universidade de aveiro

Introdução à Computação Móvel

Projeto Flutter

GroNow - Grupo 14

4 de maio de 2022

Motivation

O objetivo deste trabalho era implementar um sistema que permitisse ao utilizador encomendar e receber em sua casa encomendas de produtos alimentares de forma segura utilizando uma verificação quer por parte do entregador quer do cliente. Tem funcionalidades influenciadas por aplicações como a “UberEats” ou a “Glovo”.
Originalmente foram decididas algumas funcionalidades chave, sendo elas:

- **GPS-** A localização do entregador será partilhada com a pessoa que fez a encomenda. A localização do utilizador também será utilizada para determinar o local da entrega.
- **Mapa-** O mapa com a localização da entrega será disponibilizado ao entregador e a localização deste será disponibilizada ao utilizador que encomendou de forma a fazer *tracking* do entregador.
- **Bluetooth(beacons)-** A aplicação do entregador servirá como um *beacon* que servirá para dizer ao utilizador quando o entregador está perto do mesmo, gerando assim uma notificação ao utilizador.
- **Camera/QRCodes-** O utilizador utilizará a sua câmera para fazer scan de um QR Code, confirmando assim a entrega.
- **External Resources-** Uma API externa será utilizada para gerar os produtos da aplicação.
- **Notification**- O utilizador será notificado quando o entregador estiver perto.

Solution

A nossa solução implementa todas as funcionalidades decididas originalmente com a exceção de uma: os *External Resources*. Relativamente aos *External Resources*, a API externa não foi implementada. Após uma pesquisa intensiva de APIs relacionadas com o tema da nossa aplicação, apenas uma pequena seleção apresentava planos gratuitos para a utilização da mesma e, mesmo aquelas que apresentavam, caiam dentro de uma dos dois casos: não tinham as informações completas para a utilização na nossa aplicação; após fazer as chamadas, elas retornavam apenas um erro ou retornavam vazias.

Quanto à aplicação, esta conta com um ecrã de Boas Vindas, sendo este disponibilizado sempre que se inicia a aplicação. Após este, é-nos fornecida a página de Login (ou Registo, caso não tenhamos uma conta na aplicação) que também será fornecida sempre que a aplicação iniciar, por motivos de segurança. É no ficheiro **main.dart** onde isto tudo é inicializado. Para além destas, a aplicação possui diversas outras páginas, tal como a página do *Cart*, *Map*, *Check out*, a página de detalhes de um determinado produto e a página de perfil.

Voltando às funcionalidades, mais especificamente, o sistema de beacons.

Beacon

Para conseguir obter esta funcionalidade, foi utilizado o pacote ***flutter beacon***, assim como o ***flutter blue***, para tratar do bluetooth. Esta tem duas vertentes: o *scanner* e o *broadcaster*. Relativamente ao *scanner*, este foi atribuído à versão da aplicação do utilizador (aquele que encomenda). Ao realizar um pedido, a aplicação inicia a pesquisa por beacons, sendo apenas um específico que gera a notificação.

```
void startScanning(String courierName) async {
  if (await fb.isOn) {
    try {
      await flutterBeacon.initializeAndCheckScanning;
    } on PlatformException catch (e) {
      log("There was an error" + e.message.toString());
    }
    final regions = <Region>[];
    log("Scanning for beacons...");
    if (Platform.isIOS) {
      regions.add(Region(
          identifier: 'Apple Airlocate',
          proximityUUID: 'E2C56DB5-DFFB-48D2-B060-D0F5A71096E0'));
    } else {
      log("Device is Android");
      regions.add(Region(identifier: 'com.beacon'));
    }

    _streamRanging =
        flutterBeacon.ranging(regions).listen((RangingResult result) {});
    _streamRanging?.onData((data) {
      for (var i = 0; i < data.beacons.length; i++) {
        if (data.beacons[i].proximityUUID == courierName) {
          sendNotification();
        }
      }
    });
  }
}
```

Relativamente à versão do courier, este utiliza, para além do ***flutter blue***, o ***flutter broadcast***, que ajuda na "abertura" de um *beacon*.

```
void startBroadcast() async {
  if (await fb.isOn) {
    beaconBroadcast
        .setUUID("FF11EE22-DD33-CC44-BB55-AA6699778888")
        .setMajorId(1)
        .setMinorId(100)
        .start();
    log("Broadcasting");
  }
}
```

Map

Relativamente ao mapa, foi utilizado o pacote ***google maps flutter***, como havia sido sugerido em aulas práticas.

