

---

# **Aplicações Móveis**

## **Trabalho Prático 2 - Flutter**

---

### **Professor Responsável**

Álvaro Nuno Ferreira Silva Santos

### **Alunos**

Bruno Trindade - 2019132612

Guilherme Camacho - 2021138502

Tiago Figueiredo - 2020122664

5 de janeiro de 2025

## ÍNDICE

1	Introdução . . . . .	3
2	Organização . . . . .	4
2.1	Estrutura de Ficheiros . . . . .	4
2.2	Estrutura da Aplicação . . . . .	4
3	Funcionalidades . . . . .	11
3.1	Reutilização de componentes . . . . .	11
3.2	Persistência de Dados . . . . .	11
3.3	Gestão de Estado . . . . .	11
3.4	Integração com Serviços de Localização . . . . .	11
3.5	Design Responsivo . . . . .	12
3.6	Funcionalidades-Chave . . . . .	12

## 1 INTRODUÇÃO

Este relatório tem como objetivo facilitar a compreensão das implementações do que foi realizado para o Trabalho Prático de Flutter de Aplicações Móveis.

A aplicação permite criar, visualizar, editar e gerir **Contatos**. Inclui funcionalidades como armazenamento de informações de contato, rastreamento de localização, manipulação de imagens e exibição de um histórico de contatos recentemente editados.

## 2 ORGANIZAÇÃO

### 2.1 Estrutura de Ficheiros

A estrutura de ficheiros fica dividida em dois grandes **packages** que são:

- **models/** - Contém estruturas de dados e View Models:
  - **Contact.dart** - Modelo central para gestão de contactos.
  - **viewmodels/contact\_viewmodel.dart** - Lógica e estados para gestão dos contactos.
- **screens/** - Contém as interfaces principais da aplicação:
  - **MainScreen.dart** - Vista principal com a lista de contactos.
  - **CreateContactScreen.dart** - Vista para criar ou editar contactos.
  - **ViewContactScreen.dart** - Vista para detalhes de um contato específico.
  - **RecentlyEditedScreen.dart** - Vista para listar os contactos modificados recentemente.
- **services/** - Contém serviços utilitários:
  - **LocationService.dart** - Serviço para manipulação de localização.
  - **StorageService.dart** - Persistência de dados através do armazenamento local.
  - **NavigationService.dart** - Navegação entre as diferentes telas.
- **widgets/** - Componentes reutilizáveis para UI:
  - **ContactCard.dart** - Widget para exibição de cada contato na lista.

### 2.2 Estrutura da Aplicação

A aplicação utiliza a arquitetura **Model-View-ViewModel (MVVM)**:

- **Models** - Estruturas de dados para representar os contactos (ex.: **Contact**).
- **Views** - Interfaces e componentes visuais.
- **ViewModels** - Camada de gestão de estado e lógica comercial (ex.: **ContactViewModel**).

