

Arquiteturas Móveis



Relatório Trabalho Prático – Flutter

Realizado por:

Diogo Oliveira – 2021146037

Lara Bizarro – 2021130066

Tomás Laranjeira – 2021135060

Índice

1. Introdução	3
2. Ecrãs.....	4
2.1. main.....	4
2.2. add_contact_screen	5
2.3. contact_details_screen	6
2.4. edit_contact_screen	7
2.5. list_contact_screen	8
3. Class Contact.....	9
4. Decisões tomadas.....	10
4.1. Persistência de dados.....	10
4.2. Histórico de contactos modificados.....	10
4.3. Obtenção de localização	10
4.4. Visualização do mapa	10

1. Introdução

O presente relatório descreve o desenvolvimento de uma aplicação móvel em Flutter, que tem como objetivo a gestão de contactos pessoais, oferecendo funcionalidades para armazenamento e consulta de informações básicas, como nome, e-mail, telefone e data de nascimento.

A aplicação permite ao utilizador associar imagens aos contactos, seja a partir da galeria do dispositivo ou por meio da câmara, bem como registar e consultar as localizações geográficas dos encontros com cada contacto.

Adicionalmente, inclui-se a funcionalidade de manter um histórico autónomo dos 10 últimos contactos modificados ou com novas localizações adicionadas, promovendo uma gestão eficiente e organizada.

Esta aplicação é uma adaptação da versão originalmente desenvolvida em Kotlin durante as aulas práticas.

2. Ecrãs

2.1. main

Este ecrã mostra uma lista de todos os contactos guardados. Ao clicar no icon de lista, aparecem mais informações dos contactos. É também a partir deste ecrã que se adiciona um novo contact ou se vê o histórico dos ultimos contactos modificados.

