

Flutter

Aula 04

Prof. Dr. Rodrigo Plotze

rodrigoplotze@gmail.com



- DART
- Widgets
 - Value Widgets
 - Navigation Widgets
- Estrutura Básica

DART



The image shows a web browser window with the DartPad interface. The browser's address bar shows the URL `dartpad.dartlang.org`. The DartPad header includes the logo, a "Reset" button, the project name "long-toast-6561", and a "Samples" dropdown menu. The main area is divided into three sections: a code editor on the left, a console on the right, and a documentation panel at the bottom right. The code editor contains the following Dart code:

```
void main(){  
  print("App iniciada");  
  new App();  
  print("App finalizada");  
}  
  
class App{  
  App(){  
    print("Construtor da classe");  
  }  
}
```

A blue "RUN" button is located to the right of the code editor. The console on the right displays the output of the code:

```
App started  
Constructing a class.  
App finished
```

The documentation panel at the bottom right is currently empty. At the bottom of the interface, there are links for "Privacy notice" and "Send feedback", and a status bar indicating "no issues" and "Based on Dart SDK 2.7.1".

dartpad.dev



- Declaração de variáveis
 - Tipo seguido do identificador:
 - A palavra reservada ***var*** declara uma variável sem um tipo específico, a qual recebe um tipo dinamicamente.
 - A palavra reservada ***dynamic*** declara uma variável do tipo dinâmico com um tipo opcional

```
var x;  
  
...or...  
  
<some specific type> x;
```

```
var x = "Mel Brooks";  
String x = "Mel Brooks";
```

```
dynamic x = "Mel Books";
```



■ Tipos de dados

Type	Description
int	Integers (no decimals).
double	Decimal number (double precision).
bool	Boolean true or false.
String	Immutable string.
StringBuffer	Mutable string.
RegExp	Regular expressions.
List, Map, Set	Dart provides Collection classes.
DateTime	A point in time.
Duration	A span of time.
Uri	Uniform Resource Identifier
Error	Error information



■ Strings

The screenshot shows the DartPad web application interface. The browser address bar displays `dartpad.dartlang.org`. The application header includes the DartPad logo, a "Reset" button, the file name "long-toast-6561", and a "Samples" dropdown menu. The main editor area contains the following Dart code:

```
void main(){  
  String s1 = "Rickety Rocket";  
  String s2 = "${s1} blast off!";  
  String s3 = '$s1 blast off!';  
  print (s2);  
  print (s3);  
}
```

A blue "RUN" button is positioned to the right of the code. The right sidebar is divided into two sections: "Console" and "Documentation". The "Console" section displays the output of the program:

```
Rickety Rocket blast  
off!  
Rickety Rocket blast  
off!
```

The "Documentation" section is currently empty. The footer of the application contains links for "Privacy notice" and "Send feedback", a status indicator "1 issues show", and the version information "Based on Dart SDK 2.7.1".



■ Valores Numéricos

The screenshot shows the DartPad web interface in a browser. The URL is `dartpad.dartlang.org`. The interface has a dark theme. At the top, there's a header with the DartPad logo, a "Reset" button, the file name "long-toast-6561", and a "Samples" dropdown menu. The main area is split into three sections: a code editor on the left, a console on the right, and a documentation panel at the bottom right. The code editor contains the following Dart code:

```
void main(){  
  int i = 5;  
  double d = 5.5;  
  String si = i.toString();  
  String sd = d.toString();  
  print(i);  
  print(d);  
  print(si);  
  print(sd);  
}
```

Next to the code editor is a blue "RUN" button. The console on the right shows the output of the program:

```
5  
5.5  
5  
5.5
```

