

UMinho

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

TRABALHO PRÁTICO: PARTE 2

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

Braga, 14 de junho de 2023

Conteúdo

1	Introdução	2
2	Detalhes de implementação	2
2.1	Estrutura do projeto	2
2.1.1	actions	2
2.1.2	atoms	2
2.1.3	backgroundServices	2
2.1.4	components	3
2.1.5	screens	3
2.1.6	state	3
2.1.7	theme	3
2.2	Soluções de implementação	4
2.3	Bibliotecas/dependências utilizadas	5
2.4	Padrões de software utilizados	6
2.4.1	Provider	6
2.4.2	Padrão de Hooks	6
3	Mapa de navegação de GUI	6
4	Funcionalidades	7
5	Discussão de resultados	25
5.1	Trabalho realizado	25
5.2	Limitações	25
5.3	Testes	26
5.3.1	Software Testing	26
5.3.2	Software Analysis	27
5.4	Funcionalidades extra	28
6	Percentagem de código nativo utilizado	28
7	Gestão de projeto	30
7.1	Gestão e Distribuição de trabalho	30
8	Conclusão	30

1 Introdução

O presente relatório surge no âmbito da segunda 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 cross-platform, implicando o uso de React Native.

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

Ao desenvolver uma aplicação React Native, é crucial organizar a base de código de forma eficiente para garantir sua manutenção e escalabilidade. Além dos arquivos App.js e Wrappe-dApp.js (separados devido à lógica do Redux), a nossa implementação inclui o código nativo para iOS e Android, a fim de adicionar detalhes específicos para cada plataforma (no nosso caso, o Firebase). Também temos uma pasta chamada "assets" para armazenar os arquivos de multimédia do projeto, além de vários ficheiros de configuração de scripts. No entanto, o aspecto principal em que a estrutura pode variar entre projetos React Native é a pasta "source"(src), na qual separamos os diferentes aspectos da aplicação em diretórias lógicas. Vamos analisar mais de perto cada diretória e o seu propósito.

2.1.1 actions

A diretoria "actions" contém ficheiros que definem várias ações ou eventos que podem ocorrer dentro da sua aplicação. Por exemplo, "appData.js" e "user.js" podem definir ações relacionadas à gestão dos dados da aplicação e ações específicas do utilizador, respectivamente. Ao separar as ações em ficheiros dedicados, torna-se mais fácil geri-los e manter uma clara separação de responsabilidades. Esta é uma das pastas propostas para a utilização da biblioteca Redux.

2.1.2 atoms

A diretoria "atoms" é dedicada ao armazenamento de componentes atómicos reutilizáveis. Esses componentes geralmente são pequenos, autocontidos e representam elementos básicos da interface do utilizador. Apesar do grupo ter criado esta diretoria e ter feito um "Button.js", o grupo não explorou muito a criação deste tipo de componentes.

2.1.3 backgroundServices

A diretoria "backgroundServices" contém ficheiros relacionados com serviços em segundo plano ou utilitários. Neste projeto incluímos o arquivo "LocationTracker.js", que lida com a localização do utilizador em segundo plano.

2.1.4 components

A diretoria "components" contém componentes de interface do utilizador reutilizáveis que são usados em várias telas ou secções da aplicação. Cada componente dentro desta diretoria representa um elemento ou funcionalidade específica da interface do utilizador. Por exemplo, "BottomBar.js", "SearchBar.js" e "ToggleButton.js" são componentes reutilizáveis usados em várias telas. Ter um local central para estes componentes partilhados promove a reutilização de código e facilita a manutenção da consistência no design da aplicação.

2.1.5 screens

A diretoria "screens" consiste em componentes de tela individuais que representam diferentes visualizações ou telas da aplicação. Cada ficheiro dentro desta diretoria corresponde a uma tela específica, como "Home.js", "Profile.js" ou "Trails.js". Organizar as telas numa diretoria dedicada permitiu localizar os componentes de navegação de forma mais fácil.

2.1.6 state

Esta diretoria possui os restantes componentes relacionadas com a biblioteca Redux, nomeadamente os reducers e a store.

2.1.7 theme

A diretoria "theme" inclui ficheiros relacionados ao styling da aplicação em light mode e em dark mode. "themeContext.js" e "theme.js" definem o contexto relacionado ao tema, e têm o intuito de alterar views de forma global ao sistema. Ao separar as preocupações de styling em dark/light mode num diretório dedicado, é possível gerir e atualizar facilmente os estilos visuais da aplicação, garantindo consistência em toda a interface do utilizador.

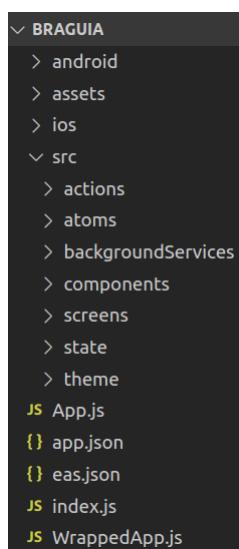


Figura 1: Estrutura do projeto

2.2 Soluções de implementação

Nesta secção iremos abordar as soluções que o grupo achou mais interessantes/complexas para satisfazer funcionalidades propostas. De forma a criar a aplicação React Native o grupo utilizou a framework Expo, que permitiu encontrar soluções para a maior parte dos problemas sem ter que escrever código nativo.

- **Persistência de dados:** De forma a preservar os dados do modelo de dados, o grupo utilizou a biblioteca Redux, que a quando feito o login do utilizador, são feitos pedidos à API de forma a guardar imediatamente todos os dados para utilização da aplicação. Com mais tempo o grupo teria guardado estes dados numa base de dados, o que teria sido uma solução mais eficiente em termos de pesquisa e updates. Para além disso são guardadas no Redux todas as preferências que o utilizador definir.
- **Pedidos à API:** Todos os pedidos feitos à API utilizam o Fetch dentro de uma função assíncrona de forma a fazer pedidos HTTP sem interromper o utilizador, possibilitando o utilizador de interagir com a aplicação enquanto o pedido executa.
- **Funcionamento offline:** De forma a que o utilizador entre na aplicação sem internet, a aplicação verifica se existe um utilizador válido nos dados persistentes de forma a efetuar o login automático. Como todos os dados estão guardados de forma persistente, não é necessário efetuar pedidos à API para navegar. Em relação ao carregamento de conteúdo multimédia, o grupo não fez nenhuma solução mas, com mais tempo, teríamos guardado todos os ficheiros numa pasta do utilizador de forma a reutilizar quando estiver offline.
- **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 biblioteca 'react-native-maps' para disponibilizar um mapa, a partir da qual fornecemos as localizações de cada ponto e configuramos algumas opções da câmara da aplicação do Maps. Este procedimento foi idêntico ao utilizado na fase anterior com a API do Google Maps.
- **Navegação por ecrãs:** De forma a possibilitar back gesture e criar ecrãs de forma organizada, o grupo utilizou um NavigationContainer onde inseriu todos os seus ecrãs. De forma a reutilizar componentes para todos os ecrãs como a ScrollView e a BottomBar, o grupo encapsulou os ecrãs em componentes que iriam reutilizar estas views.
- **Navegação de um roteiro:** Quando se inicia a navegação de um roteiro o utilizador é criado um botão que permite abrir o Google Maps no modo de navegação com todos os pontos do roteiro permitindo maior flexibilidade entre diferentes sistemas. Para além disso, um useEffect é ativado para começar a enviar pedidos à localização do utilizador, que posteriormente verifica se o utilizador se aproxima de um ponto de interesse, e caso o faça, seja lançada uma notificação que o redireciona para o ponto de interesse. Idealmente, em vez de um hook era criado um serviço *foreground* para que a aplicação funcionasse mesmo quando não estivesse aberta. Na página de navegação do roteiro tem um botão para parar a navegação e guardar a informação da viagem para o histórico do utilizador.
- **Pedido de localização:** Apesar da navegação não funcionar quando a aplicação está minimizada, os pedidos de localização foram preparados para funcionar em foreground. Aqui está o código que retorna a localização do utilizador:

```

let { status } = await Location.requestForegroundPermissionsAsync();
if (status !== 'granted') {
    console.log('Permission to access location was denied');
    return null;
}
let location = await Location.getCurrentPositionAsync({});
console.log(location);
return location;

