Universidade do Minho

Mestrado em Engenharia Informática

Tópicos de Desenvolvimento de Software

Trabalho Prático - **Parte II**

Guia Turístico: BraGuia

Grupo:

Catarina Gonçalves (PG50180)

Francisco Toldy (PG50379)

Joana Alves (PG50457)

24 de junho de 2023

Conteúdo

1 Introdução **2**

2 Detalhes da Implementação **3**
2.1 Estrutura do Projeto . . . 3
2.2 Soluções de Implementação . . . 5
2.2.1 Login . . . 5
2.2.2 Home . . . 5
2.2.3 Trails . . . 5
2.2.4 Trail . . . 5
2.2.5 Pin . . . 6
2.2.6 PinMedia . . . 6
2.2.7 Profile . . . 6
2.2.8 Trails History . . . 6
2.2.9 Points History . . . 6
2.2.10 Contacts . . . 7
2.3 Bibliotecas/Dependências Utilizadas . . . 7

3 Mapa de Navegação de GUI **8**

4 Funcionalidades **10**
4.1 Funcionalidades Utilizador *Standard* . . . 10
4.2 Funcionalidades Utilizador *Premium* . . . 10

5 Testagem **12**

6 Discussão de Resultados **13**
6.1 Trabalho Realizado . . . 13
6.2 Limitações . . . 13
6.3 Funcionalidades Extra . . . 13

7 Gestão do Projeto **14**
7.1 Gestão e Distribuição de Trabalho . . . 14
7.2 Metodologias de Controlo de Versão Utilizadas . . . 14
7.3 Reflexão sobre Performance Individual . . . 14

8 Conclusão **15**

1 Introdução

Este relatório foi desenvolvido no âmbito da unidade curricular de Tópicos de Desenvolvimento de *Software* do Mestrado em Engenharia Informática, tendo como objetivo o desenvolvimento de um guia turístico, sob a forma de uma aplicação móvel híbrida.

De acordo com o enunciado do trabalho prático, a **aplicação**, denominada "BraGuia", é uma aplicação para orientação turística que oferece roteiros turísticos aos seus utilizadores, oferecendo funcionalidades de localização e navegação geográfica, reprodução de *media* acerca de pontos de interesse, entre outros. Para além disto, de forma a obter o conteúdo para mostrar ao utilizador, a aplicação consome informação de um *backend* desenvolvido pelo docente para servir especificamente este trabalho prático.

Desta forma, recorrendo ao conhecimento adquirido ao longo do semestre relativo a desenvolvimento nativo e multi-plataforma, o projeto foi separado em duas partes distintas: **parte I** (desenvolvimento com recurso apenas a tecnologia nativa *Android*) e **parte II** (desenvolvimento multi-plataforma recorrendo à tecnologia *React Native*). Assim, o objeto de estudo deste relatório trata-se da **parte II**, onde irão ser apresentados os pormenores relativos à arquitetura, comportamento e decisões tomadas pelo grupo no desenvolvimento da mesma.

2 Detalhes da Implementação

2.1 Estrutura do Projeto

Na criação de um projeto *react-native* eram dadas as opções da utilização da ferramenta Expo e **CLI**, sendo esta última a utilizada pelo grupo. Esta decisão baseou-se na pura razão de que o Expo não funcionava nos emuladores de todos os elementos do grupo, enquanto que o CLI sim.

A estruturação de um projeto *react-native* baseado nesta ferramenta está dividido em algumas diretorias que incluem as diretorias *android* e *iOS*, no entanto, estas não foram alteradas, foi apenas adicionado conteúdo à diretoria *root* do projeto. Assim, foram criadas as seguintes diretorias:

- **Components:** Esta diretoria foi utilizada para agrupar todos os sub componentes que eram utilizados nos ecrãs principais
- **Screens:** Esta diretoria contém vários ficheiros, cada um correspondente a um ecrã da aplicação
- **Imagens:** É utilizada para agrupar todas as imagens utilizadas pelo projeto
- **Features:** Agrupa todos os *reducers* utilizados pelo **redux**
- **Store:** Possui unicamente o ficheiro que contém a configuração da *store* que vai servir de base para todo o uso do *redux*

Após várias tentativas, não foi possível implementar uma base de dados ou o *redux persist*, daí que as estruturas de dados que existem são unicamente aquelas que existem no uso do *redux*, bastante semelhantes às implementadas na fase 1.