Neste foi necessário pedir permissões de utilização da localização para ambos os utilizadores, de forma a poder localizá-los no mapa.

```
void _onMapCreated(GoogleMapController controller) async {
  mapController = controller;
  Location location = Location();
  if (await Permission.location.serviceStatus.isEnabled) {
    if (!(await Permission.location.status.isGranted)) {
      location.requestPermission();
      setState(() {});
      if (await Permission.location.isPermanentlyDenied) {
        openAppSettings();
        setState(() {});
      }
    }
  } else {
    await location.requestService();
  }
}
```

Caso o utilizador rejeitasse permanentemente a utilização da localização, a aplicação levaria o mesmo às definições de modo a ativar a permissão, uma vez que esta é

necessária para a boa utilização da aplicação.

```
Future<LatLng> _getLocation() async {
  bool locationPermission = await Permission.location.status.isGranted;
  LatLng _center = const LatLng(40.64165860185367, -8.653554472528402);
  if (locationPermission) {
    Location loc = Location();
    try {
      LocationData _locData = await loc.getLocation();
      _center =
          LatLng(_locData.latitude as double, _locData.longitude as double);
    } on Exception catch (exc, e) {
      log("THERE WAS AN EXCEPTION: " + exc.toString() + " " + e.toString());
    }
  }
  return _center;
}
```

Após esta função executar e após termos a permissão da localização podemos então proceder ao pedido.

QR Code

No que toca à funcionalidade da câmera, utilizando o pacote ***qr_code_scanner*** foi possível criar acessar a câmera logo com a funcionalidade de ler os QR Codes. Foi criado um controller que está em modo “listening” por um código de barras (Qr codes são interpretados desta forma) e assim que o controller detectar algo deste tipo na câmera atualiza o estado e mostra o resultado no fundo da página. Nesta imagem podemos ver todo o processo envolvido na criação de uma área de scan de Qr Codes.

```dart
@override
Widget build(BuildContext context) => SafeArea(
      child: Scaffold(
        appBar: AppBar(
          title: const Text("Courier Beacon"),
          backgroundColor: AppColors.primaryColor,
        ), // AppBar
        body: Stack(
          alignment: Alignment.center,
          children: <Widget>[
            buildQrView(context),
            Positioned(bottom: 10, child: buildResult()),
            Positioned(top: 10, child: buildButtons())
          ], // <Widget>[]
        ), // Stack
      ), // Scaffold
    ); // SafeArea
Widget buildQrView(BuildContext context) => QRView(
      key: qrKey,
      onQRViewCreated: onQRViewCreated,
      overlay: QrScannerOverlayShape(),
    ); // QRView

void onQRViewCreated(QRViewController controller) {
  setState(() => this.controller = controller);
  controller.scannedDataStream
      .listen((barcode) => setState(() => this.barcode = barcode));
}
```

Disponibilizamos também a funcionalidade de ativar e desativar o flash através de um botão no topo do ecrã como se pode ver na seguinte imagem .

```dart
Widget buildButtons() => Container(
      padding: EdgeInsets.symmetric(horizontal: 16),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(10),
        color: Colors.white24,
      ), // BoxDecoration
      child: Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: <Widget>[
          IconButton(
            onPressed: () async {
              await controller?.toggleFlash();
              setState(() {});
            },
            icon: FutureBuilder<bool?>(
                future: controller?.getFlashStatus(),
                builder: (context, snapshot) {
                  if (snapshot.data != null) {
                    return Icon(
                        snapshot.data! ? Icons.flash_on : Icons.flash_off); // Icon
                  } else {
                    return Container();
                  }
                }), // FutureBuilder
```