```

- **Notification Receiver:** De forma a captar quando um utilizador clica numa notificação, utilizamos um listener que é capaz de redirecionar o utilizador para um ponto de interesse conforme o seu conteúdo.

2.3 Bibliotecas/dependências utilizadas

De forma a implementarmos as funcionalidades requeridas para este trabalho prático, foi necessário instalar algumas bibliotecas/dependências (presentes no ficheiro `package.json`, tais como:

```

"dependencies": {
    "@expo/webpack-config": "^18.0.1",
    "@react-native-async-storage/async-storage": "1.17.11",
    "@react-native-firebase/app": "^18.0.0",
    "@react-navigation/native": "^6.1.6",
    "@react-navigation/native-stack": "^6.9.12",
    "expo": "~48.0.18",
    "expo-application": "~5.1.1",
    "expo-av": "~13.2.1",
    "expo-constants": "~14.2.1",
    "expo-device": "~5.2.1",
    "expo-linking": "~4.0.1",
    "expo-location": "~15.1.1",
    "expo-notifications": "~0.18.1",
    "expo-splash-screen": "~0.18.2",
    "expo-status-bar": "~1.4.4",
    "expo-updates": "~0.16.4",
    "jetifier": "2.0.0",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "react-native": "0.71.8",
    "react-native-event-listeners": "^1.0.7",
    "react-native-image-picker": "^5.4.0",
    "react-native-maps": "1.3.2",
    "react-native-safe-area-context": "4.5.0",
    "react-native-screens": "^3.20.0",
    "react-native-video": "^5.2.1",
    "react-native-web": "~0.18.10",
    "react-redux": "^8.0.7",
    "redux": "^4.2.1",
    "redux-persist": "^6.0.0",
    "redux-thunk": "^2.4.2"
},
"devDependencies": {
    "@babel/core": "^7.20.0",
    "@babel/plugin-proposal-decorators": "^7.22.3",
    "eslint": "8.42.0",

```

```
    "eslint-plugin-react": "^7.32.2"
},
```

Destas bibliotecas, destacaremos algumas:

- `eslint`: linter do React Native;
- `@react-native-firebase/app`: permite system testing da app;
- `expo-location`: permite saber a localização do dispositivo;
- `expo-notifications`: permite o envio de notificações;
- `react-native-maps`: permite utilizar mapas;
- `redux`: permite a persistência dos dados;
- ...

2.4 Padrões de software utilizados

2.4.1 Provider

O padrão Provider, em conjunto com os componentes Provider e PersistGate, é utilizado neste código para fornecer acesso à store do Redux e garantir que a store seja persistida e reidratada adequadamente.

```
const App = () => {
  return (
    <Provider store={store}>
      <PersistGate persistor={persistor} loading={null}>
        <WrappedApp/>
      </PersistGate>
    </Provider>
  );
};
```

2.4.2 Padrão de Hooks

Em vez de utilizar os métodos de ciclo de vida tradicionais, como o construtor (`constructor`), `componentDidMount` e `componentWillMount`, o grupo optou por utilizar hooks por causa da sua simplicidade, reutilização de lógica, facilidade na resolução de problemas de ciclo de vida, melhor performance e compatibilidade com o ecossistema do React.

3 Mapa de navegação de GUI

Na seguinte imagem está apresentado o mapa de navegação proveniente da aplicação de testes firebase. As páginas dos trails e dos pins não estão presentes porque o apk crasha ao abrir os trails. As imagens dos trails e dos pins estão presentes na secção de funcionalidades.

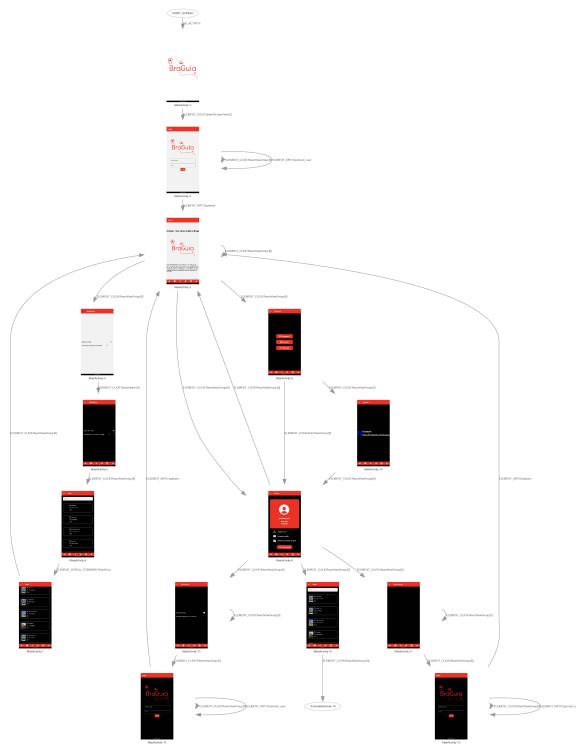


Figura 2: Mapa de navegação

4 Funcionalidades

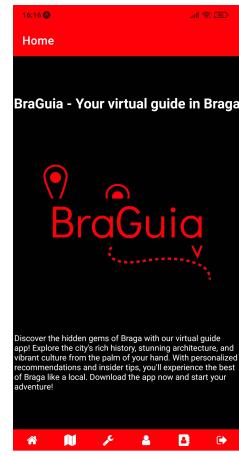
As funcionalidades contidas no sistema correspondem diretamente aos requisitos impostos para o mesmo, sendo que nesta secção o grupo optou por referir alguns dos principais requisitos.

