Widgets

Computação para Dispositivos Móveis

Prof. Gustavo Custodio
gustavo.custodio@anhembi.br

Widgets

Widgets

- Cada widget no Flutter é utilizado para realizar uma única tarefa pequena.

Widgets

- Cada widget no Flutter é utilizado para realizar uma única tarefa pequena.
  - Um widget `Text` mostra texto na tela;
  - Um widget `Padding` cria um espaço entre widgets;
  - Um widget `Scaffold` fornece uma estrutura para uma tela.

Widgets

- Um Widget sem estato (*Stateless*) é o bloco primário para construir interfaces de usuário.

Widgets

- Um Widget sem estato (*Stateless*) é o bloco primário para construir interfaces de usuário.
  - O Flutter consegue facilmente renderizar muitos widgets desse tipo sem dificuldade.
  - Widgets sem estado são imutáveis, quando criados, não podem ser mais modificados.

Widgets

- Vamos criar uma tela utilizando widgets imutáveis.

Widgets

- Vamos criar uma tela utilizando widgets imutáveis.
- Utilizaremos:

Widgets

- Vamos criar uma tela utilizando widgets imutáveis.
- Utilizaremos:
  - `Scaffold`;
  - `Container`;
  - `Padding`.

Widgets

- Vamos criar uma tela utilizando widgets imutáveis.
- Utilizaremos:
  - `Scaffold`;
  - `Container`;
  - `Padding`.
- Criaremos tudo dentro de uma classe chamada `ImmutableWidget`.

Widgets

- Crie um novo projeto em Flutter:

Widgets

- Crie um novo projeto em Flutter:
  - `flutter create exemplo-widgets`.
  - ou então utilize o atalho `Ctrl+Shift+P` do VSCode.

Widgets

- Crie um novo projeto em Flutter:
  - `flutter create exemplo-widgets`.
  - ou então utilize o atalho `Ctrl+Shift+P` do VSCode.
- Crie um arquivo `immutable_widget.dart` na pasta `lib`.

Widgets

- No arquivo `main.dart`, insira o seguinte código:

Widgets

- No arquivo `main.dart`, insira o seguinte código:

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

void main() {
  runApp(const ExemploWidgets());
}

class ExemploWidgets extends StatelessWidget {
  const ExemploWidgets({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: ImmutableWidget(),
    ); // MaterialApp
  }
}
```

Widgets

- Observe que estamos importanto o arquivo `immutable_widget.dart` para nossa classe principal, de acordo com a linha:

Widgets

- Observe que estamos importanto o arquivo `immutable_widget.dart` para nossa classe principal, de acordo com a linha:
  - `import './immutable_widget.dart';`
  - Agora vamos adicionar conteúdo a esse arquivo.

Widgets

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

class ImmutableWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.green,
      child: Padding(
        padding: EdgeInsets.all(40),
        child: Container(
          color: Colors.purple,
          child: Padding(
            padding: const EdgeInsets.all(50),
            child: Container(
              color: Colors.blue,
            ),
          ),
        ),
      ),
    );
  }
}
```

Widgets

O resultado é um retângulo de três cores.

Widgets

O resultado é um retângulo de três cores.

Containers, Columns e Rows

- O método `build` retorna um `Container` (verde)

Containers, Columns e Rows

- O método `build` retorna um `Container` (verde)
  - que contém um `Padding`
    - onde o `EdgeInsets.all(40)` define um espaço de 40 pixels nos 4 lados

Containers, Columns e Rows

- O método `build` retorna um `Container` (verde)
  - que contém um `Padding`
    - onde o `EdgeInsets.all(40)` define um espaço de 40 pixels nos 4 lados
  - que contém outro `Container` (roxo)

Containers, Columns e Rows

- O método `build` retorna um `Container` (verde)
  - que contém um `Padding`
    - onde o `EdgeInsets.all(40)` define um espaço de 40 pixels nos 4 lados
  - que contém outro `Container` (roxo)
  - que contém outro `Padding`

Containers, Columns e Rows

- O método `build` retorna um `Container` (verde)
  - que contém um `Padding`
    - onde o `EdgeInsets.all(40)` define um espaço de 40 pixels nos 4 lados
  - que contém outro `Container` (roxo)
  - que contém outro `Padding`
  - que contém um `Container` azul.

Containers, Columns e Rows

- Observe que cada widget contido dentro de outro é um parâmetro opcional chamado `child`.

Containers, Columns e Rows

- Observe que cada widget contido dentro de outro é um parâmetro opcional chamado `child`.
- Alguns widgets possuem um parâmetro opcional chamado `children`
  - ou seja, múltiplos widgets contidos nele.

Containers, Columns e Rows

- Observe que cada widget contido dentro de outro é um parâmetro opcional chamado `child`.
- Alguns widgets possuem um parâmetro opcional chamado `children`
  - ou seja, múltiplos widgets contidos nele.
  - O widget `Column` é um exemplo.

Hello World

- Observe o código do `Hello World`.

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Hello World

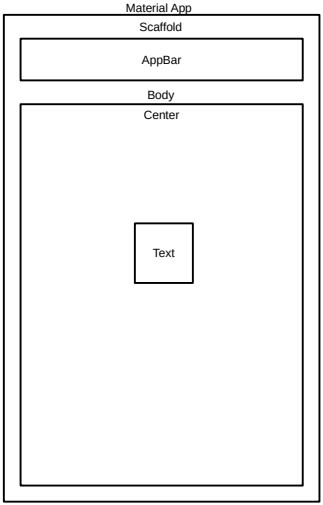
- Observe o código do `Hello World`.

