

Programação para Dispositivos Móveis

Prof. Wilson Lourenço

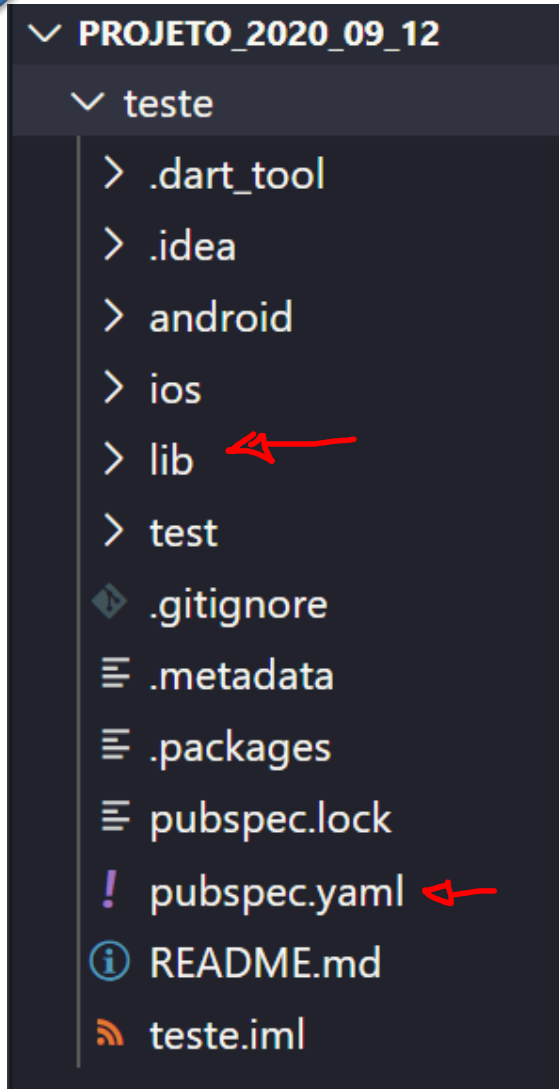
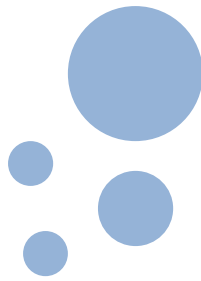


Aula 05 – Introdução ao Flutter(continuação)

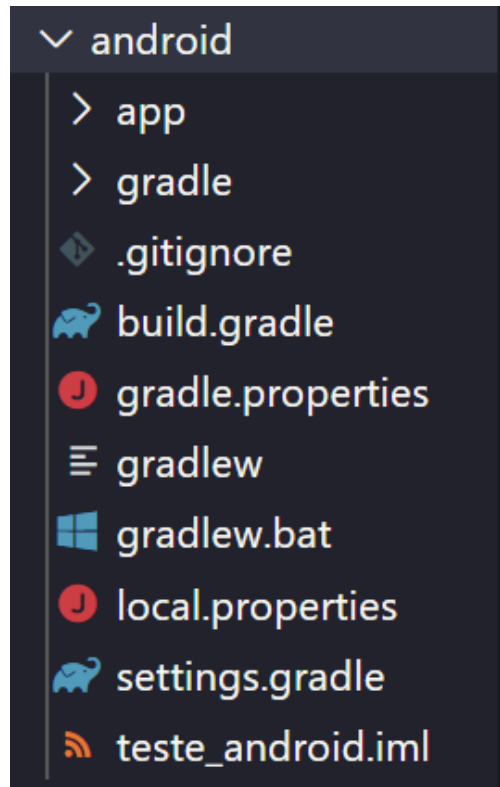
Prof. Wilson Lourenço



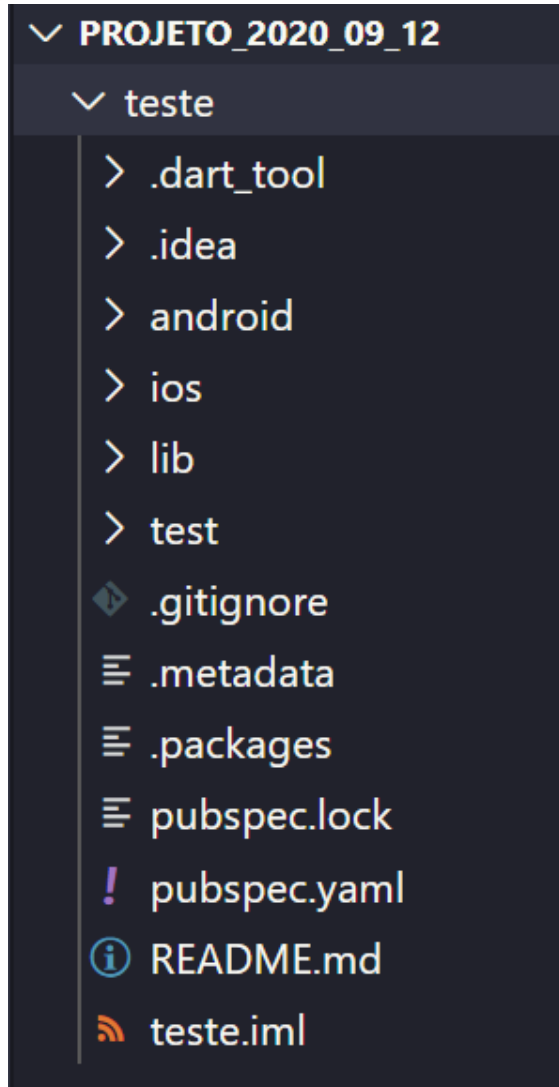
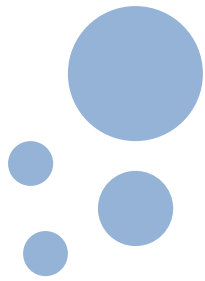
Estrutura de Arquivos



