UNIVERSIDADE da MADEIRA

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Ano Letivo 2020/2021

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Aplicações Centradas em Redes 2 º Ano/1º Semestre

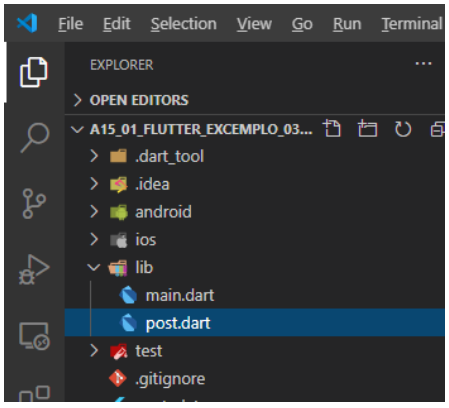
Docente: Michael Silva / Hugo Perdigão

Flutter - Exemplo 3

Objetivos:

- Utilizar Stateful widget.
- Utilizar Variáveis
- Criar uma classe para um Objeto
- Criar um widget para um objeto
- Listar Objetos

1) Criar um novo Projeto “blog”
2) Criar um novo ficheiro na pasta lib com o nome “post.dart”.



3) Criar as propriedades da classe e o construtor.

post.dart

```dart
class Post {
  int id;
  int userId;
  String title;
  String body;

  Post(int pId, int pUserId, String pTitle, String pBody) {
    this.id = pId;
    this.userId = pUserId;
    this.title = pTitle;
    this.body = pBody;
  }
}
```

4) Uma das caraterísticas da linguagem Dart é a possibilidade de incluir numa função "named parameters" ou seja atribuir um nome a cada parâmetro para que possam ser atribuídos externamente sem uma ordem específica. Isto também possibilida a atribuição do valor diretamente a uma propriedade do objeto, para isso substituir o código do construtor.

post.dart

```dart
class Post {
  int id;
  int userId;
  String title;
  String body;

  Post({this.id, this.userId, this.title, this.body});
}
```

5) Para usar este objeto na classe princial é necessário importar.

post.dart | main.dart

```dart
import 'package:flutter/material.dart';
import 'post.dart';
```

6) Criar uma stateful widget com o nome Blog e instanciar na função main.

post.dart | main.dart

```dart
import 'package:flutter/material.dart';
import 'post.dart';

Run | Debug
void main() {
  runApp(MaterialApp(
    home: Blog(),
  )); // MaterialApp
}

class Blog extends StatefulWidget {
  @override
  _BlogState createState() => _BlogState();
}

class _BlogState extends State<Blog> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

NOTA: ao contrário do stateless widget, o stateful widget é composto por duas classes, uma que cria o estado e outra que representa o estado em sí.

7) Criar um scaffold e criar os conteúdos base da App

```dart
class _BlogState extends State<Blog> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("My Blog"),
        centerTitle: true,
        backgroundColor: Colors.blueGrey[850],
      ), // AppBar
    ); // Scaffold
  }
}
```

8) Criar uma lista de Post com alguma informação. Cada posição da lista deve ser um objeto do tipo Post. (conteúdos em https://jsonplaceholder.typicode.com/posts)

post.dart | main.dart

lib > main.dart > _BlogState > posts

```dart
  @override
  _BlogState createState() => _BlogState();
}

class _BlogState extends State<Blog> {
  List<Post> posts = [
    Post(
        id: 1,
        userId: 1,
        title:
            "sunt aut facere repellat provident occaecati excepturi optio repreh
        body: "quia et suscipit\nsuscip… rem eveniet architecto"),
    Post(
        id: 2,
        userId: 1,
        title: "qui est esse",
        body:
            "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beata
    Post(
        id: 3,
        userId: 1,
        title: "ea molestias quasi exercitationem repellat qui ipsa sit aut",
        body:
            "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\n
  ];
```

9) Acrescentar a propriedade body ao scaffold e a dicionar uma column. Nessa column introduzir na propriedade children a lista de posts e utilizar a função map para iterar pelos posts. Para cada post retornar um Text com o título do post.

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text("My Blog"),
        centerTitle: true,
        backgroundColor: Colors.blueGrey[850],
      ), // AppBar
      body: Column(
        children: posts.map(
          (post) {
            return Text(post.title);
          },
        ).toList(),
      )); // Column // Scaffold
  }
}
```

10) Em dart existe uma forma mais de simplificar uma função com apenas uma linha de código.

```dart
      body: Column(
        children: posts.map((post) => Text(post.title)).toList(),
      )); // Column // Scaffold
```

11) Ao testar a aplicação, o texto não aparece organizado.

12) Na classe post criar uma função toCard que transforma um Post num Widget Card.