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

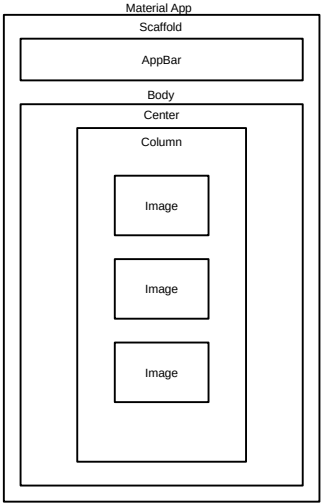


Containers, Columns e Rows

- Representamos o app com um diagrama mostrando como os widgets estão aninhados.
- Agora vamos tentar criar o seguinte app:

Containers, Columns e Rows

- Representamos o app com um diagrama mostrando como os widgets estão aninhados.
- Agora vamos tentar criar o seguinte app:



Imagens

- Antes de continuar, precisamos aprender como adicionar imagens a um proojeto.

Imagens

- Antes de continuar, precisamos aprender como adicionar imagens a um proojeto.
- Crie uma pasta chamada `images` na raiz do projeto.
- Baixe as imagens que deseja adicionar e coloque-as nessa pasta.

Imagens

- É preciso adicionar as imagens como `assets` no projeto.

Imagens

- É preciso adicionar as imagens como `assets` no projeto.
- Abra o arquivo `pubspec.yaml` e procure pela parte do arquivo:

```
1       # assets:
2       # - images/a_dot_burr.jpeg
3       # - images/a_dot_ham.jpeg
```

Imagens

- É preciso adicionar as imagens como `assets` no projeto.
- Abra o arquivo `pubspec.yaml` e procure pela parte do arquivo:

```
# assets:
# - images/a_dot_burr.jpeg
# - images/a_dot_ham.jpeg
```

- Remova o # das linhas (descomente) e substitua pelo caminho das suas imagens.
- Agora o flutter reconhecerá as imagens.

Containers, Columns e Rows

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

void main() {
 runApp(const MyApp());
}

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Colunas e Rows',
     home: Scaffold(
       appBar: AppBar(
         title: const Text('Colunas e Rows'),
       ),
       body: Center(
         child: Column(
           children: [
// Adicionando imagens na coluna
             Image.asset('images/lands_01.jpg'),
             Image.asset('images/lands_02.jpg'),
             Image.asset('images/lands_03.jpg'),
           ],
         ),
       ),
     ),
   );
 }
}
```

Containers, Columns e Rows

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Colunas e Rows',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Colunas e Rows'),
        ),
        body: Center(
          child: Column(
            children: [
  // Adicionando imagens na coluna
              Image.asset('images/lands_01.jpg'),
              Image.asset('images/lands_02.jpg'),
              Image.asset('images/lands_03.jpg'),
            ],
          ),
        ),
      ),
    );
  }
}
```

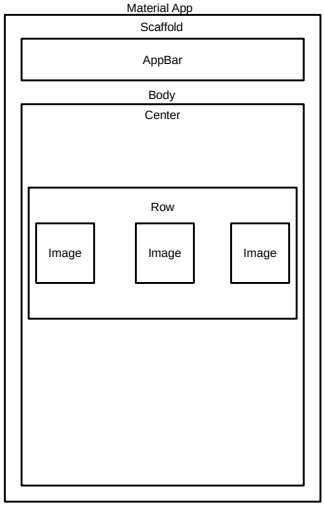
- Temos como resultado três imagens alinhadas em uma coluna.

Containers, Columns e Rows

- É também possível dispor conteúdo em uma mesma linha.

Containers, Columns e Rows

- É também possível dispor conteúdo em uma mesma linha.



Containers, Columns e Rows

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

void main() {
 runApp(const MyApp());
}

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Colunas e Rows',
     home: Scaffold(
       appBar: AppBar(
         title: const Text('Colunas e Rows'),
       ),
       body: Center(
         child: Row(
           children: [
// Adicionando imagens na coluna
             Image.asset('images/lands_01.jpg'),
             Image.asset('images/lands_02.jpg'),
             Image.asset('images/lands_03.jpg'),
           ],
         ),
       ),
     ),
   );
 }
}
```

Containers, Columns e Rows

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

void main() {
 runApp(const MyApp());
}

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Colunas e Rows',
     home: Scaffold(
       appBar: AppBar(
         title: const Text('Colunas e Rows'),
       ),
       body: Center(
         child: Row(
           children: [
 // Adicionando imagens na coluna
             Image.asset('images/lands_01.jpg'),
             Image.asset('images/lands_02.jpg'),
             Image.asset('images/lands_03.jpg'),
           ],
         ),
       ),
     ),
   );
 }
}
```

- Temos como resultado três imagens alinhadas em uma linha agora.

Exercício

- Vamos tentar colocar uma imagem seguida de um texto.
  - Use o `CircularImage` para imagens arredondadas.

Referências

Simone Alessandria and Brian Kayfitz.
Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart.
Packt Publishing Ltd, 2021.

Conteúdo

https://gustavotcustodio.github.io/dispomoveis.html

Obrigado

gustavo.custodio@anhembi.br