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

Android



(a) Página inicial



(b) Página inicial (*dark mode*)

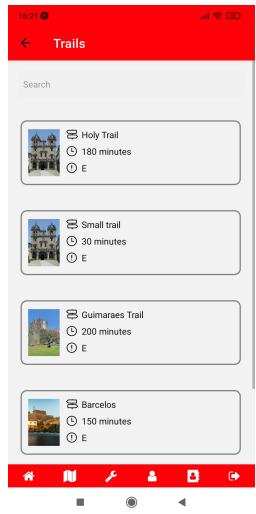
IOS



(a) Página inicial

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

Android



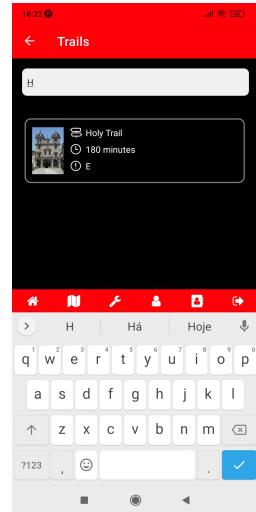
(a) Lista de roteiros



(b) Pesquisa na lista de roteiros



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

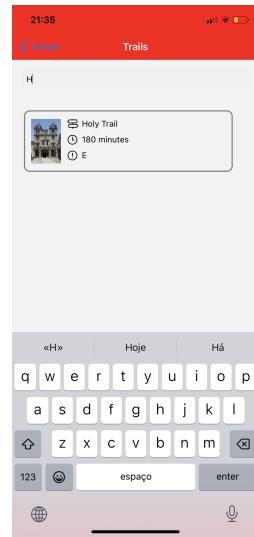


(d) Pesquisa na lista de roteiros (*dark mode*)

IOS



(a) Lista de roteiros



(b) Pesquisa na lista de roteiros

3. A aplicação deve permitir efetuar autenticação.

Android



(a) Autenticação (com credenciais erradas)



(b) Autenticação (com credenciais certas)

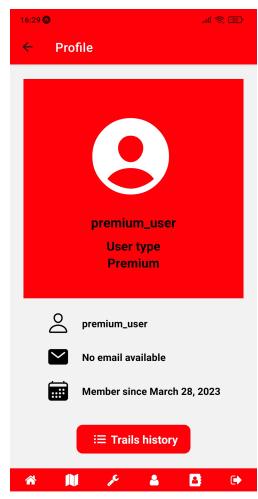
iOS



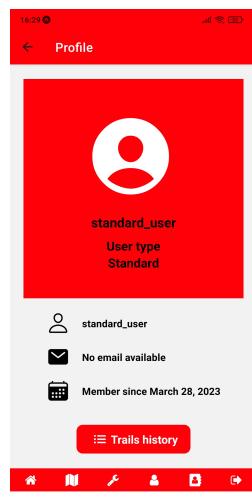
(a) Autenticação

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

Android

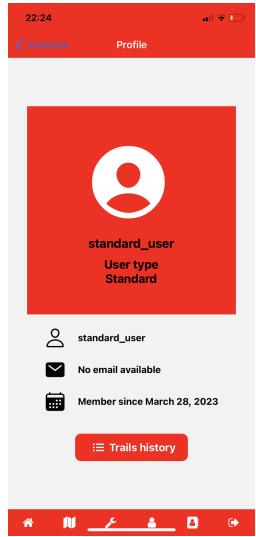


(a) Página de utilizador standard

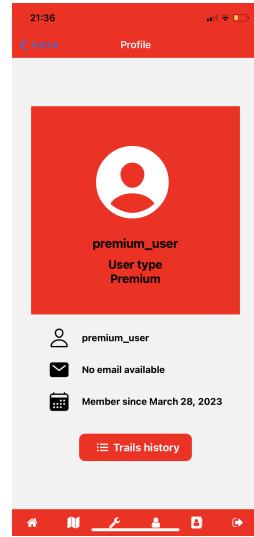


(b) Página de utilizador premium

iOS



(a) Página de utilizador standard



(b) Página de utilizador premium

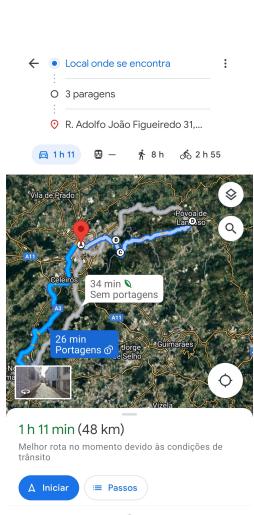
5. A aplicação deve assumir que o utilizador tem o Google Maps instalado no seu dispositivo (e notificar o utilizador que este software é necessário).

Não implementamos este requisito pois a navegação é reencaminhada para o browser e não para a aplicação do Google Maps instalada.

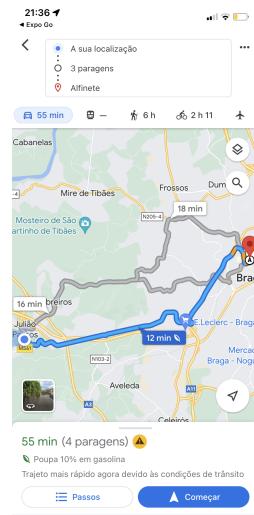
6. Para utilizadores premium, a aplicação deve possibilitar a capacidade de navegação, de consulta e descarregamento de mídia.

Este requisito foi parcialmente feito: permitimos a navegação e a consulta de conteúdo multimédia (cujas imagens de prova irão estar presentes mais a frente no relatório), mas não permitimos o seu descarregamento (a mídia não carrega sem internet).

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.



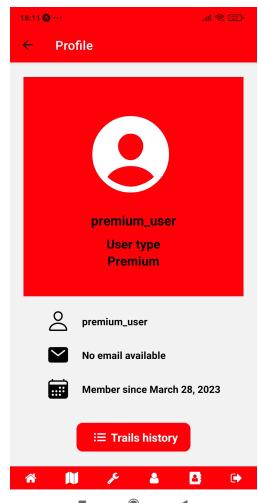
(a) Navegação Google Maps (Android)



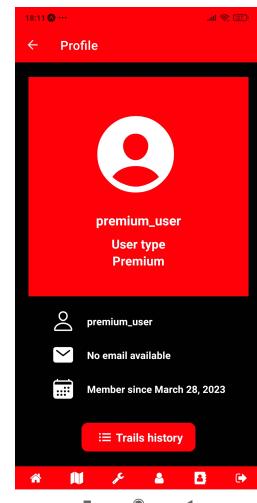
(b) Navegação Google Maps (IOS)

