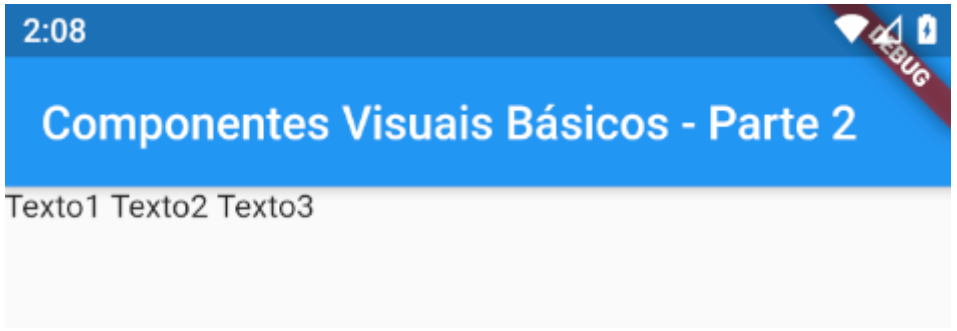
Componentes Visuais Básicos
Parte 2

Instituto Federal do Espirito Santo – Campus Colatina
Desenvolvimento para sistemas móveis - Flutter
Professor: Giovany Frossard Teixeira

Row

- É um widget que exibe seus filhos em uma matriz horizontal.
- Nos próximos exemplos alteraremos apenas o método _body().
- O código ao lado se apresenta na imagem abaixo (como só há texto no inicio da tela, foi feito recorte apenas desse pedaço da tela).




```dart
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Componentes Visuais Básicos - Parte 2',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ), // ThemeData
      home: MyHomePage(),
    ); // MaterialApp
  }
}
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Componentes Visuais Básicos - Parte 2"),
      ), // AppBar
      body: _body(),
    ); // Scaffold
  }

  _body() {
    return Row(
      children: <Widget>[
        Text("Texto1 "),
        Text("Texto2 "),
        Text("Texto3 "),
      ], // <Widget>[]
    ); // Row
  }
}
```

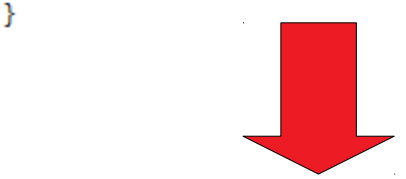
mainAxisAlignment

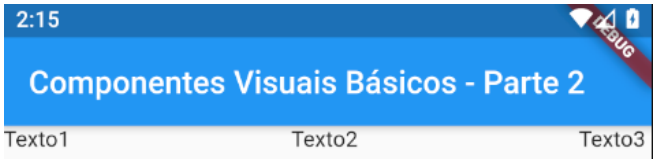
- Para o widget Row define o alinhamento horizontal.

```
_body() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.,
    children: <Widget>[
      Text("Texto1 "),
      Text("Texto2 "),
      Text("Texto3 "),
    ], // <Widget>[]
  ); // Row
}
```

```
center                    MainAxisAlignment
end                       MainAxisAlignment
spaceAround               MainAxisAlignment
spaceBetween              MainAxisAlignment
spaceEvenly               MainAxisAlignment
start                     MainAxisAlignment
values                    List<MainAxisAlignment>
nn                        expr.nn -> if (expr != null) {}
notnull                   expr.notnull -> if (expr != null) {}
null                      expr.null -> if (expr == null) {}
par                       expr.par -> (expr)
Press Enter to insert, Guia to replace
```

```
_body() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: <Widget>[
      Text("Texto1 "),
      Text("Texto2 "),
      Text("Texto3 "),
    ], // <Widget>[]
  ); // Row
}
```