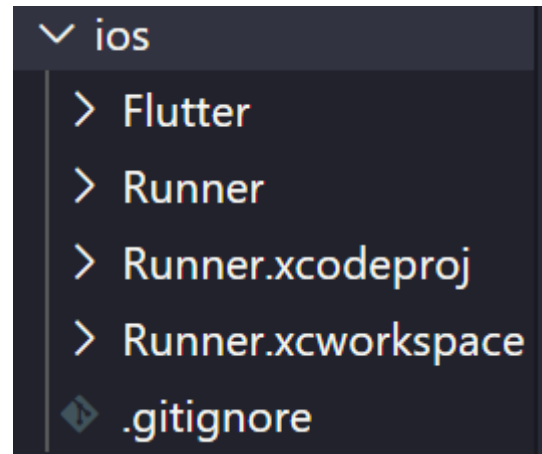
- Android → nessa pasta, temos o projeto Android nativo, criado normalmente em Java.



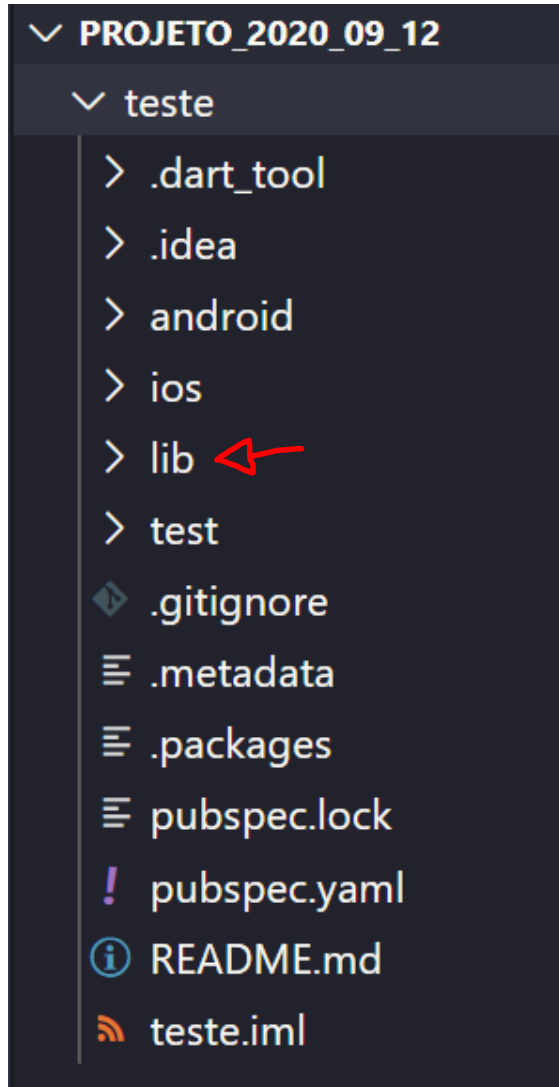
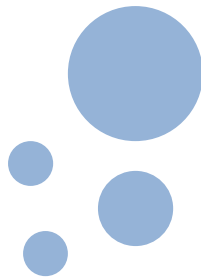
Estrutura de Arquivos



- iOS: nessa pasta, temos o projeto iOS nativo, criado em Swift.



Estrutura de Arquivos

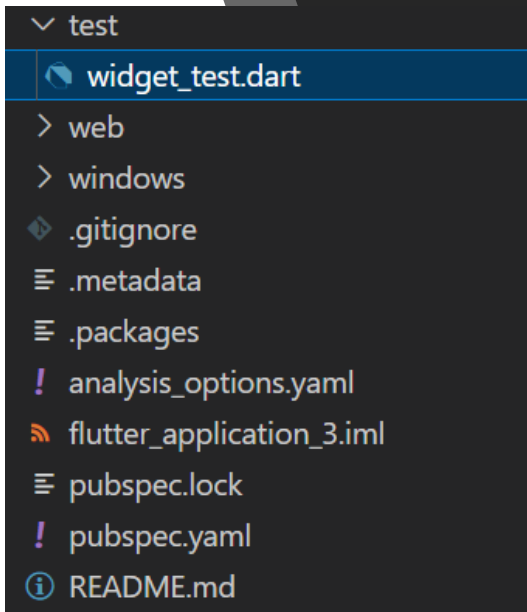


- Lib → nessa pasta, temos por padrão o arquivo main.dart, o código que inicia a nossa aplicação.

