

Navegação de Telas

Computação para Dispositivos Móveis

Prof. Gustavo Custodio
gustavo.custodio@anhembi.br

Conteúdo

- O que faremos essa aula?
 - Adicionar estado para a aplicação.
 - Interagir com botões.
 - Navegar para a próxima tela.



Navegacao de Telas

StatefulWidget

StatefulWidget

- Até o momento só utilizamos o StatelessWidget, widgets estáticos, para criar telas de apps.
 - No entanto, a maioria dos aplicativos possui alguma forma de estado.
- Existem também o StatefulWidget, um widget que **pode armazenar informação** e pode ser recriado quando um estado muda.
 - Comparado com widgets *stateless*, os widgets *stateful* possuem partes mais complexas.

StatefulWidget

- Vamos criar um aplicativo cronômetro.

StatefulWidget

- Adicione o código no `main.dart`.
- Esse código é utilizado para criar a tela principal do aplicativo, mas o conteúdo principal ficará em um `StatefulWidget`.

```
1 import 'package:flutter/material.dart';
2 import './cronometro.dart';
3
4 void main() {
5   runApp(AppCronometro());
6 }
7
8 class AppCronometro extends StatelessWidget {
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       home: Cronometro(),
12     );
13   }
14 }
```

StatefulWidget

- Crie o arquivo cronometro.dart na pasta lib e adicione o código:

```
1 import 'package:flutter/material.dart';
2 import 'dart:async';
3
4 class Cronometro extends StatefulWidget {
5   @override
6   State<Cronometro> createState() {
7     return _CronometroState();
8   }
9 }
```

StatefulWidget

- Um StatefulWidget é dividido em duas classes: o widget e o estado (State).
- Todo StatefulWidget possui um método createState() que retorna um State.
 - Os widgets StatefulWidget e State são tão dependentes um do outro que são colocados no mesmo arquivo.

State

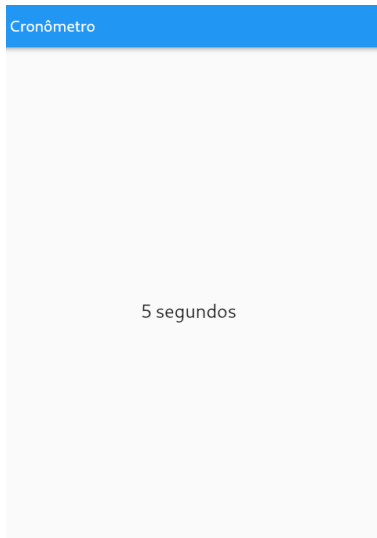
- Vamos criar a classe privada `_CronometroState` logo abaixo de `Cronometro`.

```
1 class _CronometroState extends State<Cronometro> {
2   int segundos = 0;
3   Timer? timer;
4
5   @override
6   Widget build(BuildContext context) {
7     return Scaffold(
8       appBar: AppBar(
9         title: Text('Cronometro'),
10      ),
11      body: Center(
12        child: Text(
13          '$segundos segundos',
14          style: Theme.of(context).textTheme.headline5,
15        ),
16      ),
17    );
18  }
19 }
```

State

- Ao invés de digitar o código para criar um `StatefulWidget`, existem *snippets* que podem ser utilizados no VSCode.
 - `stful` - Cria a base de uma classe que herda de `StatefulWidget`.
 - `stless` - Cria a base de uma classe que herda de `StatelessWidget`.

State



- O Timer é o widget responsável por controlar o contador de tempo.
- A cada segundo ele executa uma função passada como parâmetro.

State

- Agora precisamos adicionar o comportamento do cronômetro.
 - Primeiro definimos uma função `initState`, responsável por definir o estado inicial do cronômetro.
 - Depois definimos uma função `_onTick(Timer time)`.
 - Dentro dessa função, temos um `setState`, responsável por **atualizar** a tela.
 - Por último, temos um método `dispose`, responsável por parar o contador quando o aplicativo é fechado.

State

```
1  @override
2  void initState() {
3    super.initState();
4    segundos = 0;
5    // A cada segundo, o método _onTick é chamado
6    timer = Timer.periodic(Duration(seconds: 1), _onTick);
7  }
8
9  void _onTick(Timer time) {
10   setState(() {
11     ++segundos;
12   });
13 }
14
15 @override
16 void dispose() {
17   timer?.cancel();
18   super.dispose();
19 }
```

State

- Cada vez que o `setState` é chamado, o widget é gerado novamente.
 - Widget chama o `setState()`;
 - Widget fica marcado para ser gerado novamente.
 - Flutter chama os métodos `build` desses widgets.

Navegação

- Vamos criar uma tela inicial onde você pode clicar em um botão e ser direcionado para a tela do cronômetro.
 - Mas antes de realizar a navegação entre telas, precisamos aprender a criar botões.
- Vamos criar um botão para resetar o cronômetro.
 - Fazemos isso utilizando o `ElevatedButton`.