Notifications

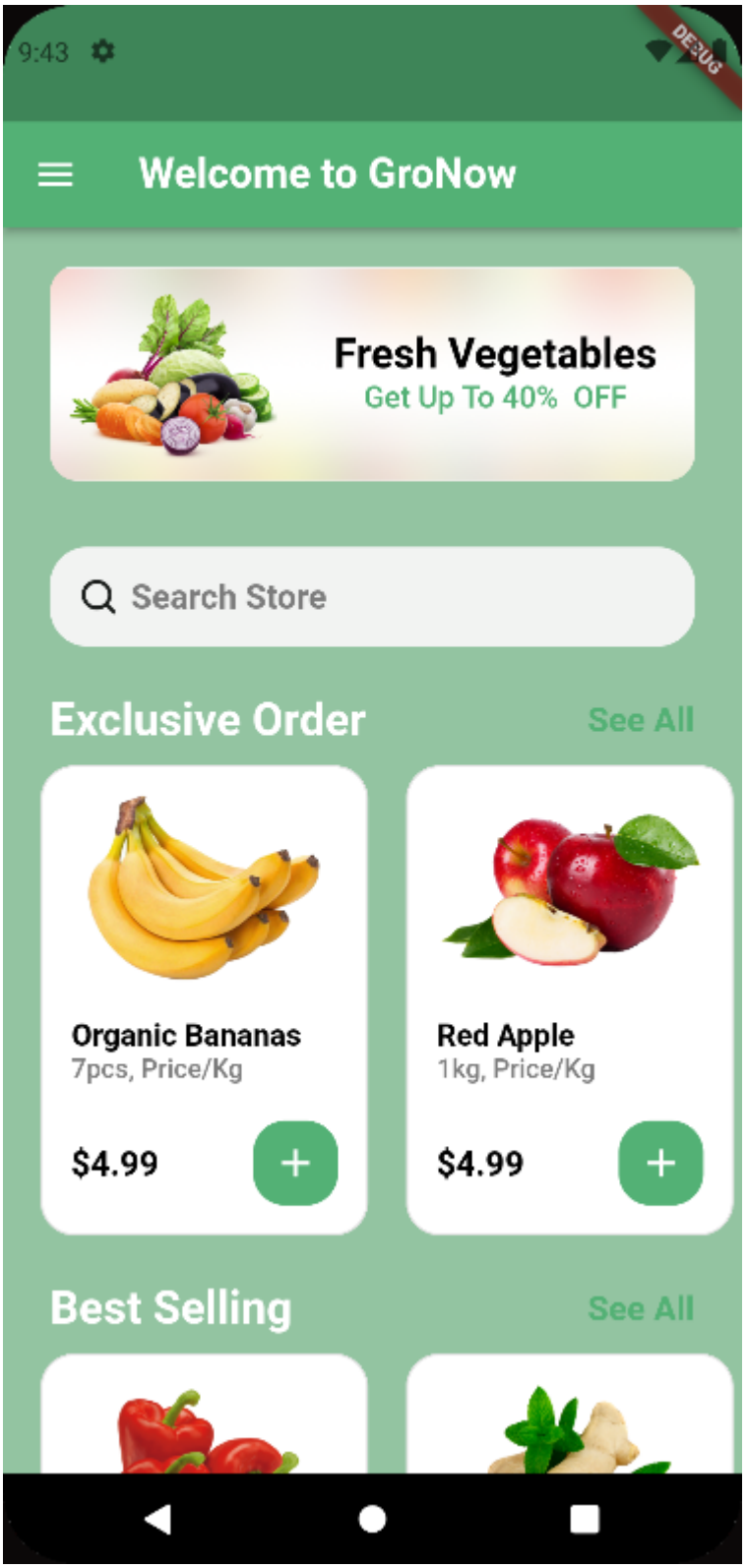
Conseguimos também implementar notificações como parte das funcionalidades dos beacons quando o utilizador se aproxima do beacon do entregador. Quando isto acontece o utilizador recebe uma notificação a dizer a sua encomenda está próxima. O pacote utilizado

foi o ***awesome notifications***. Este não está completamente desenvolvido, mas fornece as funcionalidades necessárias para o que tínhamos em mente.

