Computação Móvel
GoLoco - Flutter Aplication

Miguel Cabral[93091] , Daniel Andrade[93313]

Universidade de Aveiro
Departamento de Eletrónica Telecomunicações e Informática
13 de Novembro de 2022

1 Introdução

O presente relatório foi realizado para fundamentar o projeto realizado no âmbito da cadeira de Computação Móvel do Departamento de Eletrónica Telecomunicações e Informática, a decorrer no 1º semestre do ano letivo 2022/2023. O projeto foi desenvolvido com a finalidade de criar uma aplicação mobile usando flutter [Goo] na qual um utilizador possa procurar por pontos de interesse próximos do local onde se encontra e caminhar até eles apenas com a ajuda de uma bússola e de uma seta a apontar para a localização desse ponto de interesse. Uma vez lá, a pessoa pode ler o código QR e ver uma página detalhada sobre o local visitado. A aplicação incentiva o utilizador a caminhar, orientar-se pela cidade e explorar locais escondidos que possam existir na sua proximidade que, por outros meios, possam passar despercebidos.

2 Motivações

Com o avanço tecnológico a forma como podemos chegar a certos locais ficou muito mais facilitada, se por um lado é bom por essa facilidade por outro é mau porque ao saber a rota exata para o local, o sentido de aventura e orientação são pouco estimulados. Outra desvantagem é que essas aplicações por vezes têm somente os pontos mais conhecidos de um local e no entanto há muitos locais à nossa volta que ainda não forem partilhados por algo e às vezes que ainda não foram descobertos e isso só é possível se houver pessoas que partam à sua descoberta, o que por rotas definidas é praticamente impossível. A ideia é combinar estes dois fatores e usando a nossa aplicação, as pessoas comecem a sair das suas casas, a desfrutar da natureza, a visualizar edifícios ou monumentos escondidos, e a desenvolver outros sentidos. Esta aplicação também pode ser útil para atrair pessoas para um local remoto, tal como uma pequena aldeia, uma vez que esses locais carecem de pessoas e recompensá-los com algo é suficiente para povoar um pequeno local.

3 Implementação

3.1 Requisitos

3.1.1 Instalação das biblioteca

Após instalar todos os requisitos para o ambiente flutter [Goo] é necessário correr os seguintes comandos para instalar as bibliotecas necessárias para correr a aplicação:

```
flutter pub add permission_handler
flutter pub add google_fonts
flutter pub add animated_splash_screen
flutter pub add pedometer
flutter pub add location
flutter pub add camera
flutter pub add equatable
flutter pub add flutter_bloc
flutter pub add shared_preferences
flutter pub add google_maps_flutter
flutter pub add platform_device_id
flutter pub add sensors_plus
flutter pub add local_auth
flutter pub add flutter_launcher_icons
flutter pub add map
flutter pub add flutter_compass
flutter pub add geolocator
flutter pub add vector_math
flutter pub add qr_code_scanner
```

É ainda necessário alterar as seguintes linhas no ficheiro *"android\app\lbuild.gradle"* :

```
android {
    compileSdkVersion 33
...
defaultConfig {
        ...
        minSdkVersion 23
```

3.1.2 Instalação do Servidor NestJS

Primeiramente é necessário instalar o NodeJS [Fou] para correr o servidor NestJS [Mys], depois de instalado o NodeJS [Fou] correr o seguinte no terminal:

```
npm i -g @nestjs/cli
```

Após isso na pasta *"go-loco-api\src"* correr o seguinte: [Fou] correr o seguinte no terminal:

```
npm run start:dev
```

Para que a API funcione corretamente é ainda necessário alterar o ipv4 da variável url nos ficheiros "lib\blocs\achievement_bloc\bloc\achievement_repo.dart", "lib\blocs\markers_bloc\bloc\markers_repo.dart" e "lib\blocs\profile_bloc\bloc\profile_repo.dart" para o seu ipv4 [1].
No exemplo seguinte deve mudar o texto cortado:

String url = 'http://~~192.168.1.156~~:3000/';

[1] comando: ipconfig para windows e ifconfig em linux

3.2 Estrutura

Para uma melhor organização do projeto e também para reaproveitamento do código o nosso projeto foi estruturado como na figura 1.

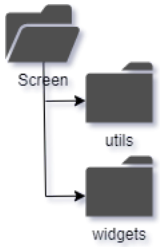

Fig. 1. Structure Schematic

Cada pasta corresponde a uma página e dentro desta podemos ter duas pastas, *"utils"* onde se encontram os ficheiros com a lógica dos widgets e *"widgets"* onde se encontram os widgets, alguns dos quais são reaproveitados em outras páginas.

3.3 Arquitetura

3.3.1 Registo e Login

A aplicação é protegida com dados biométricos usando a biblioteca "local_auth" [flu], após isso se o utilizador ainda não estiver registado é redirecionada para a página de registo, após este registo é guardada uma key na SharedPreferences do dispositivo. O Login é feito automaticamente após autenticação biométrica sempre o que utilizador reinicia a aplicação através da key.

3.3.2 BLoC

Foi utilizado uma arquitetura com base em BLoC para separar a lógica e os dados da aplicação da interface gráfica.
No BLoC dos achievements está implementada a lógica das conquistas, estas conquistas têm como base o número de passos dados.
No BLoC do map está implementada toda a lógica que está relacionada com o carregamento do mapa, o cálculo da distância ao objetivo e o ângulo entre a posição atual e o objetivo, estes dois não são mostrados na pãgina do mapa mas sim na página da bússola.
No BLoC dos markers a aplicação carrega os dados dos locais.
No BLoC do profile é onde se encontra toda a lógica relacionada com o utilizador, quer seja criar um novo perfil de utilizador ou carregar um perfil já criado através de uma key guardada na SharedPreferences do dispositivo.