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

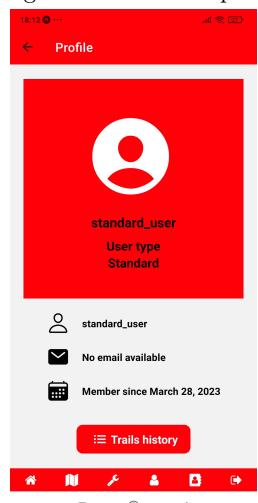
Android



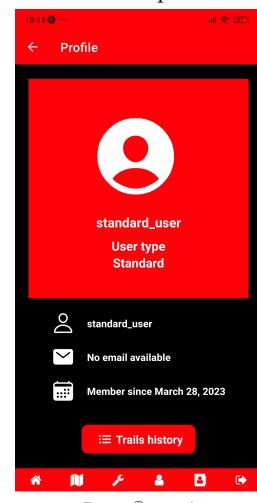
(a) Página de utilizador premium



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

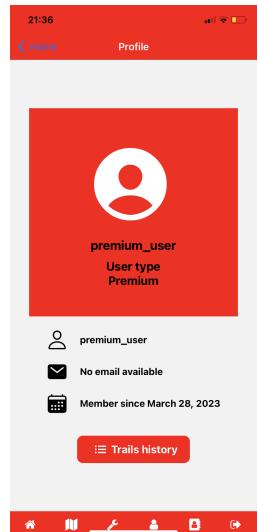


(c) Página de utilizador standard

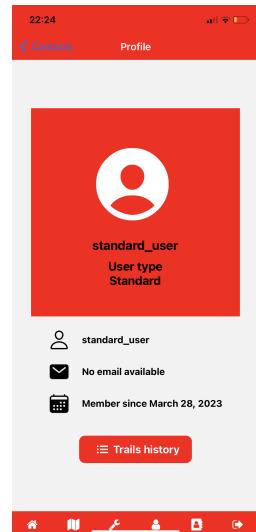


(d) Página de utilizador standard (*dark mode*)

IOS



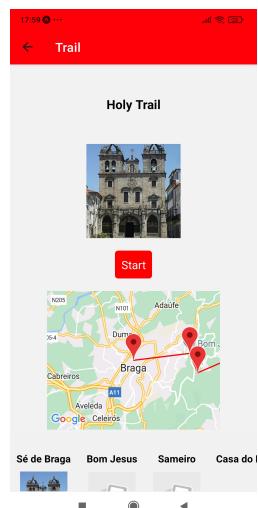
(a) Página de utilizador premium



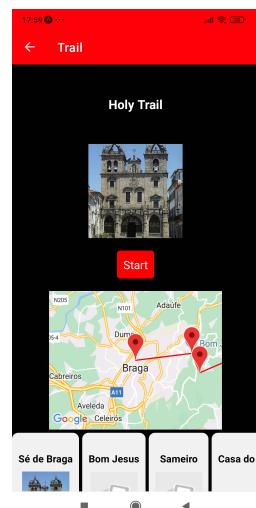
(b) Página de utilizador standard

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.

Android

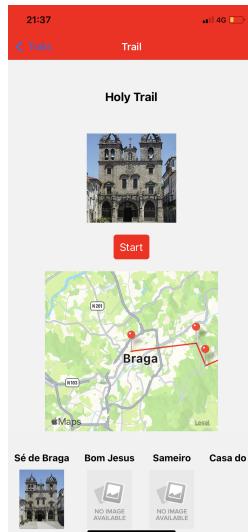


(a) Página de um roteiro



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

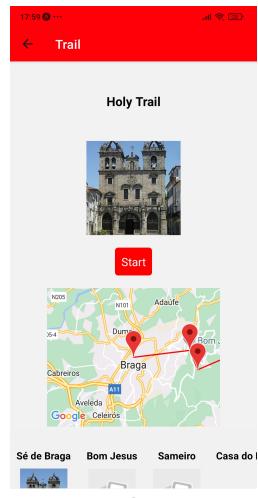
IOS



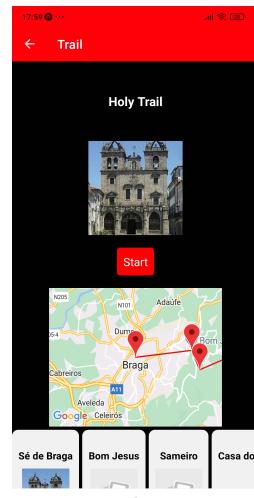
(a) Página de um roteiro

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

Android

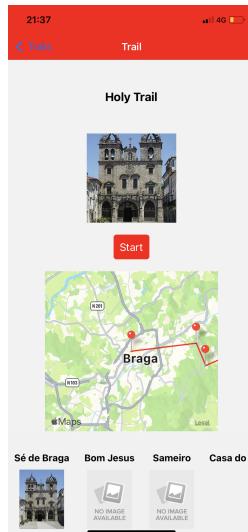


(a) Início de um roteiro



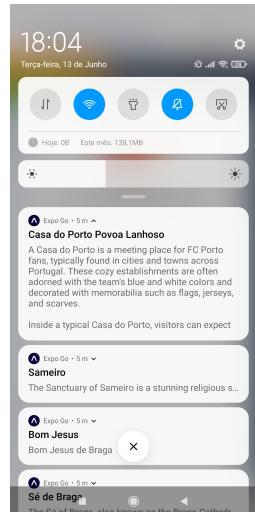
(b) Início de um roteiro (*dark mode*)

iOS



(a) Início de um roteiro

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



(a) Notificações de locais turísticos do roteiro



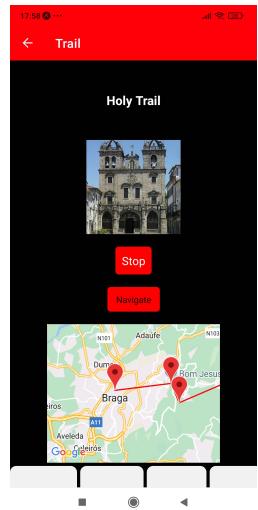
(b) Notificações de locais turísticos do roteiro (dark mode)

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

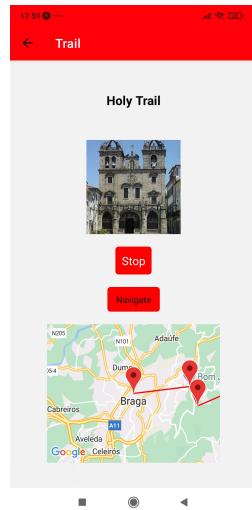
Ao receber as notificações tal como observado na figura 18a, um clique em qualquer uma redireciona para a respetiva página. Por exemplo, um clique em "Sameiro", redireciona para a página do Sameiro.