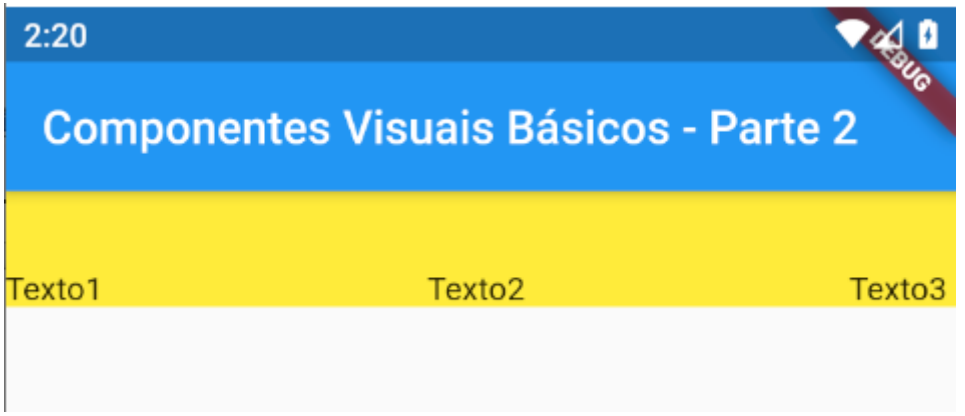




crossAxisAlignment

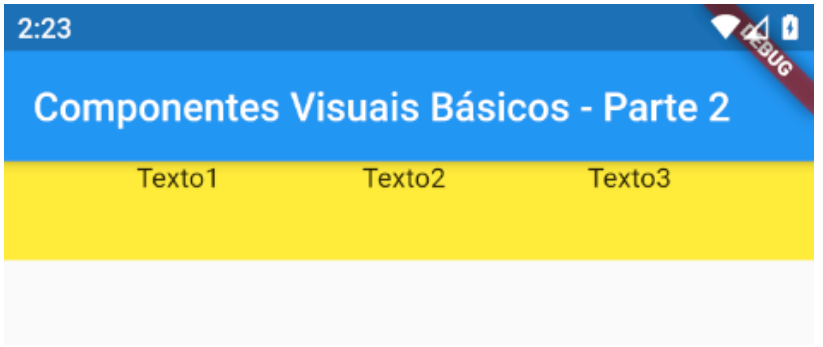
- Para o widget Row define o alinhamento vertical.

```
_body() {
  return Container(
    color: Colors.yellow,
    height: 50,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      crossAxisAlignment: CrossAxisAlignment.end,
      children: <Widget>[
        Text("Texto1 "),
        Text("Texto2 "),
        Text("Texto3 "),
      ], // <Widget>[]
    ), // Row
  ); // Container
}
```

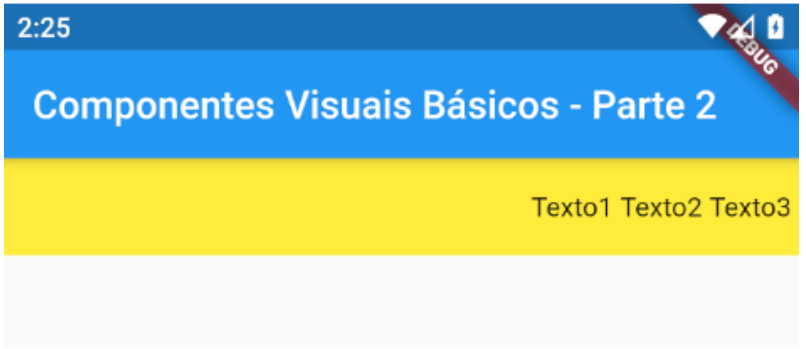


Outros exemplos

```
_body() {
  return Container(
    color: Colors.yellow,
    height: 50,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: <Widget>[
        Text("Texto1 "),
        Text("Texto2 "),
        Text("Texto3 "),
      ], // <Widget>[]
    ), // Row
  ); // Container
}
```



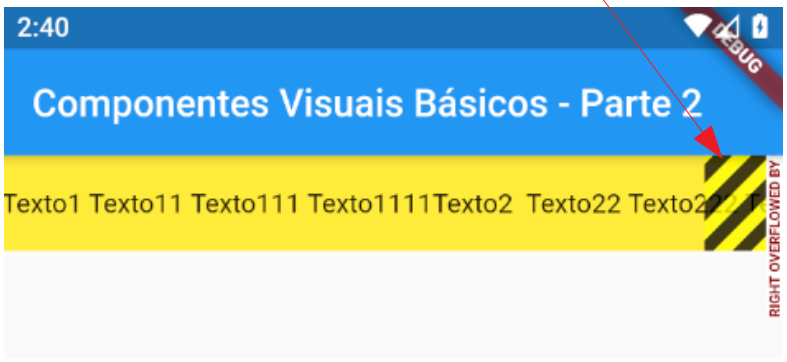

```
_body() {
  return Container(
    color: Colors.yellow,
    height: 50,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: <Widget>[
        Text("Texto1 "),
        Text("Texto2 "),
        Text("Texto3 "),
      ], // <Widget>[]
    ), // Row
  ); // Container
}
```



Excesso de widgets na Row()

- Caso sejam colocados mais widgets do que a Row é capaz de suportar o Flutter mostrará um **erro**.

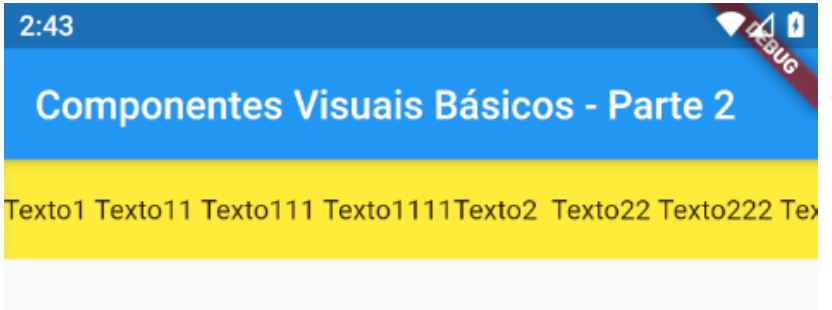
```
_body() {
  return Container(
    color: Colors.yellow,
    height: 50,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: <Widget>[
      Text("Texto1 Texto11 Texto111 Texto1111"),
      Text("Texto2  Texto22 Texto222 Texto2222"),
      Text("Texto3 "),
      ], // <Widget>[]
    ), // Row
  ); // Container
}
```

