Give&Read

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

Grupo 5: Márcia Pires, Tomás Martins

(88747) marcia.pires@ua.pt, (89286) tomasfilipe7@ua.pt

novembro de 2021

Conteúdo

1 Motivação **2**

2 Solução **3**

3 Arquitetura **5**

3.0.1 Árvore de Widgets principais 6

3.0.2 Solução para gestão de dados 7

3.0.3 Recursos/sensores externos 7

4 Resultados e conclusão **8**

5 Contribuições **9**

6 Tutorial **10**

Capítulo 1

Motivação

Desenvolvida no âmbito da cadeira de Computação Móvel, **Give&Read** é uma aplicação que visa promover o acesso à leitura de forma prática, económica e sustentável, através da partilha de livros.

Para isso, os utilizadores desta aplicação registam os livros que querem partilhar e os livros que procuram. Quando um utilizador está à procura de determinado livro e existe outro utilizador nas proximidades que quer partilhar esse mesmo livro, este último utilizador recebe essa informação. Ao tomar esse conhecimento, o utilizador pode visualizar no mapa os diversos postos de recolha (bookstops) espalhados pela cidade, dirigir-se até um deles, fazer o *check-in* do livro e deixá-lo. Assim que este livro é deixado num posto de recolha, o utilizador tem a informação de que esse livro se encontra disponível e onde, e é levado a dirigir-se até lá, para que possa então fazer o *check-out* do seu novo livro.

Capítulo 2

Solução

Para podermos atingir a visão previamente apresentada, decidimos criar uma aplicação em que cada utilizador teria guardado, numa **base de dados local**, implementada em **Hive**, uma lista com os livros que quer ler e outra lista com os livros que tem para dar. De modo a tornar a aplicação mais robusta, utilizamos o pacote **books_finder** que permite fazer pedidos à Google Books API, tendo assim acesso a um infinidade de livros.

Um utilizador, através do pacote **nearby_connections**, que se serve do bluetooth, wi-fi e localização de cada dispositivo, consegue identificar dispositivos de outros utilizadores nas proximidades e conectar-se aos mesmos numa ligação *peer-2-peer*. Desta maneira, consegue verificar quais os livros que ele próprio tem para dar que os outros utilizadores gostariam de ler.

Quando um utilizador deseja entregar um livro, apenas tem de se dirigir à *bookstop* mais próxima, informação disponível através de um mapa, obtido através do pacote **google_maps_flutter**, que nos permite aceder a um mapa da Google Maps API e que nos indica os pontos onde existe uma *bookstop* e também um ponto que indica a nossa própria localização em tempo real (obtido através do pacote **geolocator**).