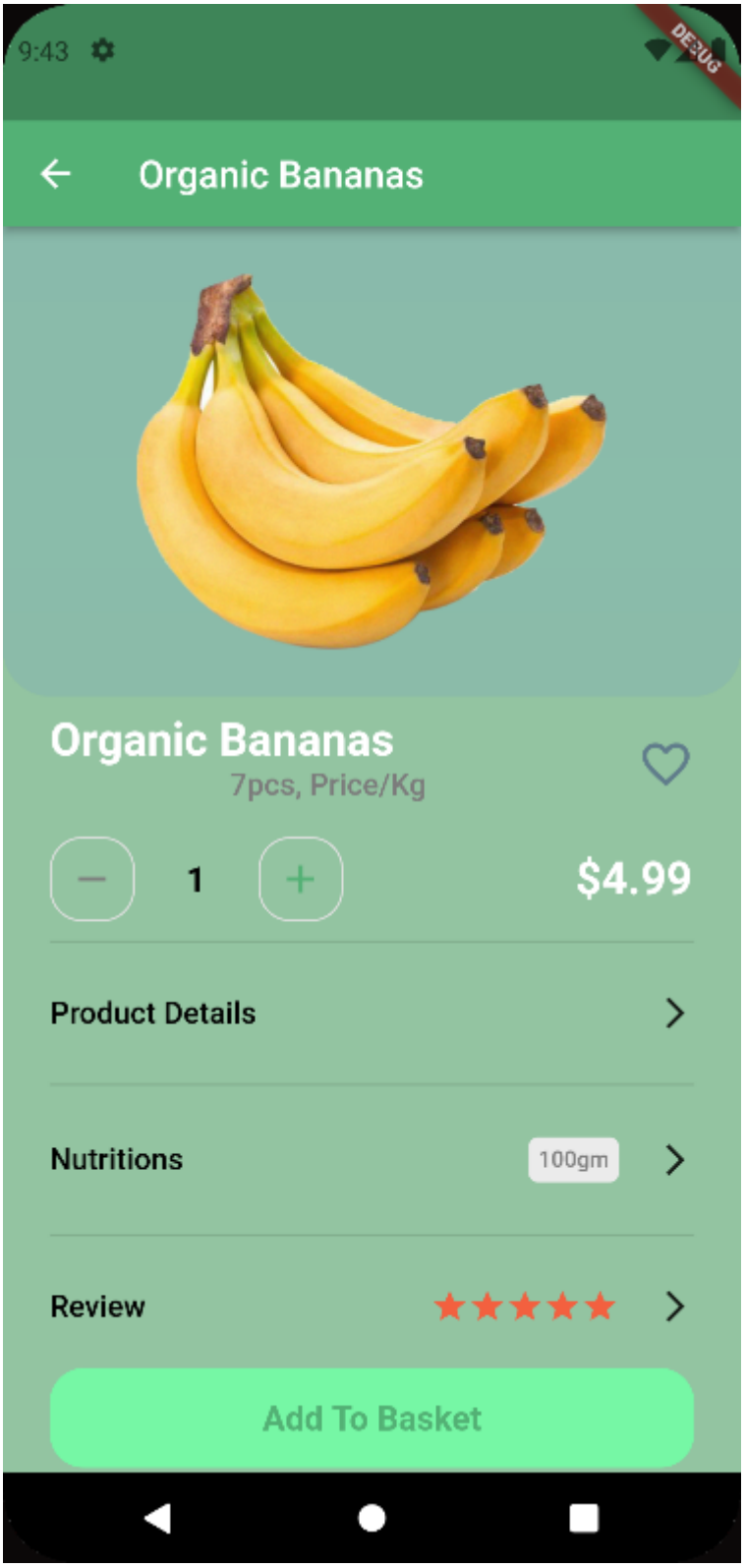
```
sendNotification() {
  AwesomeNotifications().createNotification(
      content: NotificationContent(
          id: 10,
          channelKey: 'basic_channel',
          title: 'GroNow - Arrived',
          body: 'Your Courier just arrived, go there!'));
}
```

Tutorial

Tendo em conta o público alvo desta aplicação, ela foi feita de forma a ser simples. Saltando a página de Login e de Registo, comecemos pela página inicial.