Main.dart

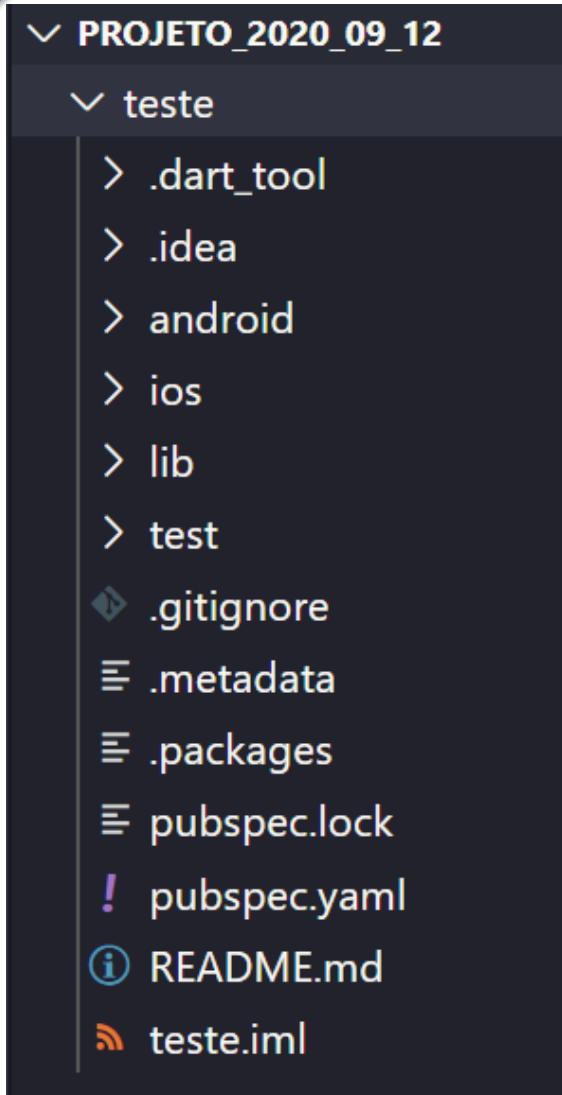
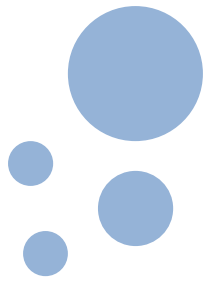
```
1  import 'package:flutter/material.dart';
2  
   Run | Debug
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    // This widget is the root of your application.
9    @override
10   Widget build(BuildContext context) {
11     return MaterialApp(
12       title: 'Flutter Demo',
13       theme: ThemeData(
14         // This is the theme of your application.
15         //
16         // Try running your application with "flutter run". You'll see the
17         // application has a blue toolbar. Then, without quitting the app, try
18         // changing the primarySwatch below to Colors.green and then invoke
19         // "hot reload" (press "r" in the console where you ran "flutter run",
20         // or simply save your changes to "hot reload" in a Flutter IDE).
21         // Notice that the counter didn't reset back to zero; the application
22         // is not restarted.
23         primarySwatch: Colors.blue,
24         // This makes the visual density adapt to the platform that you run
25         // the app on. For desktop platforms, the controls will be smaller and
```

Estrutura de Arquivos



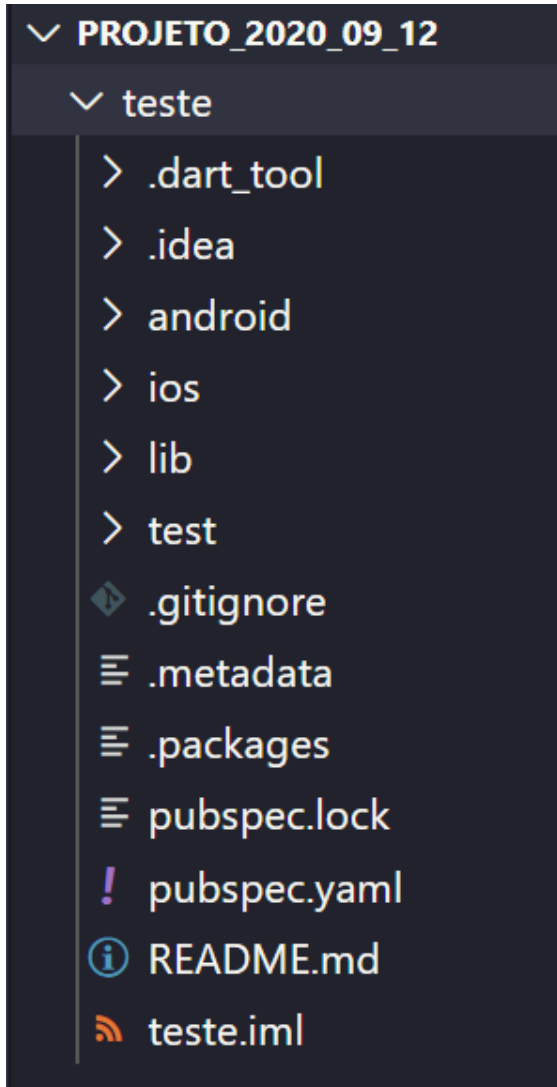
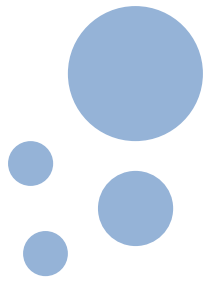
- test - Nesta pasta, temos por padrão o arquivo `widget_test.dart`, o código para testar o funcionamento dos Widgets e interação com o usuário.