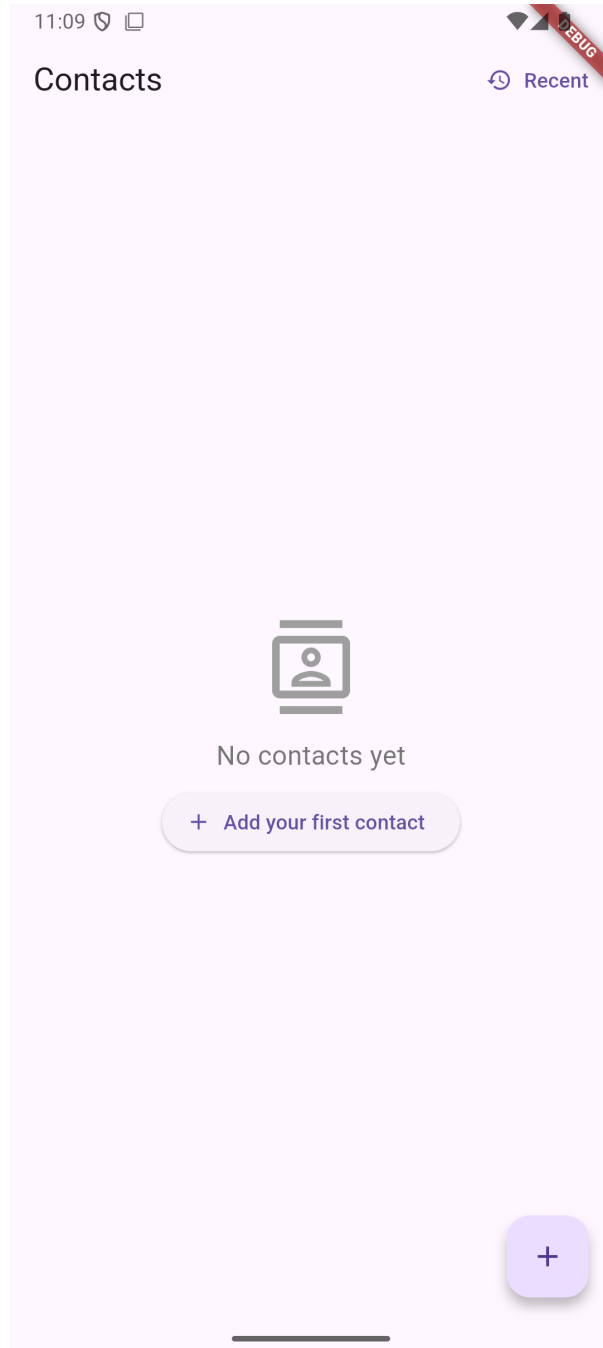


Figura 2.1: Contacts Screen

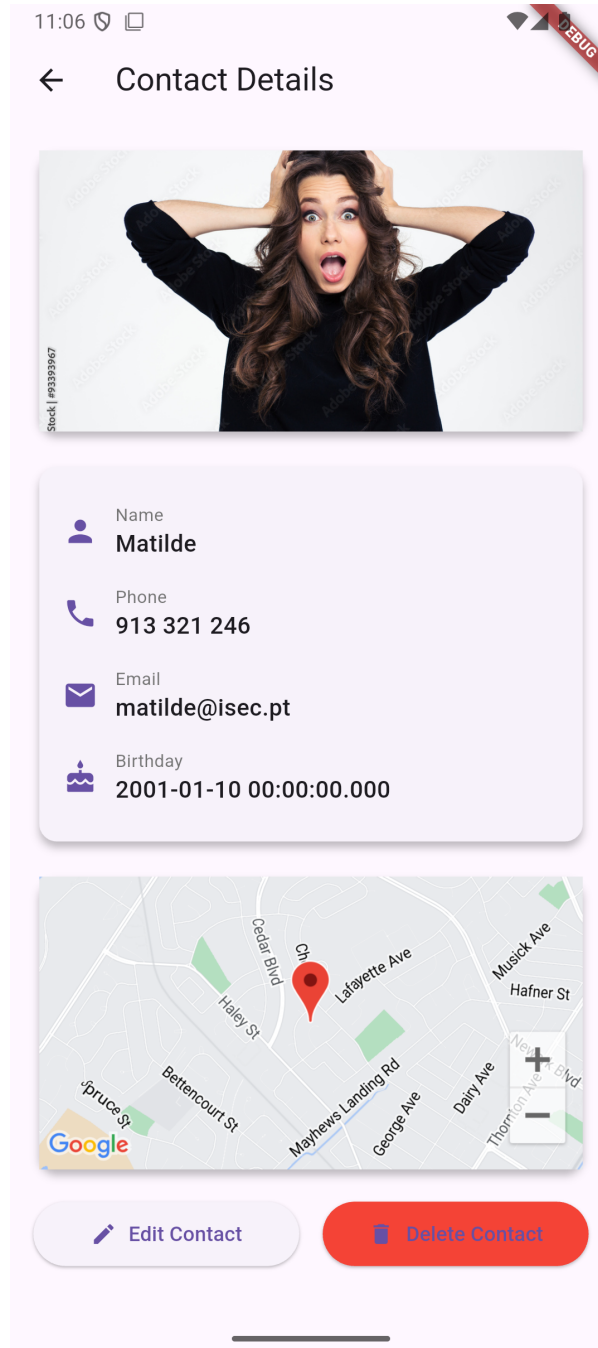






Figura 2.2: Details Screen

11:04    

← **Create Contact**



Name


 Matilde


Phone

 913 321 246

Email

 matilde@isec.pt

 Birthday: 2001-01-10

 Add Current Location

Location: Lat 37.5387, Long -122.0443

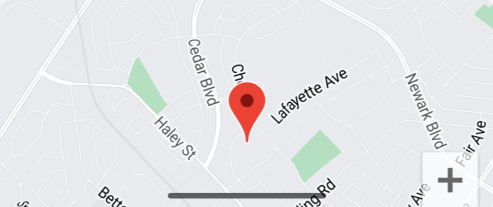


Figura 2.3: Create Screen

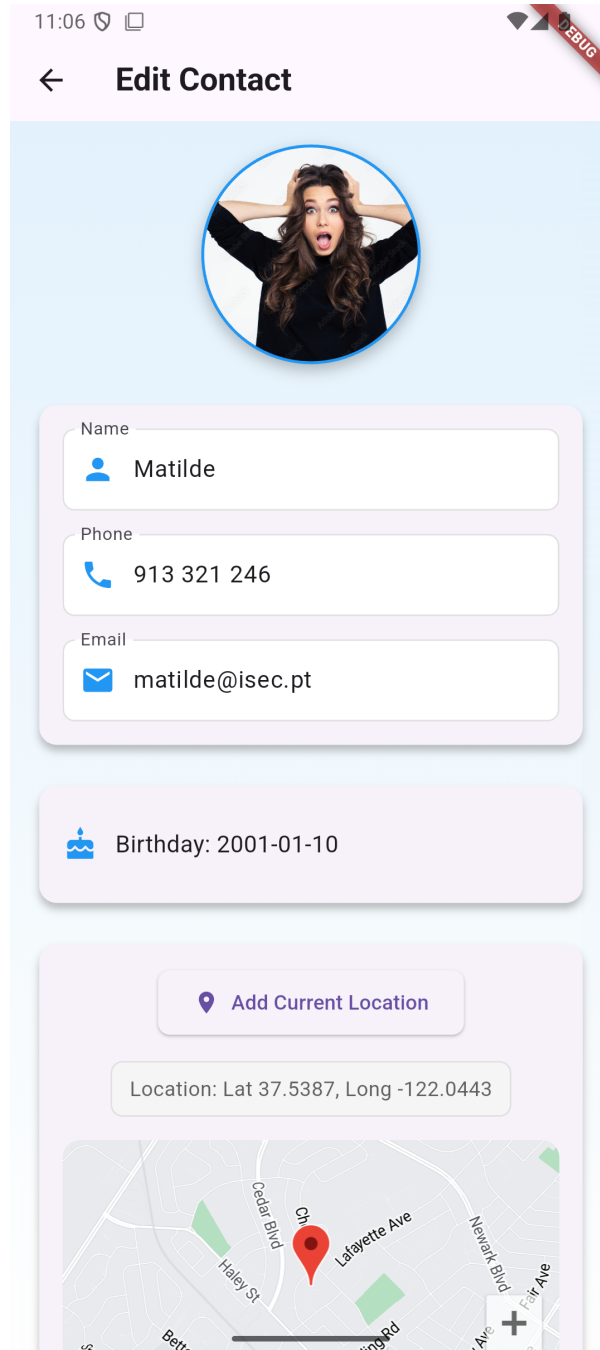


Figura 2.4: Edit Screen



## Trabalho Prático - Flutter



Figura 2.5: Recently Contacts Screen

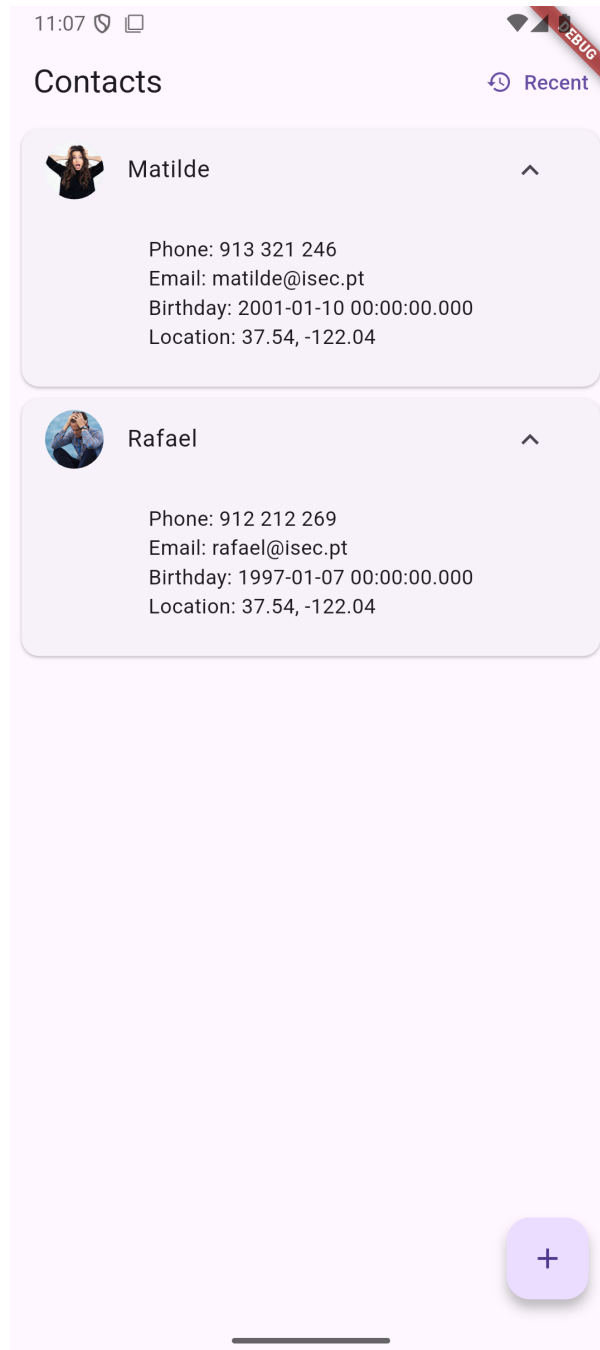


Figura 2.6: Recently Expanded Contacts Screen

## 3 FUNCIONALIDADES

### 3.1 Reutilização de componentes

A aplicação prioriza a criação de componentes reutilizáveis:

- **ContactCard** - Widget uniforme para exibir cada contato de forma consistente.
- Serviços de navegação partilhados para facilitar a transição entre telas.
- Campos reutilizáveis para formulários e layouts.

