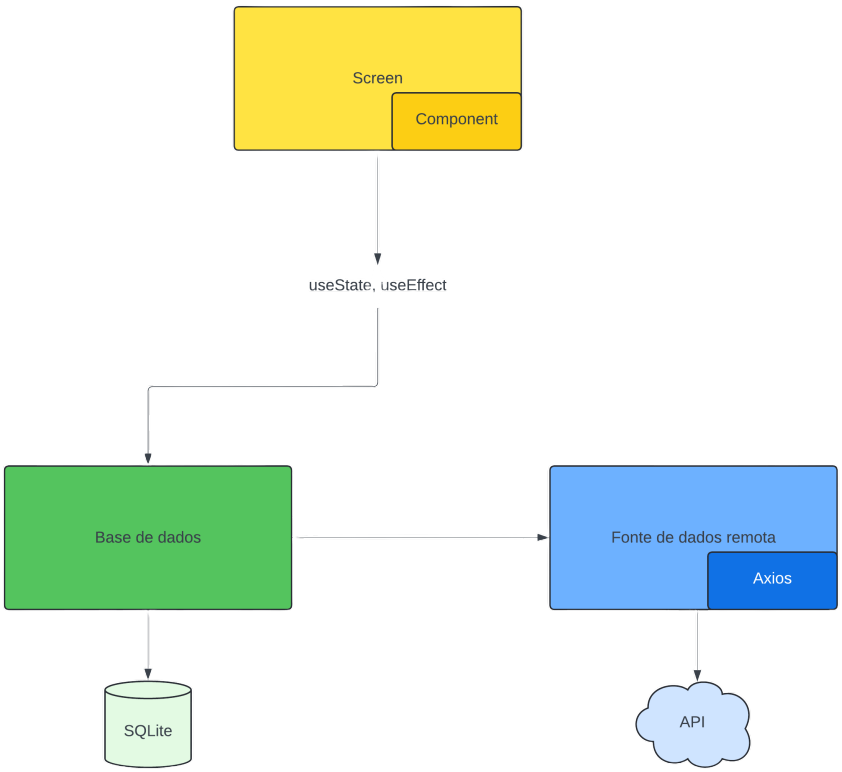

Figure 2.1: Arquitetura geral da aplicação

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 garantir a integridade dos dados na nossa implementação da base de dados em *SQLite*, seguimos diversas práticas recomendadas. Inicialmente, utilizamos a biblioteca *react-native-sqlite-storage* com suporte a *Promises*, o que assegura que todas as operações de banco de dados sejam executadas de forma assíncrona e estável.

A criação de tabelas é feita com restrições que garantem a unicidade dos registos e a consistência das relações entre as tabelas, com campos únicos e não nulos, garantindo, portanto, a inserção de dados duplicados ou vazios, mantendo a integridade dos dados armazenados.

Além disso, todas as operações de criação e manipulação de tabelas são realizadas dentro de transações, o que significa, se alguma operação falhar, todas as mudanças são revertidas, evitando que o banco de dados fique em um estado inconsistente.

2.2.2 Pedidos à API

Para realizar pedidos à API de forma eficiente e responsiva, utilizamos a biblioteca *Axios*. A implementação centraliza todas as interações com a API em uma única classe, "api.js", garantindo organização e facilidade de manutenção do código.

Configurou-se uma instância do *Axios* com uma URL base e um tempo limite para os pedidos efetuados, garantindo consistência e estabilidade nas comunicações HTTP.

2.2.3 Funcionamento offline

Para possibilitar o funcionamento offline, a aplicação utiliza o *SQLite* como base de dados local, garantindo que os dados essenciais são armazenados diretamente no dispositivo do utilizador. Os dados são armazenados de forma estruturada em tabelas definidas, como utilizadores, roteiros, pontos de interesse, conteúdo media, entre outras, conforme necessário pela lógica de negócios da aplicação.

2.2.4 Navegação entre ecrãs

A navegação entre ecrãs na aplicação é gerida através de um *TabNavigator* criado com o *react-navigation*. O *TabNavigator* permite uma transição fluida entre diferentes ecrãs da aplicação, fornecendo uma experiência de utilização intuitiva e organizada.