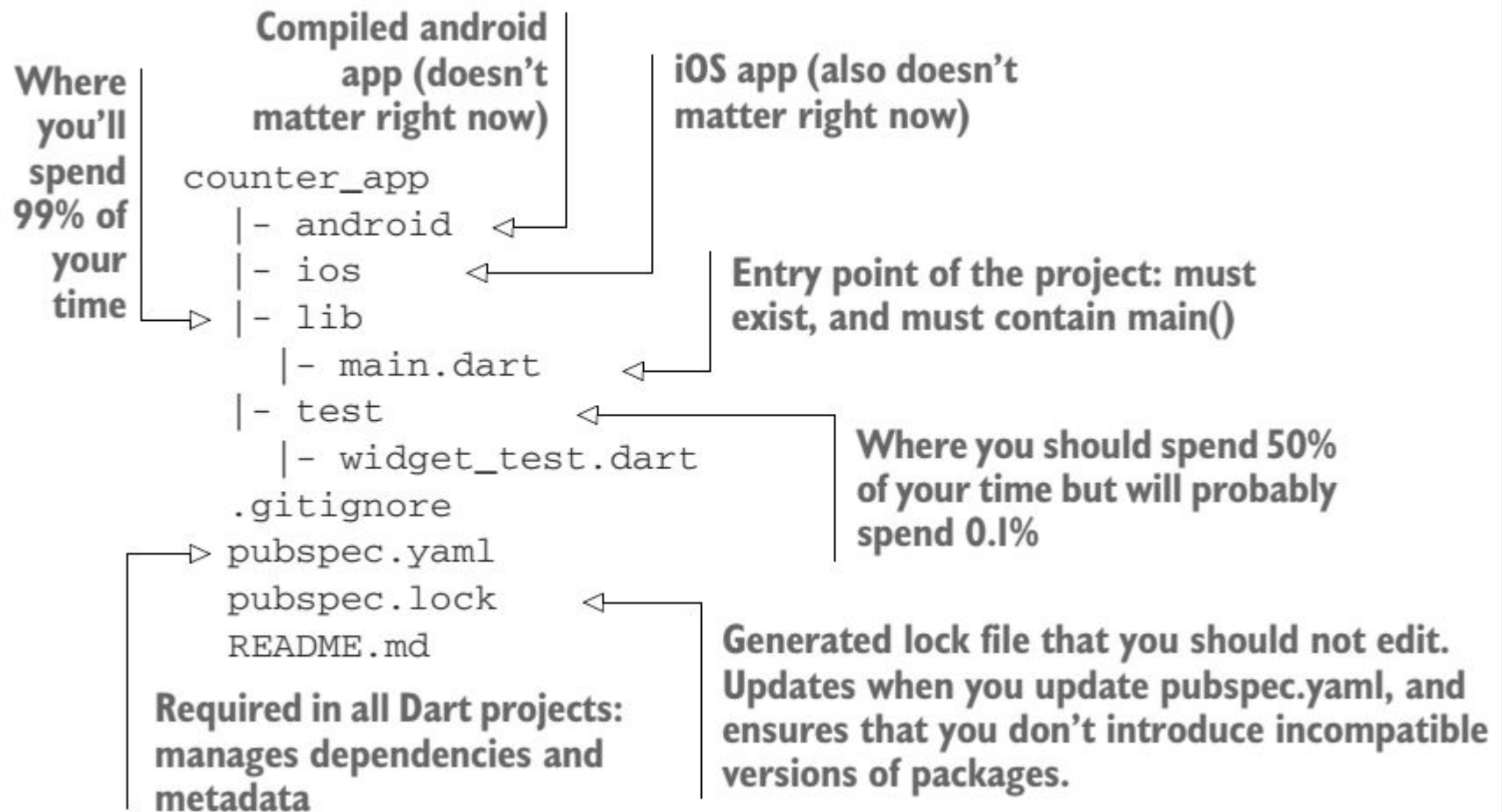
The documentation panel at the bottom right is currently empty. At the bottom of the interface, there's a footer with links for "Privacy notice" and "Send feedback", and status information: "no issues" and "Based on Dart SDK 2.7.1".

