



# Flutter

Wesley Dias Maciel

2021/02

# Prática 03

## Widgets básicos

Documentação: <https://flutter.dev/docs/development/ui/widgets-intro>

**Objetivo:** apresentar alguns widgets básicos e implementar um aplicativo simples com o widget Text.

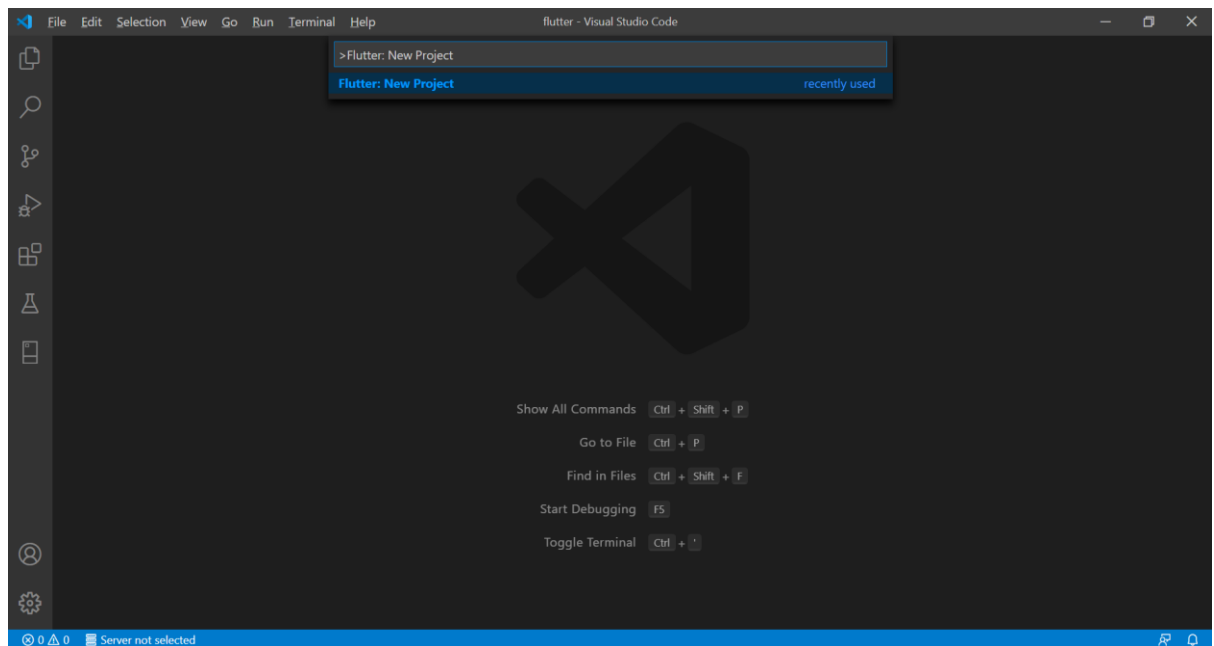
A principal tarefa de um widget é implementar uma função `build()`. A função `build()` descreve esse widget em termos de outros widgets de nível inferior na árvore de renderização. O framework Flutter constrói o widget até chegar em algum widget que represente um objeto da classe básica `RenderObject`. O objeto da classe `RenderObject` é responsável por calcular e descrever a geometria do widget.

Alguns widgets básicos:

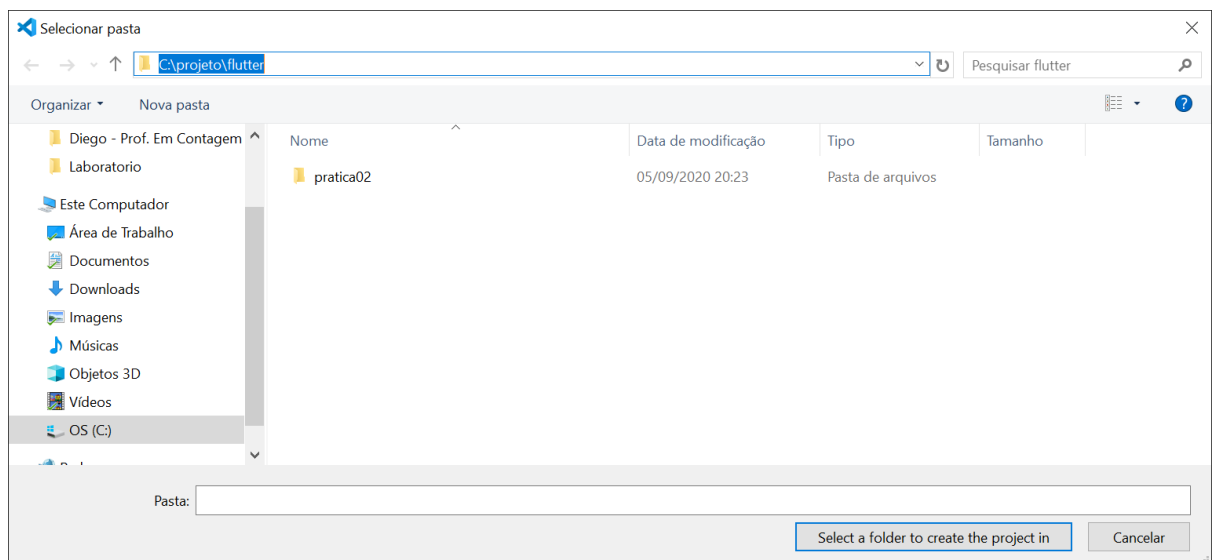
1. **Text:** permite criar texto estilizado em seu aplicativo.
2. **Row:** permite criar layouts flexíveis na direção horizontal (linha). Os objetos filhos de Row são posicionados no eixo x do plano cartesiano. Objetos da classe Row são baseados no modelo de layout **flexbox** da Web.
3. **Column:** permite criar layouts flexíveis na direção vertical (coluna). Os objetos filhos de Column são posicionados no eixo y do plano cartesiano. Objetos da classe Column também são baseados no modelo de layout **flexbox** da Web.
4. **Stack:** ao invés de ser orientado horizontal ou verticalmente, um widget Stack permite que você coloque widgets filhos uns sobre os outros. Os objetos filhos de Stack são posicionados no eixo z do plano cartesiano. Um widget Stack também permite que você use o widget **Positioned**, para determinar o posicionamento dos widgets filhos de Stack. Com o widget **Positioned**, você pode posicionar os filhos de Stack em relação à borda superior, direita, inferior ou esquerda de Stack. Objetos da classe Stack são baseados no modelo de layout de posicionamento **absolute** da Web.
5. **Container:** permite criar um elemento visual retangular. Um objeto da classe Container pode ser estilizado com um **BoxDecoration**. Objetos da classe BoxDecoration permitem criar formatações, como de plano de fundo (background), de bordas ou de sombreadimento. Um Container também pode ter margens, espaçamento lateral (padding) e restrições de tamanho. Além disso, um Container pode ser transformado em um espaço tridimensional usando uma matriz.

- 1) No Visual Studio Code, crie um novo projeto Flutter. Clique em View > Command Palette ou pressione CTRL + SHIFT + P. Na caixa de entrada, informe:

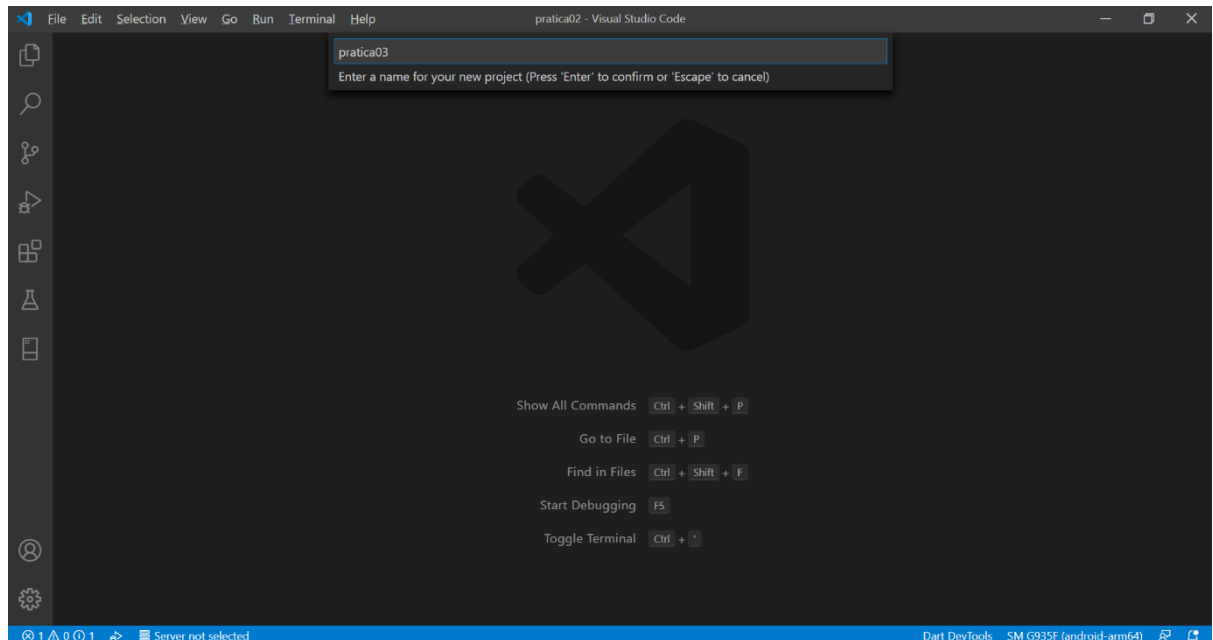
>Flutter: New Project



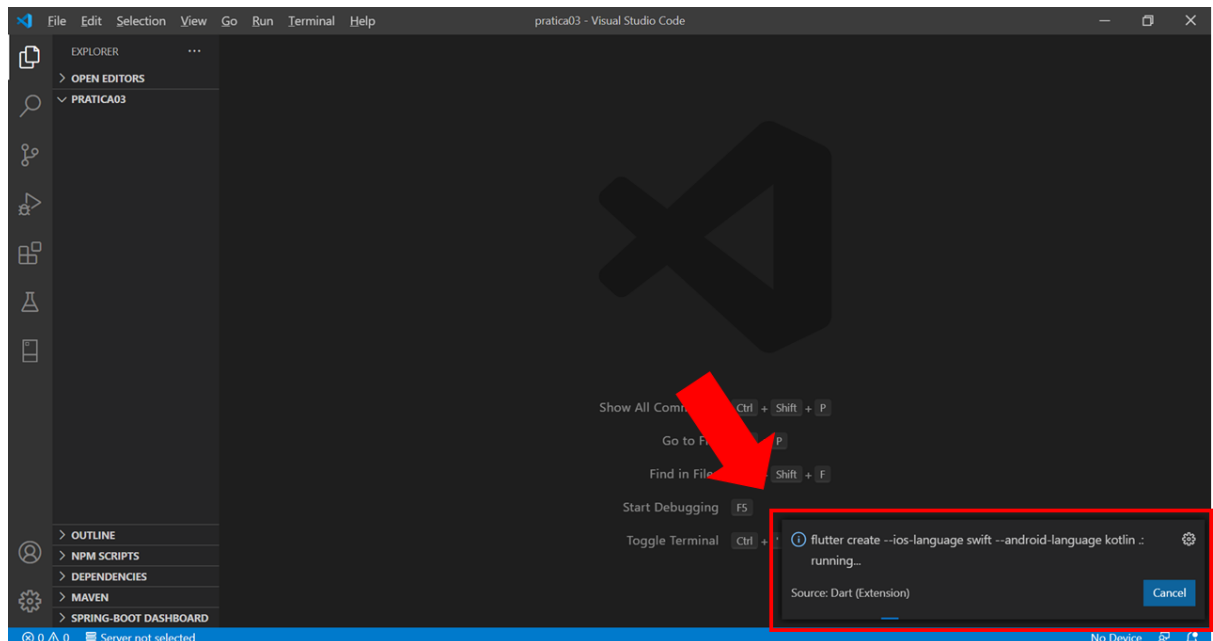
- 2) Na nova janela aberta, selecione um diretório para seu projeto. **OBS:** o caminho do diretório deve ter apenas caracteres ASCII. Além disso, o caminho do diretório não pode ter espaços em branco. Dessa forma, não use: acentos, símbolos, espaços, cedilha, etc. Exemplo: C:\projeto\flutter.