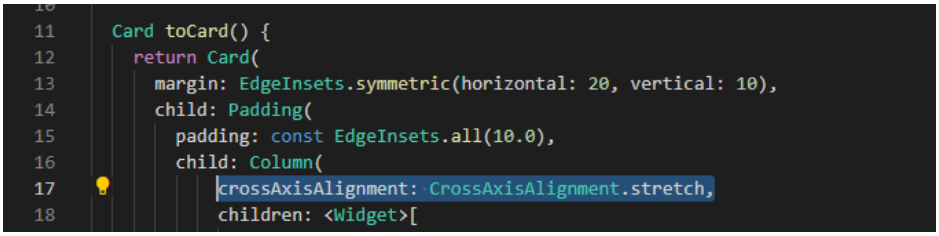
post.dart | main.dart

lib > post.dart > Post > toCard

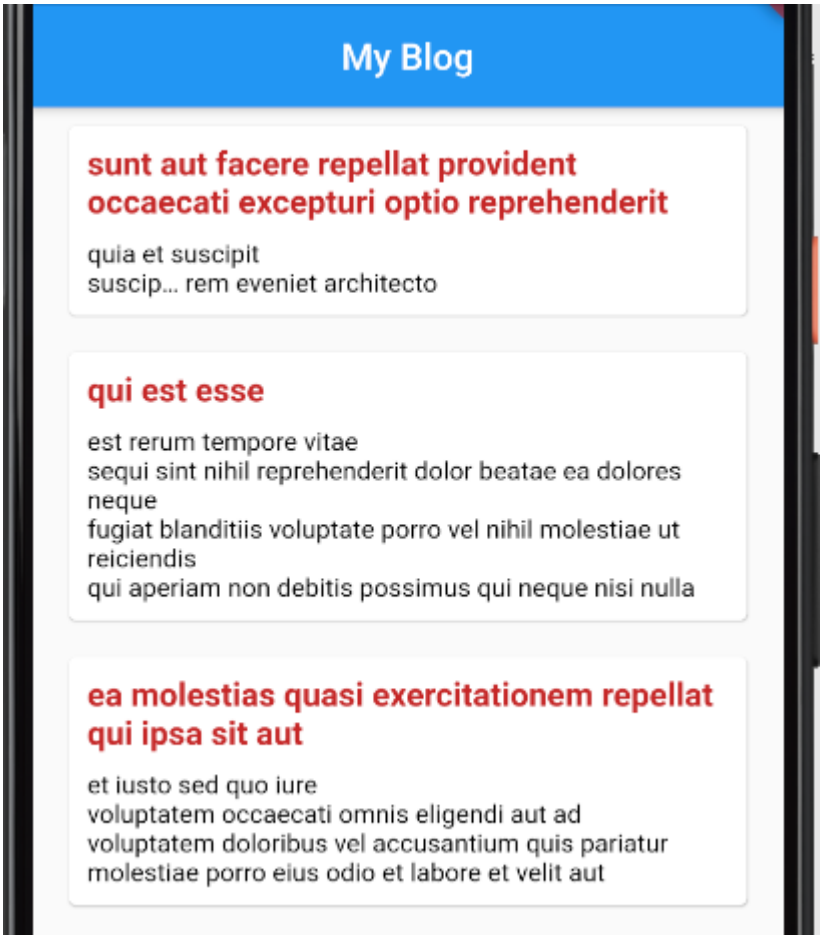
```dart
class Post {
  int id;
  int userId;
  String title;
  String body;

  Post({this.id, this.userId, this.title, this.body});

  Card toCard() {
    return Card(
      margin: EdgeInsets.symmetric(horizontal: 10, vertical: 20),
      child: Column(children: <Widget>[
        Text(
          this.title,
          style: TextStyle(
            fontSize: 18,
            color: Colors.red[800],
          ), // TextStyle
        ), // Text
        SizedBox(height: 4),
        Text(
          this.body,
          style: TextStyle(
            fontSize: 14,
            color: Colors.black,
          ), // TextStyle
        ), // Text
      ]), // <Widget>[] // Column
    ); // Card
  }
}
```

13) Para colocar padding nos objetos da colunam selecionar a Column e utilizar a lâmpada(fixes) para abrir o menu e selecionar a opção wrap with padding.

14) Além do padding, adicionar a propriedade crossAxisallignment para configurar o alinhamento dos objetos dentro da coluna, sendo que neste caso a opção stretch estica os componentes para ocuparem todo o espaço horizontal



15) Testando a aplicação, os Post aparecem organizados:



Webgrafia

https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html

https://api.flutter.dev/flutter/painting/TextStyle-class.html

https://api.flutter.dev/flutter/material/Card-class.html

https://api.flutter.dev/flutter/widgets/Column-class.html

https://api.flutter.dev/flutter/widgets/SizedBox-class.html

https://api.flutter.dev/flutter/painting/EdgeInsets-class.html