WIDGETS





■ Estrutura do projeto





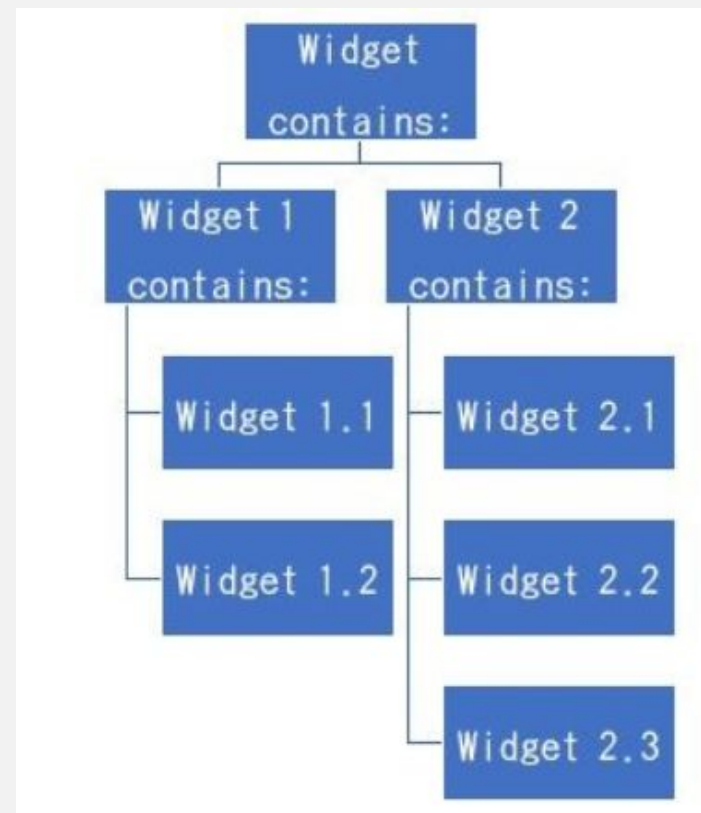
- Flutter utiliza a mesma linguagem para construção da UI e para a programação dos comportamentos

Framework	Behavior expressed in ...	UI expressed in ...
Xamarin	C#	XAML
React Native	JavaScript	JSX
NativeScript	JavaScript	XML
Flutter	Dart	Dart



- Widgets são os blocos de construção da UI.
- A criação de uma UI é realizada por meio de uma composição de *widgets*

Widget Tree □





■ ***Built-in Flutter Widgets***

- Conjunto completo de widgets.
- As aplicações são construídas a partir da composição de widgets prontos.
- Alguns exemplos de widgets:

They fall into these major categories:

- Value widgets
- Layout widgets
- Navigation widgets
- Other widgets



■ *Value Widgets*

- São usados exibir valores para o usuário e obter valores do usuário no App.

Checkbox	FormField	RefreshIndicator
CircularProgressIndicator	Icon	RichText
Date & Time Pickers	IconButton	Slider
DataTable	Image	Switch
DropDownButton	LinearProgressIndicator	Text
FlatButton	PopupMenuButton	TextField
FloatingActionButton	Radio	Tooltip
FlutterLogo	RaisedButton	
Form	RawImage	



▪ *Layout Widgets*

- Utilizados para controlar como os widgets são apresentados na tela.

Align	FittedBox	Padding
AppBar	Flow	PageView
AspectRatio	FractionallySizedBox	Placeholder
Baseline	GridView	Row
BottomSheet	IndexedStack	Scaffold
AppBar	IntrinsicHeight	Scrollable
Card	IntrinsicWidth	Scrollbar
Center	LayoutBuilder	SingleChildScrollView
Column	LimitedBox	SizedBox
ConstrainedBox	ListBody	SizedOverflowBox
Container	ListTile	SliverAppBar
CustomMultiChildLayout	ListView	SnackBar
Divider	MediaQuery	Stack
Expanded	NestedScrollView	Table
ExpansionPanel	OverflowBox	Wrap



■ *Navigation Widgets*

- Quando seu aplicativo tem várias cenas ("telas", "páginas"), você precisará de alguma maneira de se mover entre eles.
- Eles controlam como o usuário vê uma cena e depois passa para o próximo.
- Geralmente isso é feito quando o usuário toca em um botão.