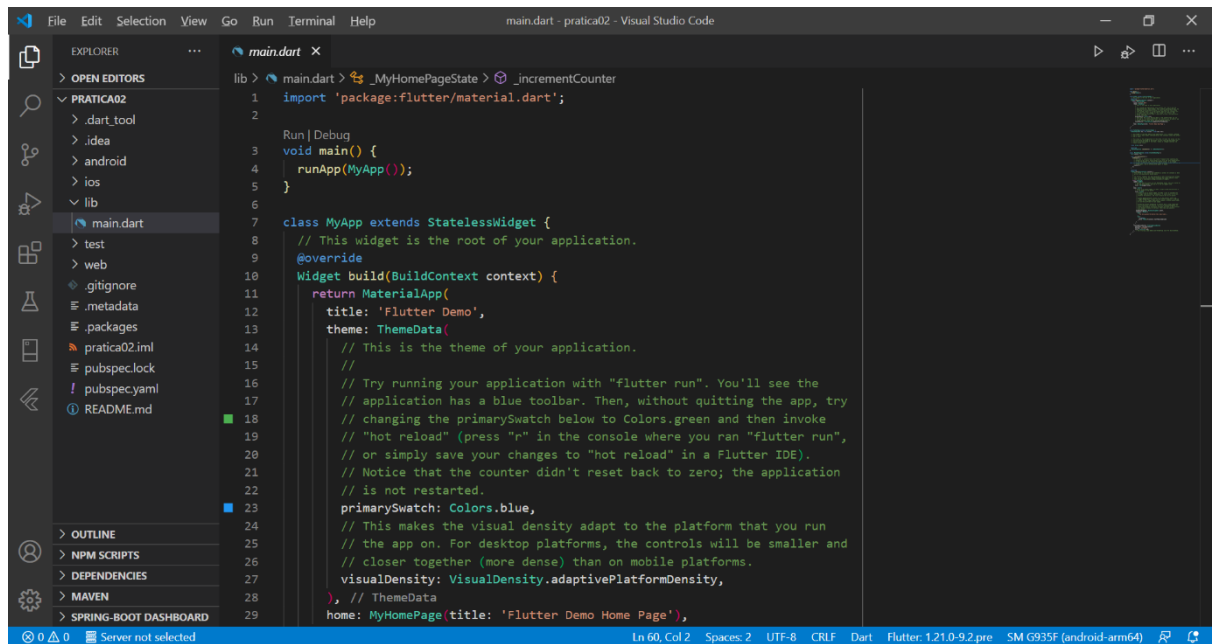
- 3) Na nova caixa de entrada, informe um nome para seu projeto. Exemplo: pratica03. Em seguida, pressione a tecla ENTER. **OBS:** o nome do projeto deve ter apenas caracteres ASCII (<https://pt.wikipedia.org/wiki/ASCII>). Além disso, o nome do projeto não pode ter espaços em branco. Dessa forma, não use: acentos, símbolos, espaços, cedilha, etc.



- 4) Aguarde a finalização do processo de criação de seu projeto.



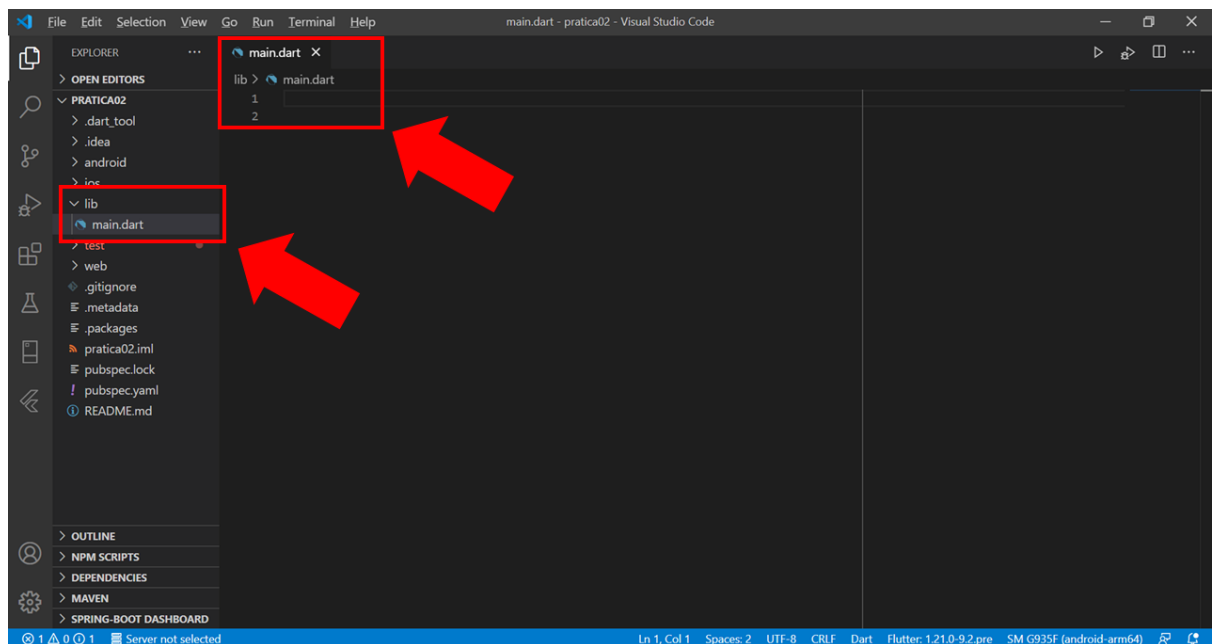
- 5) Após a criação de seu projeto, observe a estrutura de diretórios criada e o código inicial gerado como exemplo em lib/main.dart.



```

lib > main.dart > _MyHomePageState > _incrementCounter
1 import 'package:flutter/material.dart';
2
3 Run | Debug
4 void main() {
5   runApp(MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   // This widget is the root of your application.
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter Demo',
14      theme: ThemeData(
15        // This is the theme of your application.
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
26        // closer together (more dense) than on mobile platforms.
27        visualDensity: VisualDensity.adaptivePlatformDensity,
28      ), // ThemeData
29      home: MyHomePage(title: 'Flutter Demo Home Page'),
30    );
31  }
32 }
  
