



# Flutter - Aula 1

Exercícios 1 a 5



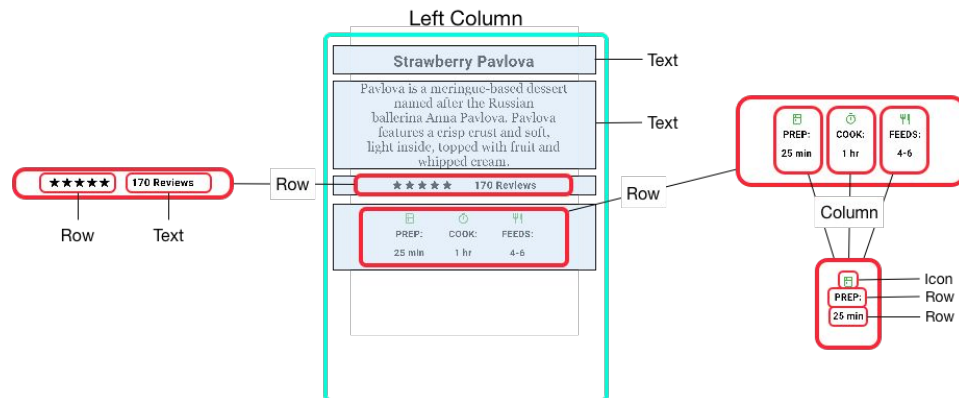
- Vamos seguir o passo a passo no site do Flutter;



- Android Studio; [Ver mais](#)
- Dependências; [Ver mais](#)
- Flutter CLI - Flutter Command-Line Tool; [Ver mais](#)

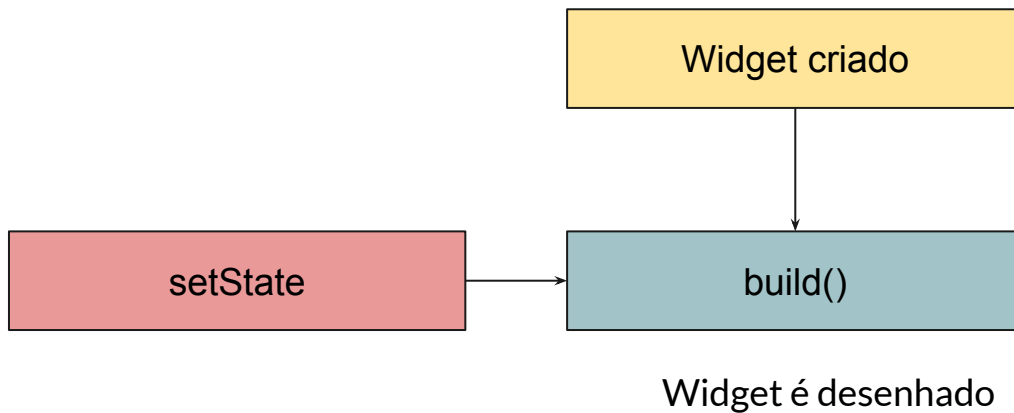


- Componentes;
- Como um conjunto de Lego;
- Widgets de Layout: Servem para determinar a estrutura de layout;
- Widgets de Interface;






- StatefulWidget: Possui controle de estados;
- StatelessWidget: Não possui controle de estados;
- setState;





- Indica que o App utiliza Material Design
- Primeiro Widget chamado
- Configure temas

```
1  import 'package:flutter/material.dart';
2
3  >> void main() {
4      runApp(const MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8      const MyApp({Key? key}) : super(key: key);
9
10     @override
11     Widget build(BuildContext context) {
12         return MaterialApp(
13             theme: ThemeData(
14                 brightness: Brightness.dark,
15                 primaryColor: Colors.blueGrey
16             ), // ThemeData
17             debugShowCheckedModeBanner: false,
18             home: Scaffold(
19                 appBar: AppBar(
20                     title: const Text('MaterialApp Theme'),
21                 ), // AppBar
22                 body: Text("La Belle de Jour")
23             ), // Scaffold
24         ); // MaterialApp
25     }
26 }
27
```