AlertDialog

BottomNavigationBar

Drawer

MaterialApp

Navigator

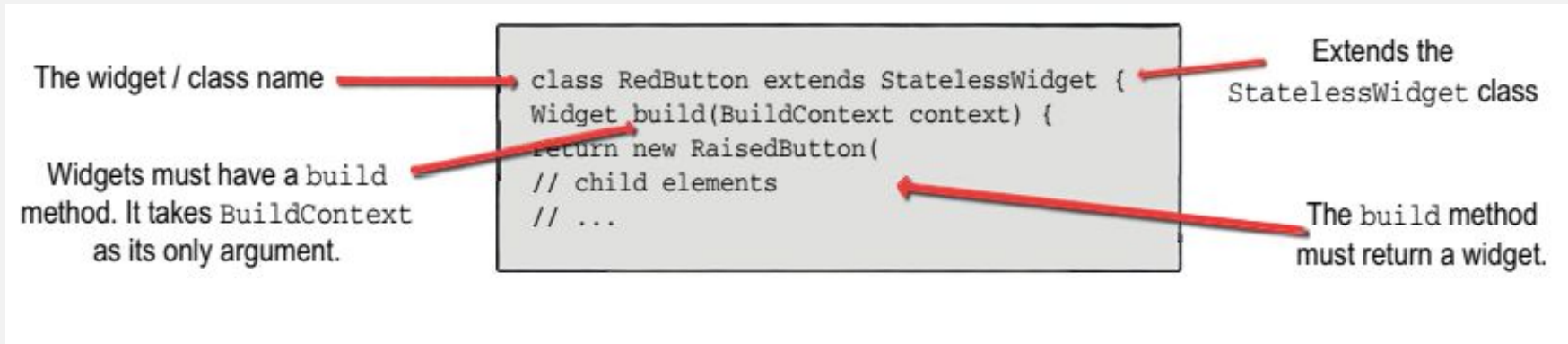
SimpleDialog

TabBar

TabBarView



- Todos os widgets devem conter um método denominado ***build***, o qual deve retornar um outro widget.





Hello Flutter



```
main.dart X
lib > main.dart > ...
1  import "package:flutter/material.dart";
2
3  void main() => runApp(App());
4
5  class App extends StatelessWidget {
6    @override
7    Widget build(BuildContext context) {
8      return new MaterialApp(
9        title: "Meu Primeiro App",
10       home: Container(
11         child: Text("Hello Flutter!"),
12       ));
13   }
14 }
```