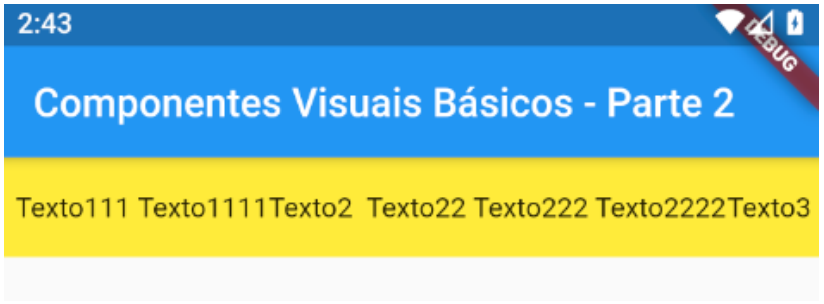


SingleChildScrollView

- Uma forma de resolver esse problema é usando um **SingleChildScrollView**.

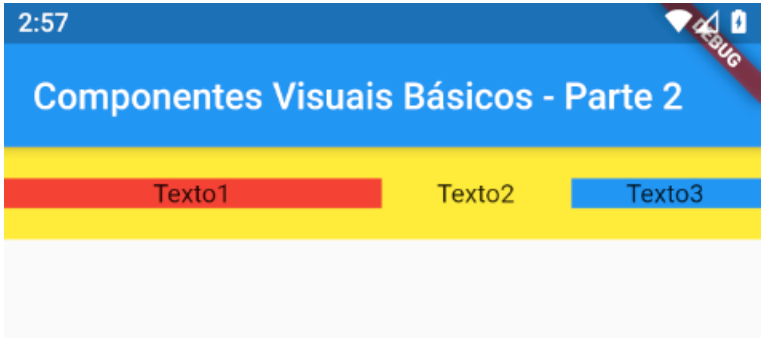
```
_body() {
  return Container(
    color: Colors.yellow,
    height: 50,
    child: SingleChildScrollView(
      scrollDirection: Axis.horizontal,
      child: Row(
        mainAxisAlignment: MainAxisAlignment.end,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: <Widget>[
          Text("Texto1 Texto11 Texto111 Texto1111"),
          Text("Texto2  Texto22 Texto222 Texto2222"),
          Text("Texto3 "),
        ], // <Widget>[]
      ), // Row
    ), // SingleChildScrollView
  ); // Container
}
```





Expanded

- Para dividir o espaço disponível na Row é possível utilizar o widget **Expanded**. Com esse widget os espaços são divididos de forma proporcional aos pesos.