```

6) Apague o código inicial gerado como exemplo em lib\main.dart.



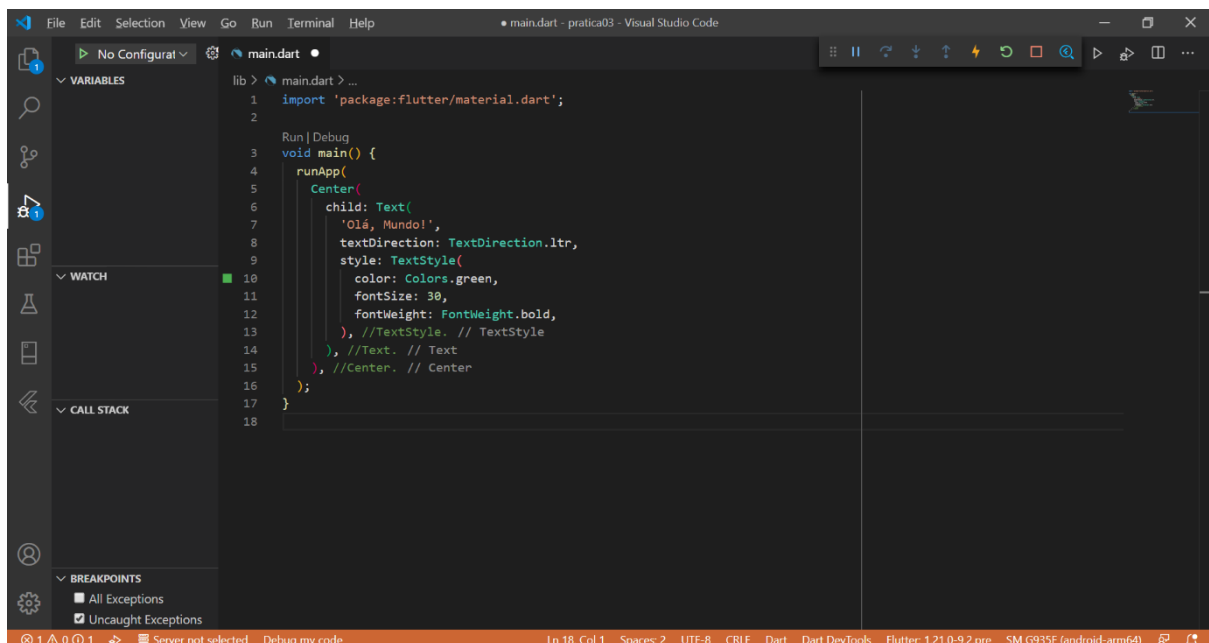
## Text

7) Um aplicativo Flutter mínimo possui uma função principal main() que simplesmente chama a função runApp() com um widget.