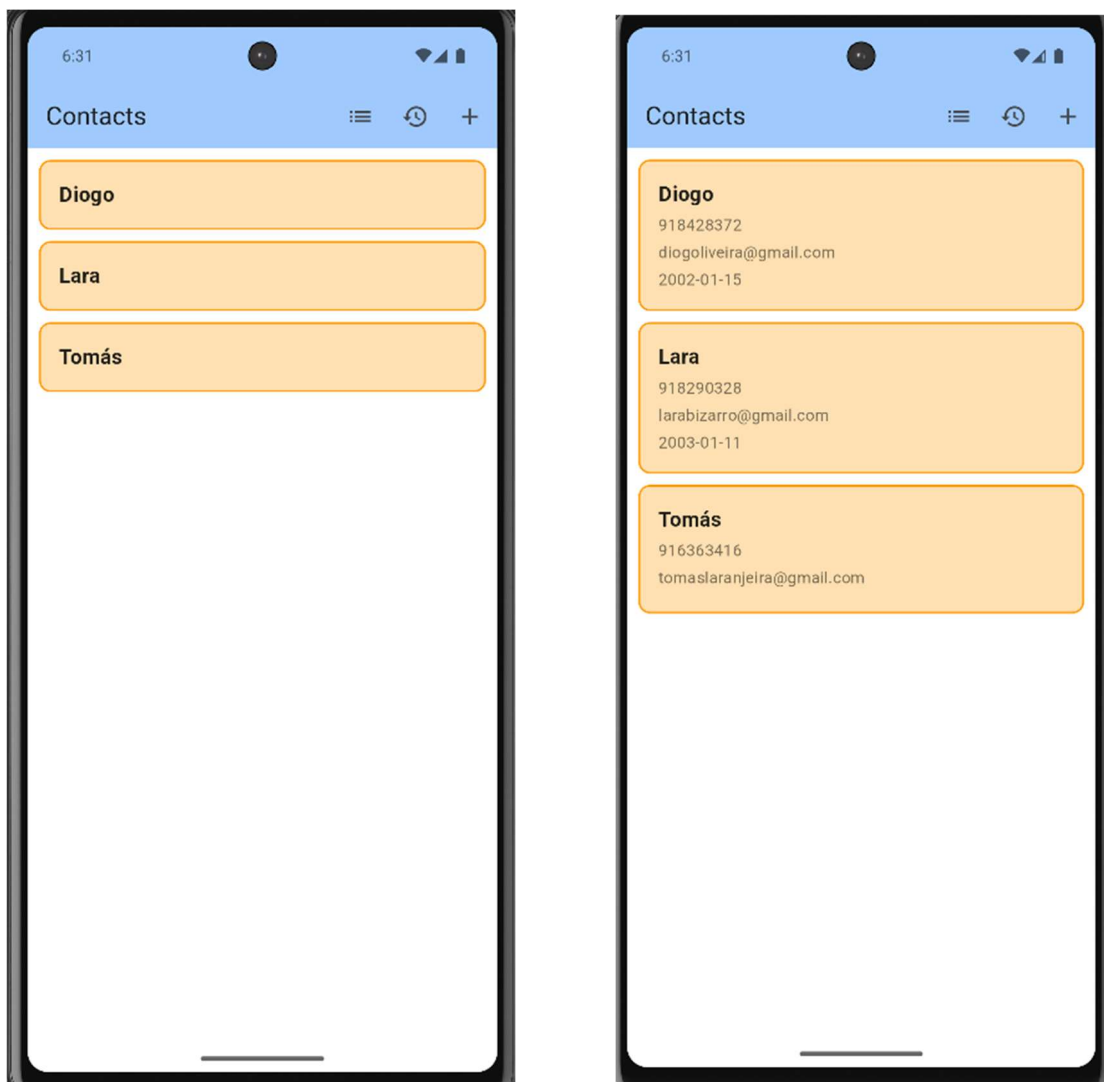
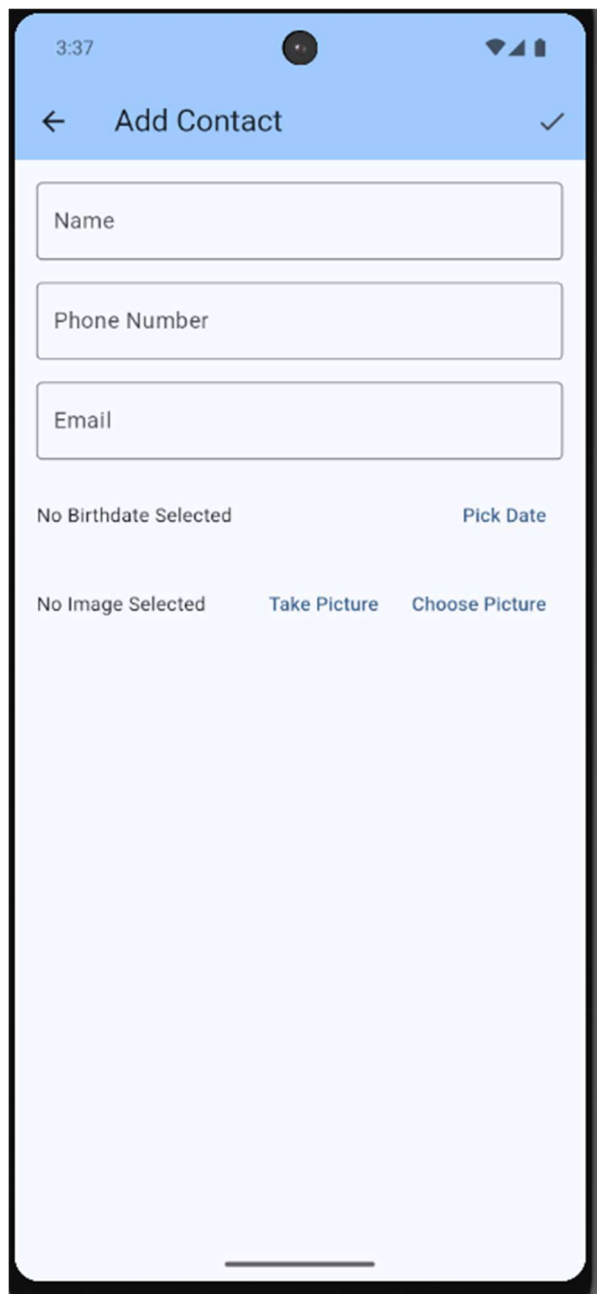


Figura 1- main

2.2. add_contact_screen

Este ecrã apresenta os utilizadores com os vários campos a preencher para criar um novo contacto. Os campos *Name*, *Phone Number* e *Email* são de preenchimento obrigatório enquanto que a seleção de data de nascimento e de imagem são opcionais.



