
Arquiteturas Móveis
2022/2023

# Trabalho Prático II

## Grupo 11

# Relatório Técnico

Realizado por
Diogo Pascoal 2018019825
Leonardo Sousa 2019129243
Rodrigo Costa 2020133365

# Índice

# 1. Introdução

Desenvolvimento de uma aplicação em Flutter simples, para a apresentação da ementa semanal da cantina do ISEC, tirando partido de uma API facultada aos alunos.

Esta aplicação permite consultar a ementa semanal, apresentando para cada dia informação sobre: sopa, prato de peixe, prato de carne, prato vegetariano e sobremesa. Também permite que um utente altere o menu previsto, submetendo uma atualização da informação presente na aplicação.

Do mesmo modo, na ementa é possível confirmar se um dos dias foi atualizado, sendo destacado a verde uma ementa que tenha sido atualizada.

Está implementado nesta aplicação a persistência da última ementa recebida pela API, mas aso o utilizador queira garantir que tem a informação mais recente do servidor, o mesmo pode manualmente atualizar a lista.

Para além de obter as imagens do servidor, esta aplicação também permite atualizar a imagem da ementa, assim como apenas permitir que o utilizador edite a informação caso se encontre perto da cantina.

# 2. Ecrãs

## 2.1. main

Este é apenas um ecrã de introdução, com um botão para acesso às ementas e a um pequeno ecrã de créditos.


*Figura 1 - main*

## 2.2. ementa_screen

Ecrã principal deste trabalho prático, conta com ScrollView que contém, para além de Divider's e algumas SizedBox para formatação, uma ListView com a ementa disponível.


*Figura 2 - Ementa Semanal*

## Classes

Para a representação da informação, foram criadas duas classes: **FoodClass** e **EmentaClass**

### FoodClass

Esta classe representa a ementa de 1 dia, seja a original ou atualizada, sendo que toda a informação acerca da mesma se encontra presente:

- img – String com o caminho no servidor para obtenção da imagem
- weekDay – String que indica o dia da semana a que se refere a mesma ementa
- soup – String que indica o prato de Sopa
- fish – String que indica o prato de Peixe
- meat – String que indica o prato de Carne

- vegetarian – String que indica o Prato Vegetariano
- desert – String que indica a sobremesa

A estrutura baseia-se no JSON Schema fornecido, mais especificamente no menu_post_shema, sendo que este representava 1 ementa apenas (sendo esta o padrão para enviar uma ementa atualizada).

**EmentaClass**

Baseando-se no menu_get_shema.json, esta classe apenas tem 2 instâncias de FoodClass (já que esta representa o resto da informação proveniente do verbo GET da API).
Estas duas instâncias referem-se à ementa Original, e à ementa Atualizada (update)

**Obtenção da Informação**

Para obter a ementa, é chamada a função _fetchCantinaEmentas. Esta irá em primeiro lugar verificar se existe uma instância da ementa guardada nas **SharedPreferences**. Caso exista, irá obter a sua informação a partir da última ementa guardada. Caso contrário, irá proceder à obtenção da ementa via API.

Através do endpoint definido em url_menu, a aplicação irá fazer um pedido GET à API e criar um mapa da resposta. Graças ao Schema disponibilizado junto com a API, sabemos a estrutura desta mesma mensagem.
Após a criação do mapa, iremos percorrer cada par presente no mapa de modo a organizar segundo os próximos dias.

Para facilitar a tradução de JSON para uma instância da classe EmentaClass, foi definido uma factory **fromJson**, que traduz a String numa instância pronta a usar. Do mesmo modo a própria factory da EmentaClass é composta por duas chamadas ao fromJson da FoodClass, permitindo formatar a informação numa instância da classe pronta a usar.

Visto que obtivemos uma atualização da ementa, iremos guardar neste momento uma cópia para as SharedPreferences.

Após uma lista completa de todas as ementas disponíveis, são filtradas as ementas que não têm uma versão atualizada, assim como as ementas que são uma atualização, de modo a apresentar de diferente modo as ementas originais e atualizadas na lista de ementas.

**Apresentação da Informação**

Após a obtenção das ementas, cada uma das mesmas será apresentada numa ListView.