Assim sendo, os tipos de dados guardados estão distribuídos por 4 *reducers*: *reducer* da App, do Histórico, dos *Trails* e do *User*.

- **App Reducer:** Os dados relativos à app são guardados da seguinte forma, notando que os *socials*, *contacts* e *partners* são inseridos exatamente como são obtidos através do *fetch* após a transformação em JSON.

```
app: {
    app_name: ***,
    app_desc: ***,
    app_landing_page_text: ***,
}
socials: [ *** ]
contacts: [ *** ]
partners: [ *** ]
```

- **Trail Reducer:** Relativamente aos *Trails*, estes também são maioritariamente guardados como são obtidos do fetch, no entanto com alguns campos adicionais:

```
trails: [
    {
        ***,
```

```
        state: "running"/"toStart",
        start_date: ***
    }
]
points: [ *** ]
```

- **History Reducer:** Relativamente aos históricos, estes são criados unicamente através da aplicação, e daí têm o seguinte formato:

```
trail_id_count: ***
pin_id_count: ***
trails: [{
        id: ***,
        trail_id: ***,
        nrPins: ***,
        last_pin_name: ***,
        last_pin_id: ***,
        state: "completed"/"onHold"/"canceled"
        pinsDone: [],
        start_date: ***,
    }]
points: [{
    id: ***,
    pin_id: ***
}]
```

  É de notar ainda, que cada histórico de um *trail* vai ter associado uma lista de *pins* que consiste numa lista de nomes dos pontos de interesse que adicionou ao histórico.

- **User Reducer:** Finalmente, a informação relativa ao *user* e aos *cookies* vai ser guardada no seguinte formato:

```
user: {
    ***
}
isPremium: true/false,
coookies:  {
    token: ***,
    session: ***
}
```

2.2 Soluções de Implementação

Nesta secção serão então apresentadas as soluções de implementação do *react native*, endereçando especificamente cada um dos ecrãs que existem na aplicação:

2.2.1 Login

Este ecrã tem como objetivo permitir ao utilizador *Premium* e *Standard* fazer a sua autenticação. Isto é conseguido ao executar um pedido *fetch* cujo *body* é um JSON com os dados introduzidos pelo utilizador. A resposta é processada, sendo armazenadas as *cookies* importantes e executada a mudança para a página inicial.

2.2.2 Home

Este ecrã representa a página inicial da aplicação, onde, através do *redux*, obtemos a informação da mesma provinda do *backend*. No entanto, caso essa informação ainda não esteja presente no *redux*, é efetuado um *fetch* à API para assim a obter. Para além disto, é também neste ecrã que é realizado o primeiro *fetch* da informação do utilizador, sendo, de seguida, armazenada no *redux*.

2.2.3 Trails

De modo a implementar a apresentação de todos os *trails* existentes, foi obtido através do *redux* os *trails*, juntamente com a utilização da *FlatList* para permitir uma listagem mais eficiente e menos pesada no servidor web.

2.2.4 Trail

Este ecrã é dedicado a apresentar ao utilizador toda a informação relevante sobre o *trail* que escolheu. Consoante o tipo de utilizador, a aplicação recorre a *conditional rendering* para mostrar ou não certos componentes da página. Caso o utilizador seja *Premium*, será mostrado ao mesmo os botões para iniciar e parar o roteiro, seguido da *thumbnail* do trilho, seguido de informação básica do roteiro. Depois, é criado, recorrendo à API do *Google Maps*, um pequeno mapa do centro do ecrã com os vários marcadores dos pontos do roteiro bem como uma *Polyline* a demonstrar o percurso que o utilizador irá fazer. Por fim, temos a listagem dos vários pontos do roteiro, cada elemento com a funcionalidade de redirecionar para a página do ponto de interesse. O utilizador *Standard* não tem acesso às funcionalidades relativas à navegação, ou seja o mapa dentro da aplicação e a possibilidade de iniciar e parar roteiros.