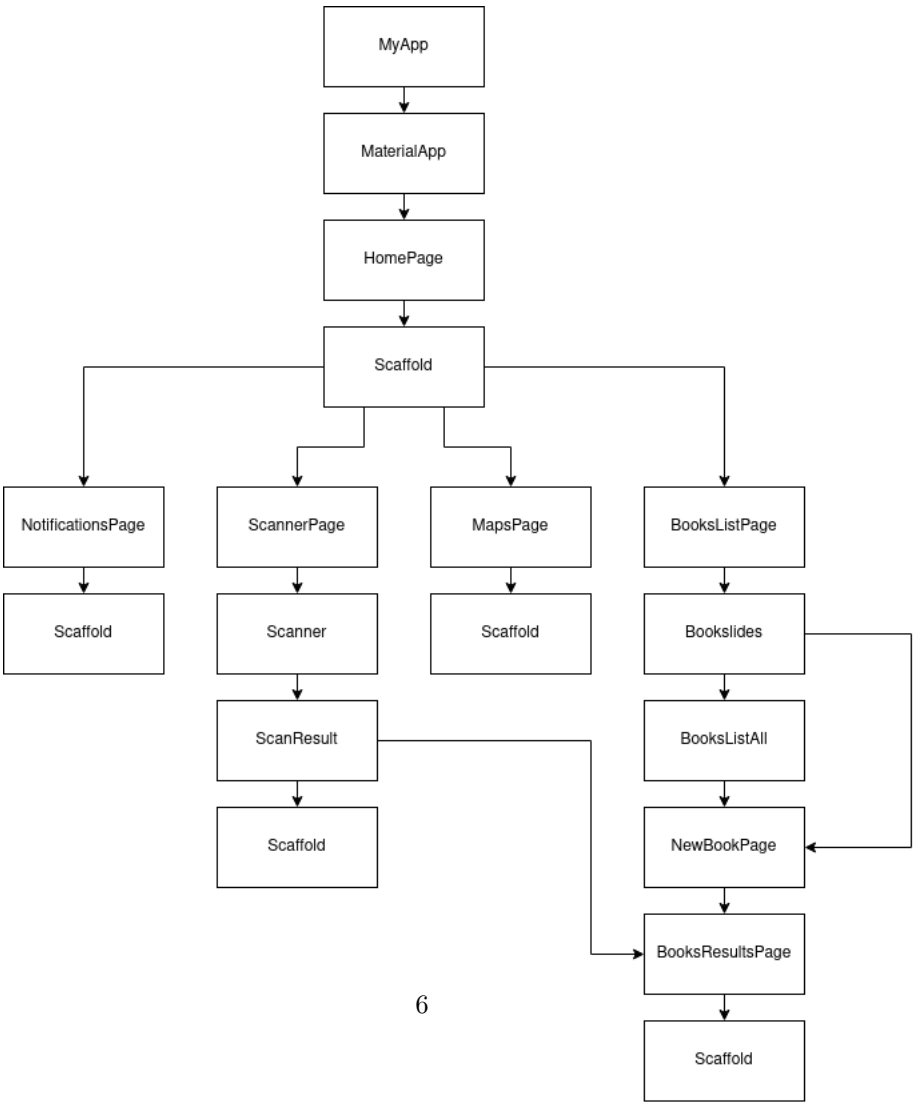
Uma vez na *bookstop*, estão disponíveis dois QRCode: um para que possa fazer o depósito de um livro (*check-in*) e outro para que possa fazer a recolha de um livro (*check-out*). Para implementar o **scanner** de QR code, foi utilizado o pacote **qr_code_scanner**. Para fazer então o *check-in* de um livro, o utilizador lê o QRCode correspondente e indica qual o livro que quer deixar. Esta informação sobre o registo de livros presentes em cada *bookstop* é guardada numa **base de dados central**, implementada em **Firebase**.

Um utilizador, sempre que quer verificar se alguma *bookstop* tem disponível um livro que queira ler, basta entrar no separador correspondente na aplicação, que irá fazer um pedido à base de dados central do Firebase e comparar os livros presentes em cada *bookstop* com a sua própria lista de livros para ler. Irá apresentá-los numa lista, bem com a informação de qual *bookstop* contém o livro desejado e terá também um botão que leva o utilizador à localização da mesma no mapa. De seguida, esse mesmo utilizador tem de se deslocar ao ponto de recolha e tal como o utilizador que depositou o livro, este terá de ler um QRCode, desta vez, o correspondente ao levantamento (*check-out*) de um livro.

Capítulo 3

Arquitetura

3.0.1 Árvore de Widgets principais



3.0.2 Solução para gestão de dados

De maneira a acedermos a dados externos que estejam ao alcance de qualquer *widget*, recorremos a duas diferentes soluções:

- **Firebase**: através desta plataforma pudemos utilizar a funcionalidade de **Realtime Database**, que nos permite guardar dados em *cloud* e obtê-la em tempo real, funcionando assim como a nossa base de dados central. Uma vez que todos os utilizadores precisam ter conhecimento sobre os livros disponíveis em cada *bookstop* em tempo real, consideramos esta a melhor opção, utilizando-a de uma forma bastante simples - uma tabela em que cada entrada nos indica de qual *bookstop* se trata e qual o livro em questão. Desta forma, podemos facilmente acrescentar novos livros a diferentes *bookstops* através do método *set()* ou, para remover livros, filtramos as entradas da tabela pelo livro que queremos eliminar, da respetiva bookstop, e depois usamos o método *remove()* para efetivamente remover.