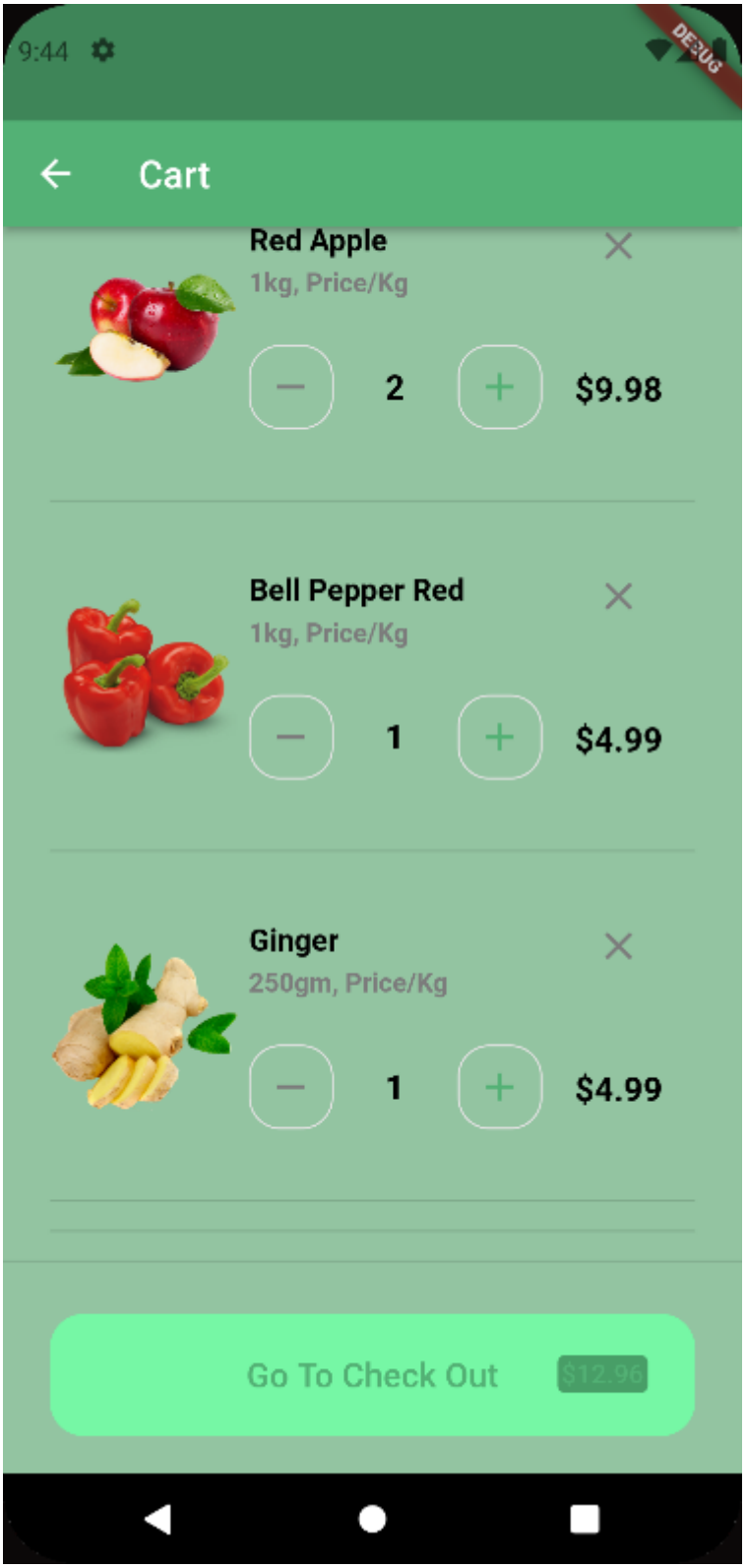


Nesta são apresentados os produtos mais comuns e mais relevantes para os clientes. Ao clicarmos na *card* das bananas, somos levados à página dos detalhes.