Relativamente ao início e fim do roteiro, o utilizador tem a possibilidade de, após início do roteiro, adicionar cada ponto visitado ao histórico visitando a página do mesmo. De forma a cobrir possíveis esquecimentos devido ao serviço de *foreground* não enviar notificações com base na localização, no momento em que o utilizador para o roteiro, é apresentado um menu onde este pode adicionar os pontos do roteiro que não tenham sido até esse momento adicionados ao histórico.

2.2.5 Pin

Esta página é responsável por apresentar toda a informação de um dado ponto de interesse. Assim, recebe como *props*, tanto o ponto de interesse, como o *id* do *trail* a que pertence, sendo este último necessário para marcar o ponto como visitado e adicionar o mesmo ao histórico.

Para além disto, esta página recorre a *conditional rendering*, de acordo com o tipo de utilizador autenticado, para apenas fazer *display* aos elementos corretos. Assim, no caso do utilizador *standard*, temos apenas a visualização da *thumbnail* do ponto, assim como o seu nome, descrição e propriedades. Já no caso do utilizador *premium*, para além das referidas, possui também a possibilidade de aceder à página de *media* do ponto (caso este possua algum tipo de *media*) e, por último, pode também adicionar o mesmo ao histórico.

Relativamente ao histórico, é utilizado o *redux* para adicionar o ponto ao histórico de pontos visitados, sendo fornecido o identificador do ponto, o seu nome e o *trail* a que corresponde.

2.2.6 PinMedia

Esta página é responsável por fazer *display* a todos os elementos de *media* de um determinado ponto de interesse. Desta forma, recebe como *props* a lista de *medias* do ponto, e, para cada elemento da mesma, constrói o *template* necessário através do componente auxiliar **MediaListSection**. Desta forma, a *media* é apresentada sequencialmente, possuindo, no final da página, um botão para a funcionalidade de *download* de toda a *media*.

2.2.7 Profile

Esta página é responsável por apresentar toda a informação relativa ao utilizador autenticado. Para isso, recorre ao *redux* para obter a informação do mesmo, utilizando *conditional rendering* para diferenciar os dois tipos de utilizadores. No caso do *standard*, são apresentados o seu nome completo, nome de utilizador, email (caso possua) e o seu tipo de utilizador. No caso do utilizador *premium*, para além das funcionalidades antes referidas, também apresenta dois botões de acesso aos dois tipos de histórico: *trails* e pontos de interesse visitados.

Para além disto, também possui um botão de *log out*, permitindo a correta desautenticação do utilizador e redirecionamento do mesmo para a página de *log in*.

2.2.8 Trails History

De modo a implementar a apresentação de todos os *trails* adicionados ao histórico, foi obtido através do *redux* estes mesmos *trails*, juntamente com a utilização da *FlatList* para permitir uma listagem mais eficiente. Além disso, é permitido também aceder à informação do ponto de interesse ao pressionar no ponto desejado. Um extra definido, foi a apresentação de estado do histórico de cada *trail*, dependendo se este foi iniciado/completado/cancelado, sendo este último apenas possível quando o utilizado tenta finalizar o *trail* sem ter passado por pelo menos metade dos pontos do trilho, incluindo o último ponto.

2.2.9 Points History

De modo a implementar a apresentação de todos os pontos adicionados ao histórico, foi obtido através do *redux* os estes mesmos pontos, juntamente com a utilização da *FlatList* para permitir uma listagem mais eficiente. Além disso, é permitido também aceder à informação do ponto de interesse ao pressionar o ponto desejado.

2.2.10 Contacts

De modo a implementar a apresentação de todos os contactos existentes, foi obtido através do *redux* os contactos, juntamente com a utilização da *FlatList* para permitir uma listagem mais eficiente e menos pesada no servidor web.

2.3 Bibliotecas/Dependências Utilizadas

De modo a implementar todas as funcionalidades requisitadas foi necessário utilizar ferramentas que precisam de ser instaladas e posteriormente importadas como dependências. Desta forma, apresentamos as bibliotecas e dependências principais utilizadas:

- **Redux:** utilizado para a gestão do estado global da aplicação.
- **@react-navigation:**
  - **bottom-tabs:** utilizado para a criação e gestão da *bottom navigation bar*.
  - **stack:** utilizado para gestão de navegação entre ecrãs que não pertenciam à barra de navegação.
- **@supersami/rn-foreground-service:** Utilizada para possibilitar a criação de um serviço *foreground* no momento em que um roteiro é iniciado. Possui várias funcionalidades de manipulação do serviço criado.
- **react-native:**
  - **fs:** utilizado para auxiliar no *download* dos ficheiros *media* dos pontos de interesse, ao fornecer acesso ao sistema de ficheiros do dispositivo.
  - **geolocation-service:** utilizada para obter a localização do utilizador.
  - **maps:** permite a criação de um MapView na página do Trail. Além da criação do mapa permite também adicionar Markers e a Polyline descritiva da rota a percorrer.
  - **sound:** utilizado para lidar com a gestão de áudios dos pontos de interesse.
  - **video:** utilizado para lidar com a gestão de vídeos dos pontos de interesse.
  - **vector-icons:** utilizado para apresentar a mesma *font* de ícones de modo a manter a consistência na aplicação.

3 Mapa de Navegação de GUI

Nesta secção serão apresentados dois mapas de navegação da aplicação, um para cada tipo de utilizador. Os mapas demonstram as opções de navegação que o utilizador *Premium* e o utilizador *Standard* podem recorrer, sendo que as indicações das opções de navegação resultantes da *nav bar* foram apenas demonstradas no ecrã principal para reduzir a redundância. Os mapas também estarão disponibilizados nos anexos:

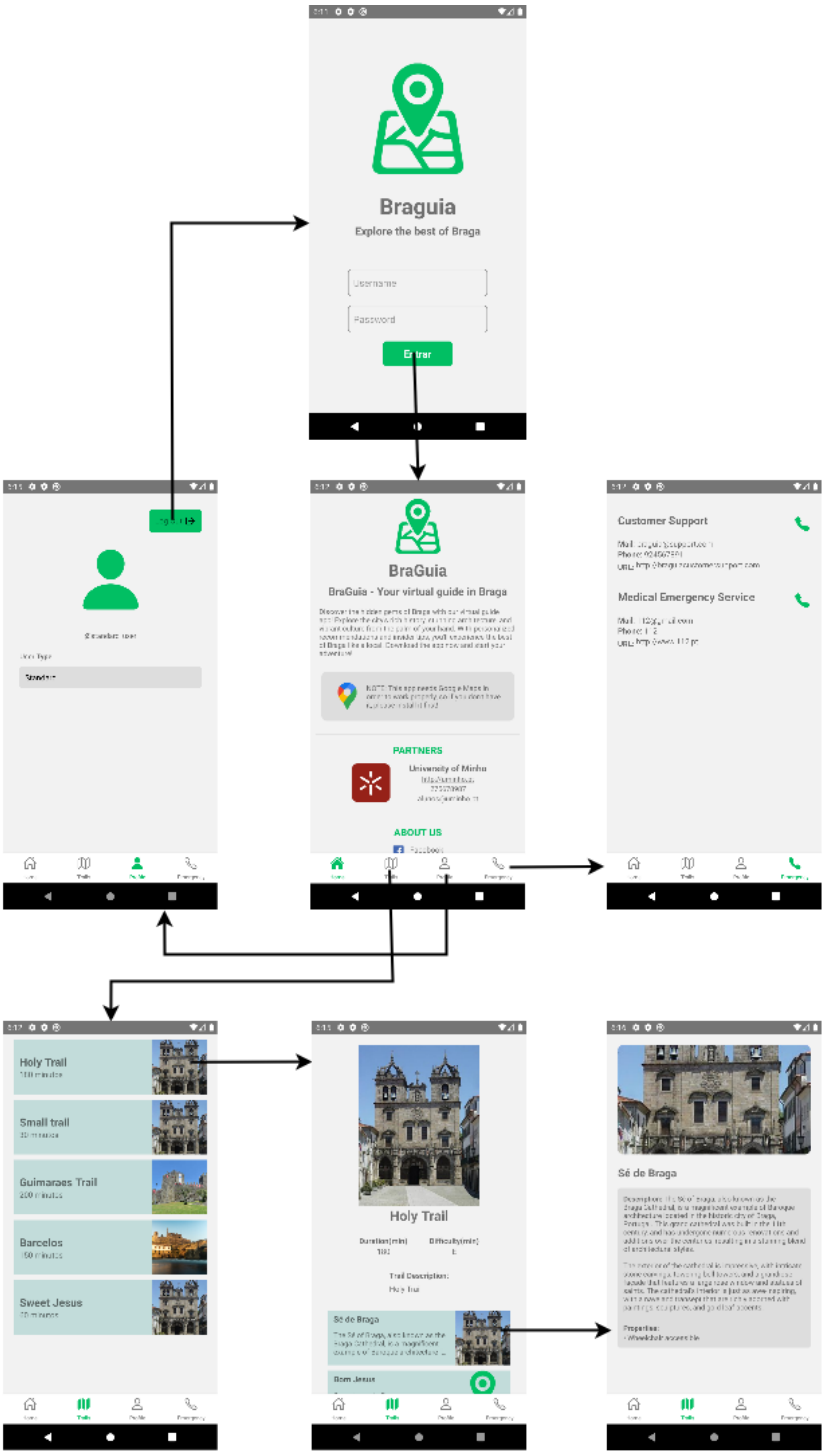

Figura 1: Mapa de Navegação utilizador *Standard*

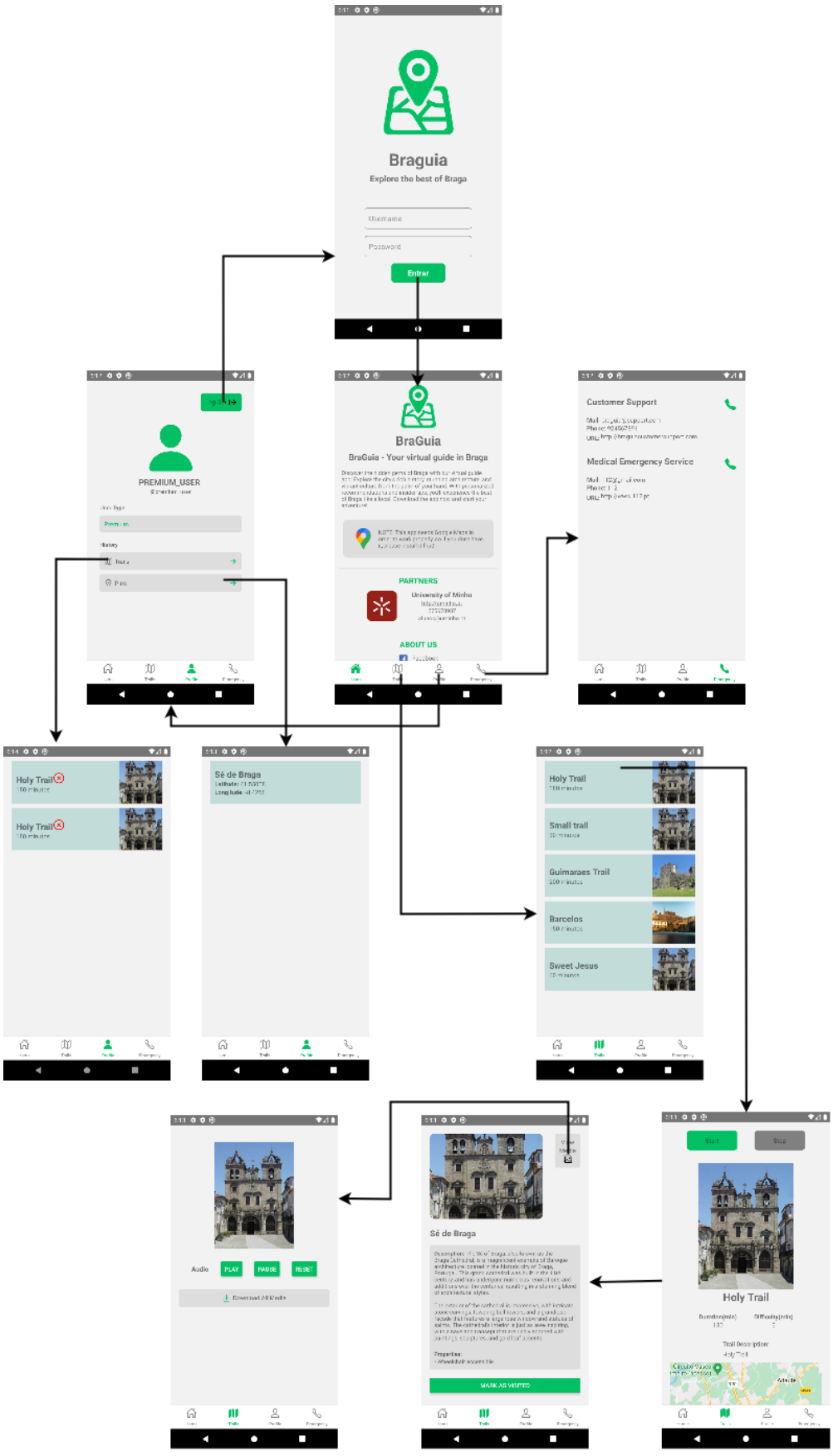

Figura 2: Mapa de Navegação utilizador *Premium*

4 Funcionalidades

Nesta secção iremos apresentar as diversas funcionalidades da aplicação, sendo estas funcionalidades obrigatórias provindas dos requisitos do enunciado. Desta forma, começamos pelas funcionalidades específicas de um utilizador *standard*, notando que o utilizador *premium* é uma extensão deste, isto é, herda as suas funcionalidades.

4.1 Funcionalidades Utilizador *Standard*

Para os utilizadores *standard*, isto é, os utilizadores com **menos** privilégios na aplicação, são oferecidas as seguintes funcionalidades, notando que todas as funcionalidades exceto a autenticação são apenas acessíveis a utilizadores previamente **autenticados**:

- **Autenticação:** Os utilizadores poderão realizar a sua autenticação, com os seus dados correspondentes.
- **Informações da Aplicação:** Na página inicial, qualquer utilizador irá poder visualizar toda a informação da aplicação.
- **Listagem de *Trails*:** Todos os utilizadores têm acesso a uma listagem de todos os *trails* do sistema, incluindo o seu nome, imagem representativa e duração esperada (em minutos).
- **Informação de um *Trail*:** Todos os utilizadores conseguem visualizar a informação de um *trail*, a sua imagem representativa, nome, duração, dificuldade, descrição e pontos de interesse do mesmo.
- **Informação de um Ponto de Interesse:** Todos os utilizadores, através da página de um *trail*, conseguem aceder à página de um ponto de interesse em específico, onde conseguem visualizar a sua imagem (caso a possua), nome, descrição e propriedades.