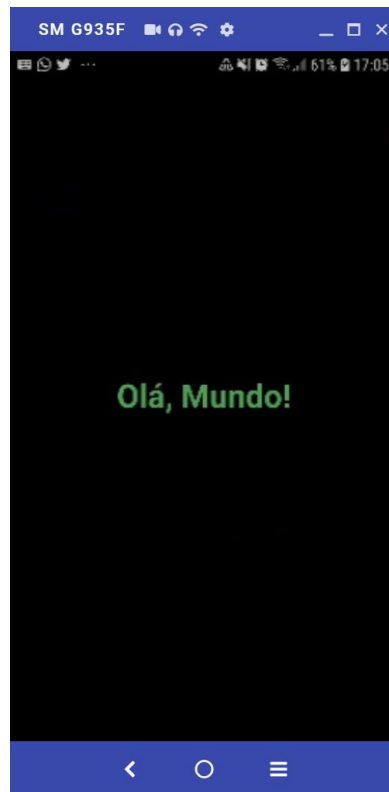
- 8) No exemplo abaixo, a função `runApp()` é chamada com um widget para texto. O aplicativo apresenta na tela um texto centralizado, na cor verde, negrito e com tamanho de 30 pixels.

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

void main() {
  runApp(
    Center(
      child: Text(
        'Olá, Mundo!',
        textDirection: TextDirection.ltr,
        style: TextStyle(
          color: Colors.green,
          fontSize: 30,
          fontWeight: FontWeight.bold,
        ), //TextStyle.
      ), //Text.
    ), //Center.
  );
}
```



- 9) Pressione a tecla F5, para iniciar a depuração, debug, do código. Aguarde a finalização do processamento do Gradle. Numa janela do Google Chrome ou numa tela do Vysor (<https://vysor.io/>), observe a saída gerada.

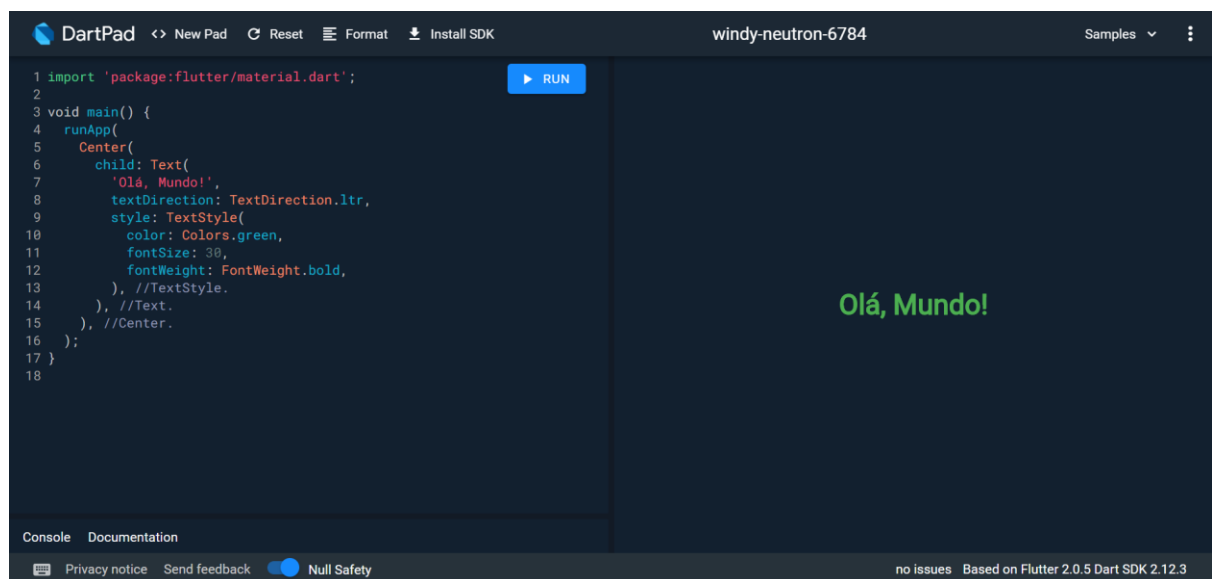


**OBS:**

Caso o framework Flutter dispare uma mensagem de erro no arquivo de teste, **test/main.dart**, apague todo o conteúdo do arquivo de teste.

10) Execute o código também no DartPad:

<https://dartpad.dev/>



11) Altere o código, incluindo um plano de fundo branco ao texto.

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

void main() {
  runApp(
    Center(
      child: Text(
        'Olá, Mundo!',
        textDirection: TextDirection.ltr,
        style: TextStyle(
          color: Colors.green,
          fontSize: 30,
          fontWeight: FontWeight.bold,
          backgroundColor: Colors.white,
        ), //TextStyle.
      ), //Text.
    ), //Center.
  );
}
```

