Prática 01

Introdução

Documentação: https://flutter.dev/docs/development/ui/widgets-intro

Objetivo: apresentar um código mínimo para entendimento do framework Flutter.

OBS: para executar os exemplos desta prática, use:

a. https://idx.google.com

b. https://zapp.run/, ou
c. https://dartpad.dev/, ou;
d. https://flutlab.io/editor, ou;
e. https://flutterstudio.app/, ou;
f. https://codemagic.io/;
g. Visual Studio Code, apenas se já tiver o Flutter instalado em sua máquina.

1) Flutter é um framework de desenvolvimento de interface de usuário criado pela Google com base na linguagem de programação Dart.
   - No framework Flutter, os módulos básicos para construção de uma interface são chamados de widgets.
   - Os widgets são construídos usando uma estrutura que se inspira no React.
   - A aparência de um widget é descrita por uma configuração e pelo estado atual do widget.
   - Quando o estado de um widget muda, ele reconstrói sua descrição:
     - O widget determina a diferença entre a descrição atual e a anterior, para determinar as mudanças mínimas necessárias.

main()

2) Função principal e obrigatória em um algoritmo em linguagem de programação Dart, responsável por iniciar a execução do programa.

```dart
void main() {
  print('Olá, Mundo!');
}
```

Parâmetros

3) Parâmetro Posicional Obrigatório:

```dart
class  Pessoa  {
  String  nome;
  String curso;

  Pessoa(this.nome, this.curso);

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa('Ana', 'Tecnologia da Informação');
  pessoa.imprimir();
}
```

4) Parâmetro Posicional Opcional:

```dart
class Pessoa {
  String nome;
  String? curso;

  Pessoa(this.nome, [this.curso]);

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa('Ana', 'Tecnologia da Informação');
  pessoa.imprimir();

  pessoa = Pessoa('Bruna');
  pessoa.imprimir();
}
```

5) Parâmetro Posicional Opcional com Valor Default (Padrão):

```dart
class Pessoa {
  String nome;
  String? curso;

  Pessoa(this.nome, [this.curso = 'Banco de Dados']);

  void imprimir() {
    print('$nome: $curso');
  }
```

```
}

void main() {
  Pessoa pessoa = Pessoa('Ana', 'Tecnologia da Informação');
  pessoa.imprimir();

  pessoa = Pessoa('Bruna');
  pessoa.imprimir();
}
```

6) Parâmetro Nominal Opcional:

```
class Pessoa {
  String? nome;
  String? curso;

  Pessoa({this.nome, this.curso});

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa(nome: 'Ana', curso: 'Tecnologia da Informação');
  pessoa.imprimir();

  pessoa = Pessoa(nome: 'Bruna');
  pessoa.imprimir();
}
```

7) Parâmetro Nominal Opcional com Valor Default (Padrão):

```
class Pessoa {
  String? nome;
  String? curso;

  Pessoa({this.nome, this.curso = 'Banco de Dados'});

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa();
  pessoa.imprimir();
```

```
  pessoa = Pessoa(nome: 'Ana', curso: 'Tecnologia da Informação');
  pessoa.imprimir();

  pessoa = Pessoa(nome: 'Bruna');
  pessoa.imprimir();

  pessoa = Pessoa(curso: 'Tecnologia da Informação');
  pessoa.imprimir();
}
```

8) Parâmetro Nominal Requerido (Obrigatório):

```
class  Pessoa  {
  String  nome;
  String curso;

  Pessoa({required this.nome, required this.curso});

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa(nome: 'Ana', curso: 'Tecnologia da Informação');
  pessoa.imprimir();
}
```

9) Parâmetro Posicional e Nominal:

```
class Pessoa {
  String nome;
  String? curso;

  Pessoa(this.nome, {this.curso});

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa('Ana', curso: 'Tecnologia da Informação');
  pessoa.imprimir();

  pessoa = Pessoa('Bruna');
  pessoa.imprimir();
}
```

10) Parâmetro Posicional e Nominal com Valor Default:

```
class Pessoa {
  String nome;
  String? curso;

  Pessoa(this.nome, {this.curso = 'Banco de Dados'});

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa('Ana', curso: 'Tecnologia da Informação');
  pessoa.imprimir();

  pessoa = Pessoa('Bruna');
  pessoa.imprimir();
}
```

11) Parâmetro Posicional e Nominal Requerido (Obrigatório):

```
class Pessoa {
  String nome;
  String? curso;

  Pessoa(this.nome, {required this.curso});

  void imprimir() {
    print('$nome: $curso');
  }
}

void main() {
  Pessoa pessoa = Pessoa('Ana', curso: 'Tecnologia da Informação');
  pessoa.imprimir();
}
```

Aplicativo Mínimo

12) Um aplicativo Flutter mínimo possui uma função principal main() que simplesmente chama a função runApp() com um widget.