- **Informações de Perfil:** Todos os utilizadores têm acesso à informação do seu perfil, contendo um atalho para executar o *log out*.
- **Contactos de Emergência:** Todos os utilizadores têm acesso a uma página com todos os contactos de emergência da aplicação.

4.2 Funcionalidades Utilizador *Premium*

Para os utilizadores *premium*, e lembrando que estes são uma extensão dos utilizadores *standard*, a aplicação oferece as seguintes funcionalidades:

- **Informação de um *Trail*:** Para além de todas as informações disponíveis referidas acima, a aplicação adiciona a apresentação interativa de todo o percurso do *trail*, assim como a possibilidade de iniciar e parar esse mesmo percurso, obtendo estatísticas sobre o mesmo, nomeadamente o tempo decorrido e o número de locais visitados.
- **Informação de um Ponto de Interesse:** Para além das referidas acima, para um utilizador *premium* é ainda adicionada a funcionalidade de visualização de *media* do ponto de interesse e possibilidade de marcar o mesmo como visitado.

- **Download Media:** Na página de *media* de um ponto de interesse, os utilizadores *premium* têm a possibilidade de fazer *download* da mesma, permitindo uma melhor experiência em uso *offline*.
- **Histórico:** Os utilizadores *premium* têm acesso ao seu histórico de *trails* (sejam estes completados com sucesso ou não) e de pontos de interesse visitados.

5 Testagem

Todo o processo de desenvolvimento foi suportado por testagem da aplicação em emuladores via *Android Studio*. Para isso, foram utilizadas duas configurações:

- Google Pixel 3 - API 29
- Google Pixel 5 - API 29

Infelizmente não foi possível ao grupo efetuar quaisquer testes em ambiente *iOS* devido ao facto de nenhum membro possuir nem *iPhone* nem um computador *Apple* para poder suportar os testes utilizando *XCode*.