O *TabNavigator* é configurado no componente *tabNavigator.js*, onde cada aba representa um ecrã diferente da aplicação, como os ecrãs *"Home"*, *"Trails"*, *"Pins"*, *"Maps"*, *"Profile"* e *"Settings"*. Estas abas são definidas utilizando o componente *Tab.Screen* do *react-navigation*. Cada *Tab.Screen* é associado a um componente que representa o conteúdo do ecrã correspondente.

O código faz uso de useContext para obter o tipo de utilizador a partir do AuthContext, permitindo assim condicionar a exibição de certas abas, como a aba "Maps", que só está disponível para utilizadores Premium.

2.2.5 Navegação por um roteiro

A navegação por um roteiro é efetuada a partir da informação recebida da tabela dos *edges*. Esta informação é processada, sendo que são extraídas as coordenadas relativas a cada ponto de interesse. De seguida, através do módulo *Linking* do *React Native*, é fornecido um link da API do Google Maps que redireciona o utilizador que quer começar o roteiro para a aplicação do Google Maps, com o roteiro completo, desde a localização atual do utilizador até ao ponto de interesse final do roteiro, passando por pontos de interesse intermédios se tal se verificar.

2.2.6 Pedido de localização

Antes um utilizador poder sequer receber atualizações acerca da sua localização, através do serviço, este utilizador tem de fornecer permissões de acesso à localização e de publicar notificações por parte do dispositivo. Assumindo que estas permissões são garantidas pelo utilizador, e, quando o utilizador ativa o serviço de localização, é criada uma notificação permanente que indica ao utilizador que a sua localização está a ser monitorizada.

A monitorização da localização é efetuada a partir da biblioteca *react-native-geolocation-service* e concretamente a partir da funcionalidade *Geolocation*. Esta funcionalidade vai buscar a localização do utiizador - latitude e longitude - em intervalos de 10 segundos. Na fase inicial de cada intervalo, o utilizador recebe uma ou mais notificações a indicar que está perto de um ou mais pontos de interesse, se tal se verificar de acordo com o raio de escuta, circular, à volta dos pontos de interesse recebidos pela API.

2.2.7 Notificações

As notificações do nosso projeto são criadas com recurso à biblioteca *Notifee*, que oferece capacidades ricas de criação de notificações diversas. No ficheiro relacionado às notificações, apresentamos 3 funções: uma que cria uma notificação permanente, própria ao serviço de localização; outra que cria uma notificação que indica que o utilizador está perto de um ponto de interesse; e, por fim, outra que remove uma notificação de acordo com o seu identificador, criada com o propósito de remover a notificação do serviço de localização quando este é desativado.

De notar que a notificação permanente e as notificações para os pontos de interesse são criadas em canais com identificadores diferentes, e as notificações para os pontos de interesse apresentam um identificador igual ao identificador do ponto de interesse correspondente.

2.3 Bibliotecas/dependências utilizadas

2.3.1 Axios

Cliente HTTP baseado em *Promises*. É isomórfico e utiliza o módulo HTTP nativo do *Node* do lado do servidor.

2.3.2 Moment

Biblioteca de datas em *JavaScript* para validar, manipular e formatar datas.

2.3.3 Notifee

Biblioteca rica em *features* de notificações *Android* e *iOS* para *React Native*.

2.3.4 React Native Async Storage

Um sistema de armazenamento de dados *'key-value'* assíncrono, desencriptado e persistente para *React Native*.

2.3.5 React Native Encrypted Storage

Uma solução segura de armazenamento para aplicações em *React Native* que permite a desenvolvedores guardar dados sensíveis, como *tokens* de autenticação, *passwords*, entre outras informações confidenciais, apenas no dispositivo do utilizador. Utiliza criptografia para proteger os dados guardados, garantindo que, mesmo que o dispositivo seja comprometido, os dados mantêm-se impossíveis de serem lidos sem as chaves de descodificação.

2.3.6 React Native Geolocation Service

Biblioteca que fornece recursos de geolocalização para aplicações em *React Native*. Permite que desenvolvedores acedam aos dados de localização do dispositivo - que incluem latitude, longitude, altitude, velocidade, direção e muito mais. Este serviço é essencial para aplicações que requerem funcionalidades baseadas na localização, tais como mapas, navegação, monitorização de localização e *geofencing*.