### 3.2 Persistência de Dados

Os dados são armazenados localmente utilizando **SharedPreferences**:

```
static Future<void> saveContacts(List<Contact> contacts) async {  
    final prefs = await SharedPreferences.getInstance();  
    final List<Map<String, dynamic>> contactJsonList = contacts.map((contact) =>  
        await prefs.setString('contacts', json.encode(contactJsonList)));  
}
```

### 3.3 Gestão de Estado

A aplicação utiliza o **ChangeNotifier** para a gestão do estado de forma reativa:

```
class ContactViewModel extends ChangeNotifier {  
    List<Contact> _contactsList = [];  
    // ... métodos de gestão de estado  
    void toggleCard(String contactId) {  
        // ... lógica para alternar estados  
        notifyListeners();  
    }  
}
```

### 3.4 Integração com Serviços de Localização

A aplicação permite a captura da localização através de permissões no dispositivo:

```
static Future<Position?> getCurrentLocation() async {  
    bool serviceEnabled = await Geolocator.isLocationServiceEnabled();  
    // ... lógica de manipulação de localização  
}
```

## 3.5 Design Responsivo

Adapta o layout para orientações diferentes do ecrã (paisagem ou retrato):

```
bool isLandscape = MediaQuery.of(context).orientation == Orientation.landscape;  
// Adaptação condicional do layout com base na orientação
```

## 3.6 Funcionalidades-Chave

### Gestão de Contactos

- Criação de novos contactos.
- Edição de contactos existentes.
- Eliminação de contactos.
- Visualização dos detalhes de cada contato.

### Manipulação de Imagens

- Escolha de foto de perfil.
- Armazenamento e apresentação da imagem do contato.

### Funcionalidades de Localização

- Captura da localização atual.
- Exibição da localização no mapa.
- Salvamento da localização junto ao contato.

### Atividades Recentes

- Listagem de contactos recentemente editados.
- Acesso rápido às últimas modificações.

### Persistência de Dados

- Armazenamento local das informações dos contactos.
- Histórico de contactos recentes armazenado para consultas futuras.