- No exemplo abaixo, a função runApp() é chamada com um widget para texto.

- A propriedade textDirection da classe Text recebe o valor ltr, left to right, indicando que a escrita na tela deve ocorrer da esquerda para a direita.
    - **OBS**: se a direção do texto não for herdada de um widget pai, é necessário informar a direção no widget Text.

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

void main() {
  runApp(
    const Text(
      'Olá, Mundo!',
      textDirection: TextDirection.ltr,
    ), //Text.
  ); //runApp.
}
```

13) Alterar a cor do texto através do parâmetro style:

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

void main() {
  runApp(
    const Text(
      'Olá, Mundo!',
      textDirection: TextDirection.ltr,
      style: TextStyle(
        color: Colors.white,
      ),
    ), //Text.
  ); //runApp.
}
```

14) Alterar o código, para que o texto seja escrito da direita para a esquerda:

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

void main() {
  runApp(
    const Text(
      'Olá, Mundo!',
      textDirection: TextDirection.rtl,
      style: TextStyle(
        color: Colors.red,
      ),
    ), //Text.
  ); //runApp.
}
```

15) Os widgets são inseridos, formando uma árvore, tree, em que um widget torna-se filho de outro. No exemplo abaixo, o widget Text torna-se filho, child, do widget Center.

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

void main() {
  runApp(
    const Center(
      child: Text(
        'Olá, Mundo!',
        textDirection: TextDirection.ltr,
        style: TextStyle(
          color: Colors.red,
        ),
      ), //Text.
    ), //Center.
  );
}
```

OBS: o framework Flutter força o widget raiz a cobrir toda a tela.

Exercício

Em linguagem de programação Dart, escreva um algoritmo que:

1) Possua uma classe para armazenar os dados dos clientes de uma loja.
2) Possua uma classe para armazenar os dados dos produtos de uma fábrica.

No framework Flutter, escreva o algoritmo de um aplicativo mínimo que:

3) Apresente uma mensagem na tela.
4) Aplique estilos de formatação à mensagem impressa.
5) Centralize a mensagem na tela.

Classe MaterialApp

Documentação: https://api.flutter.dev/flutter/material/MaterialApp-class.html

Objetivo: apresentar a classe MaterialApp.

Uma instância da classe MaterialApp é um widget que envolve vários outros widgets que são normalmente necessários para aplicativos de design material (Material Design: https://material.io/design, https://materializecss.com/).

16) O código abaixo exemplifica o uso da classe MaterialApp.
- A função runApp () recebe uma instância da classe MaterialApp como parâmetro.
- O parâmetro home do widget MaterialApp recebe um objeto da classe Home.
- A classe Home é uma classe sem estado, porque ela estende a classe StatelessWidget (https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html). Isso significa que a classe Home não sofre alteração, seu estado não muda.

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

void main() {
  runApp(
    MaterialApp(
      home: Home(),
    ),
  );
}

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const Text(
      "Olá, Mundo!!!",
    );
  }
}
```

- A classe Home sobrescreve, @override, o método build () da classe StatelessWidget.
- O método build() recebe como parâmetro um objeto context da classe BuildContext.
  - Os objetos da classe BuildContext armazenam uma referência para o widget pai do widget corrente.
  - Dessa forma, os objetos da classe BuildContext possuem uma referência para o contexto em que o widget corrente deve ser criado.
- Além disso, o método build() retorna um widget. No caso, ele retorna um widget Text.

17) A classe Home é criada pelo desenvolvedor e poderia ter qualquer outro nome, como PaginaInicial, por exemplo:

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

void main() {
  runApp(
```

```
    MaterialApp(
      home: PaginaInicial(),
    ),
  );
}

class PaginaInicial extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const Text("Olá, Mundo!!!");
  }
}
```