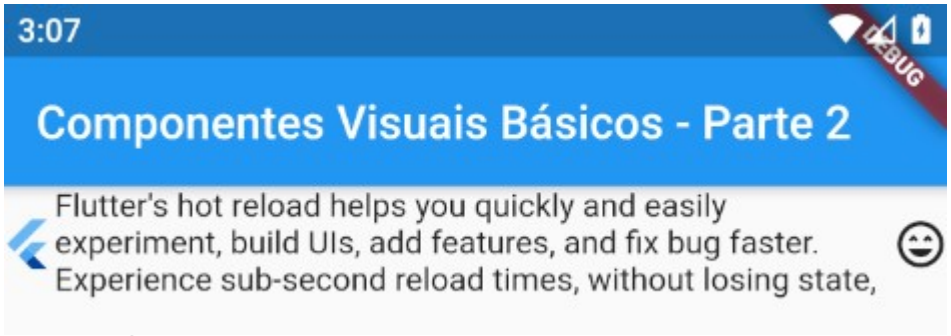


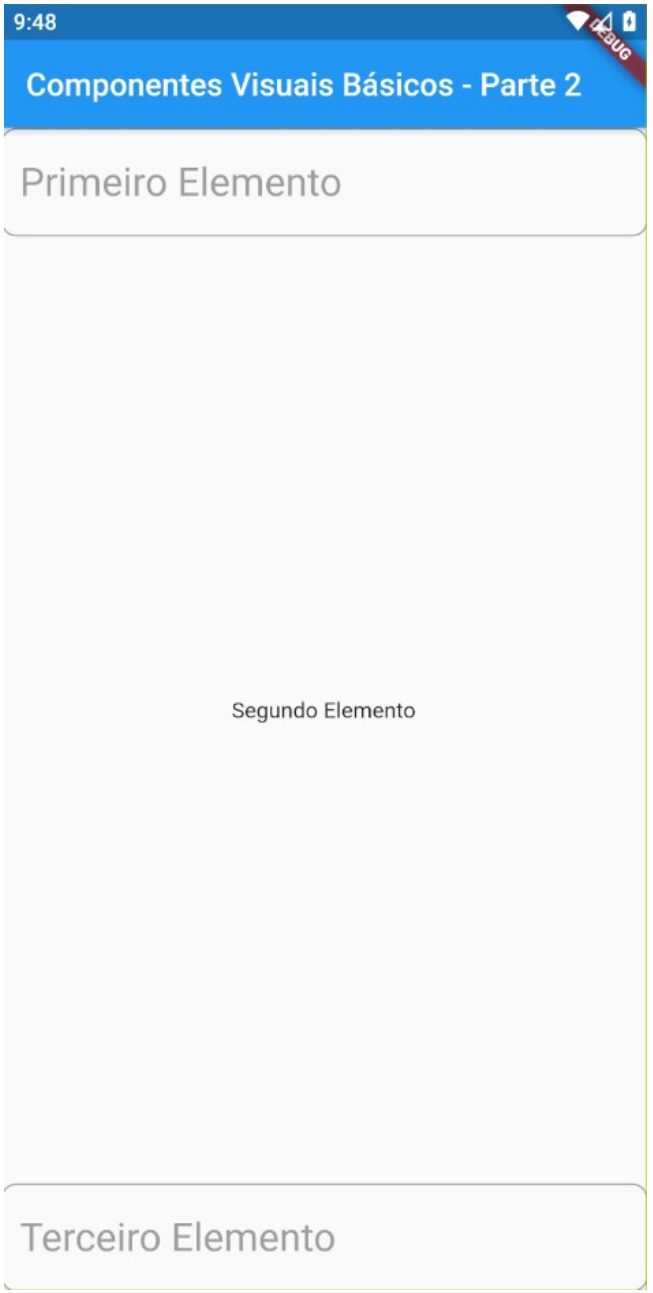

```
_body() {
  return Container(
    color: Colors.yellow,
    height: 50,
    child: Row(
      children: <Widget>[
      Expanded(
          flex: 2,
          child: Container(
            color: Colors.red,
            child: Text(
              "Texto1",
              textAlign: TextAlign.center,
            ), // Text
          ), // Container
      ), // Expanded
      Expanded(child:Text("Texto2", textAlign: TextAlign.center,)),
      Expanded(child:Container(
          color: Colors.blue,
          child: Text(
            "Texto3 ",
            textAlign: TextAlign.center,), // Text
        ) // Container
      ), // Expanded
      ], // <Widget>[]
    ), // Row
  ); // Container
}
```

Falta de espaço – usando Expanded

- Esse exemplo foi obtido na própria documentação do Flutter - https://api.flutter.dev/flutter/widgets/Row-class.html . É altamente recomendável estudar a documentação dos widgets para melhor utilizá-los.

```
_body() {
  return Container(
    height: 50,
    child: Row(
      children: <Widget>[
        const FlutterLogo(),
        const Expanded(
          child: Text(
              "Flutter's hot reload helps you quickly and easily experiment, "
                  "build UIs, add features, and fix bug faster. "
                  "Experience sub-second reload times, without losing state, "
                  "on emulators, simulators, and hardware for iOS and Android."),
        ), // Expanded
        const Icon(Icons.sentiment_very_satisfied),
      ], // <Widget>[]
    ), // Row
  ); // Container
}
```





Column

- É um widget que exibe seus filhos em uma matriz vertical.
- mainAxisAlignment, para Column, define o alinhamento vertical. Já o crossAxisAlignment define o alinhamento horizontal (notar que é ao contrário do widget Row).
- No exemplo temos a sensação que o texto ocupou todo o espaço disponível (ver próximo slide).

```
_body() {
  return Container(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: <Widget>[
        CampoEdicao("Primeiro Elemento"),
        Text("Segundo Elemento"),
        CampoEdicao("Terceiro Elemento"),
        ], // <Widget>[]
      ), // Column
    ); // Container
}
```

Column

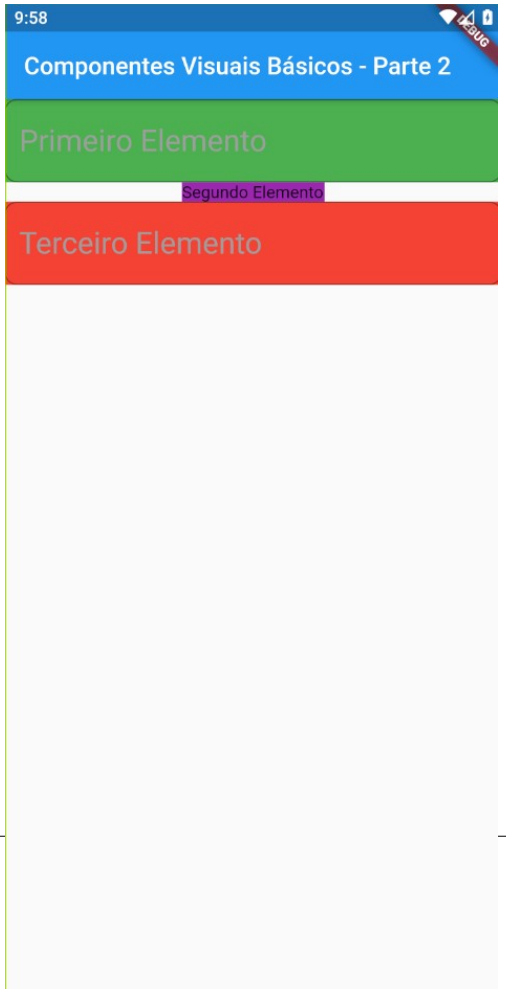
- No exemplo vemos que o **Text** ocupa apenas o espaço necessário para a sua exibição. Como foi usado **spaceBetween** o Column distribuiu seus elementos na vertical colocando o primeiro e o último nos extremos e o segundo elemento no meio.

```
_body() {
  return Container(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: <Widget>[
        Container(
            color: Colors.green,
            child: CampoEdicao("Primeiro Elemento")), // Container
        Container(color: Colors.purple, child: Text("Segundo Elemento")),
        Container(
            color: Colors.red,
            child: CampoEdicao("Terceiro Elemento")), // Container
      ], // <Widget>[]
    ), // Column
  ); // Container
}
```

