Cento Universitário UNA

Usabilidade, Desenvolvimento Web, Mobile e Jogos

Graduação – TI e Engenharias

Práticas de Laboratório

Carlos Augusto dos Santos Pinheiro, Cristiano de Macedo Neto, Diego Augusto de Faria Barros, Wesley Dias Maciel

2020/02

Centro Universitário UNA
Graduação – TI e Engenharias
Usabilidade, Desenvolvimento Web, Mobile e Jogos
Prática de Laboratório
Carlos Augusto dos Santos Pinheiro, Cristiano de Macedo Neto, Diego Augusto de Faria Barros, Wesley Dias Maciel
2020/02

Flutter

Centro Universitário UNA
Graduação – TI e Engenharias
Usabilidade, Desenvolvimento Web, Mobile e Jogos
Prática de Laboratório
Carlos Augusto dos Santos Pinheiro, Cristiano de Macedo Neto, Diego Augusto de Faria Barros, Wesley Dias Maciel
2020/02

Prática 09

Estado

Documentação: https://flutter.dev/docs/development/ui/widgets-intro, https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html, https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html

Objetivo: apresentar as classes StatelessWidget e StatefulWidget.

No framework Flutter, a ideia central é que você construa sua interface a partir de widgets. A principal tarefa de um widget é implementar uma função build (). A função build () descreve esse widget em termos de outros widgets de nível inferior na árvore de renderização.

O framework Flutter constrói o widget até chegar em algum widget que represente um objeto da classe básica RenderObject. O objeto da classe RenderObject é responsável por calcular e descrever a geometria do widget.

Um widget descreve sua aparência a partir de sua configuração e estado atuais. Quando o estado de um widget muda, o widget reconstrói essa descrição, a fim de determinar as mudanças mínimas necessárias na árvore de renderização subjacente para a transição de um estado para o próximo.

Os widgets são descritos por classes. Ao escrever um aplicativo, você normalmente criará novos widgets que serão subclasses de StatelessWidget ou StatefulWidget. Seu widget será subclasse de StatelessWidget, se não precisar gerenciar algum estado. Por outro lado, seu widget será subclasse de StatefulWidget, caso faça a gerência de algum estado.

Para converter um widget que não gerencia estado em outro que gerencia, você precisa fazer duas coisas:

Centro Universitário UNA
Graduação – TI e Engenharias
Usabilidade, Desenvolvimento Web, Mobile e Jogos
Prática de Laboratório
Carlos Augusto dos Santos Pinheiro, Cristiano de Macedo Neto, Diego Augusto de Faria Barros, Wesley Dias Maciel
2020/02

1) Converter a classe que estende StatelessWidget em uma classe de estado. Esta classe será responsável por determinar o estado do widget e terá que estender a classe State.

    a. A classe State possui um método setState (). Você tem que usar o método setState () para alterar o estado interno de um objeto da classe State. Então, crie um método para promover a alteração de estado. Depois, passe esse método como parâmetro de setState ().

2) Criar uma nova classe que estende StatefulWidget. Essa classe terá que criar e retornar a classe de estado que você criou no passo anterior através do método createState ().

OBS: é uma boa prática de programação que o nome da classe de estado seja igual ao nome da classe que estende StatefulWidget concatenado com o string “State”. No exemplo abaixo, a classe que estende StatefulWidget chama-se “PaginaInicial” e a classe que gerencia estado chama-se “PaginaInicial**State**”:

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

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

class PaginaInicial extends StatefulWidget {
  PaginaInicialState createState() {
    return PaginaInicialState();
  }
}

class PaginaInicialState extends State<PaginaInicial> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Página Inicial")),
    );
  }
}
```

Na definição da classe PaginaInicialState, a instrução State<PaginaInicial> informe que a classe é um estado para PaginaInicial:

```
class PaginaInicialState extends State<PaginaInicial>
```

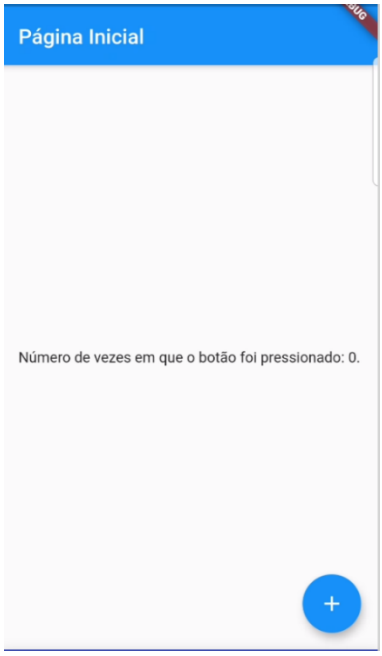
1) Crie um novo projeto Flutter com o exemplo abaixo, usando:
    a. Visual Studio Code, ou;
    b. https://dartpad.dev/, ou;
    c. https://flutlab.io/, ou;
    d. https://flutterstudio.app/, ou;
    e. https://codemagic.io/.

Sem Estado

2) O exemplo abaixo tenta alterar o estado de um widget ao clicar no botão apresentado na tela. Um método para incremento de uma variável foi implementado para promover a alteração de estado. Esse método foi associado ao evento de clicar no botão, onPressed. Entretanto, a alteração de estado não ocorre, porque a classe estende StatelessWidget.

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

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

class Home extends StatelessWidget {
  int numeroVezes = 0;

  void cliqueDoBotao() {
    numeroVezes = numeroVezes + 1;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Página Inicial"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

```
          children: [
            Text("Número de vezes em que o botão foi pressionado: $numeroVezes
."),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: cliqueDoBotao,
        child: Icon(Icons.add),
      ),
    );
  }
}
```



Com Estado

3) Para conseguirmos implementar a mudança de estado, o primeiro passo é converter a classe que estende StatelessWidget em uma classe de estado. Esta classe será responsável por determinar o estado do widget e terá que estender a classe State.
    a. **OBS**: temos que usar o método setState () para alterar o estado interno do objeto da classe State. O método que incrementa o valor da variável, cliqueDoBotao (), promove a alteração de estado. Então, passamos esse método como parâmetro de setState ().

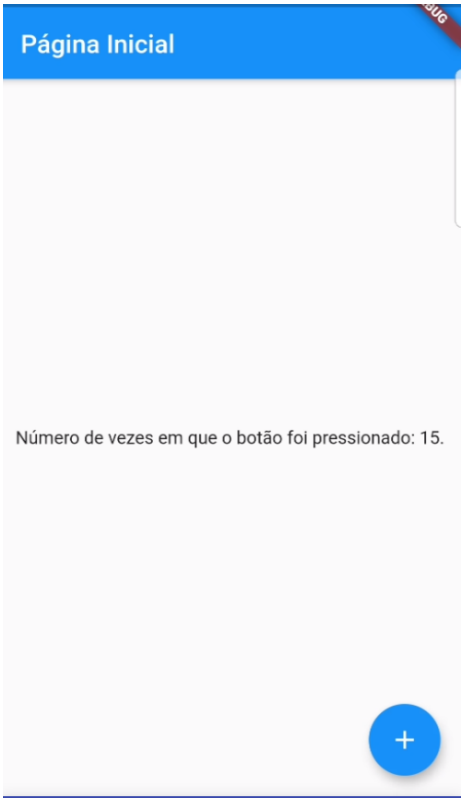
```dart
import 'package:flutter/material.dart';

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

class HomeState extends State<Home> {
  int numeroVezes = 0;

  void cliqueDoBotao() {
    numeroVezes = numeroVezes + 1;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Página Inicial"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
                "Número de vezes em que o botão foi pressionado: $numeroVezes.
"),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(cliqueDoBotao);
        },
        child: Icon(Icons.add),
      ),
    );
  }
}
```

4) Agora, temos que criar uma nova classe que estende StatefulWidget. Essa classe cria e retorna a classe de estado que criamos no passo anterior através do método createState ().

Centro Universitário UNA
Graduação – TI e Engenharias
Usabilidade, Desenvolvimento Web, Mobile e Jogos
Prática de Laboratório
Carlos Augusto dos Santos Pinheiro, Cristiano de Macedo Neto, Diego Augusto de Faria Barros, Wesley Dias Maciel
2020/02

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  int numeroVezes = 0;

  void cliqueDoBotao() {
    numeroVezes = numeroVezes + 1;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Página Inicial"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text("Número de vezes em que o botão foi pressionado: $numeroVezes
."),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(cliqueDoBotao);
        },
        child: Icon(Icons.add),
      ),
    );
  }
}
```