3.3.3 Página Inicial

O painel principal está dividido em 2 secções:

1. O cabeçalho onde temos pequena mensagem de boas-vindas seguida do nome do utilizador.
2. Os dados onde o utilizador pode acompanhar o seu progresso em termos de últimas descobertas e conquistas desbloqueadas. Ao clicar no botão *"Procurar Locais"*, aparece uma nova página, a página do mapa.

3.3.4 Bússola

Com o objetivo de guiar o utilizador até ao local utilizámos uma bússola utilizando a biblioteca *"smooth_compass"* [AF], dentro da mesma temos a distância desde o ponto onde se encontra o utilizador até ao local e uma seta que aponta para o local, estes dois são calculados no bloc do map como referido na secção 3.3.2.

3.3.5 Código QR

Com o objetivo de obter a verificação que o utilizador chegou ao local, cada local tem o seu próprio código QR, para a sua leitura foi usada a biblioteca *"qr_code_scanner"* [ZB].

3.3.6 API

Com o objetivo de a aplicação usar dados remotos foi criada uma API utilizando a linguagem typescript [Mic] a correr num servidor NestJS [Mys]. Foram criados métodos para criar perfis de utilizador, para retornar perfis, para retornar markers e para retornar achievements. Estes dados ficam guardados em memória.

3.3.7 Diagrama de Arquitetura

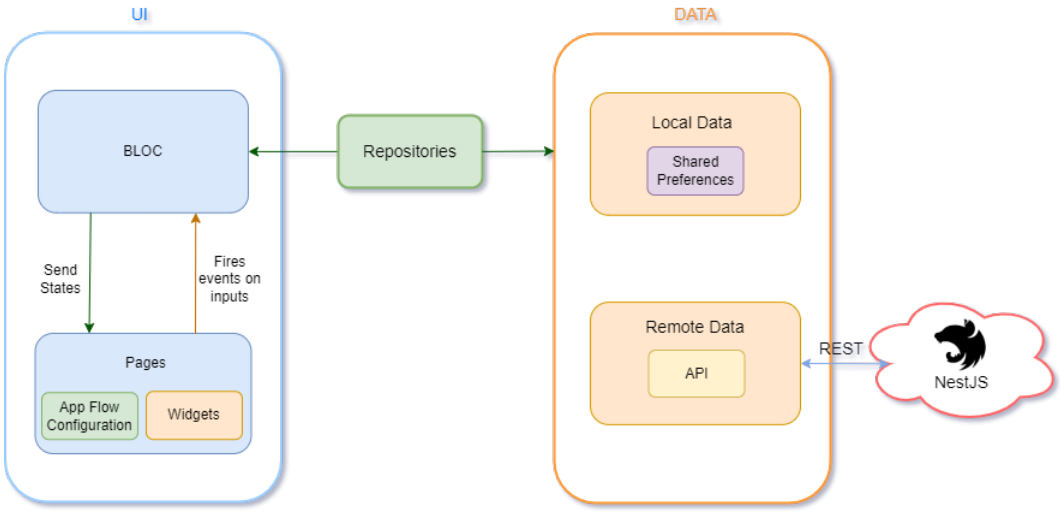

Fig. 2. Architecture Diagram

4 Simulação

Para a simulação primeiramente deve descarregar o repositório disponível na secção 6, após isso deve iniciar o servidor NestJS [Mys] a API como descrito na secção 3.1.2. Após isso na pasta do projeto deve correr o seguinte:

```
flutter run
```

De notar que não é possível simular a aplicação num emulador.
Se tudo correr sem erros deve ver no seu dispositivo uma página a pedir para colocar a sua impressão digital.

5 Discussão

Analisando os resultados obtidos, podemos concluir que praticamente todos os objetivos impostos no inicio do desenvolvimento foram concluídos com exceção de uma incompatibilidade com a biblioteca usada para ler os códigos QR que não lê. No entanto no que diz respeito às features foram implementadas mais que as pensadas no inicio, como a bússola, o cálculo da distância e o ângulo ao objetivo e um sistema de recompensas com a utilização do pedómetro.
Numa futura atualização poderia ser usada uma base de dados para guardar os dados locais e adicionar a opção do utilizador criar um novo local.

6 Repositório

https://github.com/AnBapDan/cm_project

References

[AF] Alihadi and Fadiktk. *smooth compass*. URL: https://pub.dev/packages/smooth_compass. (accessed: 12.11.2022).

[flu] flutter.dev. *local auth*. URL: https://pub.dev/packages/local_auth. (accessed: 11.11.2022).

[Fou] OpenJS Foundation. *Node.js*. URL: https://nodejs.org. (accessed: 10.11.2022).

[Goo] Google. *Flutter*. URL: https://flutter.dev/. (accessed: 07.11.2022).

[Mic] Microsoft. *typescript*. URL: https://www.typescriptlang.org/. (accessed: 12.11.2022).

[Mys] Kamil Mysliwiec. *NestJS*. URL: https://nestjs.com. (accessed: 10.11.2022).

[ZB] Zxing and Mike Buss. *qr code scanner*. URL: https://pub.dev/packages/qr_code_scanner. (accessed: 12.11.2022).