3:37

← Add Contact ✓

Name

Phone Number

Email

No Birthdate Selected Pick Date

No Image Selected Take Picture Choose Picture

Figura 2 - add_contact_screen

2.3. contact_details_screen

Este ecrã mostra toda a informação relativa a um contacto, incluindo um mapa com marcadores nas localizações em que o utilizador se encontrou com este contacto, para guardar a localização atual basta clicar no icone de marcador no canto superior direito.

Caso o utilizador queira editar alguma informação relativa ao contacto basta clicar no icone de edição no canto superior direito o que o levará para um diferente ecrã.

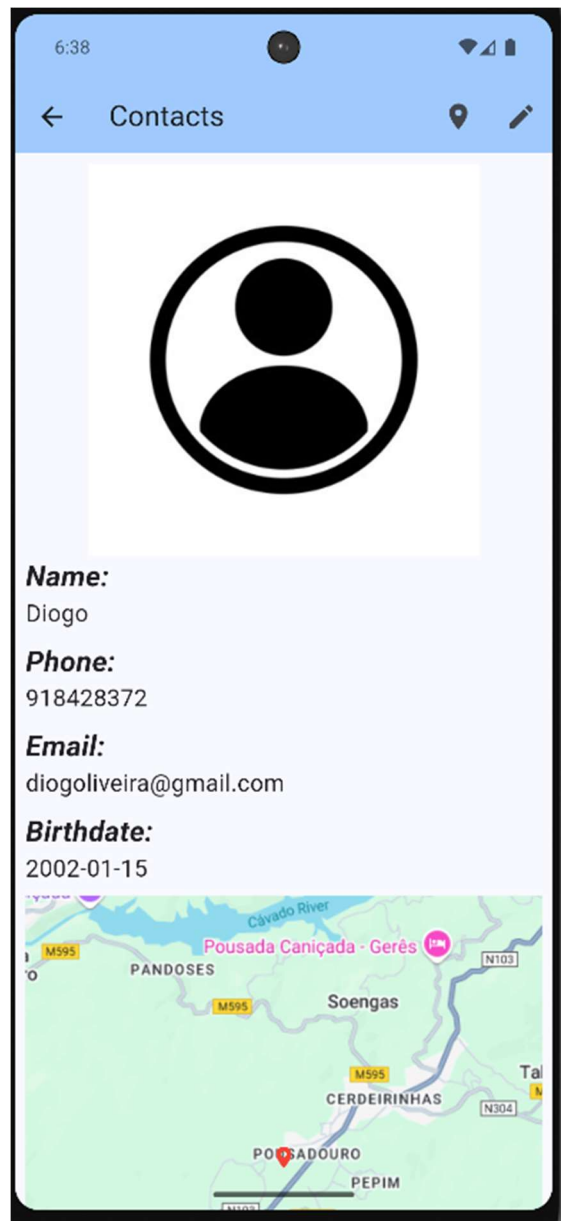


Figura 3 - contact_details_screen

2.4. edit_contact_screen

Este ecrã é semelhante ao *add_contact_screen*, no entanto os campos já aparecem preenchidos com a informação pré-existente.

4:04

← Edit Contact ✓

Name
Diogo

Phone Number
916229321

Email
diogoliveira@gmail.com

Birthdate: 2002-01-15 Pick Date

No Image Selected Take Picture Choose Picture

Figura 4 - edit_contact_screen

2.5. list_contact_screen

Este ecrã mostra uma lista dos ultimos contactos que foram modificados. Quer seja a alteração de um campo ou a inserção de uma nova localização.