- 
- Estrutura padrão de layout
  - appBar
  - body
  - floatingActionButton

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(const MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    const MyApp({Key? key}) : super(key: key);
9
10   @override
11   Widget build(BuildContext context) {
12     return MaterialApp(
13       theme: ThemeData(
14         brightness: Brightness.dark,
15         primaryColor: Colors.blueGrey
16       ), // ThemeData
17       debugShowCheckedModeBanner: false,
18       home: Scaffold(
19         appBar: AppBar(
20           title: Text("Título"),
21         ), // AppBar
22         body: Center(
23           child: Text("Centro da tela"),
24         ), // Center
25         floatingActionButton: FloatingActionButton(
26           onPressed: () => print("Apertou botão"),
27           child: Icon(Icons.access_time_rounded),
28         ), // FloatingActionButton
29       ), // Scaffold
30     ); // MaterialApp
31   }
32 }
```



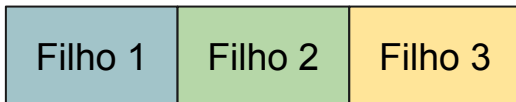
- Componente básico de layout
- padding
- margin

```
child: Container(  
  child: Text(":"), style: TextStyle(color: Colors.black, fontSize: 32)),  
  padding: EdgeInsets.all(32),  
  decoration: BoxDecoration(  
    shape: BoxShape.circle,  
    color: Colors.amberAccent,  
  ), // BoxDecoration  
), // Container
```





- Mostra os widgets filhos horizontalmente



```
body: Row(  
  children: [  
    Text("Texto 1"),  
    Text("Texto 2"),  
    Text("Texto 3")  
  ],  
), // Row
```



- Mostra os widgets filhos verticalmente



```
body: Column(  
  children: [  
    Text("Texto 1"),  
    Text("Texto 2"),  
    Text("Texto 3")  
  ],  
), // Column
```



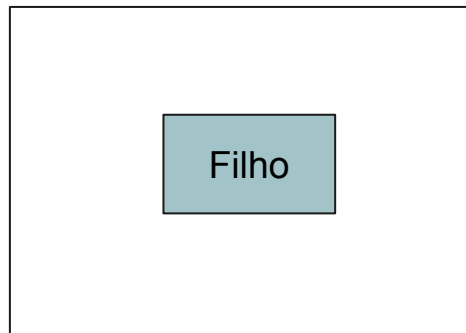
- Centraliza o widget filho na tela



```
- body: Center(  
  | child: Text("Texto centralizado"),  
  |  
), // Center
```



- Adiciona um espaço ao redor do widget filho
- EdgeInsets



```
children: [  
  Padding(  
    padding: const EdgeInsets.fromLTRB(0, 0, 12, 0),  
    child: Container(  
      decoration: BoxDecoration(color: Colors.blue),  
      height: 32,  
      width: 32,  
    ), // Container  
  ), // Padding
```



- Texto

```
dy: Text("Flutter é legal :)",  
  style: TextStyle(  
    color: Colors.amberAccent,  
    backgroundColor: Colors.white,  
    fontWeight: FontWeight.bold,  
    fontFamily: "Courier New",  
    fontSize: 48)), // TextStyle, Text
```



Crie uma tela, utilizando a estrutura padrão de layout (Scaffold).

Nesta tela, deve existir um texto centralizado (o que está escrito fica a seu critério).

Esse texto deve estar em negrito, com um tamanho de fonte igual a 24.

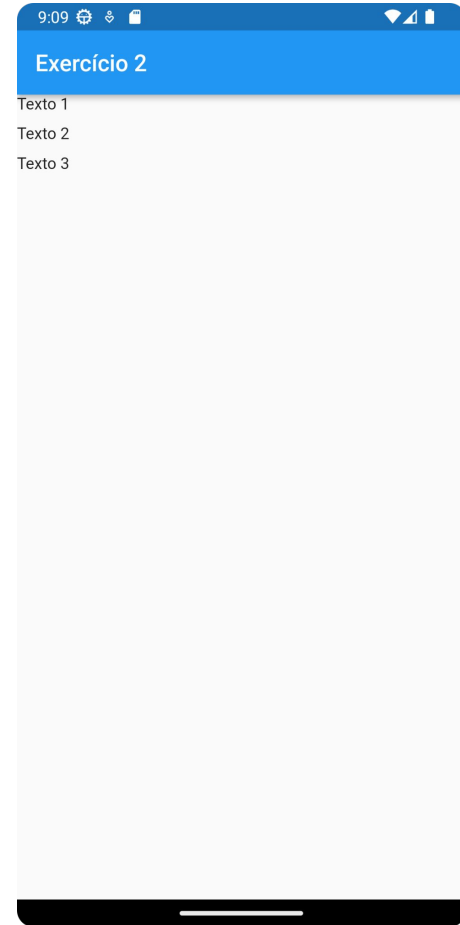




Crie uma tela, utilizando a estrutura padrão de layout (Scaffold).

Nesta tela, deverão ser exibidos três textos, sendo exibidos um em cima do outro.

Adicione um espaço no texto do meio, para que os três não fiquem muito colados.





- TextButton
- ElevatedButton
- OutlinedButton

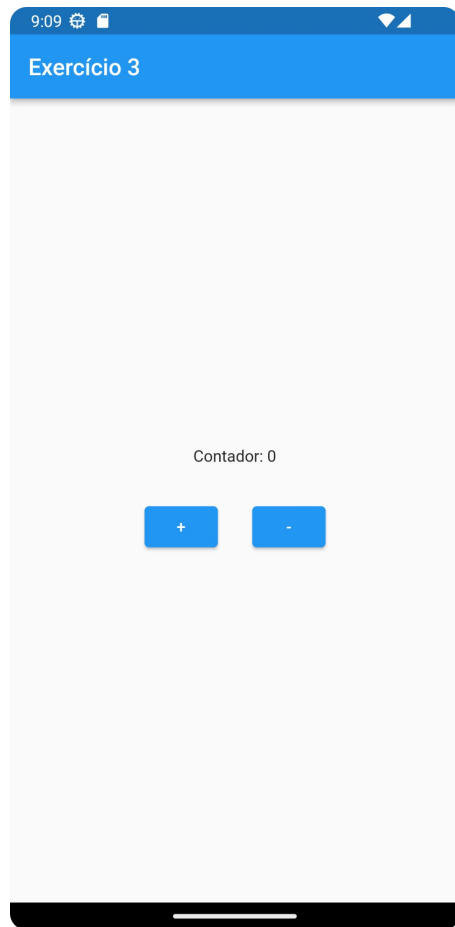
```
TextButton(  
  onPressed: () {  
    print("Cliquei em um TextButton");  
  },  
  child: Text("TextButton")), // TextButton  
  