2.3.7 React Native Maps

Biblioteca que fornece um componente de mapa customizável para aplicações *iOS* e *Android*. Permite aos desenvolvedores integrar mapas interativos em aplicações móveis, utiliza SDKs de mapas nativos para oferecer recursos de desenho de mapas de alto desempenho e ricos em recursos. A biblioteca permite uma personalização extensiva, incluindo o estilo, o posicionamento da câmara e o tratamento de vários eventos de mapas.

2.3.8 React Native Permissions

Uma API de permissões unificada para *React Native* para *iOS*, *Android* e *Windows*.

2.3.9 React Native Safe Area Context

Fornece uma API flexível para aceder a informações de inserção da 'área segura' do dispositivo. Isto permite posicionar o seu conteúdo de forma adequada em torno de detalhes, barras de estado, indicadores de início e outros elementos de interface de utilizador.

2.3.10 React Native SQLite Storage

Plugin nativo do *SQLite3* para *Android*, *iOS* e *Windows*.

2.3.11 React Native Vector Icons

Oferece a capacidade de inserir ícones *vector* customizáveis, com caraterísticas de versatilidade, nas aplicações em *React Native*, em componentes tão variados como botões, logótipos, e barras de *tab* ou navegação.

2.3.12 React Navigation

Biblioteca para *React Native* que permite a desenvolvedores criar interfaces navegáveis em aplicações móveis. Fornece um conjunto de componentes e APIs para implementar vários tipos de padrões de navegação, como navegadores *stack*, navegadores *tab*, navegadores *drawer* e muito mais. É altamente personalizável e suporta estruturas de navegação simples e complexas, tornando-o uma escolha versátil para fazer a gestão a navegação de aplicações.

2.4 Padrões de software utilizados

2.4.1 Padrões estruturais

Facade

O padrão *Facade* está presente em todos os *Providers* da aplicação, e é essencial para partilhar dados e funcionalidades entre componentes de forma eficiente e gerir o estado da aplicação de maneira centralizada.

```
function App () {
    const isDarkMode = useColorScheme() === 'dark';
      const theme = isDarkMode ? DarkTheme : DefaultTheme;

    useEffect(() => {
        LogBox.ignoreLogs(["EventEmitter"]);
    }, []);

    return (
        <DbContextProvider>
            <GeofenceProvider>
                <LocationProvider>
                    <AuthProvider>
                        <NavigationContainer theme={theme}>
                            <TabNavigator />
                        </NavigationContainer>
                    </AuthProvider>
                </LocationProvider>
            </GeofenceProvider>
        </DbContextProvider>
    );
  };
```

- *DbContextProvider*: Fornece o contexto do banco de dados *SQLite* para a aplicação.
- *GeofenceProvider* e *LocationProvider*: Fornecem funcionalidades relacionadas à geolocalização e *geofences*.
- *AuthProvider*: Fornece o contexto de autenticação e *tokens* de autenticação para gerir o estado de autenticação do utilizador.

2.4.2 Padrões comportamentais

Observer

Os *Hooks* do *React* seguem o padrão comportamental *Observer*, isto é, permitem a implementação de código reutilizável e modular que podem entrar nos componentes de ciclo de vida e gestão de estado do *React.*

O grupo optou por adotar este módulo, com recurso a funcionalidades como o *useState*, *useEffect*, entre outros, devido à sua simplicidade, permitindo uma sintaxe mais concisa e fácil de entender. Além disso, e como foi referido previamente, este módulo facilita a reutilização de lógica entre componentes, uma vez que permitem extrair e compartilhar funcionalidades complexas de forma mais modular.

Outro benefício significativo dos *Hooks* é a melhor resolução de problemas relacionados ao ciclo de vida, uma vez que proporcionam um controle mais granular sobre o momento de execução de determinadas ações, como efeitos colaterais e atualizações de estado, contribuindo para um código mais claro e menos propenso a erros, além de melhorar a performance geral da aplicação, pois evita renderizações desnecessárias e otimiza o uso de recursos.

Além disso, os *Hooks* são altamente compatíveis com o ecossistema do *React*, o que significa que podem ser integrados facilmente com outras bibliotecas e ferramentas populares dentro da *framework*.