Estrutura de Arquivos



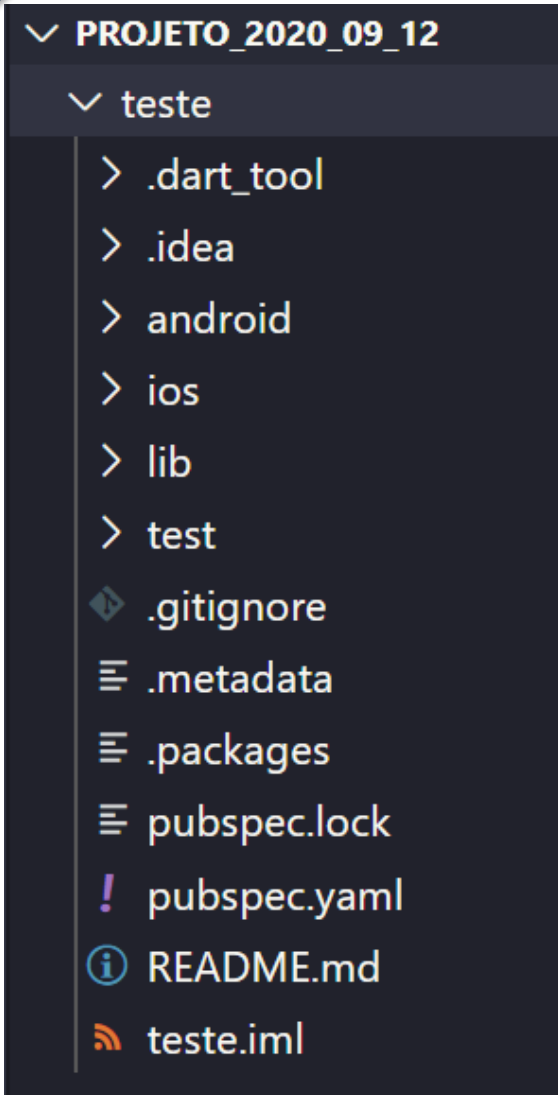
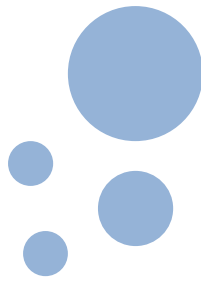
- .gitignore → controle de versão, onde especificamos que o gitignore não será versionado.

Estrutura de Arquivos



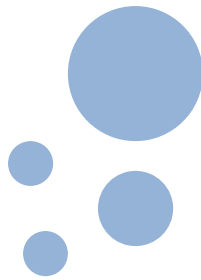
- .metadata → propriedades específicas do projeto Flutter.
- É responsável por fornecer os dados para atualizações do framework.

Estrutura de Arquivos



- .packages → é aqui que o SDK do Flutter salva as urls das dependências que ele necessita mais frequentemente.

Estrutura de Arquivos



✓ PROJETO_2020_09_12

✓ teste

> .dart_tool

> .idea

> android

> ios

> lib

> test

🔍 .gitignore

≡ .metadata

≡ .packages

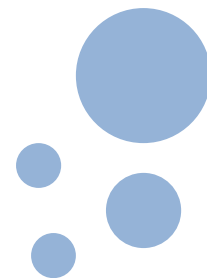
≡ pubspec.lock

! pubspec.yaml

📘 README.md

📡 teste.iml

- Pubspec.yaml → coração das dependências e controle do aplicativo.
- Onde especificamos as dependências/extensões
- Fontes/imagens/videos