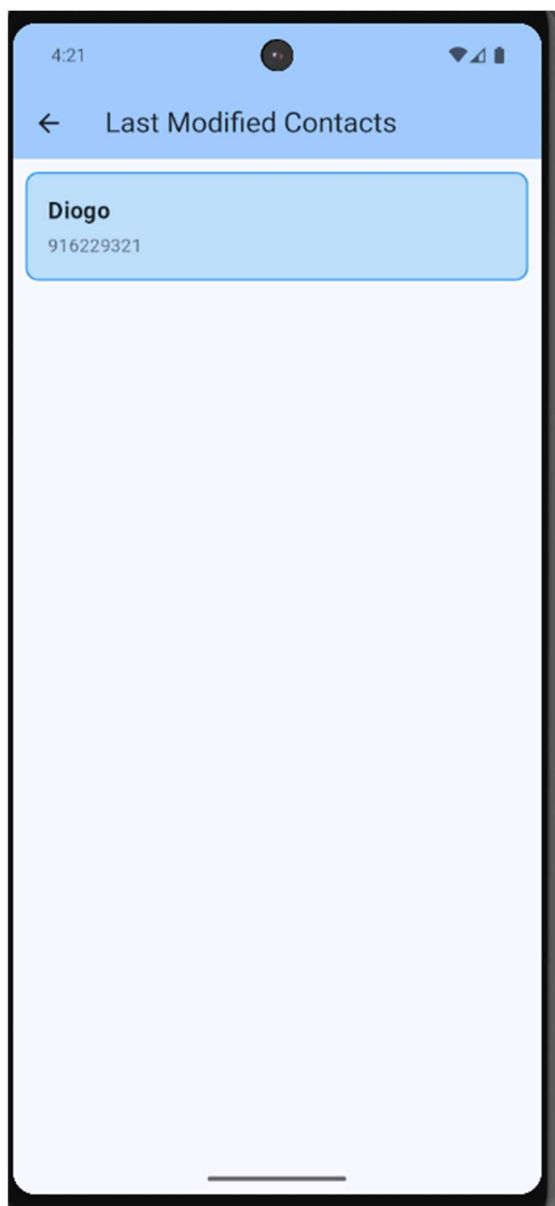


Figura 5 - list_contact_screen

3. Class Contact

Optámos por criar um ficheiro *contact.dart* apenas com a class *Contact* de forma a ser mais fácil a sua utilização em vários outros ficheiros.

Esta classe tem variáveis em que guarda as informações de cada contacto anteriormente referidas. Existem também métodos *toJson* e *fromJson* para auxiliar a guardar e obter os dados quando a aplicação é encerrada.

```
class Contact {
  final String name;
  final String number;
  final String email;
  final DateTime? birthdate;
  final String? imagePath;
  List<LatLng>? locationData;

  Contact({
    required this.name,
    required this.number,
    required this.email,
    this.birthdate,
    this.imagePath,
    this.locationData,
  });

  Map<String, dynamic> toJson() {
    return {
      'name': name,
      'number': number,
      'email': email,
      'birthdate': birthdate?.toIso8601String(),
      'imagePath': imagePath,
      'locationData': locationData?.map((LatLng) {
        return {'latitude': latLng.latitude, 'longitude': latLng.longitude};
      }).toList(),
    };
  }

  factory Contact.fromJson(Map<String, dynamic> json) {
    return Contact(
      name: json['name'],
      number: json['number'],
      email: json['email'],
      birthdate: json['birthdate'] != null
        ? DateTime.tryParse(json['birthdate'])
        : null,
      imagePath: json['imagePath'],
      locationData: json['locationData'] != null
        ? (json['locationData'] as List<dynamic>).map((location) {
            return LatLng(
              location['latitude'] as double,
              location['longitude'] as double,
            ); // LatLng
          }).toList()
        : [],
    ); // Contact
  }
}
```

Figura 6 - Class Contact

4. Decisões tomadas

4.1. Persistência de dados

Para o armazenamento de dados dos contactos optámos pela utilização de um ficheiro de formato json guardado na diretoria temporária do android.

4.2. Histórico de contactos modificados

Como indicado no enunciado para a criação do histórico de contactos modificados foi utilizada a biblioteca *shared_preferences*.

4.3. Obtenção de localização

Apesar do enunciado pedir a utilização da biblioteca *location* para a obtenção de localização, durante as aulas práticas o docente informou que também seria aceite o uso da biblioteca *geolocator*, por isso optámos pela utilização da alternativa.

4.4. Visualização do mapa

Inicialmente optámos pela utilização da biblioteca *google_maps_flutter*, no entanto tivemos dificuldades na implementação o que nos levou à utilização da biblioteca *flutter_map*.

5. Conclusão

O desenvolvimento da aplicação "Contacts" em Flutter permitiu consolidar os conceitos abordados nas aulas práticas, adaptando uma solução inicial em Kotlin para Flutter. A aplicação criada cumpre os requisitos estabelecidos, proporcionando ao utilizador uma interface intuitiva e funcionalidades avançadas, como a gestão de contactos com integração de localização e suporte de imagens.

Um dos principais desafios enfrentados durante o desenvolvimento foi a apresentação do mapa na aplicação. Apesar da liberdade na escolha das bibliotecas para esta funcionalidade, ajustar as marcações das localizações e garantir uma integração fluida com a interface revelou-se mais complexo do que o previsto.