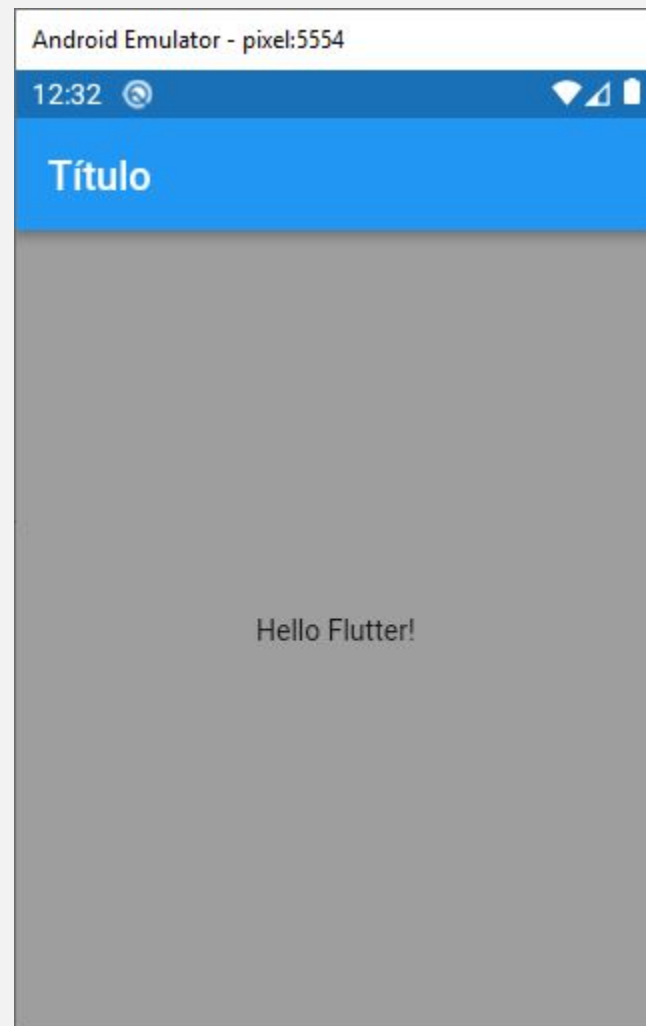
Hello Flutter



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

void main() => runApp(App());