- **Hive**: por outro lado, existem dados que só são necessários localmente e, portanto, utilizamos este pacote de Flutter, escrito em puro Dart, para servir de base de dados local. Havendo necessidade de guardar a lista de livros para ler e lista de livros para dar localmente, foram feitas então duas diferentes tabelas exatamente para esse efeito: uma tabela que guarda os livros para ler do seu utilizador, e uma tabela que guarda os livros para dar do seu utilizador. Desta forma podemos sempre ter disponível na nossa aplicação estas listas de livros, onde facilmente podem ser adicionados novos livros, através do método *add()*, ou removidos através do método *delete()*. Tal como na solução anterior, ambas as ações podem ser observadas em tempo real.

3.0.3 Recursos/sensores externos

A nível de recursos, foram utilizados os seguintes:

- API da Google Books, acedida através do pacote books_finder, que nos disponibiliza uma vasta quantidade de livros e os seus detalhes, e que usamos para que os utilizadores possam pesquisar por determinado livro ou determinada *keyword* e obter exatamente o livro pretendido.

- API da Google Maps, que, através do pacote google_maps_flutter, nos retorna um mapa, marcado com as *bookstops* existentes e ainda a localização atual do utilizador.

- Uso da câmara do telemóvel para efetuar o *scanner* dos QRCodes.

- Bluetooth, wi-fi e localização, necessários para utilizar o pacote nearby_connections que é aquele que nos permite efetuar ligações peer-to-peer e assim comunicar entre diferentes dispositivos móveis.

Capítulo 4

Resultados e conclusão

Finalizado este projeto, podemos concluir que fomos capazes de cumprir os nossos objetivos iniciais de maneira bastante satisfatória, terminando com uma aplicação bastante sólida, robusta e intuitiva, capaz de satisfazer todas as necessidades para o seu bom funcionamento.

- As listas locais de livros para ler e livros para dar, tal como previsto, funcionam sem nenhum problema e de forma intuitiva. Ainda assim, a API utilizada para aceder a um *dataset* de livros poderia ser melhor, uma vez que a Google Books, apesar de oferecer uma grande quantidade de livros, a qualidade da informação obtida não é a melhor nem a mais rigorosa;
- O mapa funciona bem e de acordo com o necessário;
- O QRCode também funciona tal como pretendido;
- A conexão peer-to-peer, que permite a comunicação entre dois dispositivos móveis também funciona bem e satisfaz as necessidades da aplicação. No entanto, o plano inicial era utilizar somente o bluetooth mas o Flutter ainda não disponibiliza nenhum pacote capaz de fazer esta ligação, suportando até ao momento apenas ligações entre dispositivos com Bluetooth Low Energy (através dos pacotes flutter_blue e flutter_bluetooth_serial) e não Bluetooth Classic, como exigem os smartphones;
- Em vez de notificações, tal como inicialmente previsto também, optámos por ter um separador onde o utilizador pode consultar quais *bookstops* têm o seu livro, apenas por questões práticas de conciliação de tempo.

Posto isto, apesar de algumas soluções não serem as inicialmente idealizadas, conseguimo-os adaptar e oferecer a mesma qualidade a que nos tínhamos proposto, através de muita pesquisa e tentativa-erro. O balanço final do nosso projeto é muito positivo pois além de termos uma aplicação bastante funcional, sentimos que conseguimos aprender e aprofundar diversos conceitos complexos e não tão complexos de Flutter, que era acima de tudo o nosso maior objetivo.

Capítulo 5

Contribuições

O trabalho foi desde início dividido de igual forma entre os dois membros do grupo, uma vez que ambos ambicionavam aprender o máximo possível. Assim sendo, a contribuição individual de cada um é de 50%.