2.4.3 Modelo da base de dados

A imagem abaixo mostra o esquema da nossa base de dados resultante do desenvolvimento do projeto.

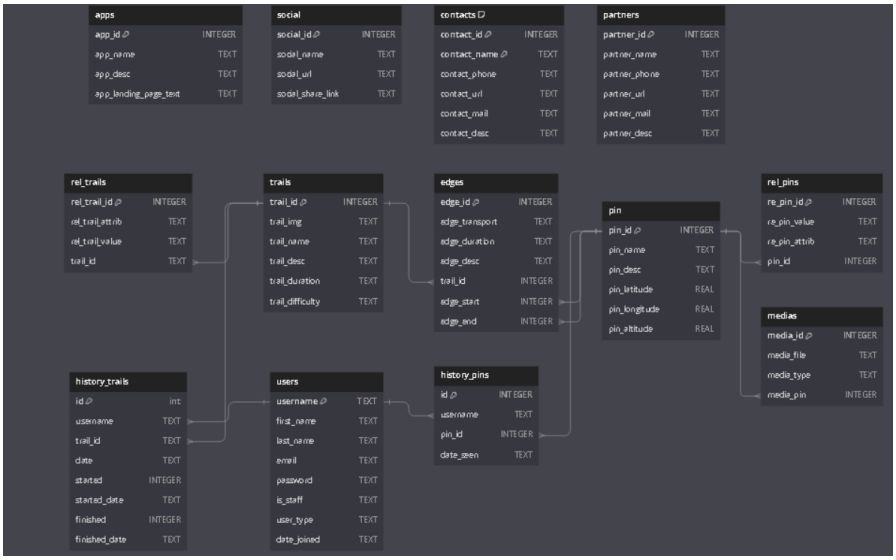

Figure 2.2: Modelo concetual da base de dados

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.

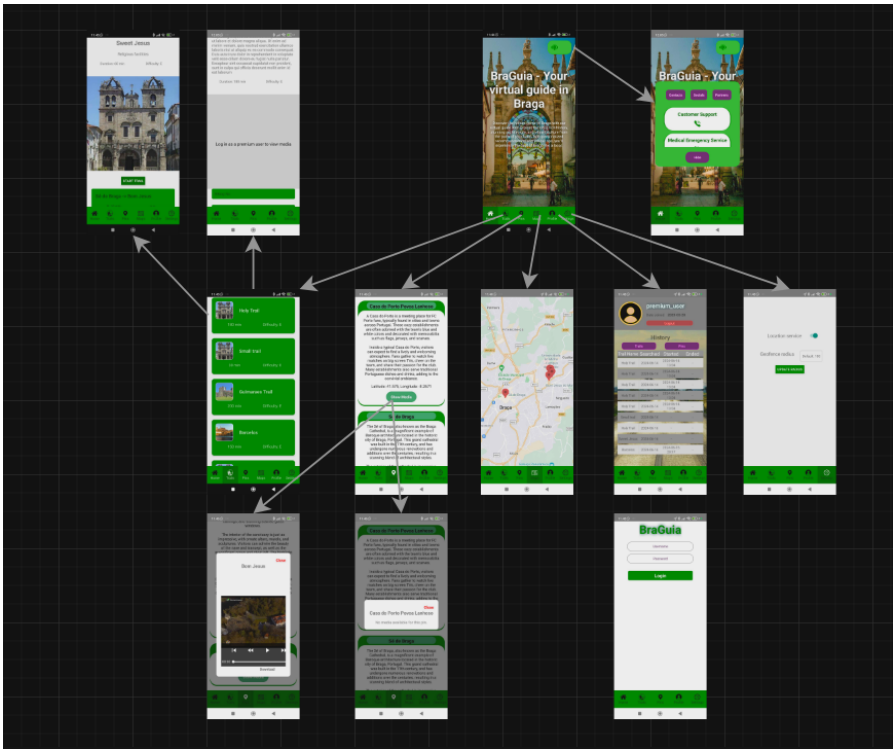

Figure 3.1: Mapa de Navegação

Chapter 4

Funcionalidades

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;
- A aplicação deve possuir um menu com definições que o utilizador pode manipular;

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 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 duas e seguiu a arquitetura recomendada para *React Native*.