ElevatedButton(  
  onPressed: () => print("Cliquei em um ElevatedButon"),  
  child: Text("ElevatedButton")), // ElevatedButton  
  
OutlinedButton(  
  onPressed: clicouOutlineButton, child: Text("OutlineButton")) // OutlinedButton
```





Crie uma tela com um contador e dois botões.

- Um botão (+) irá incrementar o contador;
- Um botão (-) irá decrementar o contador;
- Exiba o contador em tela;





- Caixa de texto
- Controllers

```
width: 300,  
  
child: TextField(  
  controller: editController,  
  obscureText: false,  
  decoration: InputDecoration(  
    border: OutlineInputBorder(),  
    labelText: "Meu texto"  
  ), // InputDecoration  
) // TextField
```



Vamos finalmente criar a nossa calculadora...

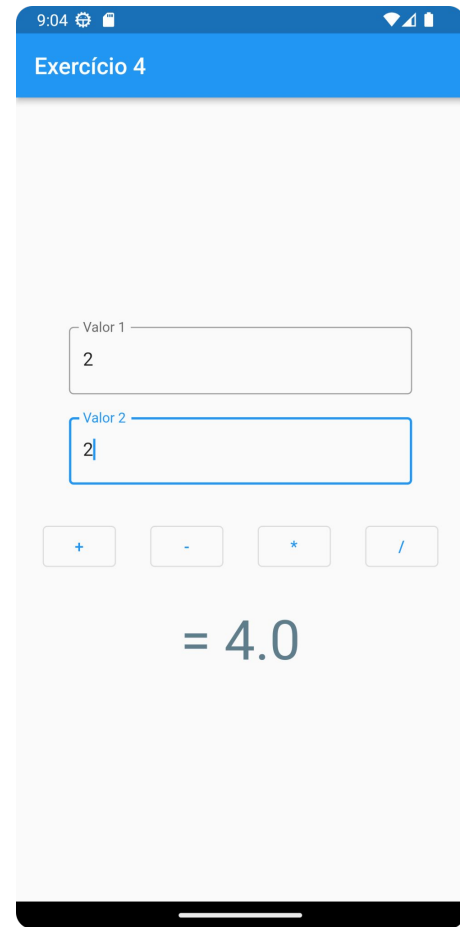
Ela deve possuir dois TextFields em campo, com os valores a calcular e quatro botões:

+

-

\*

/





- Exiba imagens no seu app
- JPEG, PNG, GIF, WebP, BMP, WBMP
- Imagens da internet, com o NetworkImage
- Imagens dos seus assets, com AssetImage

```
children: [  
  Image(  
    image: AssetImage("assets/happy_santaclaus.png"),  
  ), // Image  
  Image(  
    image: NetworkImage("https://www.iconbolt.com/preview/facebook/remix-icon-fill/flutter.svg"),  
  ), // Image  
],
```



Vamos fazer um app de Natal!  
Para isso, crie um app que exiba uma foto do papai noel,  
e uma foto de uma árvore de natal.