✓ PROJETO_2020_09_12

✓ teste

> .dart_tool

> .idea

> android

> ios

> lib

> test

📁 .gitignore

≡ .metadata

≡ .packages

≡ pubspec.lock

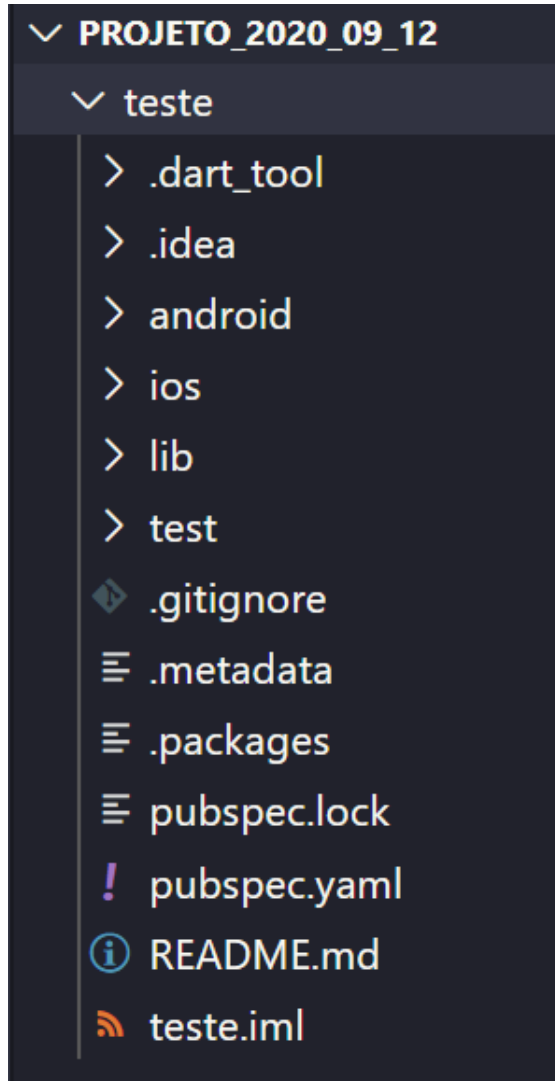
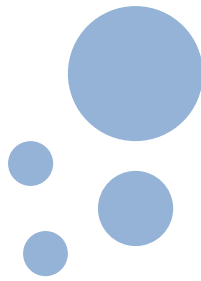
! pubspec.yaml

📄 README.md

📄 teste.iml

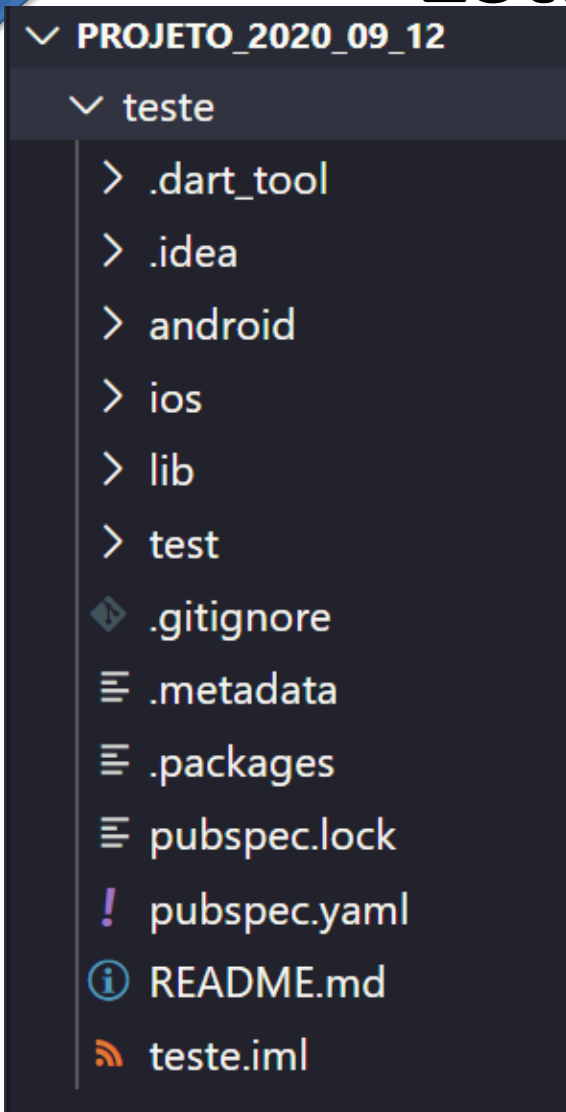
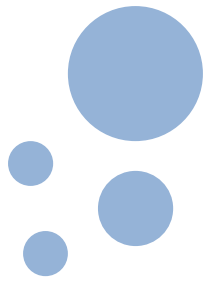
- Pubspec.lock → manter a compatibilidade das dependências

Estrutura de Arquivos



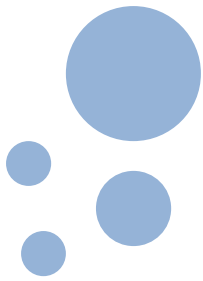
- README.md → Descritivo para criar anotações sobre o aplicativo e dicas de instalação

Estrutura de Arquivos



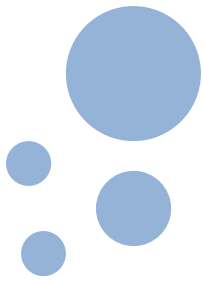
- Teste.iml → é um facilitador para o Dart se comunicar com o interpretador Java na hora de gerar o aplicativo para o Android.
- É gerado automaticamente pelo Flutter não deve ser editado.