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 garantir que a notificação que indica que o utilizador está perto de um ponto de interesse redireciona o utilizador para esse mesmo ponto de interesse, quando clicada. A segunda limitação indica que o projeto não inclui um *preview* do itinerário para cada roteiro específico. De referir que o grupo tentou implementar estas funcionalidades mas sem grande sucesso, devido à biblioteca *notifee* não oferecer suporte robusto para o redirecionamento de notificações quando clicadas (em *Android* utilizariam-se Intents para este efeito); o segundo tópico referido não foi possível de ser realizado devido a limitações nas *WebViews* do React Native.

Outras limitações do projeto incluem a inconsistência do serviço de localização em *background*, sendo que esta funcionalidade proposta no enunciado não ficou 100 por cento implementada. Na secção do menu com definições 'maleáveis', o raio de escuta dos pontos de interesse também apresenta inconsistências.

Também de referir que não efetuamos testes para o sistema operativo iOS, devido ao facto de nenhum dos membros do grupo apresentar um computador com MacOS.

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 navegador Tab, com a opção "Maps". Este mapa foi implementado com recurso à biblioteca *react-native-maps*, com um componente *MapView*, e apresenta 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. Relativamente ao histórico, o grupo decidiu desenvolver esta funcionalidade com o objetivo de adicionar funcionalidades futuras. O histórico presente não mostra apenas os pontos de interesse que encontrou ou os roteiros que um dado utilizador pesquisou, é possível ao mesmo saber quando começou a rota e quando terminou. O propósito seria informar o utilizador de se um dado roteiro foi concluído e o tempo que o mesmo demorou a terminá-lo.

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 ficou inicialmente responsável por tratar de todo o processo de passagem dos roteiros desde a API até a um ecrã que mostra a lista dos roteiros bem como um ecrã que mostra cada roteiro com maior pormenor, passando por criar um *draft* da base de dados e utilizar os *Hooks* para gestão de ciclo de vida de componentes e variáveis.

O aluno pg53944 inicialmente ficou responsável pelas funcionalidades de *login* e *logout* bem como a gestão de sessão do utilizador e a amostragem das informações gerais da aplicação no ecrã, seguindo o processo referido anteriormente relativo à base de dados e gestão de ciclo de vida. O aluno pg52690 demorou a comunicar com qual parte iria responsabilizar-se inicialmente, acabando por ficar responsável pelo registo do histórico dos utilizadores.

Deste modo, cada aluno ia comunicando as funcionalidades que implementou/tentou implementar.

6.2 Eventuais metodologias de controlo de versão utilizadas

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 implementou o navegador *tab*, uma versão inicial da base de dados com recurso a *SQLite*, bem como gestão de ciclo de vida para o ecrã que mostra uma lista responsiva de roteiros e para o ecrã que mostra um único roteiro em maior pormenor. De seguida, trabalhou nas funcionalidades mais relacionadas com

a localização da aplicação, tais como a criação de um mapa do *react-native-maps*, o serviço de localização, as geofences com recurso a este serviço, o redirecionamento para a aplicação do Google Maps para iniciar um roteiro, as notificações relativas ao serviço de localização e às entradas nas geofences.

O aluno pg53944 desenvolveu as funcionalidades de *login*, *logout* e garantiu a qualidade da gestão de sessão através do uso de *cookies* e distinção entre funcionalidades normais ou de utilizadores *premium*, criou o ecrã que mostra a informação geral da aplicação com um botão acessível aos contactos de emergência, criou um ecrã que mostra a informação relativa a todos os pontos de interesse com detalhe, incorporou o conteúdo *media* no projeto, implementou o aviso de ser necessário o Google Maps para usufruir das capabilidades completas da aplicação, implementou também um ecrã de perfil do utilizador, em que algumas informações sobre o mesmo são dispostas, e fez algum trabalho de estética.

O aluno pg52690 implementou o histórico dos roteiros e pontos de interesse visitados por um utilizador, reestruturou os estilos da aplicação num ponto central e também fez algumas melhorias gráficas, nomeadamente no ecrã do *login* perfil do utilizador.