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

Android

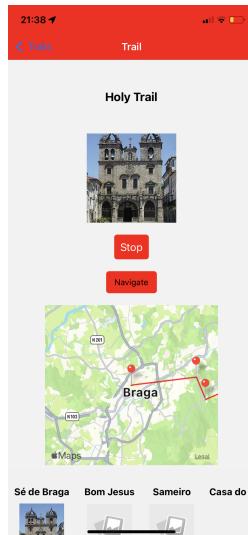


(a) Interrupção de um roteiro



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

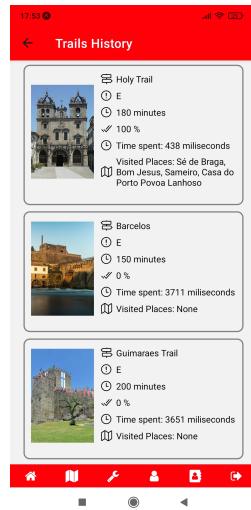
IOS



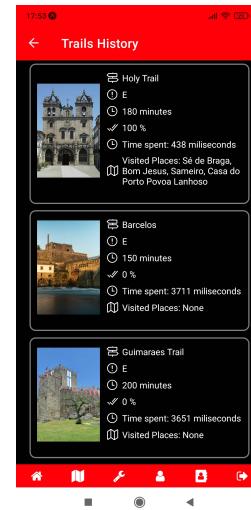
(a) Interrupção de um roteiro

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

Android

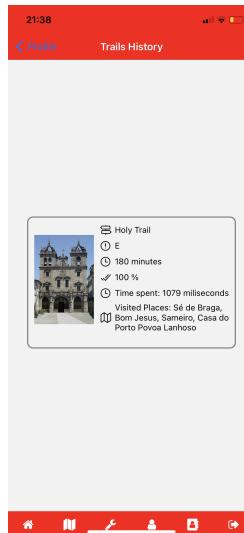


(a) Lista de roteiros visitados



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

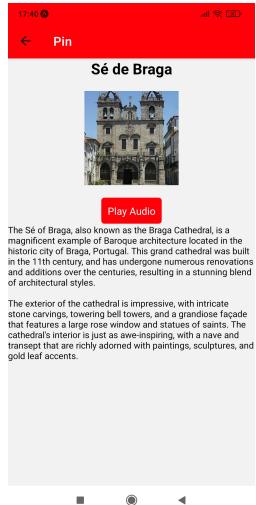
IOS



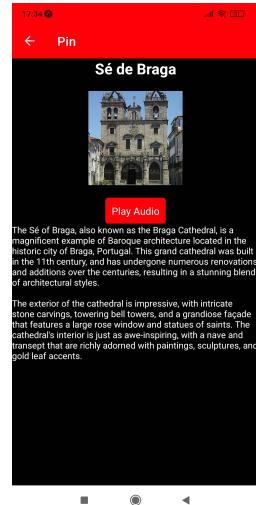
(a) Lista de roteiros visitados

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.

Android

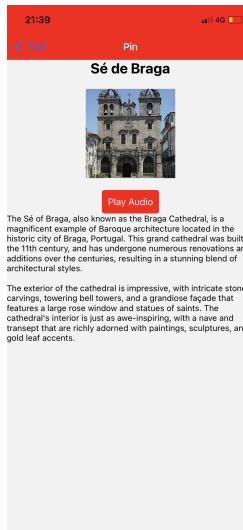


(a) Página de um ponto turístico



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

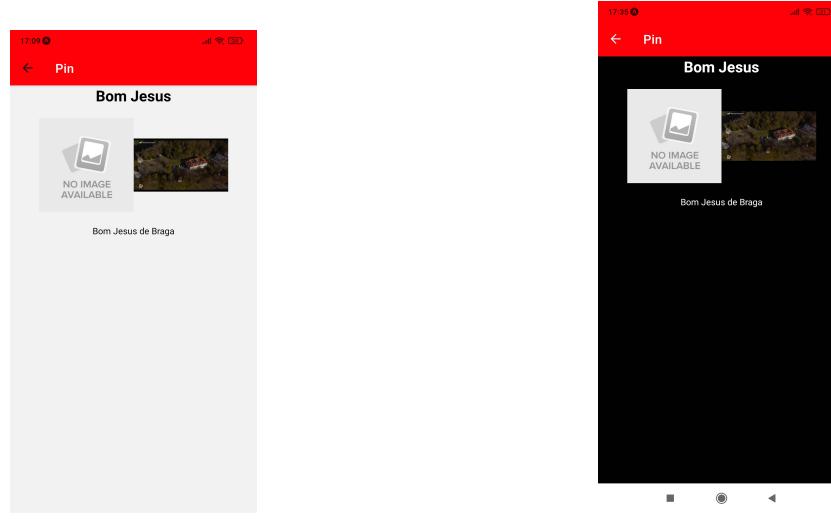
IOS



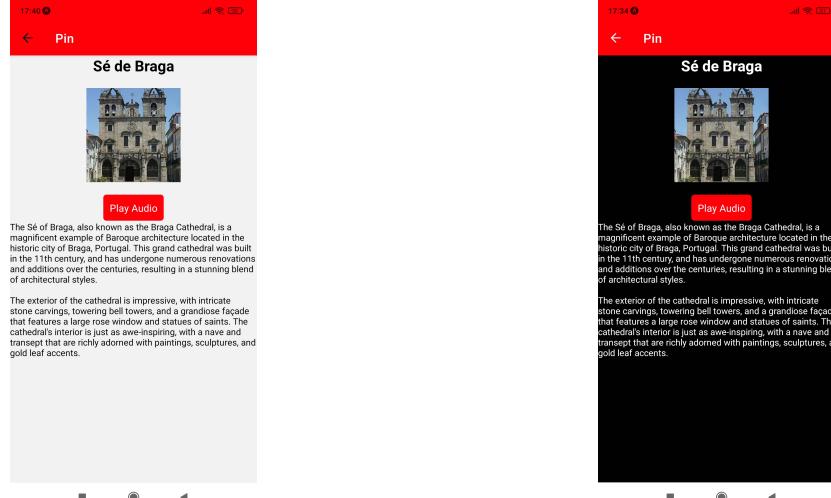
(a) Página de um ponto turístico

16. A aplicação deve ter a capacidade de apresentar e produzir 3 tipos de mídia: voz, imagem e vídeo. Note que o vídeo presente na versão IOS não é o mesmo do Android porque IOS não suporta vídeo no formato mp3.