Widgets



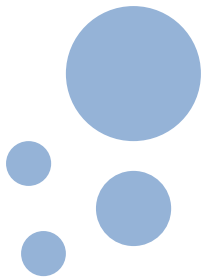
- Os Widgets são o que você vai utilizar para construir o seu aplicativo.
- Exemplos:
 - Lista
 - Botão
 - Barra de Busca
 - E todo e qualquer outro tipo de elemento que você possa precisar criar a interface gráfica do aplicativo.

Widgets



- Duas modalidades
- 1. Personalizados
 - Criar ou baixar de algum desenvolvedor
- 2. Básicos
 - Padrão vem no SDK Flutter
 - Botões
 - Campos de texto
 - Ícones
 - Lista

Widgets



- Trabalham como padrão reativa inspirada nos moldes do React
- Descrevem como a interface gráfica e estados da aplicação devem funcionar.
- Quando o estado de um Widget muda, é recriado a sua “Existência”, mudando apenas o que foi alterado.
 - Realizando o mínimos de alterações no árvore de componentes da página.
 - Compilação JIT (modo de desenvolvimento)

- <https://flutter.dev/docs/reference/widgets>

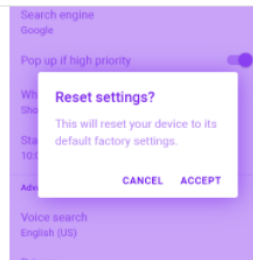
Widget of the Week playlist



AbsorbPointer

A widget that absorbs pointers during hit testing. When absorbing is true, this widget prevents its subtree from receiving pointer events by terminating hit testing at itself. It still consumes space during layout and paints its child as usual. It just prevents its children from being the target of located events, because it returns true from `RenderBox.hitTest`.

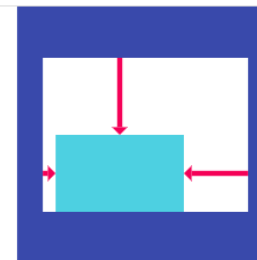
[Documentation](#)



AlertDialog

Alerts are urgent interruptions requiring acknowledgement that inform the user about a situation. The `AlertDialog` widget implements this component.

[Documentation](#)



Align

A widget that aligns its child within itself and optionally sizes itself based on the child's size.

[Documentation](#)



Abc

Text

A run of text with a single style.

[Documentation](#)

Text class

A run of text with a single style.

The `Text` widget displays a string of text with single style. The string might break across multiple lines or might all be displayed on the same line depending on the layout constraints.

The `style` argument is optional. When omitted, the text will use the style from the closest enclosing `DefaultTextStyle`. If the given style's `TextStyle.inherit` property is true (the default), the given style will be merged with the closest enclosing `DefaultTextStyle`. This merging behavior is useful, for example, to make the text bold while using the default font family and size.

Sample



This example shows how to display text using the `Text` widget with the overflow set to `TextOverflow.ellipsis`.

Hello, Ruth! How are you?

Hello, Ruth!...

```
Text(  
  'Hello, $_name! How are you?',  
  textAlign: TextAlign.center,
```



TextBox class

A rectangle enclosing a run of text.

This is similar to [Rect](#) but includes an inherent [TextDirection](#).

Constructors

[TextBox.fromLTRB](#)(double left, double top, double right, double bottom, [TextDirection](#) direction)

Creates an object that describes a box containing text.

const

Properties

[bottom](#) → double

The bottom edge of the text box.

final

[direction](#) → [TextDirection](#)

The direction in which text inside this box flows.

final

[end](#) → double

The [right](#) edge of the box for left-to-right text; the [left](#) edge of the box for right-to-left text. [...]

read-only

[hashCode](#) → int

The hash code for this object. [...]

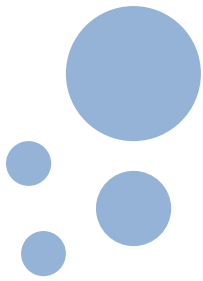
read-only, override

[left](#) → double

The left edge of the text box, irrespective of direction. [...]

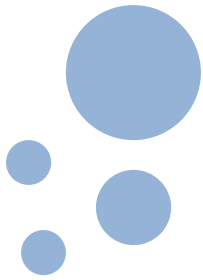
final

Material Design



- Material Design = modelo de componentes e estilização criado e especificado pelo Google para a construção de interfaces gráficas.
- `import 'package:flutter/material.dart';`