Chapter 7

Conclusão

Tratando-se de um guia turístico para uma aplicação em *React Native*, 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, como, por exemplo, a estrutura do projeto em *React Native* recomendada. No entanto, seria interessante abordar tópicos em que a aplicação não ficou polida, como garantir a consistência da funcionalidade do serviço de localização em segundo plano, bem como garantir que a notificação que indica que o utilizador está perto de um ponto de interesse redireciona o utilizador para esse mesmo ponto de interesse, quando clicada; e incluir um *preview* do itinerário para cada roteiro específico.

Gostávamos, ainda, de ter incluído alguns testes, bem como testar a aplicação para iOS.

Acreditamos que cumprimos com o nosso trabalho e que temos um protótipo bastante sólido (para *React Native*) se fosse eventualmente incluído no mercado de trabalho.

Bibliography

[1] Axios. *Promise-based HTTP Client for Node.js*. URL: https://axios-http.com/docs/intro.

[2] GitHub. *React Native Maps*. URL: https://github.com/react-native-maps/react-native-maps/blob/master/docs/installation.md.

[3] Moment. *Parse, validate, manipulate, and display dates and times in JavaScript*. URL: https://momentjs.com/.

[4] React Native. *Debugging Basics*. URL: https://reactnative.dev/docs/debugging.

[5] React Native. *Direct Manipulation*. URL: https://reactnative.dev/docs/direct-manipulation.

[6] React Native. *Learn the Basics*. URL: https://reactnative.dev/docs/tutorial.

[7] React Native. *Networking*. URL: https://reactnative.dev/docs/network.

[8] React Native. *Props*. URL: https://reactnative.dev/docs/props.

[9] React Native. *State*. URL: https://reactnative.dev/docs/state.

[10] React Native. *Style*. URL: https://reactnative.dev/docs/style.

[11] React Native. *Using a ScrollView*. URL: https://reactnative.dev/docs/using-a-scrollview.

[12] React Navigation. *Routing and navigation for Expo and React Native apps*. URL: https://reactnavigation.org/.

[13] Notifee. *A feature rich notification library for React Native*. URL: https://notifee.app/.

[14] NPM. *React Native Async Storage*. URL: https://www.npmjs.com/package/@react-native-async-storage/async-storage.

[15] NPM. *React Native Encrypted Storage*. URL: https://www.npmjs.com/package/react-native-encrypted-storage.

[16] NPM. *React Native Geolocation Service*. URL: https://www.npmjs.com/package/react-native-geolocation-service.

[17] NPM. *React Native Permissions*. URL: https://www.npmjs.com/package/react-native-permissions/v/2.2.0.

[18] NPM. *React Native Safe Area Context*. URL: https://www.npmjs.com/package/react-native-safe-area-context.

[19] NPM. *React Native SQLite Storage*. URL: https://www.npmjs.com/package/react-native-sqlite-storage.

Chapter 8

Anexos

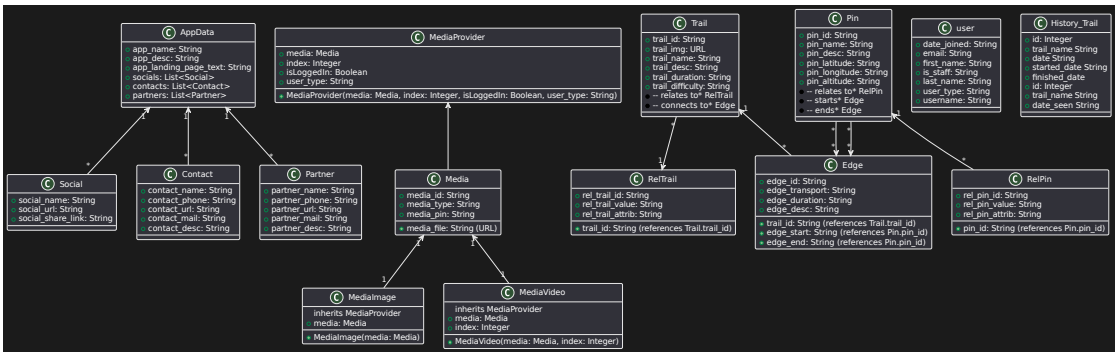

Figure 8.1: Diagrama de classes