Android



(a) Produção de conteúdo multimédia de vídeo

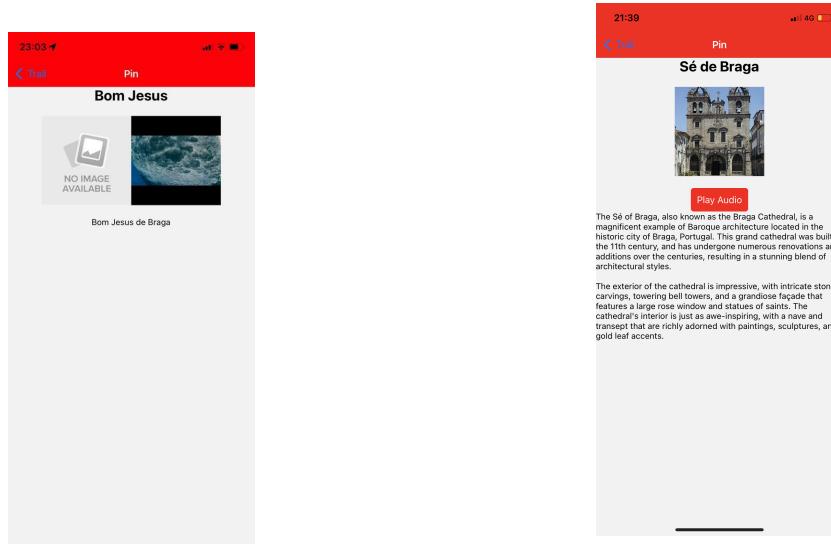


(c) Produção de conteúdo multimédia de imagem e áudio

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

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

IOS

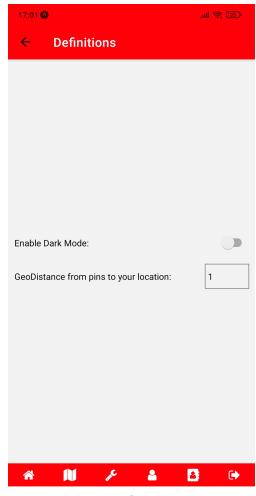


(a) Produção de conteúdo multimédia de vídeo

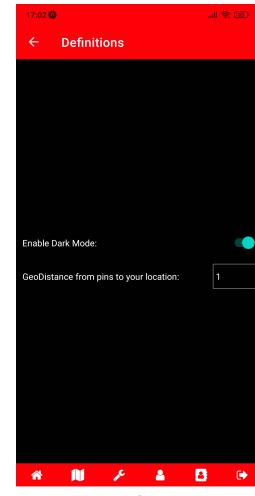
(b) Produção de conteúdo multimédia de imagem e áudio

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

Android

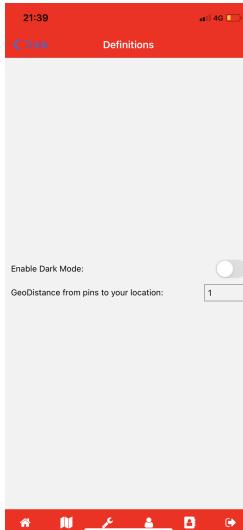


(a) Definições da aplicação



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

IOS



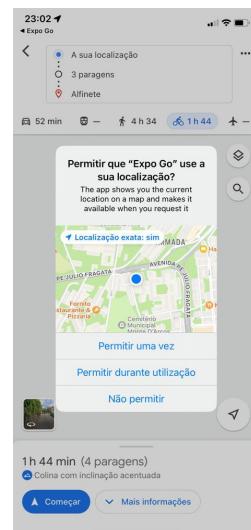
(a) Definições da aplicação

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

Neste requisito, não permitimos que a aplicação ligue e desligue o serviço de localização. No entanto, quando o utilizador abre o Google Maps, é-lhe pedido que ligue esse mesmo serviço.



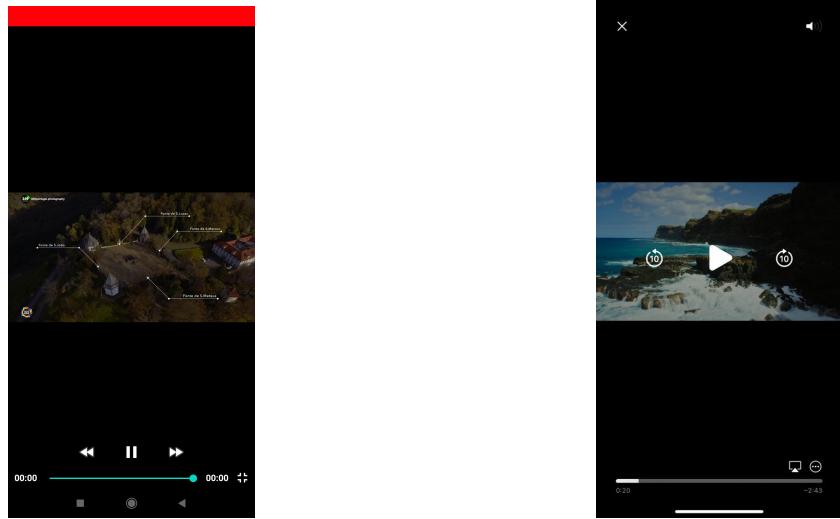
(a) Ligação da localização (Android)



(b) Ligação da localização (IOS)

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.

Esta funcionalidade foi testada e funciona corretamente em Android, mas, em IOS, não conseguimos correr os vídeos do backend. Para efeitos de teste, usamos um vídeo do tipo MP4 e aí já funciona corretamente.

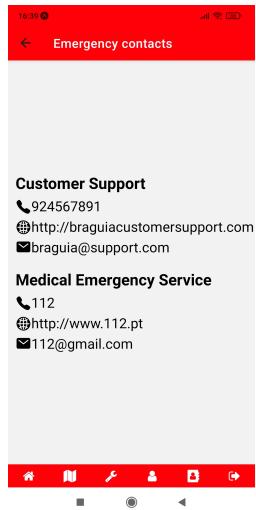


(a) Página de um ponto turístico a correr um vídeo
guardado localmente (Android)

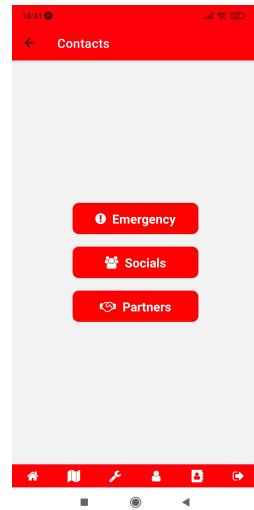
(b) Página de um ponto turístico a correr um vídeo
guardado localmente (IOS)

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.