class App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      debugShowCheckedModeBanner: false,
      title: "Meu Primeiro App",
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Título'),
        ),
        body: Center(
          child: Text("Hello Flutter!")
        ),
        backgroundColor: Colors.grey,
      ));
  }
}
```





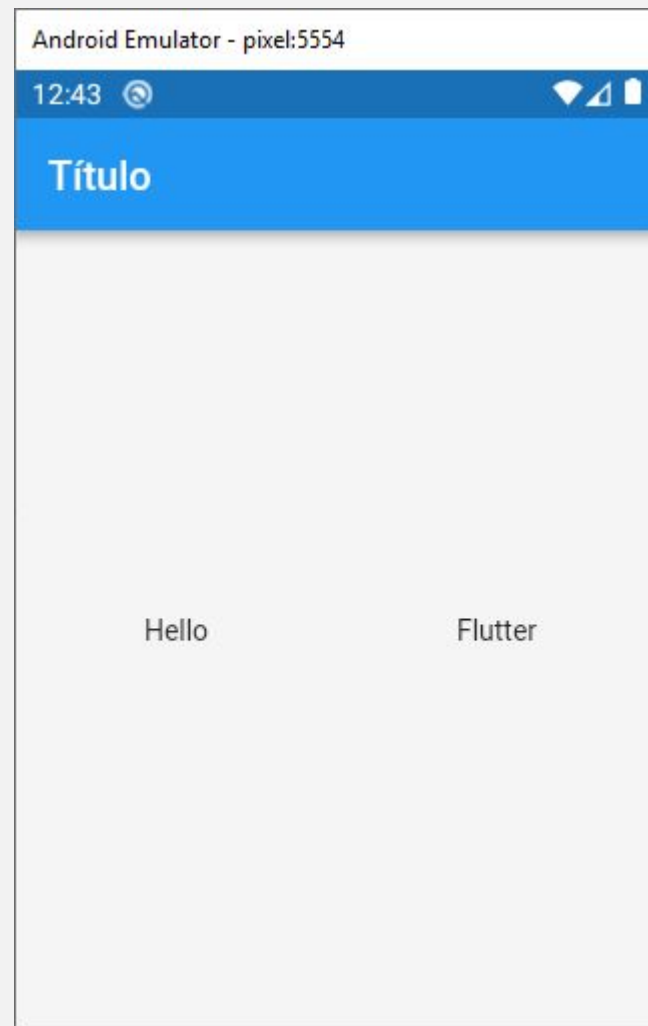
Hello Flutter



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

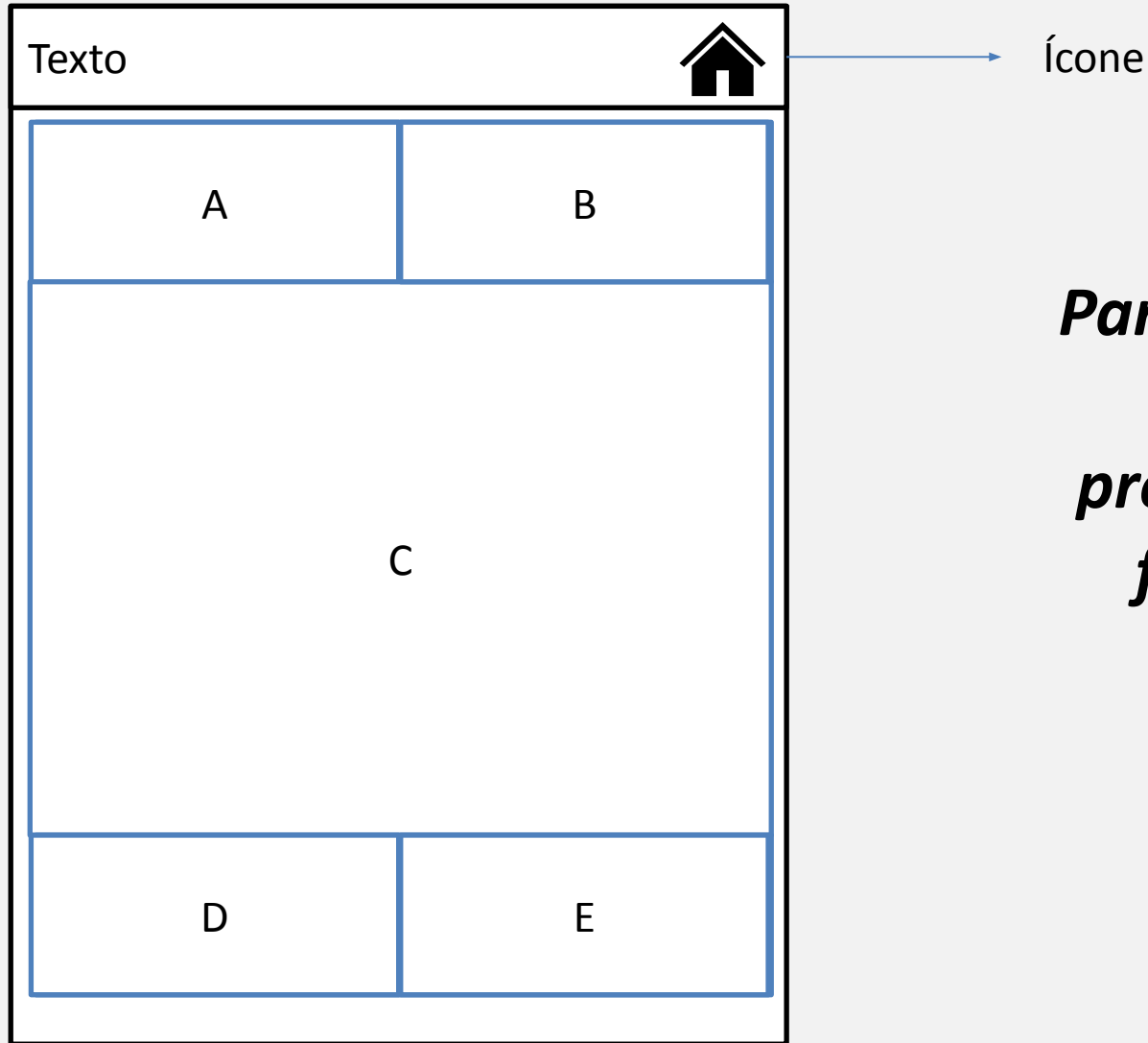
void main() => runApp(App());

class App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      debugShowCheckedModeBanner: false,
      title: "Meu Primeiro App",
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Título'),
        ), // AppBar
        body: Center(child: Row(
          children: <Widget>[
            Expanded(
              child: Text('Hello', textAlign: TextAlign.center),
            ), // Expanded
            Expanded(
              child: Text('Flutter', textAlign: TextAlign.center),
            ), // Expanded
          ], // <Widget>[]
        ), // Row
      ), // Center
      backgroundColor: Colors.grey[100],
    )); // Scaffold // MaterialApp
  }
}
```



ATIVIDADE PRÁTICA

Elabore um App baseado no seguinte layout:



***Para cada widget
especificar
propriedades de
formatação.***