Column

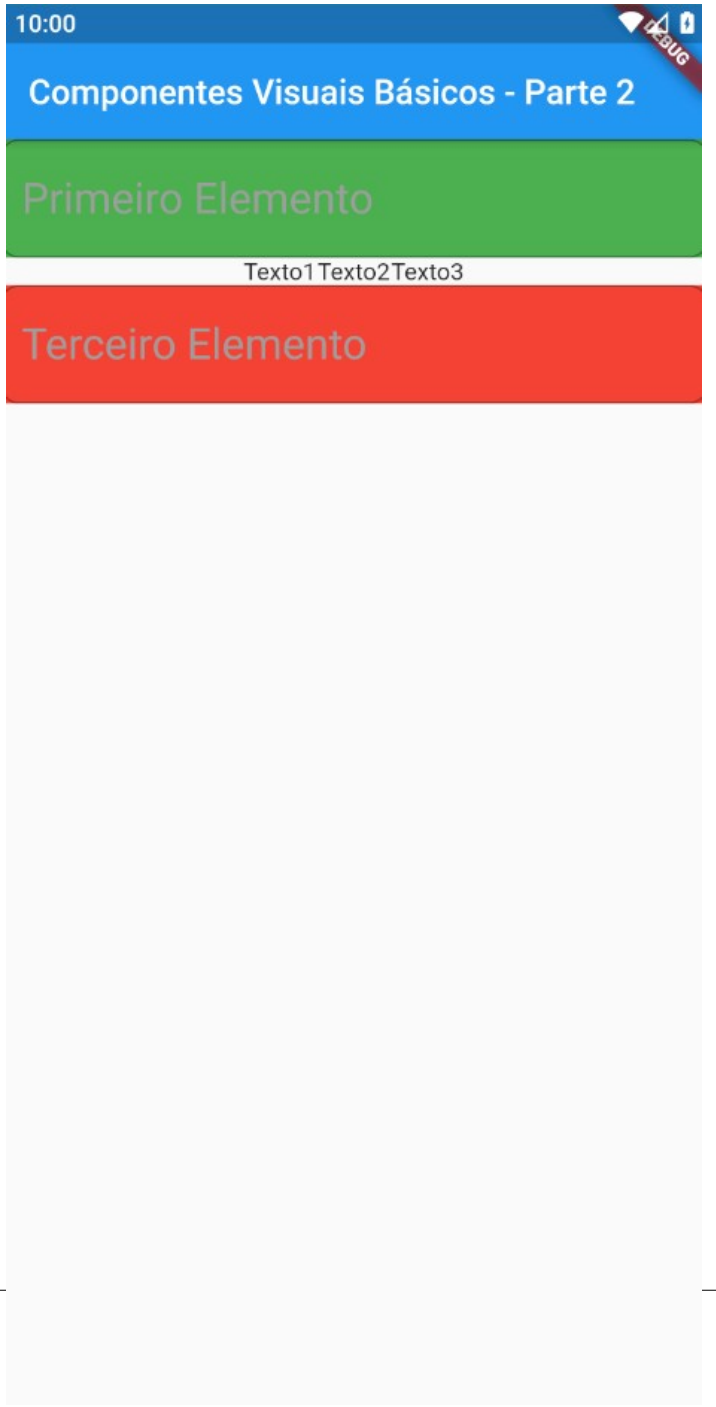
- Trocando de **spaceBetween** para **start** é possível ver que os widgets ocupam apenas o espaço necessário para a sua exibição.

```
_body() {
  return Container(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.start,
      children: <Widget>[
        Container(
            color: Colors.green,
            child: CampoEdicao("Primeiro Elemento")), // Container
        Container(color: Colors.purple, child: Text("Segundo Elemento")),
        Container(
            color: Colors.red,
            child: CampoEdicao("Terceiro Elemento")), // Container
      ], // <Widget>[]
    ), // Column
  ); // Container
}
```