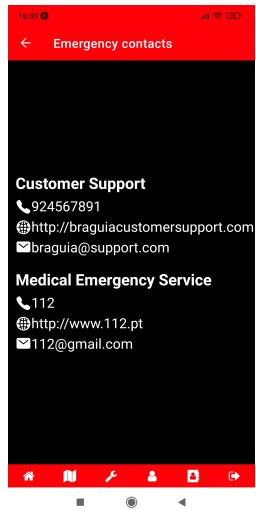
Android



(a) Página de contactos de emergência



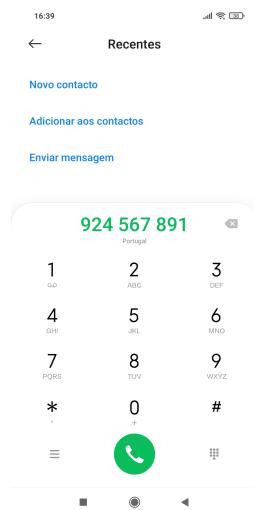
(b) Página de contactos



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

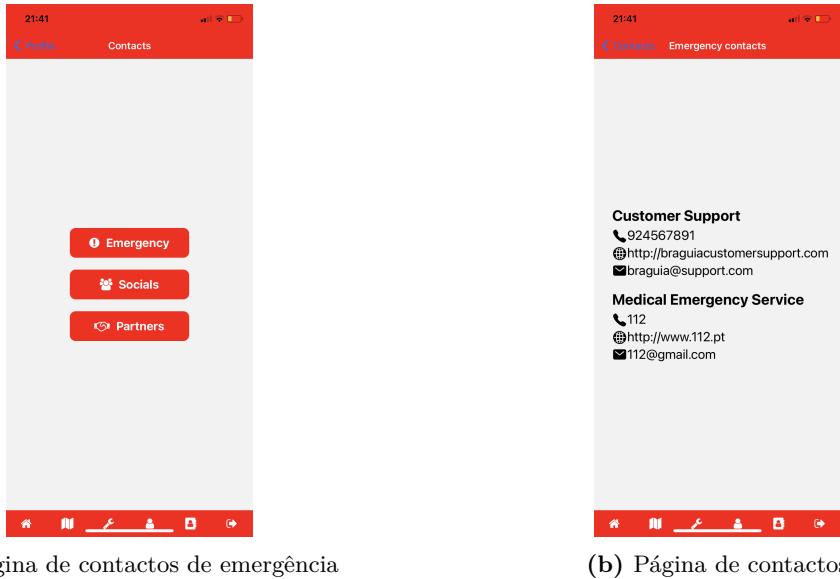


(d) Página de contactos (dark mode)



(e) Chamada telefónica

IOS



5 Discussão de resultados

5.1 Trabalho realizado

O grupo conseguiu fazer uma aplicação básica cross-platform a maior parte dos requisitos propostos. Apesar de não ter implementados as melhores soluções, durante o relatório o grupo foi apresentando soluções que teria implementado caso tivesse mais tempo. Com isto, consideramos que a nossa aplicação tem uma estrutura muita organizada, facilitando a leitura e expansão. Consideramos também que a utilização do Navigationcontainer facilita também a expansão da aplicação. O grupo limitou-se a escrever a grande maioria do código em react (à exceção da implementação do firebase para android) o que permitiu que a aplicação corresse em ambos Android e IOS sem problemas. A solução contém as configurações que em junção com o github actions automatizam a criação do apk para que a equipa não o tenha que fazer manualmente a cada atualização.

5.2 Limitações

Foram identificadas alguns erros e limitações na nossa aplicação:

- Nem todo o backend foi tida em conta na aplicação - por exemplo, a lista de pontos turísticos de um roteiro não tem contém os parâmetros `edge_transport`, `edge_duration` e `edge_desc`;
- Não foi implementado um serviço foreground para a navegação;
- Em IOS os vídeos no formato mp3 não são carregados;
- Quando o utilizador dá logout, a sua informação é perdida;

- Existe um bug onde é possível serem lançadas mais do que uma notificação do mesmo ponto de interesse.
- No apk, ao abrir a página de navegação, a aplicação crasha (apesar de na build do expo funcionar sem problemas). Tivemos problemas em corrigir este problema a tempo porque o grupo teve dificuldades a gerar o apk inicialmente e teve dificuldades em detetar o erro na plataforma de testes (o grupo não conseguiu encontrar o erro nos logs). Para além disto, como o grupo gerava o apk com a framework do expo, haviam builds que demoravam até duas horas para completar, dificultando ainda mais a tarefa de correção deste erro.

5.3 Testes

5.3.1 Software Testing

De forma a executar testes na plataforma firebase o grupo teve que acrescentar componentes relativos ao firebase na pasta de código de android nativa. Com isto o grupo consegui criar robots que percorrem toda a aplicação de forma a fazer E2E testing. De seguida estão algumas execuções feitas na plataforma. O grupo não fez testes com mais dispositivos porque o erro de crashar ao abrir um trail (no apk) também acontecia nos seus dispositivos físicos sendo que o grupo se focou mais em corrigir o erro existente em vez de detetar novos erros. De forma a criar o apk de forma automática, o grupo utilizou o github actions que gera um apk na plataforma expo sempre que vê um push para o repositório.

Matriz de teste	Identificador ⓘ	Tipo de teste	Horário de inicio	Total de dispositivos	Problemas
matrix-2e2wgt7ptkfs	—	Robo	há 21 horas	1	—
matrix-n66tu5tgokiba	—	Robo	há 1 dia	1	—
matrix-2s0acq191u3p6	—	Robo	há 1 dia	1	—
matrix-1evyzkdwdma6	—	Robo	há 1 dia	1	—
matrix-ak2dtss9ye99a	—	Robo	há 1 dia	1	—

Figura 33: Execução de testes do firebase

O grupo não fez testes E2E para IOS porque não consegui gerar o seu executável na plataforma expo. Quando tentamos criar o executável foi nos pedida uma licença paga, sendo que o grupo se limitou a testar a aplicação no seu dispositivo físico com a build do expo. De seguida está a imagem que comprova este problema.

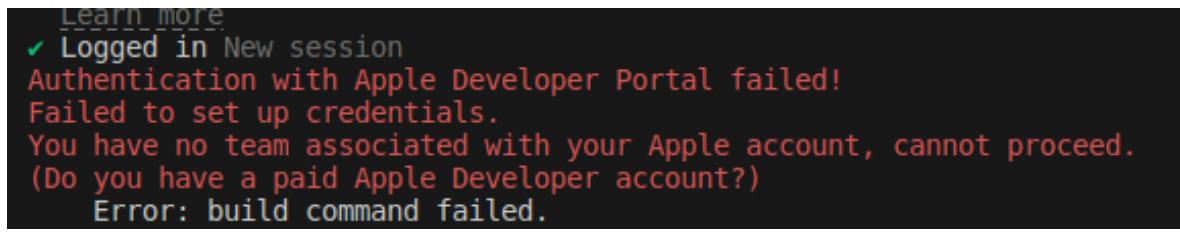


Figura 34: Erro ao criar executável IOS

5.3.2 Software Analysis

Nesta secção, iremos descrever o nosso processo de análise de software. Para tal, recorremos ao ESLint de forma a podermos identificar warnings e eventuais erros no código da nossa aplicação - quer na lógica do programa, quer no layout dos componentes/screens.