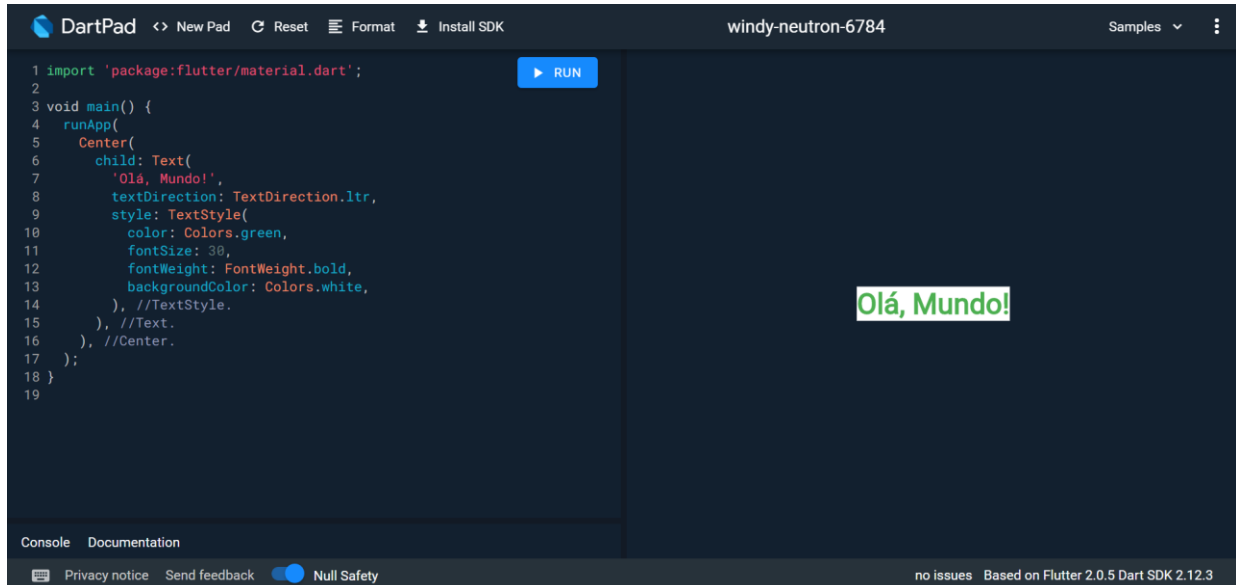
12) Clique em File > Save ou pressione CTRL + S para salvar a alteração. Caso necessário, clique no botão de Hot Reload ou no botão de Restart. Veja a alteração numa janela do Google Chrome ou na tela do Vysor:





13) Execute o código também no DartPad:

<https://dartpad.dev/>



14) Altere o código, para que o aplicativo apresente uma interpolação de string. Através da interpolação de string, você consegue acessar o valor de uma variável ou de uma expressão dentro de uma string. No exemplo, o aplicativo apresenta o texto interpolado com a variável nome.

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

void main() {
  String nome = "Ana";

  runApp(
    Center(
      child: Text(
        'Olá, $nome!',
        textDirection: TextDirection.ltr,
        style: TextStyle(
          color: Colors.green,
          fontSize: 30,
          fontWeight: FontWeight.bold,
          backgroundColor: Colors.white,
        ), //TextStyle.
      ), //Text.
    ), //Center.
  );
}
```



## RichText

O widget RichText exibe texto que possui trechos com estilos diferentes. O texto a ser exibido é descrito usando uma árvore de objetos TextSpan. Cada TextSpan possui um estilo associado que é aplicado naquela subárvore.

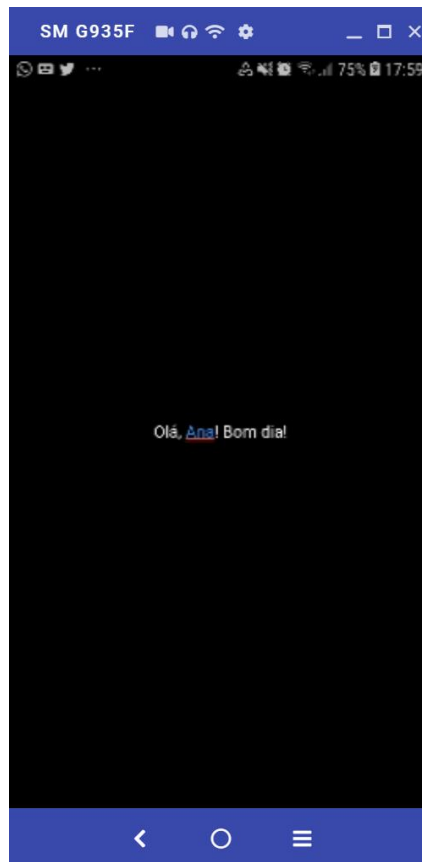
- 15) O exemplo abaixo altera o código, para que apenas parte do texto receba um estilo determinado. No exemplo, apenas o valor da variável nome recebe formatação. Observe que um widget TextSpan é incluído com um vetor filho. O vetor filho possui dois outros widgets TextSpan.

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

void main() {
  String nome = "Ana";

  runApp(
    Center(
      child: RichText(
        textDirection: TextDirection.ltr,
        text: TextSpan(
          text: "Olá, ",
```

```
children: [  
  TextSpan(  
    text: '$nome',  
    style: TextStyle(  
      color: Colors.blue,  
      decoration: TextDecoration.underline,  
      decorationColor: Colors.red,  
      decorationStyle: TextDecorationStyle.double,  
    ),  
  ),  
  TextSpan(  
    text: '! Bom dia!',  
  ),  
],  
,  
,  
,  
,  
);  
}
```