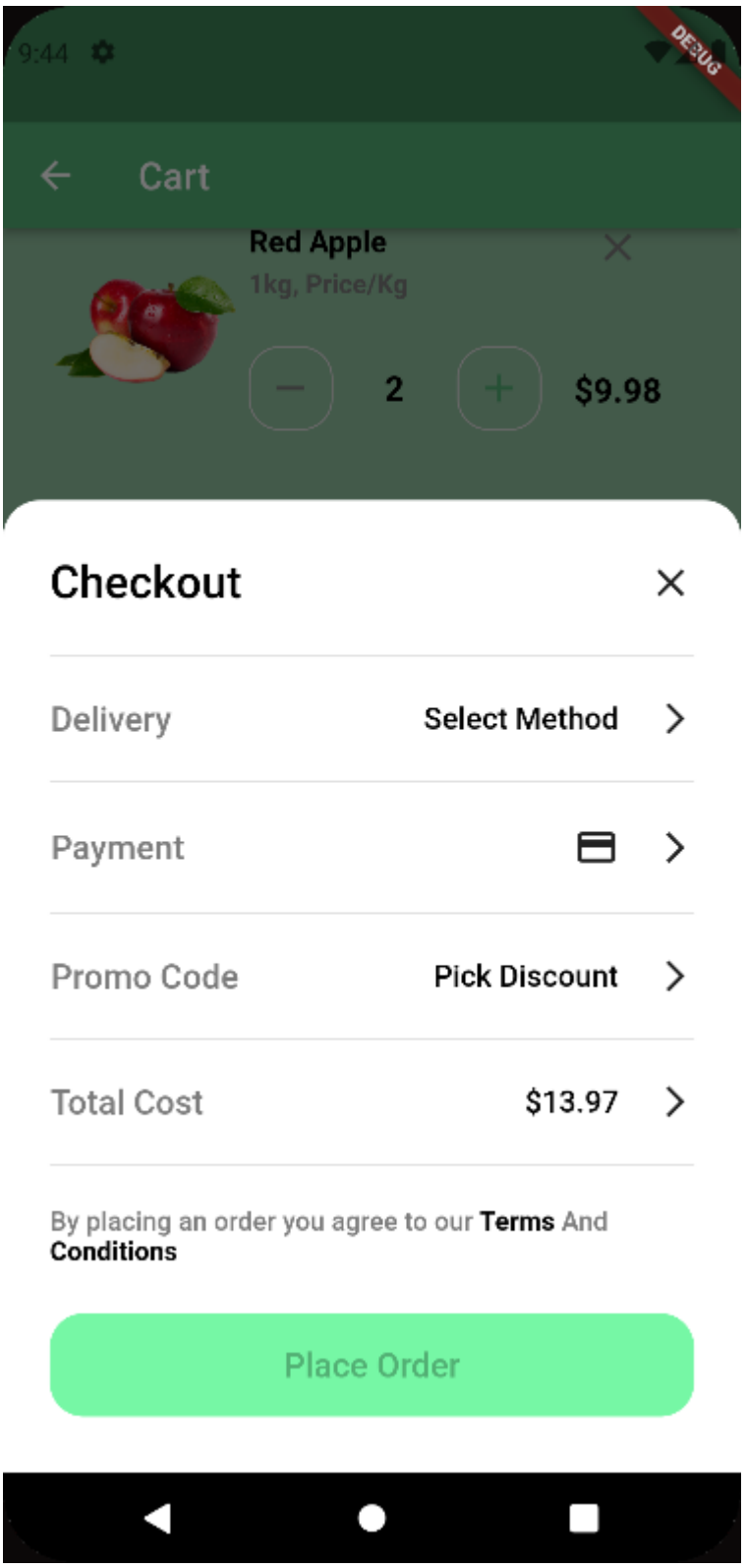


Mais uma vez, temos uma página relativamente simples, com o preço, a quantidade que o cliente deseja e um botão para adicionar ao carrinho. O botão de coração era um teste que estava a ser realizado à altura que foi retirado o print.