Uma vez que ambos os membros trabalharam com todas as tecnologias usadas, a responsabilidade da implementação de cada *feature* é repartida pelos dois, no entanto podemos dizer que a Márcia Pires se focou mais no *design* e usabilidade da aplicação, na integração da Google Books API e da Google Maps API, na utilização do Scanner e implementação da base de dados *Hive*. Por sua vez, o Tomás Martins focou-se mais na implementação do bluetooth que posteriormente foi substituída pela implementação de uma ligação peer-2-peer, através da combinação do uso de bluetooth, wi-fi e localização, e na implementação da base de dados *Firebase*.

Figura 5.1: Márcia Pires

Figura 5.2: Tomás Martins

Capítulo 6

Tutorial

Para inicializar a aplicação, apenas é necessário correr o comando:

```
$ flutter run
```

Em seguida, um pequeno tutorial de boa utilização da aplicação final.

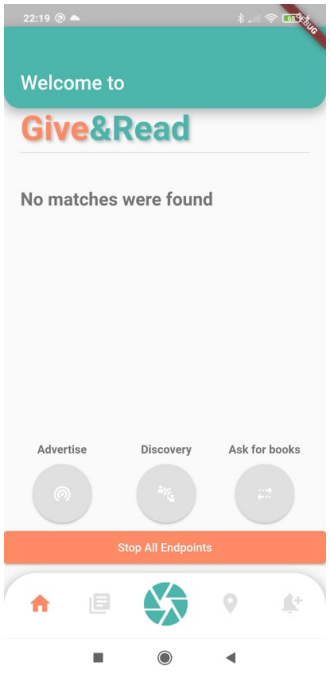


Figura 6.1: Página inicial

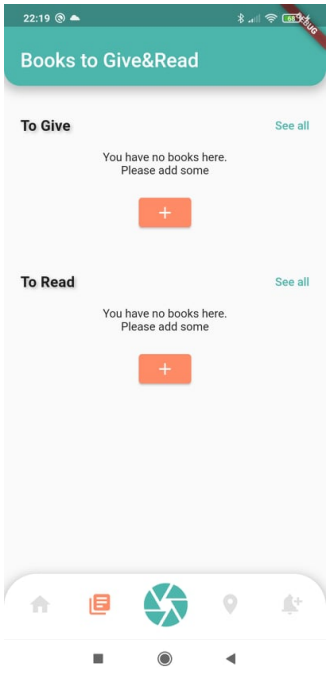


Figura 6.2: Aba de lista de livros To Give e To Read iniciais

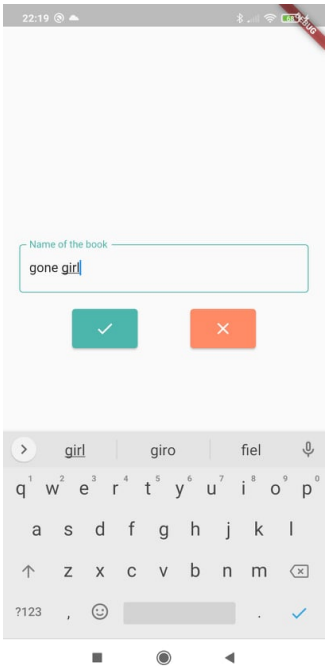


Figura 6.3: Adicionar livro a uma lista

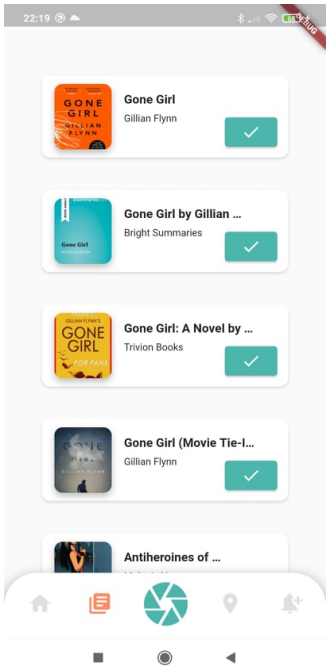


Figura 6.4: Resultado da pesquisa

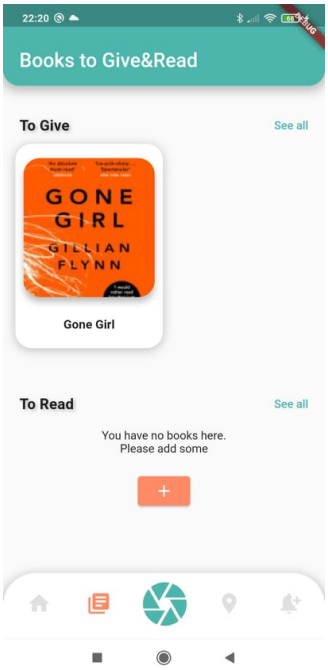


Figura 6.5: Livro adicionado à lista

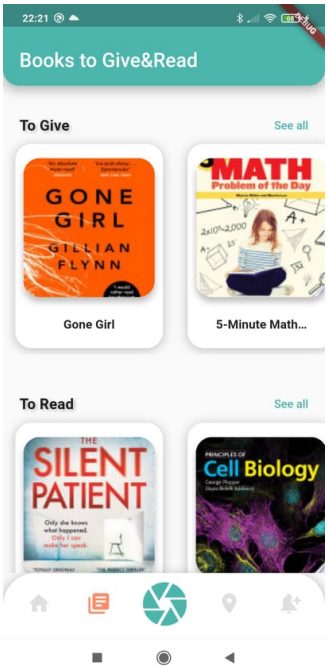


Figura 6.6: Listas mais completas

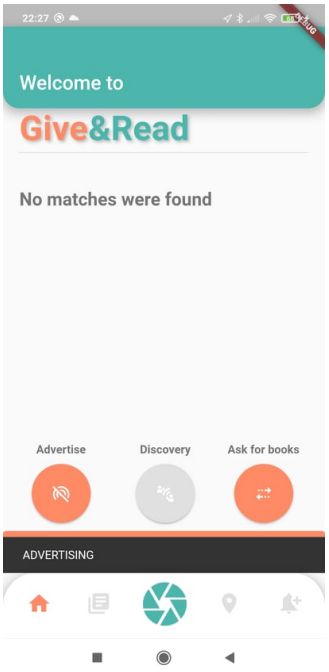


Figura 6.7: Conectar a dispositivos próximos: advertise/discovery

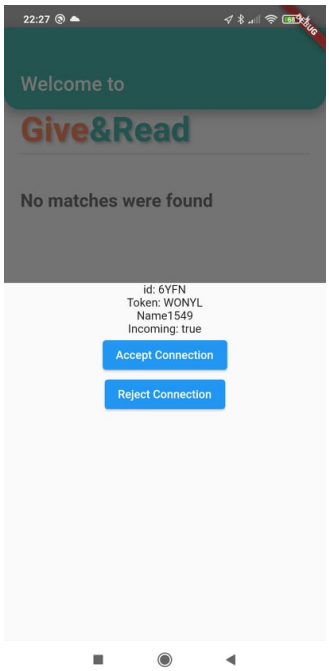


Figura 6.8: Encontrado dispositivo, aceitar conexão

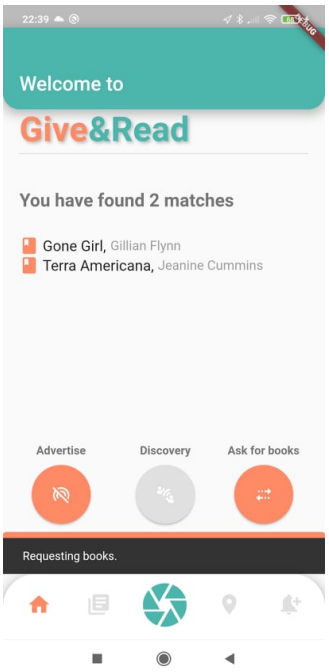