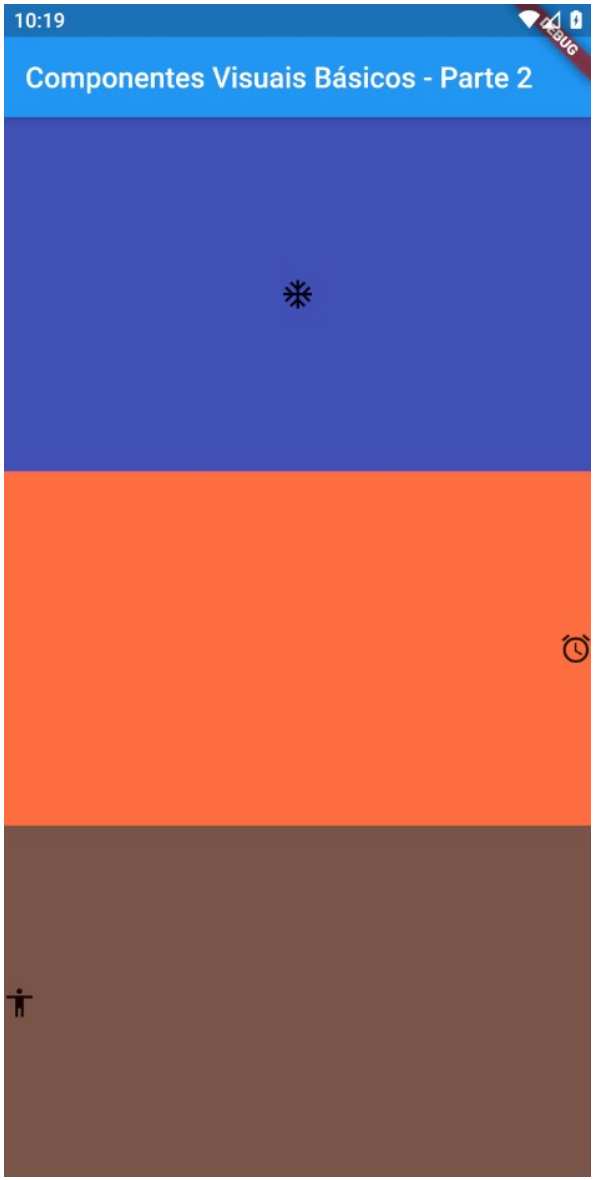
Column – com uma Row

```
_body() {
  return Container(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.start,
      children: <Widget>[
      Container(
          color: Colors.green,
          child: CampoEdicao("Primeiro Elemento")), // Container
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
        Text("Texto1"),
        Text("Texto2"),
        Text("Texto3"),
        ], // <Widget>[]
      ), // Row
      Container(
          color: Colors.red,
          child: CampoEdicao("Terceiro Elemento")), // Container
      ], // <Widget>[]
    ), // Column
  ); // Container
}
```



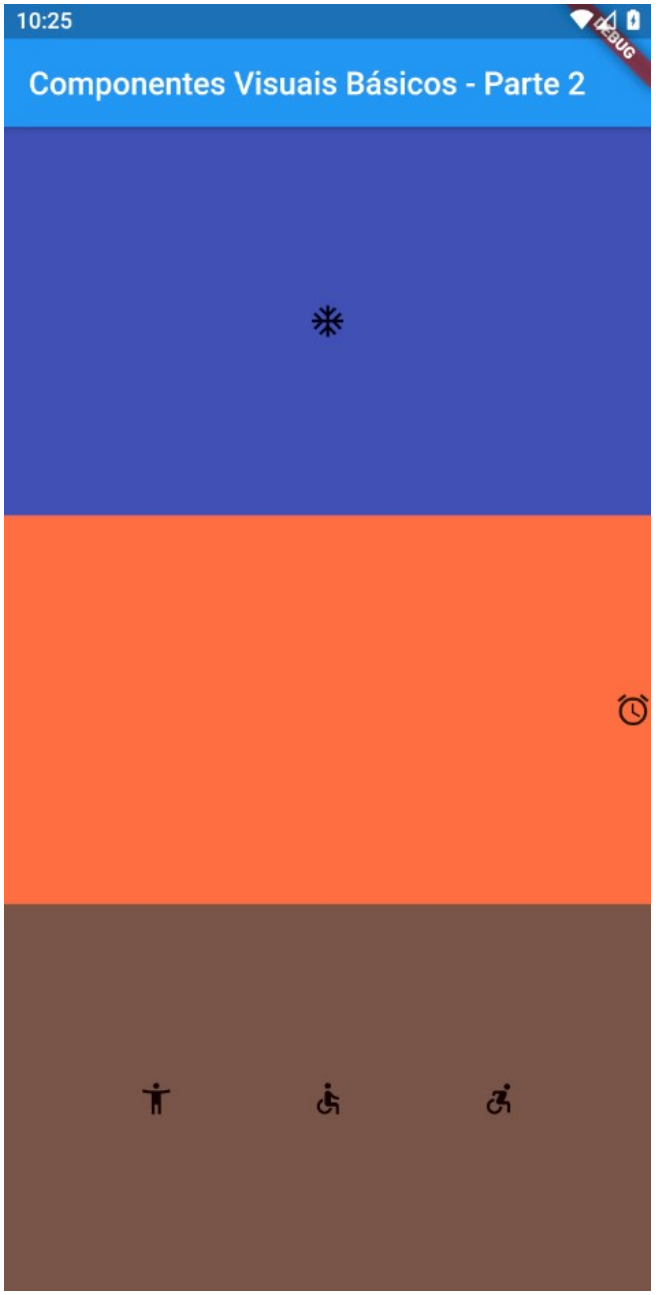
Column - Expanded

```
50  _body() {
51    return Container(
52      child: Column(
53        children: <Widget>[
54          Expanded(
55            child: Container(
56              color: Colors.indigo,
57              alignment: Alignment.center,
58              child: Icon(Icons.ac_unit))), // Container, Expanded
59          Expanded(
60            child: Container(
61              color: Colors.deepOrangeAccent,
62              alignment: Alignment.centerRight,
63              child: Icon(Icons.access_alarm))), // Container, Expanded
64          Expanded(
65            child: Container(
66              color: Colors.brown,
67              alignment: Alignment.centerLeft,
68              child: Icon(Icons.accessibility))), // Container, Expanded
69        ], // <Widget>[]
70      ), // Column
71    ); // Container
72  }
```