Além disso, foi utilizado *ESLint* para fazer uma análise estática de todo o código, de forma a poder ser feita uma redução dos erros que surgiram na avaliação. Em conjunção com o *ESLint*, foi também utilizado o *Prettier* de forma a formatar o código automaticamente para melhor legibilidade de acordo com regras previamente estabelecidas pela ferramenta.

6 Discussão de Resultados

6.1 Trabalho Realizado

Concluído o período de desenvolvimento desta versão da aplicação BraGuia, verificamos uma implementação quase total de todas as funcionalidades requisitadas. A aplicação consegue proporcionar as várias funcionalidades necessárias para conseguir oferecer uma experiência completa, tanto a utilizadores *premium* como *standard*, com esforços feitos para reduzir o efeito negativo das limitações que serão mencionadas na secção seguinte. Além do nível funcional, foi mantido o mais possível o design intuitivo criado na versão da aplicação desenvolvida na fase anterior, com algumas mudanças positivas em pontos que o grupo achou possível e/ou necessário.

6.2 Limitações

Como mencionado anteriormente na estruturação do projeto, não foi possível a implementação da base de dados nem do *redux persist*, daí resultando numa falta de persistência de dados entre instâncias da aplicação.

Relativamente ao serviço de notificações com base na localização, este não foi implementado. Apesar de ser possível a criação do serviço de *Foreground* e tendo a aplicação a capacidade de fazer verificações periódicas da localização do utilizador, estas verificações não continuavam com a aplicação minimizada, havendo um *timeout* depois de um determinado número de segundos. No entanto, foi criada a notificação *foreground* no momento em que o utilizador inicia o roteiro, de forma a servir de lembrete ao utilizador que a aplicação está ativa e está um roteiro em curso, incentivando o mesmo a aceder manualmente a cada página dos pontos de interesse do roteiro.

6.3 Funcionalidades Extra

Ao longo do desenvolvimento do projeto, para além das funcionalidades obrigatórias, o grupo decidiu inovar alguns aspetos do mesmo ao disponibilizar aos utilizadores funcionalidades extra. Deste modo, passamos a enumerar as mesmas:

- Apresentação do **estado** (iniciado/cancelado/finalizado) de cada *trail* na página de Histórico.

7 Gestão do Projeto

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

7.1 Gestão e Distribuição de Trabalho

A gestão e distribuição do trabalho realizado foi bastante uniforme ao longo do desenvolvimento, que envolveu igualmente todos os elementos do grupo através de métodos de distribuição de tarefas e responsabilidades a cada elemento.

7.2 Metodologias de Controlo de Versão Utilizadas

Nesta segunda fase do trabalho prático, contrariamente à anterior, o grupo implementou uma metodologia de desenvolvimento **feature branch**, o que significa que, para cada funcionalidade ou mudança no código, era criada uma *branch* específica para o mesmo, sofrendo um *merge* para a *branch* principal aquando do término do desenvolvimento da mesma.

Para além disto, em alguns casos foram criados *pull requests* com essas mesmas *branches*, tendo como objetivo os elementos do grupo realizarem *code reviews* dos mesmos.

7.3 Reflexão sobre Performance Individual

Ao longo do desenvolvimento do projeto todos os elementos tiveram uma participação ativa tanto na idealização como no desenvolvimento, permitindo um rápido desenvolvimento sem problemas de comunicação. Assim, todos os elementos estariam com uma avaliação igual em relação à sua performance.

8 Conclusão

Após a realização desta segunda parte do projeto, o grupo encontra-se razoavelmente satisfeito com o trabalho desenvolvido, tendo sido alcançados a maioria dos objetivos estabelecidos quer pelo docente, como pelo próprio grupo.

Dado que nenhum membro do grupo tinha experiência prévia em desenvolvimento *mobile cross-platform*, nomeadamente na tecnologia *React Native*, foi, tal como na primeira parte, um processo de desenvolvimento com as suas dificuldades devido à novidade que era trabalhar neste ambiente. Consideramos que isso, tal como era de esperar, atrasou partes do desenvolvimento, resultando numa entrega final com algumas limitações.

Em suma, a segunda fase deste trabalho serviu para aprofundarmos o conhecimento em relação a desenvolvimento *cross-platform* e utilização da tecnologia *React Native* para o efeito.