Navegação de Telas

Navegação

Navegação

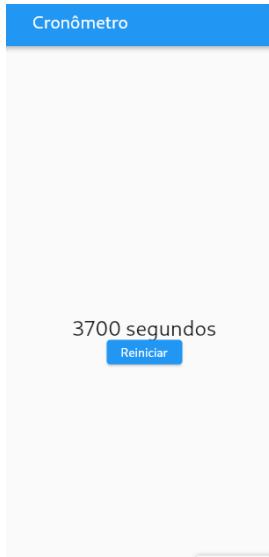
- Antes de criar o botão, vamos adicionar um widget Column em `_CronometroState`
 - Dessa forma, podemos adicionar o contador e um botão na mesma tela.

```
1 body: Column(  
2   // Alinha os elementos da coluna no centro da tela  
3   mainAxisAlignment: MainAxisAlignment.center,  
4   children: <Widget>[  
5     Center(  
6       child: Text(  
7         '$segundos segundos',  
8         style: Theme.of(context).textTheme.headline5,  
9       ),  
10    ),  
11    Center(  
12      child: _criarBotaoDeResetar(context),  
13    ),  
14  ],  
15 ),
```

Navegação

```
1  Widget _criarBotaoDeResetar(BuildContext context) {  
2    return ElevatedButton(  
3      onPressed: () {  
4        setState(() {  
5          segundos = 0;  
6        });  
7      },  
8      child: Text('Resetar'),  
9    );  
10 }
```

Navegação



Navegação

- O atributo `onPressed` determina o que acontece quando o botão é pressionado.
 - No nosso caso, ele chama o método `setState` que zera o contador de segundos.
 - O `onPressed` está associado a uma função que é passada como parâmetro.
 - Quando o botão é pressionado, ele executa a função a qual ele está associado.

Navegação

- O Flutter utiliza uma classe chamada Navigator para gerenciar as telas do aplicativo.
 - O Navigator mantém um histórico de todas as telas chamadas e a ordem delas.

Navegação

- Dentro do arquivo `main.dart` crie uma classe chamada `TelaPrincipal`.

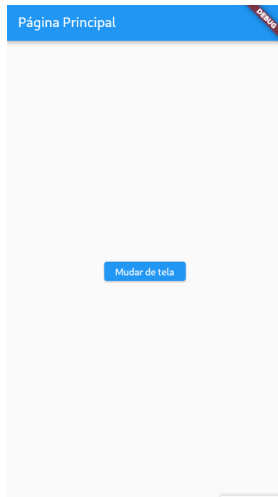
```
1 class TelaPrincipal extends StatelessWidget {  
2   @override  
3   Widget build(BuildContext context) {  
4     return Scaffold(  
5       appBar: AppBar(  
6         title: Text('Página Principal'),  
7       ),  
8       body: Center(  
9         child: ElevatedButton(  
10          child: Text('Mudar de tela'),  
11          onPressed: () {  
12            Navigator.push(  
13              context,  
14              MaterialPageRoute(builder: (context) => Cronometro()),  
15            );  
16          },  
17        ),  
18      ),  
19    );  
20  }  
21 }
```

Navegação

- E altere a classe AppCronometro:

```
1  class AppCronometro extends StatelessWidget {  
2      Widget build(BuildContext context) {  
3          return MaterialApp(  
4              home: TelaPrincipal(),  
5          );  
6      }  
7  }
```

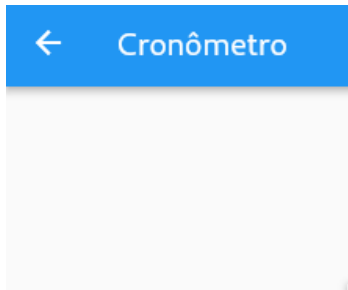
Navegação



Navegação

- Navigators funcionam como uma pilha:
 - Você pode adicionar componentes na pilha com push.
 - Você pode remover componentes da pilha com pop.
- Utilizamos o push para adicionar a tela com o cronômetro na lista de telas.
 - Agora vamos utilizar o pop para criar um botão de voltar.

Navegação



Navegação

- Vamos criar um método `_criarBotaoDeVoltar`.

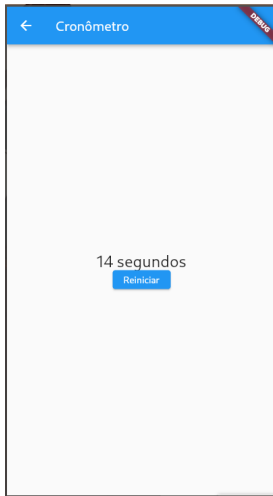
```
1  Widget _criarBotaoDeVoltar(BuildContext context) {  
2    return IconButton(  
3      icon: Icon(Icons.arrow_back),  
4      onPressed: () {  
5        Navigator.pop(context);  
6      },  
7    );  
8  }
```

Navegação

- E vamos adicionar um botão no AppBar na tela do cronômetro:

```
1  @override
2  Widget build(BuildContext context) {
3    return Scaffold(
4      appBar: AppBar(
5        title: Text('Cronometro'),
6        leading: _criarBotaoDeVoltar(context),
7      ),
8      ...
```

Navegação



Navegação

- Além do `ElevatedButton`, existe também o `IconButton`.
 - Ele é uma imagem que reage a toques do usuário.
 - Além do botão de voltar, é possível criar botões de edição de texto, de salvar, etc.
- Usando o conhecimento que temos de navegação agora, podemos criar um menu de navegação.

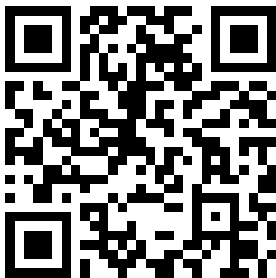
Referências

 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