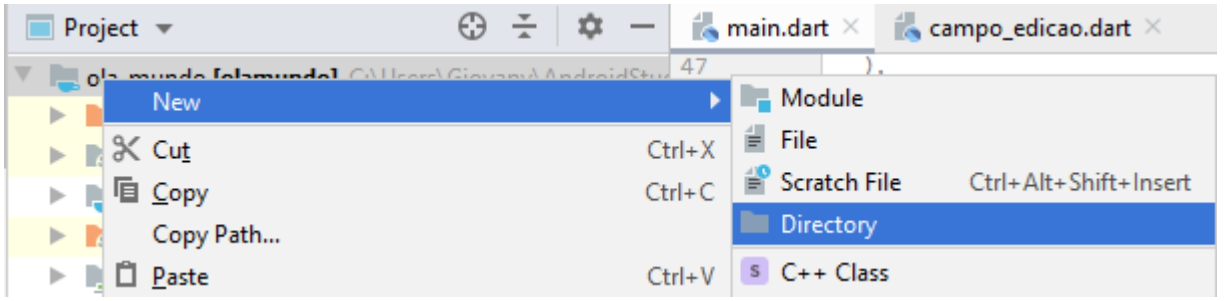
Column – Row e Expanded

```dart
_body() {
  return Container(
    child: Column(
      children: <Widget>[
        Expanded(
          child: Container(
            color: Colors.indigo,
            alignment: Alignment.center,
            child: Icon(Icons.ac_unit))), // Container, Expanded
        Expanded(
          child: Container(
            color: Colors.deepOrangeAccent,
            alignment: Alignment.centerRight,
            child: Icon(Icons.access_alarm))), // Container, Expanded
        Expanded(
          child: Container(
            color: Colors.brown,
            alignment: Alignment.centerLeft,
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: <Widget>[
                Icon(Icons.accessibility_new),
                Icon(Icons.accessible),
                Icon(Icons.accessible_forward),
                ], // <Widget>[]
              ), // Row
            ), // Container
          ), // Expanded
        ], // <Widget>[]
      ), // Column
    ); // Container
}
```



Stack

- Possibilita colocar widgets um sobre o outro.
- Esse widget é útil para sobrepor vários filhos de uma maneira simples, por exemplo, um texto sobre uma imagem.
- Para trabalhar com imagens uma das formas é utilizando a pasta **assets** (essa é uma pasta padrão para armazenamento e obtenção de imagens num aplicativo).
- Para utilizar a pasta assets é necessário primeiramente criá-la no projeto:



Pasta assets

- O próximo passo é configurar o acesso a essa pasta no arquivo **pubspec.yaml** (esse é um dos arquivos mais importantes no Flutter, nele fazemos a inclusão de plugins e diversas outras configurações).
- Nesse arquivo é necessário modificar um trecho que está comentado:

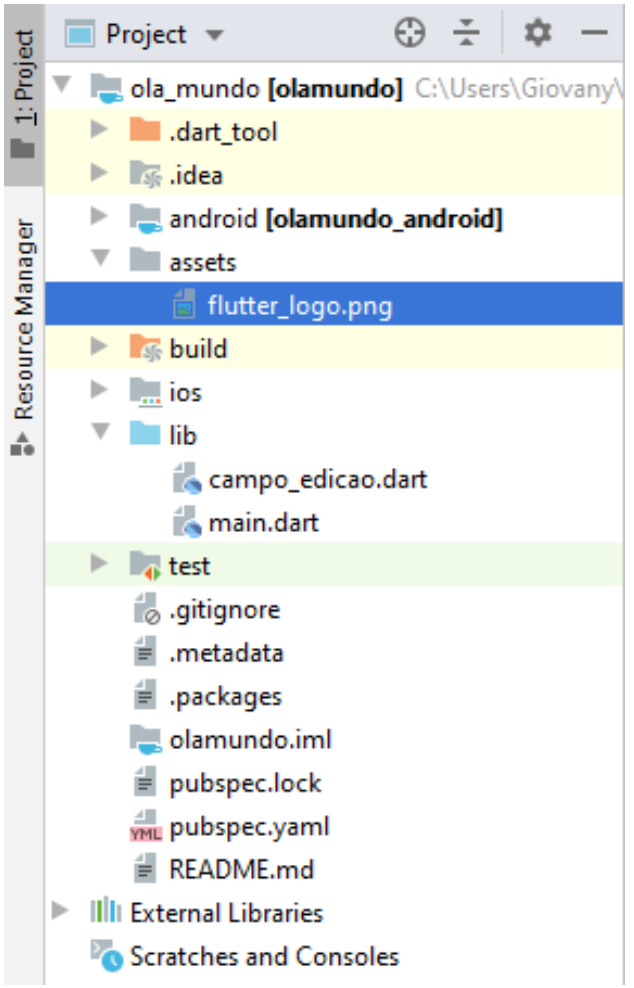
```
# To add assets to your application, add an assets section, like this:
# assets:
#   - images/a_dot_burr.jpeg
#   - images/a_dot_ham.jpeg
```