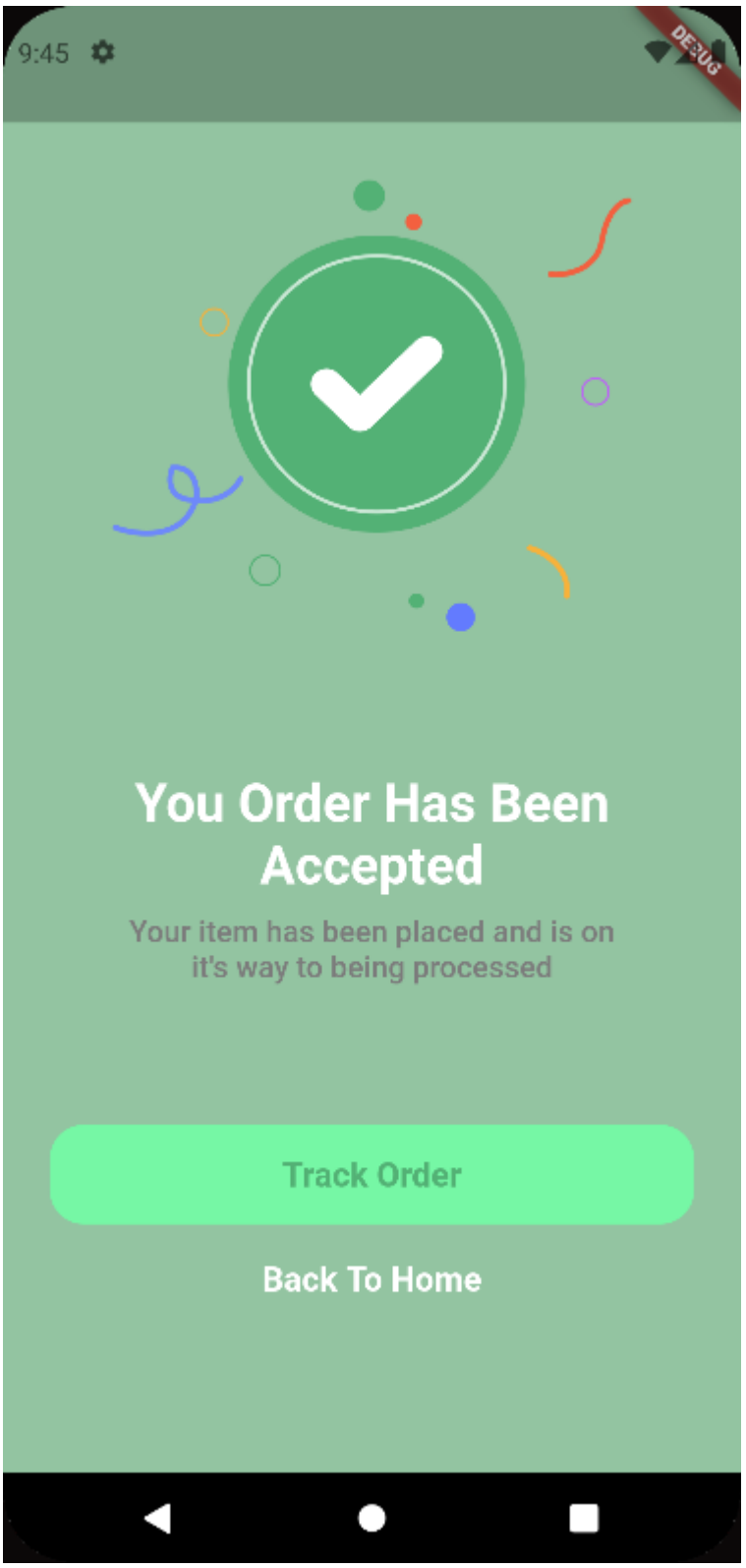
Passando ao carrinho, neste são apresentados os produtos que foram adicionados a este, assim como o preço total que o cliente vai pagar.