Material Design



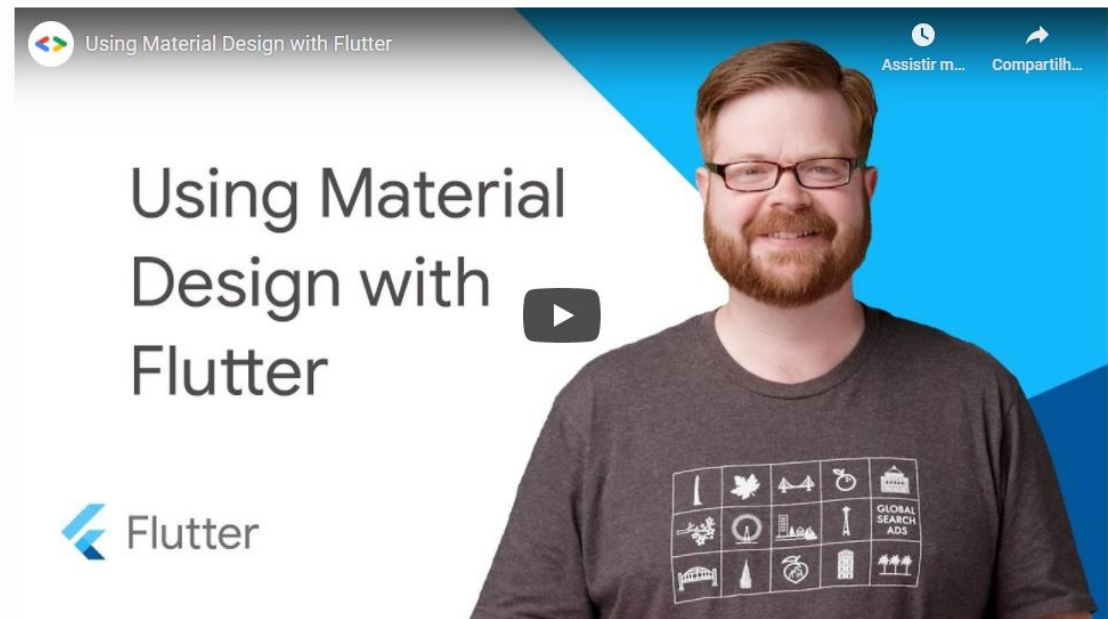
- O Navegador permite fazer a transição sem problemas entre as telas do seu aplicativo no modelo de pilha.
- Cada nova página aberta vai para o topo da pilha. A última página a abrir e a primeira a ser fechada.

- <https://api.flutter.dev/flutter/material/material-library.html>

material library

Flutter widgets implementing Material Design.

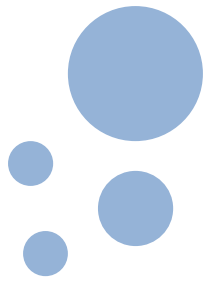
To use, import `package:flutter/material.dart`.



See also:

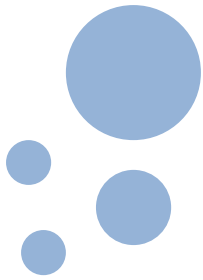
- flutter.dev/widgets for a catalog of commonly-used Flutter widgets.
- material.io/design for an introduction to Material Design.

Material Design



```
main.dart X
lib > main.dart > main
1  import 'package:flutter/material.dart';
2
   Run | Debug | Profile
3  void main() {
4    runApp(const MaterialApp(
5      home: TutorialHome(),
6    )); // MaterialApp
7  }
8
9  class TutorialHome extends StatelessWidget {
10    const TutorialHome({Key? key}) : super(key: key);
11
12    @override
13    Widget build(BuildContext context) {
14      return const Scaffold(
15        body: Center(
16          child: Text('Olá mundo!'),
17        ), // Center
18      ); // Scaffold
19    }
20  }
```


Material Design



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

```
void main() {  
  runApp(const MaterialApp(  
    home: TutorialHome(),  
  ));  
}
```

```
class TutorialHome extends StatelessWidget {  
  const TutorialHome({Key? key}) : super(key: key);
```

```
  @override
```

```
  Widget build(BuildContext context) {  
    return const Scaffold(  
      body: Center(  
        child: Text('Olá mundo!'),  
      ),  
    );  
  }  
}
```

- <https://api.flutter.dev/flutter/cupertino/cupertino-library.html>

cupertino library

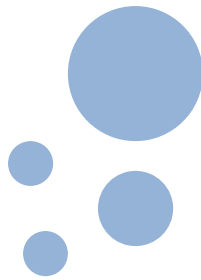
Flutter widgets implementing the current iOS design language.

To use, import `package:flutter/cupertino.dart`.



Classes

Scaffold



- O elemento scaffold implementa a estrutura básica do layout do Material Design.
- É Através do scaffold que pode especificar uma toolbar, configurar o texto, ícones, alinhamentos
- Especificamos o corpo da página body
- Para utilizar o scaffold é necessária a implementação do AppBar e do Body

- <https://api.flutter.dev/flutter/material/Scaffold-class.html>

Scaffold class

Implements the basic material design visual layout structure.

This class provides APIs for showing drawers, snack bars, and bottom sheets.

To display a snackbar or a persistent bottom sheet, obtain the [ScaffoldState](#) for the current [BuildContext](#) via [Scaffold.of](#) and use the [ScaffoldState.showSnackBar](#) and [ScaffoldState.showBottomSheet](#) functions.

Interactive App

Sample code

This example shows a [Scaffold](#) with a body and [FloatingActionButton](#). The body is a [Text](#) placed in a [Center](#) in order to center the text within the [Scaffold](#). The [FloatingActionButton](#) is connected to a callback that increments a counter.

Sample Code

You have pressed the button 0 times.