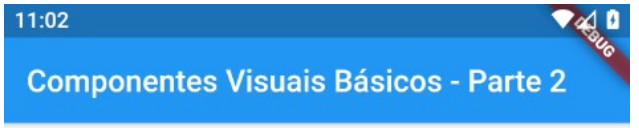
```
# To add assets to your application, add an assets section, like this:
assets:
  - assets/
```


Colocando uma imagem em assets

- Agora é necessário colocar a imagem na pasta **assets** (podemos fazer o mesmo procedimento com quantas imagens desejarmos).
- Para acessar essa imagem utilizamos o seguinte código:

Image.asset(caminho_imagem)





Voltando a Stack

- Primeiramente utilizaremos a opção **StackFit.expand** para fazer a pilha ocupar todo o espaço disponível.
- O segundo passo será fazer a imagem expandir (mesmo que a deforme) utilizando a opção **BoxFit.fill**.

```
_body() {
  return Stack(
    fit: StackFit.expand,
    children: <Widget>[
      Image.asset("assets/flutter_logo.png", fit: BoxFit.fill,),
    ], // <Widget>[]
  ); // Stack
}
```


Empilhando ...

- Agora colocaremos um texto sobre a imagem.
- Uma questão importante é o primeiro Container (que possui **alignment: Alignment.bottomCenter**). Esse Container não possui cor (é transparente) e por isso permite a exibição da imagem que está sob ele. Se não tivermos esse Container, apesar de limitarmos o Container interno a 40 de altura, a sua cor irá se propagar e a imagem não será exibida (o **StackFit.expand** irá expandi-lo).
- No slide seguinte serão apresentadas telas do emulador com o Container externo (primeira imagem) e sem o Container externo (segunda imagem) para ilustrar a explicação do parágrafo anterior.

```
_body() {
  return Stack(
    fit: StackFit.expand,
    children: <Widget>[
      Image.asset("assets/flutter_logo.png", fit: BoxFit.fill,),
      Container(
        alignment: Alignment.bottomCenter,
        child: Container(
          alignment: Alignment.center,
          height: 40,
          color: Colors.green,
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text(
                "Flutter é muito legal",
                style: TextStyle(
                  fontSize: 20,
                  color: Colors.white,
                ), // TextStyle
              ), // Text
            ], // <Widget>[]
          ), // Row
        ), // Container
      ), // Container
    ], // <Widget>[]
  ); // Stack
}
```



11:24
DEBUG
Componentes Visuais Básicos - Parte 2
Flutter é muito legal

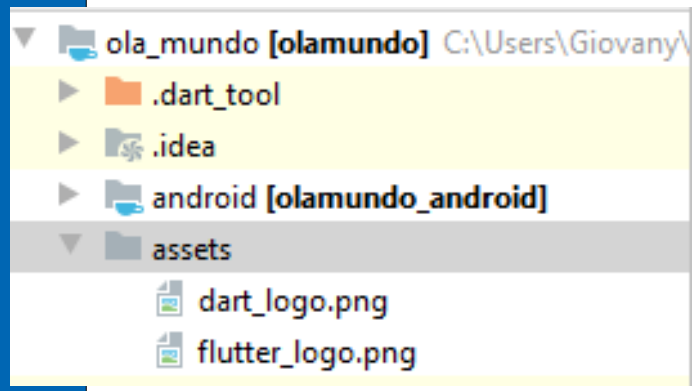

11:24
DEBUG
Componentes Visuais Básicos - Parte 2
Flutter é muito legal

PageView

- É um widget que gera páginas roláveis. O comportamento desse widget é similar a um carrossel e muito utilizado para exibição de imagens.
- Utilizaremos 2 imagens nesse carrossel.
- Utilizaremos a opção **BoxFit.cover** nas imagens para não deformá-las (essa opção expande um recorte da imagem mantendo as proporções).

PageView

```
_body() {
  return PageView(
    children: <Widget>[
      Image.asset("assets/flutter_logo.png", fit: BoxFit.cover),
      Image.asset("assets/dart_logo.png", fit: BoxFit.cover),
    ], // <Widget>[]
  ); // PageView
}
```



Dúvidas?