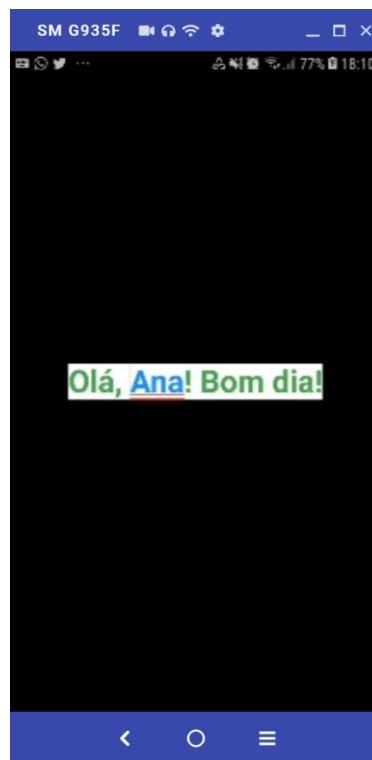


16) O próximo exemplo altera o código, para que os filhos herdem a formatação do primeiro TextSpan (widget pai).

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

void main() {
  String nome = "Ana";

  runApp(
    Center(
      child: RichText(
        textDirection: TextDirection.ltr,
        text: TextSpan(
          text: "Olá, ",
          style: TextStyle(
            color: Colors.green,
            fontSize: 30,
            fontWeight: FontWeight.bold,
            backgroundColor: Colors.white,
          ), //TextStyle.
        children: [
          TextSpan(
            text: '$nome',
            style: TextStyle(
              color: Colors.blue,
              decoration: TextDecoration.underline,
              decorationColor: Colors.red,
              decorationStyle: TextDecorationStyle.double,
            ),
          ),
          TextSpan(
            text: '! Bom dia!',
          ),
        ],
      ),
    ),
  );
}
```



17) O exemplo seguinte altera o código novamente, incluindo uma quebra de linha ao texto.

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

void main() {
  String nome = "Ana";

  runApp(
    Center(
      child: RichText(
        textDirection: TextDirection.ltr,
        text: TextSpan(
          text: "Olá, ",
          style: TextStyle(
            color: Colors.green,
            fontSize: 30,
            fontWeight: FontWeight.bold,
            backgroundColor: Colors.white,
          ), //TextStyle.
        children: <TextSpan>[
          TextSpan(
            text: '$nome',
            style: TextStyle(
              color: Colors.blue,
```

```
        decoration: TextDecoration.underline,  
        decorationColor: Colors.red,  
        decorationStyle: TextDecorationStyle.double,  
      ),  
    ),  
    TextSpan(  
      text: '! \nBom dia!',  
    ),  
  ],  
),  
,  
,  
,  
);  
}
```



## Exercício

- 1) Altere o exemplo desta prática, para que ele apresente a tela abaixo. O nome da pessoa e o dia da semana devem ser lidos de uma variável através de interpolação.



Dica:

```
runApp(  
  Center(  
    child: RichText(  
      textDirection: TextDirection.ltr,  
      textAlign: TextAlign.center,  
      text: TextSpan(...
```