O builder desta view gera uma ListTile que, verificando se o elemento a apresentar também se encontra na lista de ementas atualizadas, apresenta a ementa com um fundo a branco caso seja a original ou um fundo a verde caso seja uma versão atualizada.

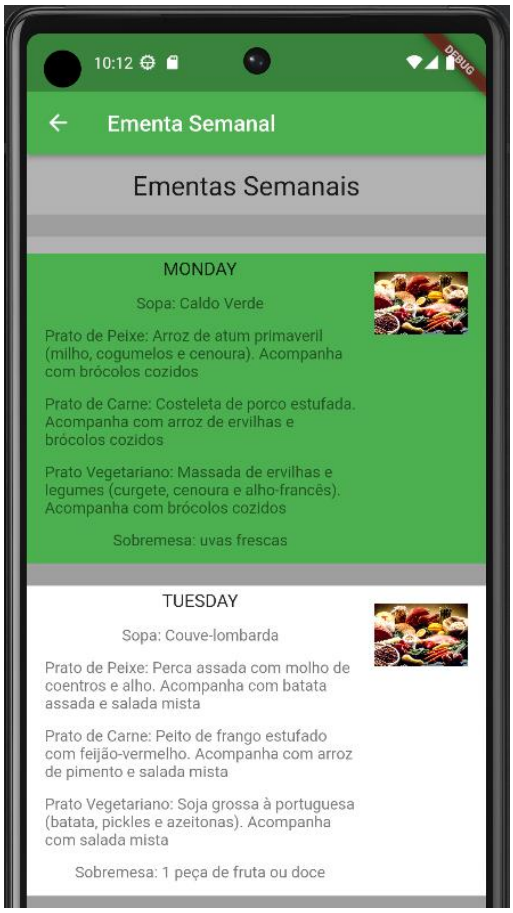


*Figura 3 - Ementa Semanal com uma ementa atualizada e outra original*

Por fim, é feito um pedido para obter a imagem à API através do endpoint definido em url_image. Caso este não a tenha ou exista um erro ao obter a fotografia, é apresentada uma imagem default.

Ao manter premido uma das ementas, o utilizador acede ao ecrã de edição da ementa.

## 2.3. edit_screen

Por fim, este é o ecrã onde podemos atualizar uma ementa.


*Figura 4.1 - Editar ementa*


*Figura 4.2 - Atualizar Imagem*

Neste ecrã é possível verificar qual era a ementa anterior, e inserir valores atualizados destes mesmos campos.
Este ecrã é construído recebendo um FoodClass enviado como argumento a partir da Ementa Semanal.

### Localização para atualização

A aplicação irá verificar se o utilizador se encontra na cantina (ou próximo da mesma) através do plugin location. Em primeiro lugar, irá verificar se tem permissão e acesso à localização, sendo que o utilizador não poderá atualizar a ementa se o mesmo não der permissão para aceder à localização.

```
var location = Location();
_serviceEnabled = await location.serviceEnabled();
if (!_serviceEnabled) {
  _serviceEnabled = await location.requestService();
  if (!_serviceEnabled) {
    setState(() {
      _canEdit = false;
    });
  }
}

_permissionGranted = await location.hasPermission();
if (_permissionGranted == PermissionStatus.denied) {
  _permissionGranted = await location.requestPermission();
  if (_permissionGranted != PermissionStatus.granted) {
    setState(() {
      _canEdit = false;
    });
    _canEdit = false;
  }
}
```

*Figura 5 - Obtenção do serviço de localização*

Caso consiga aceder ao serviço, iremos obter a última localização fornecida pelo serviço e verificar se o utilizador se encontra perto da cantina.

```
_locationData = await location.getLocation();

//if latitude and longitude are in a certain range, allow editing
if (_locationData.latitude! > EditScreen.latCantina - 0.005 &&
    _locationData.latitude! < EditScreen.latCantina + 0.005 &&
    _locationData.longitude! > EditScreen.altCantina - 0.005 &&
    _locationData.longitude! < EditScreen.altCantina + 0.005) {
  setState(() {
    _canEdit = true;
  });
} else {
  setState(() {
    _canEdit = false;
  });
}
```