18) As vírgulas no código geram uma identação que destaca cada widget e seus parâmetros. A remoção das vírgulas gera uma versão mais compacta do código:

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

void main() {
  runApp(MaterialApp(home: Home()));
}

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const Text("Olá, Mundo!!!");
  }
}
```

Arrow Syntax

19) A sintaxe de flecha, ou de seta, (arrow syntax) é usada quando o escopo da função, procedimento ou método possui apenas uma instrução. Exemplo: a função main () abaixo pode ser chamada através da sintaxe de flecha:

O algoritmo:

```
int func(x) {
  return x * 2;
}

void main() {
  print(func(3));
}
```

Na sintaxe de flecha, poderia ser escrito na forma:

```
int func(x) => x * 2;
void main() => print(func(5));
```

Um outro exemplo da sintaxe de flecha:

```
void func() => print(5);
void main() => func();
```

Então:

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const Text("Olá, Mundo!!!");
  }
}
```

Ou ainda:

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) => const Text("Olá, Mundo!!!");
}
```

Classe Scaffold

Documentação: https://api.flutter.dev/flutter/material/Scaffold-class.html

Objetivo: apresentar a classe Scaffold.

A classe Scaffold implementa a estrutura básica de layout visual do material design. Ela possui 3 parâmetros opcionais de destaque: uma barra (appBar), um corpo (body) e um botão (floatingActionButton).

20) O código abaixo exemplifica o uso da classe Scaffold.

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const Scaffold();
  }
}
```

Ou

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) => const Scaffold();
}
```

21) O exemplo anterior apresenta uma faixa indicativa de debug. Atribuir o valor false ao atributo debugShowCheckedModeBanner do widget MaterialApp oculta a faixa:

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

void main() => runApp(
      MaterialApp(
        home: Home(),
        debugShowCheckedModeBanner: false,
      ),
    );

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const Scaffold();
  }
}
```

22) O exemplo seguinte inclui uma barra (appBar).

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

void main() => runApp(MaterialApp(home: Home()));
```

```dart
class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(appBar: AppBar());
  }
}
```

23) O exemplo abaixo inclui um título na barra.

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
    );
  }
}
```

24) Incluir um ícone (leading) na barra.

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
        title: const Text('Página Inicial'),
      ),
    );
  }
}
```

25) Incluir um corpo (body) com um texto.

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

void main() => runApp(MaterialApp(home: Home()));
```

```
class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
        title: const Text('Página Inicial'),
      ),
      body: const Text('Olá, Mundo!!!'),
    );
  }
}
```

26) Incluir um botão (floatingActionButton).

- Nesse exemplo, não há uma funcionalidade associada ao botão, porque o parâmetro onPressed do botão recebe o valor null.
- Ao manter o botão pressionado, ele apresenta a mensagem (tooltip) “Exemplo de botão”.
- O botão possui um ícone, o sinal de adição (+).

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
        title: const Text('Página Inicial'),
      ),
      body: const Text('Olá, Mundo!!!'),
      floatingActionButton: const FloatingActionButton(
        onPressed: null,
        tooltip: 'Exemplo de botão',
        child: Icon(Icons.add),
      ),
    );
  }
}
```

27) Aplica cor azul ao fundo do corpo (body).

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

void main() => runApp(MaterialApp(home: Home()));
```

```
class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
        title: const Text('Página Inicial'),
      ),
      body: const Text('Olá, Mundo!!!'),
      backgroundColor: Colors.blue[900],
      floatingActionButton: const FloatingActionButton(
        onPressed: null,
        tooltip: 'Exemplo de botão',
        child: Icon(Icons.add),
      ),
    );
  }
}
```

28) Aplicar estilo ao texto escrito no corpo (body):

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
        title: const Text('Página Inicial'),
      ),
      body: const Text(
        'Olá, Mundo!!!',
        style: TextStyle(
          color: Colors.white,
          fontSize: 30,
          fontWeight: FontWeight.bold,
        ),
      ),
      backgroundColor: Colors.blue[900],
      floatingActionButton: const FloatingActionButton(
        onPressed: null,
        tooltip: 'Exemplo de botão',
        child: Icon(Icons.add),
      ),
    );
```