ESLint Report	
5 problems (5 errors, 0 warnings) - Generated on Tue Jun 13 2023 17:33:10 GMT+0100 (Hora de verão da Europa Ocidental)	
[+] C:\Users\utilizador\Desktop\BraGuia\.eslintrc.js	1 problem (1 error, 0 warnings)
[+] C:\Users\utilizador\Desktop\BraGuia\App.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\WrappedApp.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\babel.config.js	1 problem (1 error, 0 warnings)
[+] C:\Users\utilizador\Desktop\BraGuia\index.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\metro.config.js	3 problems (3 errors, 0 warnings)
[+] C:\Users\utilizador\Desktop\BraGuia\src\actions\appData.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\actions\user.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\atoms\Button.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\backgroundServices\LocationTracker.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\components\BottomBar.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\components\NotificationManager.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\components\PinsSlide.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\components\SearchBar.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\components\TailItem.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\components\ToggleButton.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\screens\Contacts.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\screens\Definitions.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\screens\Emergency.js	0 problems
[+] C:\Users\utilizador\Desktop\BraGuia\src\screens\History.js	0 problems

Figura 35: Relatório obtido pelo ESLint

Além disso, também recorremos novamente ao Firebase Test Lab para analisarmos a performance da nossa aplicação aquando dos testes efetuados pelos seus robots.

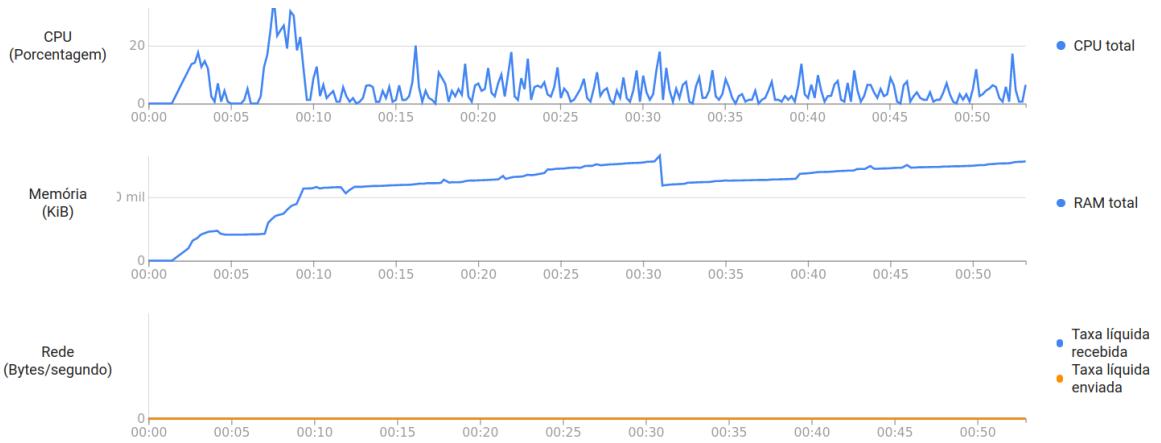


Figura 36: Gráfico de performance obtido no firebase

5.4 Funcionalidades extra

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

- Semelhante ao efetuado no TP1, criamos uma imagem do Google Maps que desenha o trajeto a efetuar do roteiro;
- Semelhante ao efetuado no TP1, implementamos uma barra de pesquisas que nos devolve um roteiro ao escrevermos o seu nome (ou parte dele);
- Semelhante ao efetuado no TP1, implementamos uma opção de ativar o *dark mode* na aplicação ao criarmos um *switch* nas definições;
- Na lista de contactos (emergência, parceiros ou redes sociais), adicionamos a opção de abrir URL no browser do dispositivo e de começar o envio de uma mensagem e-mail com o constado na lista de contactos.

6 Percentagem de código nativo utilizado

Nesta secção, iremos fazer uso do `scc` (Sloc cloc and code) para contabilizarmos a percentagem de código nativo utilizado neste trabalho prático. Uma vez que este *software* não nos devolvem percentagens em concreto, dividimos o repositório em duas pastas: uma que contém o código nativo (para Android e IOS) e outra com o código escrito em React Native e executamos o programa `scc` em cada uma das diretórias, obtendo os seguintes resultados:

Language	Files	Lines	Blanks	Comments	Code	Complexity
XML	10	124	4	23	97	0
JSON	6	220	0	0	220	0
Java	4	239	28	48	163	7
Gradle	3	291	15	32	244	0
Properties File	2	58	11	32	15	0
gitignore	2	45	5	11	29	0
Batch	1	91	21	0	70	15
C Header	1	7	2	0	5	0
Objective C	1	10	3	0	7	0
Objective C++	1	67	13	12	42	2
Prolog	1	14	2	0	12	0
Shell	1	240	28	110	102	13
Swift	1	4	0	4	0	0
Total	34	1410	132	272	1006	37
Estimated Cost to Develop (organic) \$27,184						
Estimated Schedule Effort (organic) 3.50 months						
Estimated People Required (organic) 0.69						
Processed 50924 bytes, 0.051 megabytes (SI)						

Figura 37: Output para o código nativo

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	29	2409	243	16	2150	83
JSON	2	70	0	0	70	0
Total	31	2479	243	16	2220	83
Estimated Cost to Develop (organic) \$62,411						
Estimated Schedule Effort (organic) 4.79 months						
Estimated People Required (organic) 1.16						
Processed 69581 bytes, 0.070 megabytes (SI)						

Figura 38: Output para o código React Native

Assim, de forma a obtermos a percentagem de código nativo, iremos efetuar o seguinte cálculo:

$$\frac{1410}{1410 + 2479} * 100 = 36.26\%$$

No entanto, este valor não é muito relevante, pois os módulos nativos que temos presentes apenas foram necessários para implementarmos o Firebase na versão em Android.

7 Gestão de projeto

7.1 Gestão e Distribuição de trabalho

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

- Luís Silva (pg50564)
 - Layout dos componentes e screens;
 - Serviços/notificações;
 - Dados persistentes com o Redux;
 - Pedidos à API;
 - Navegação;
 - Testes;
 - GitHub Actions
- Simão Cunha (a93262)
 - Layout dos componentes e screens;
 - Layout e lógica da *Bottom navbar*
 - Layout e lógica dos contactos (emergency, partners e socials);
 - Layout e lógica da página de perfil;
 - Layout da página do histórico;
 - Ativação/desativação do *dark mode*;
 - Aplicação do ESLint;
- Gonçalo Pereira (a93168)
 - Layout dos componentes e screens;
 - Navegação de componente por ecrãs;
 - Tratamento do conteúdo multimédia;
 - Integração do mapa na página do roteiro;
 - Lógica de lista de roteiros e pesquisa de roteiros pela barra de pesquisas;

8 Conclusão

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

Foi-nos imposta a utilização da linguagem React Native para elaborarmos a aplicação, na qual utilizamos de framework Expo devido a sua facilidade de utilização e fast reload. Com isto observamos que a utilização de linguagens cross-platform facilita o processo de criação de uma aplicação para várias plataformas.

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, de forma a dar-mos importância a processos de CI/CD no futuro da nossa vida profissional.