*Figura 6 - Obtenção da localização e verificação da proximidade da cantina*

Esta verificação é efetuada ao criar o widget, pelo que a falta de localização ou esta não estiver dentro dos parâmetros definidos, irá mostrar um SnackBar a indicar que o utilizador tem que estar na cantina para poder editar a ementa.

```
@override
void initState() {
  super.initState();
  WidgetsBinding.instance.addPostFrameCallback((_) async {
    await checkIfLocationCorrect().then((value) => {
          !_canEdit
              ? ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
              ├── content: Text("Tem que estar na cantina para editar!")))  // SnackBar
              : null
        });
  });
 }
}
```

*Figura 7 - Inicialização do widget*

**Fotografia da ementa**

A aplicação permite tirar uma fotografia a partir da camera através do plugin camera. Através do widget default presente na documentação deste mesmo plugin, é possível tirar uma foto sendo que esta será no momento de enviar a atualizar codificada para base64, segundo o schema de menu_post_shema.json.

```
Future<void> _getNewEmentaImage(BuildContext context) async {
  final cameras = await availableCameras();
  var image = await Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => CameraPage(
        cameras: cameras,
      ),  // CameraPage
    ),  // MaterialPageRoute
  );

  if (!mounted) return;

  setState(() {
    _isNewImage = true;
    _isVisible = true;
    imagemEmenta = Image.file(File(image.path));
    List<int> imageBytes = File(image.path).readAsBytesSync();
    String base64Image = base64Encode(imageBytes);
    _ementa.img = base64Image;
  });
}
```

*Figura 8 - Obtenção de fotografia a partir da camera*

Após obtenção da foto, esta mesma fica disponível para visualização no próprio ecrã de edição.

*Figura 9 - Fotografia atualizada na edição*


*Figura 10 - Ecrã para tirar fotografia*

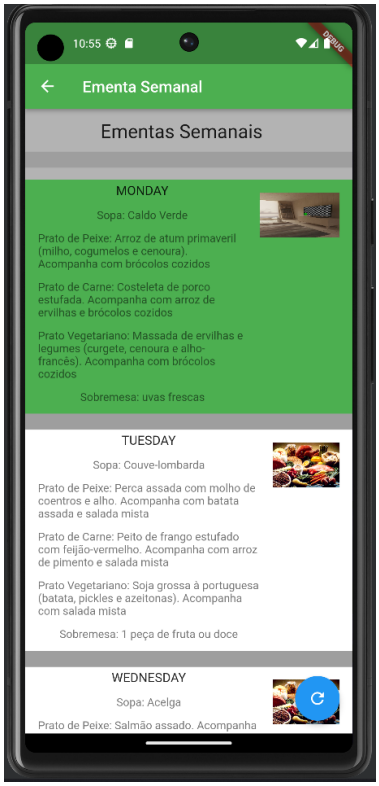


*Figura 11 - Ementa atualizada*

**Atualizar informação**

Para atualizar a informação no servidor, usamos o endpoint url_menu mas com o verbo POST. Quando o botão de editar é pressionado, é chamada a função editDone() que irá codificar um JSON segundo o Schema referido antes enviar um pedido POST ao servidor, garantindo nos headers que o Content-Type está definido para application/json, com o charset=UTF-8, sendo o body a codificação em JSON da informação.

Por fim, a aplicação volta à lista de ementas.

```
Future<void> editDone() async {
  var image;
  if (_isNewImage) {
    image = _ementa.img;
  } else {
    image = "null";
  }
  http.Response a = await http.post(
    Uri.parse(url_menu),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, String>{
      'img': image,
      'soup': _ementa.soup,
      'fish': _ementa.fish,
      'meat': _ementa.meat,
      'vegetarian': _ementa.vegetarian,
      'desert': _ementa.desert,
      'weekDay': _ementa.weekDay,
    }),
  );
  Navigator.of(context).pushNamed(EmentaScreen.routeName);
}
```

*Figura 12 - Envio da informação para o servidor*

### 2.4. creditos_screen

Por fim, existe um pequeno ecrã com a informação dos elementos que participaram no desenvolvimento deste projeto.


*Figura 13 - Ecrã Creditos*