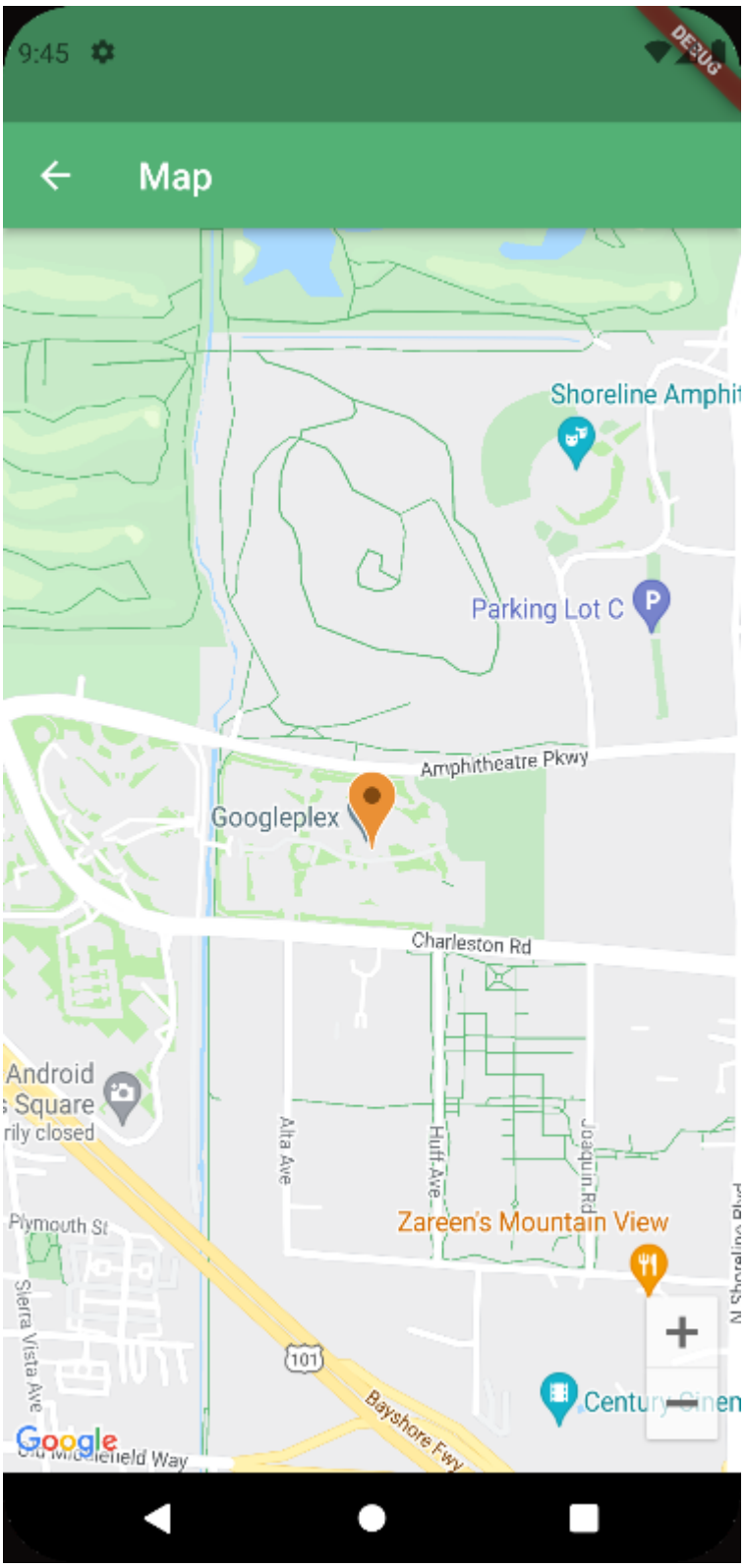
Na altura do print existiam ainda alguns bugs visuais que eventualmente foram melhorados. Ao clicarmos no botão do Check Out, é-nos apresentado uma seção na qual o utilizador pode selecionar os detalhes do mesmo.



Ao confirmar o pedido, somos levados a uma página de sucesso que, posteriormente, nos pergunta se queremos voltar à Homepage ou se queremos ir para o mapa.



Relativamente ao mapa, o seu layout é relativamente simples, não havendo muito a explicar.



Por último, temos a página da câmera, onde aparece um leitor de QR Codes, que irá ler o código da encomenda e que terminará a mesma.

Achievements and Issues

Conseguimos conectar dois telemóveis utilizando beacons e bluethoot, e também integrar completamente na aplicação a funcionalidade de ler códigos QR. A solução é

também capaz de localizar o utilizador no google maps, apesar de não termos conseguido ligar os dois pontos por uma rota.

Tivemos problemas a encontrar uma boa API para os produtos com a informação que necessitamos, e portanto consideramos o google Maps como o nosso recurso externo. Apesar dos produtos serem estáticos todas as interações que normalmente se fariam com uma API estão disponíveis e funcionais. O código está preparado para receber uma API adequada apesar de não termos encontrado uma.

Contribution

Ambos trabalhamos com o mesmo grau de esforço neste projeto, e portanto achamos justa a divisão de 50/50 na avaliação.

Conclusion

Com este projeto conseguimos com sucesso aprender a integrar ferramentas numa aplicação mobile bem como desenvolver uma UI bastante satisfatória. Apesar de não ter todas as funcionalidades que queríamos integrar, o resultado final está dentro das nossas expectativas.

Além disso, com este trabalho aprendemos a utilizar o flutter com uma profundidade bastante razoável, consideramos a aprendizagem desta ferramenta uma mais valia no nosso percurso.

Team

Paulo Pereira 98430

Fábio Martins 98119