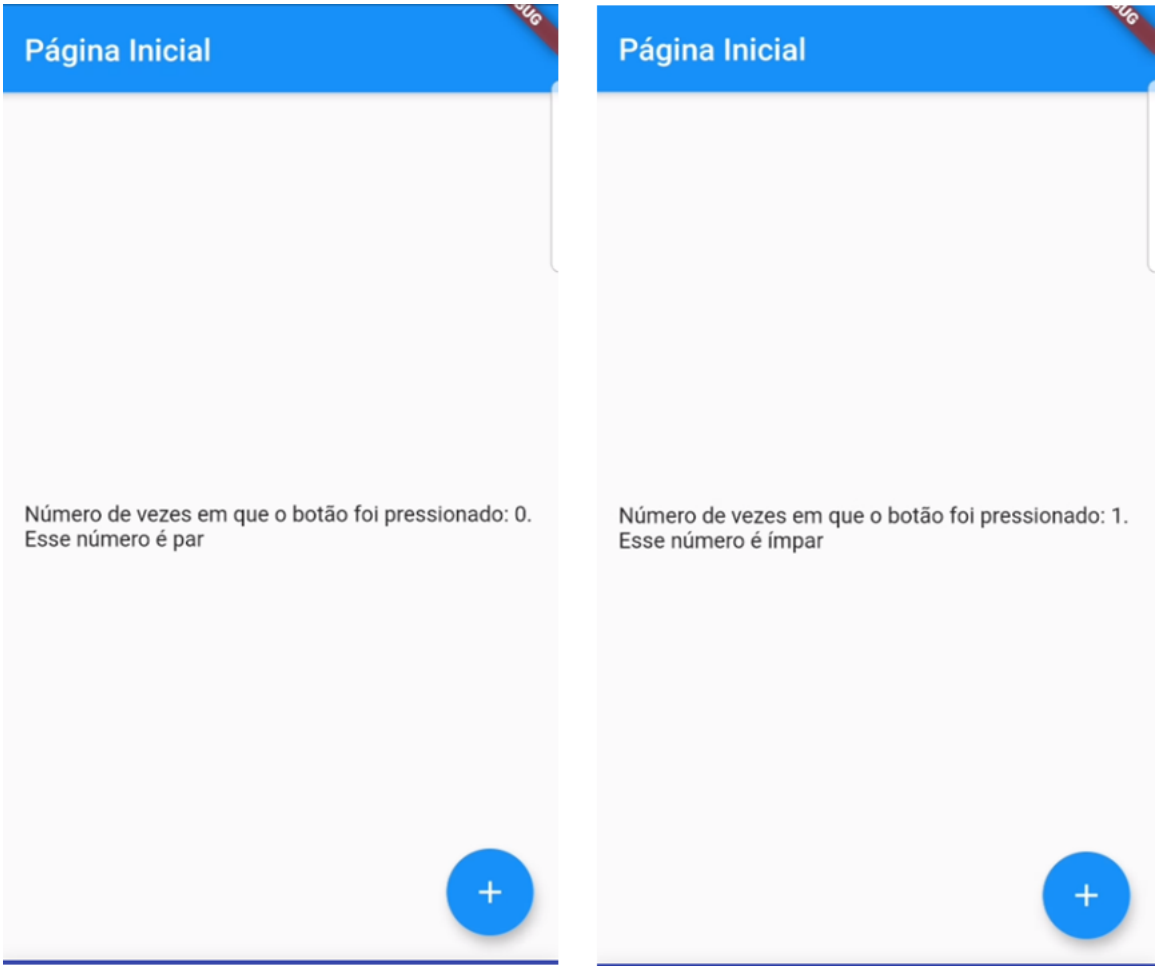
Agora, quando executamos o programa, o widget apresenta as alterações de estado:



Exercício

1) Altere o exemplo desta prática, para que ele informe se o número de vezes em que o botão foi pressionado é par ou ímpar.

Centro Universitário UNA
Graduação – TI e Engenharias
Usabilidade, Desenvolvimento Web, Mobile e Jogos
Prática de Laboratório
Carlos Augusto dos Santos Pinheiro, Cristiano de Macedo Neto, Diego Augusto de Faria Barros, Wesley Dias Maciel
2020/02



Dica:

```
  int numeroVezes = 0;
  String mensagem = "Número de vezes em que o botão foi pressionado: 0.\nEsse
número é par";

  void cliqueDoBotao() {
    numeroVezes = numeroVezes + 1;
    mensagem = "Número de vezes em que o botão foi pressionado: $numeroVezes.\
nEsse número é ${numeroVezes % 2 == 0 ? "par" : "ímpar"}";
  }
```