Figura 6.9: Ask for books para conhecer livros que posso dar

Figura 6.10: Verificar no mapa *bookstop* próxima

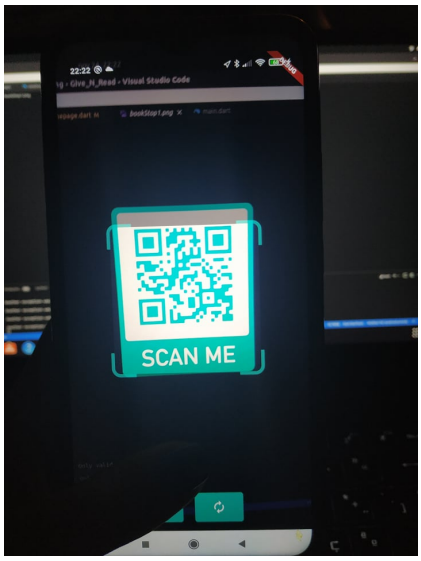


Figura 6.11: Na *bookstop*, scannear o QRCode para deixar livro

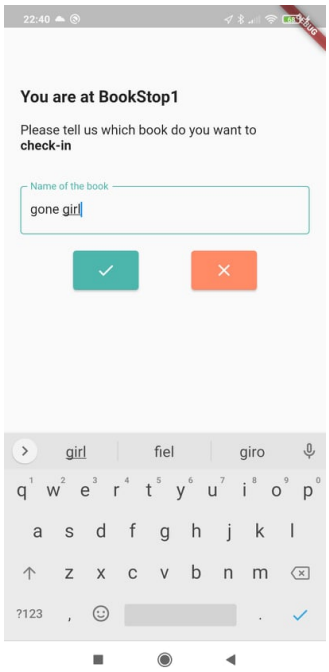


Figura 6.12: Identificar o livro que quer deixar

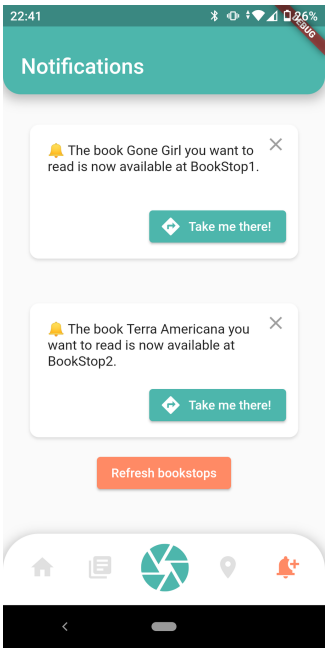


Figura 6.13: Utilizador 2 verifica na aba de notificações que o livro que quer ler, está disponível em determinada *bookstop*

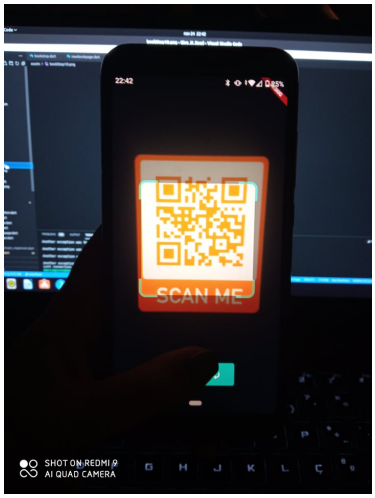


Figura 6.14: Utilizador 2, já na *bookstop*, faz scan do QR para levar o livro consigo

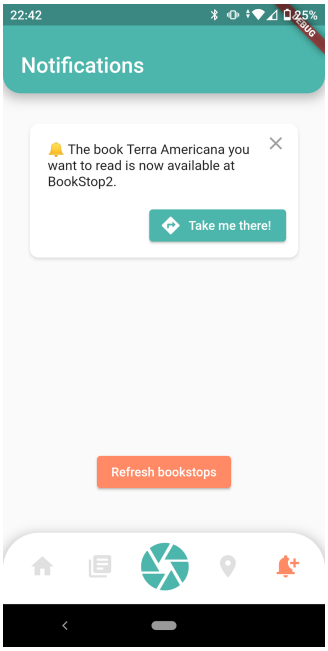


Figura 6.15: Utilizador 2, verifica na aba de notificações que o livro do qual fez *check-out*, já não se encontra na *bookstop*