```
  }
}
```

29) Centralizar o botão (floatingActionButton):

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
        title: const Text('Página Inicial'),
      ),
      body: const Text(
        'Olá, Mundo!!!',
        style: TextStyle(
          color: Colors.white,
          fontSize: 30,
          fontWeight: FontWeight.bold,
        ),
      ),
      backgroundColor: Colors.blue[900],
      floatingActionButton: const FloatingActionButton(
        onPressed: null,
        tooltip: 'Exemplo de botão',
        child: Icon(Icons.add),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
    );
  }
}
```

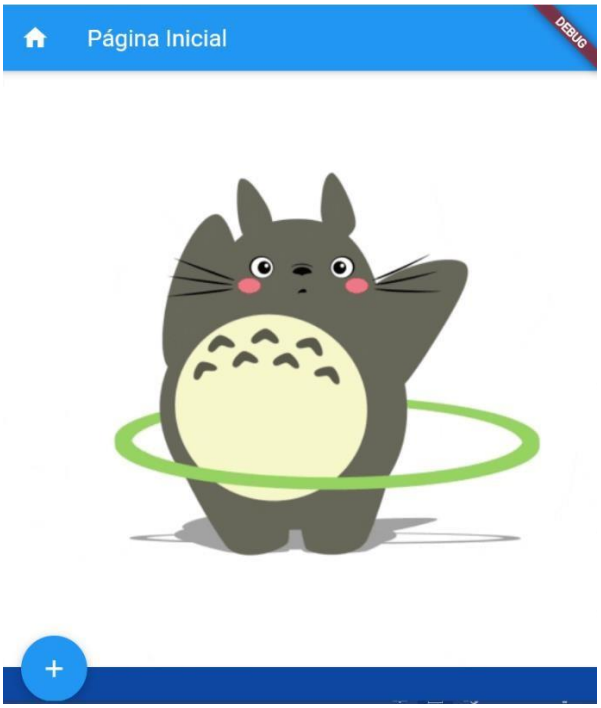
30) Alinhar o botão (floatingActionButton) à esquerda:

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

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
```

```
        title: const Text('Página Inicial'),
      ),
      body: const Text(
        'Olá, Mundo!!!',
        style: TextStyle(
          color: Colors.white,
          fontSize: 30,
          fontWeight: FontWeight.bold,
        ),
      ),
      backgroundColor: Colors.blue[900],
      floatingActionButton: const FloatingActionButton(
        onPressed: null,
        tooltip: 'Exemplo de botão',
        child: Icon(Icons.add),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.startDocked,
    );
  }
}
```

31) Incluir uma imagem no corpo (body):

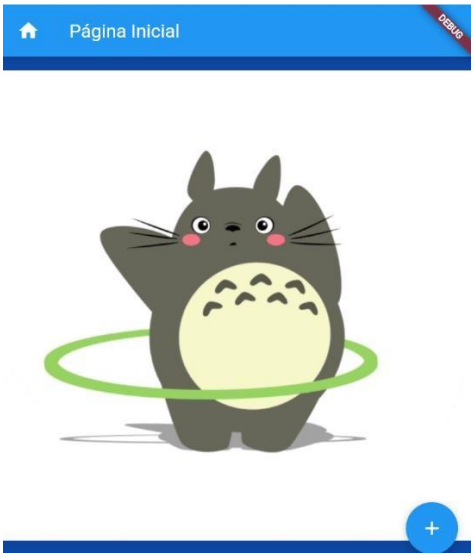
```
import 'package:flutter/material.dart';

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: const Icon(Icons.home),
        title: const Text('Página Inicial'),
      ),
      body: Image.network(
        'https://media.giphy.com/media/pt0EKLDJmVvlS/giphy.gif',
      ),
      backgroundColor: Colors.blue[900],
      floatingActionButton: const FloatingActionButton(
        onPressed: null,
        tooltip: 'Exemplo de botão',
        child: Icon(Icons.add),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.startDocked,
    );
  }
}
```



Exercício

6) Altere o exemplo desta prática, para que ele apresente a tela abaixo.

7) No exercício anterior, altere:
    a. O ícone do leading no widget AppBar().
    b. O texto do título no widget AppBar().
    c. O texto do body no widget AppBar(), para que apresente seu nome na cor amarela.
    d. O ícone do widget FloatingActionButton().
    e. O texto no tooltip do widget FloatingActionButton().
8) Altere o último exemplo desta prática, para que ele centralize a imagem na tela. Exemplo:


Página Inicial
DEBUG
+