

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

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

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

1. Importação de pacotes

Importa o pacote principal de widgets e componentes visuais do Flutter baseado no *Material Design*. Sem ele, widgets como *Scaffold*, *AppBar* e *FloatingActionButton* não estariam disponíveis.

2. Função principal da aplicação

- **main()** é o ponto de entrada da aplicação.
- **runApp()** inicializa o Flutter e renderiza o widget passado como parâmetro.
- **MyApp()** é o primeiro widget da árvore, marcado como constante para otimizar performance.

3. Classe MyApp – Estrutura raiz da aplicação

- **MyApp** estende **StatelessWidget**, indicando que seu conteúdo não muda durante a execução.
- O construtor com **super.key** permite o uso de chaves no widget.

```

@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Flutter Demo',
    theme: ThemeData(
      colorScheme: ColorScheme.fromSeed(seedColor:
Colors.deepPurple),
    ),
    home: const MyHomePage(title: 'Aula 01 -
Flutter'),
  );
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required
this.title});
  final String title;

```

Método build

MaterialApp é o widget principal que configura:

- **title**: identificador da aplicação.
- **theme**: tema global da interface. Aqui é criado a partir de uma cor-semente.
- **home**: widget inicial da aplicação. Nesse caso, MyHomePage.

4. Classe MyHomePage – Tela principal (Stateful)

- **StatefulWidget** é usado porque haverá mudança de estado na tela (o contador).
- O campo **title** é obrigatório e será usado na **AppBar**.

<pre>@override State<MyHomePage> createState() => _MyHomePageState(); }</pre> <pre>class _MyHomePageState extends State<MyHomePage> { int _counter = 0; void _incrementCounter() { setState(() { _counter++; }); } @override Widget build(BuildContext context) { return Scaffold(</pre>	<p>Criando o estado</p> <p>Retorna a instância da classe de estado <code>_MyHomePageState</code>, que conterá a lógica dinâmica.</p> <p>5. Classe <code>_MyHomePageState</code> – Controle do estado</p> <ul style="list-style-type: none">• <code>_counter</code> é uma variável privada usada para armazenar o valor atual do contador. <p>Função de incremento</p> <ul style="list-style-type: none">• <code>setState()</code> notifica o Flutter de que houve alteração no estado.• Isso causa a reconstrução da interface para refletir o novo valor. <p>6. Construção da interface da tela</p> <p>Scaffold oferece estrutura visual padrão: AppBar, Body, Floating Button etc.</p>
---	--

```

appBar: AppBar(
  backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
  title: Text(widget.title),
),

body: Center(
  child: Column(
    mainAxisAlignment: .center,
    children: [
      const Text('Pressione o botão algumas
vezes:'),
      Text(
        '$_counter',
        style:
Theme.of(context).textTheme.headlineMedium,
      ),
    ],
  ),
),

```

AppBar

- Usa a cor inversa do tema.
- Exibe o título enviado a MyHomePage.

Corpo da tela

- **Center**: centraliza seu filho.
- **Column**: organiza os widgets verticalmente.
- Primeiro texto: estático.
- Segundo texto: exibe o valor atual do contador usando interpolação (\$_counter).

<pre>floatingActionButton: FloatingActionButton(onPressed: _incrementCounter, tooltip: 'Increment', child: const Icon(Icons.add),),); } }</pre>	FloatingActionButton <ul style="list-style-type: none">• Botão flutuante fixo no canto inferior direito.• Ao ser pressionado, chama <code>_incrementCounter</code>.• Ícone de “+” representando incremento.
--	--

Resumo geral do funcionamento

- A aplicação inicia carregando MyApp.
- MyApp configura o tema e define MyHomePage como tela inicial.
- MyHomePage controla um contador que começa em zero.
- O valor do contador é exibido na tela.
- Ao clicar no botão flutuante, o contador aumenta e a interface é atualizada.