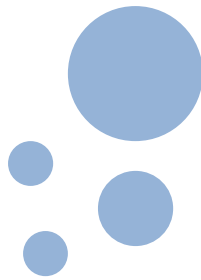


```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MaterialApp(
6     home: TutorialHome(),
7   )); // MaterialApp
8
9   class TutorialHome extends StatelessWidget {
10     const TutorialHome({Key? key}) : super(key: key);
11
12     @override
13     Widget build(BuildContext context) {
14       return Scaffold(
15         appBar: AppBar(
16           leading: const IconButton(
17             onPressed: null,
18             tooltip: 'Menu de navegação',
19             icon: Icon(Icons.menu),
20           ), // IconButton
21         title: const Text('Tutorial Home'),
22       ), // AppBar
23       body: const Center(
24         child: Text('Olá mundo!'),
25       ), // Center
26     ); // Scaffold
27   }
28 }
```

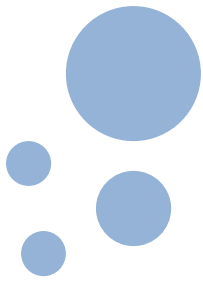


Olá mundo!

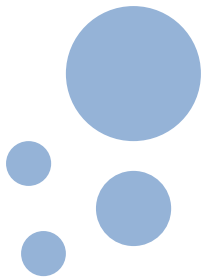
Stateless Widget



- Por padrão os widgets são sem estado.
- O stateless Widget não possibilita alterações dinâmicas, totalmente estático.
- Criado para estruturas estáticas nos aplicativos (telas, menus), tudo o que não envolva inputs dos usuários, acessos a APIs e coisas mutáveis.

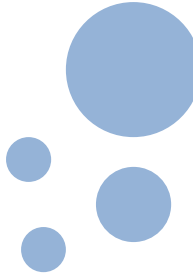



- Todo widget precisa ser declarado como uma classe e estender a classe `StatelessWidget` ou `StatefulWidget`.
- Todo Widget tem um método chamado `build` que retorna os elementos

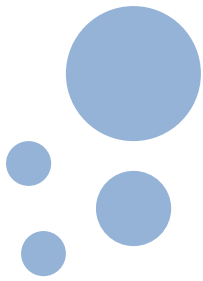


```
main.dart X
lib > main.dart > main
1  import 'package:flutter/material.dart';
2
   Run | Debug | Profile
3  void main(List<String> args) {
4      runApp(const MyWidget());
5  }
6
```

- Digite st, e seleccione stateless widget



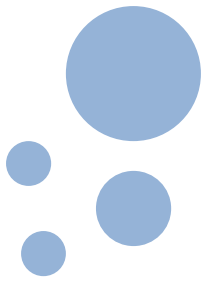
```
7 class MyWidget extends StatelessWidget {  
8   const MyWidget({Key? key}) : super(key: key);  
9  
10  @override  
11  Widget build(BuildContext context) {  
12    return MaterialApp(  
13      theme: ThemeData(primaryColor: Colors.blue),  
14      home: Container(),  
15    ); // MaterialApp  
16  }  
17 }
```



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

```
void main(List<String> args) {  
  runApp(const MyWidget());  
}
```

```
class MyWidget extends StatelessWidget {  
  const MyWidget({Key? key}) : super(key: key);
```

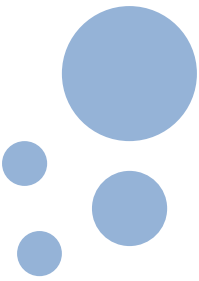


```
@override
```

```
Widget build(BuildContext context) {  
  return MaterialApp(  
    theme: ThemeData(primaryColor: Colors.blue),  
    home: Container(),  
  );  
}  
}
```

- debugShowCheckedModeBanner: false,

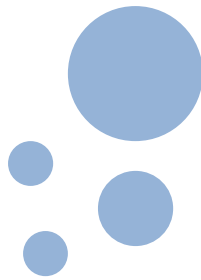
```
7  class MyWidget extends StatelessWidget {  
8    const MyWidget({Key? key}) : super(key: key);  
9  
10   @override  
11   Widget build(BuildContext context) {  
12     return MaterialApp(  
13       debugShowCheckedModeBanner: false,  
14       theme: ThemeData(primaryColor: Colors.blue),  
15       home: Container(),  
16     ); // MaterialApp  
17   }  
18 }
```



Vídeo:

- <https://www.youtube.com/watch?v=s7VNmqw6wLM>

Referências



Flutter Framework

Desenvolva aplicações móveis
no Dart Side!



 Casa do
Código

LEONARDO H. MARINHO

Dicas para Estudo



Seja “CURIOSO”:

Procure revisar o que foi estudado.
Pesquise as referências
bibliográficas.



Seja “ANTENADO”:

Leia a próxima aula.



Seja
“COLABORATIVO”:

Traga assuntos relevantes para a sala
de aula.
Participe da aula.
Proponha discussões relevantes
sobre o conteúdo.



Prof. Wilson Lourenço

