



Componentes de Interface

Parte 3

Profa. Ivre Marjorie
(ivre@pucminas.br)

O que vamos aprender nessa aula?

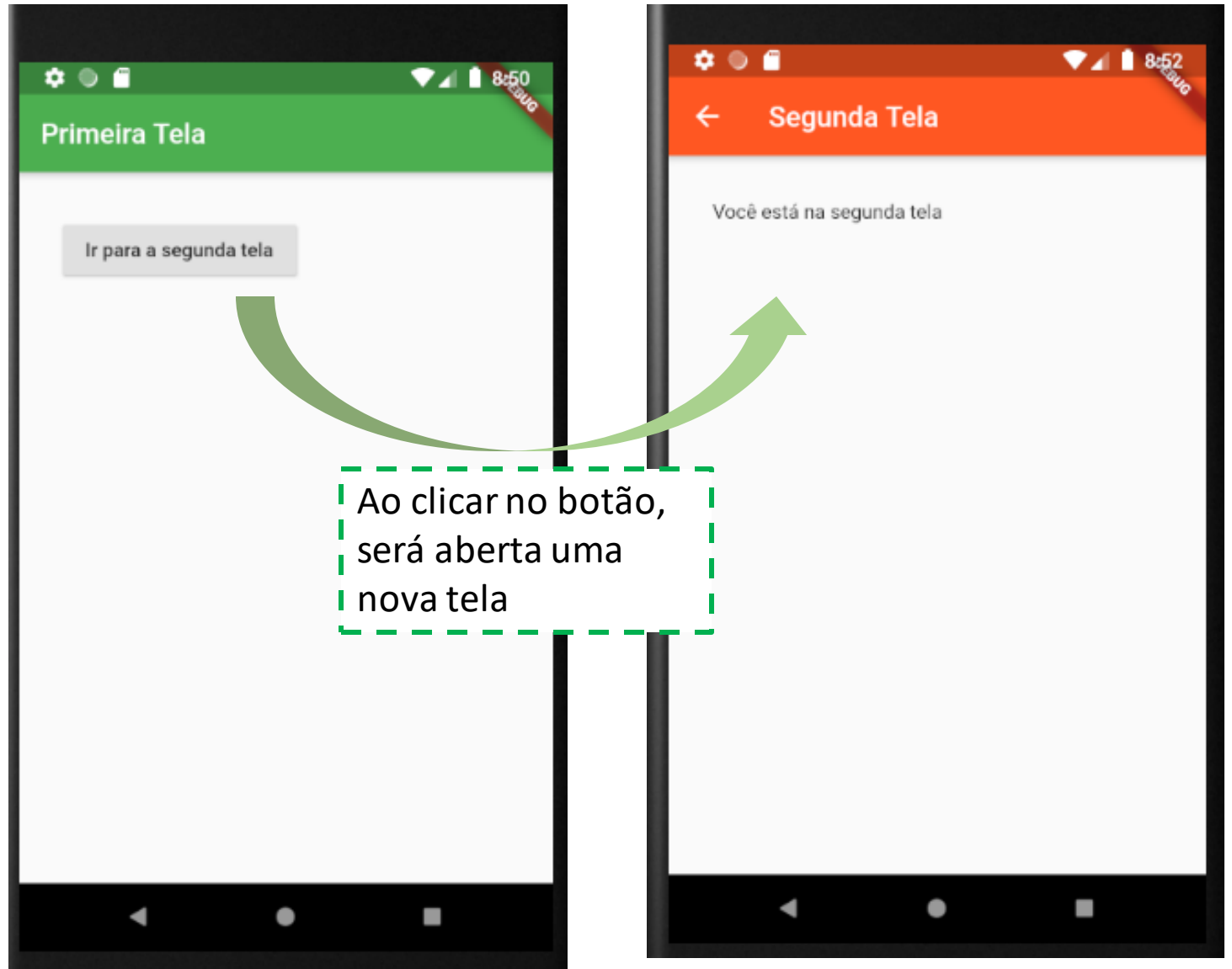
- Navigator
- MainAxisAlignment
- CrossAxisAlignment

Navigator

- Classe responsável por fazer a navegação entre uma tela e outra
- O método `push()` abre uma nova tela sem fechar a anterior (contexto)
- O método `pop()` abre uma nova tela fechando a anterior (contexto)

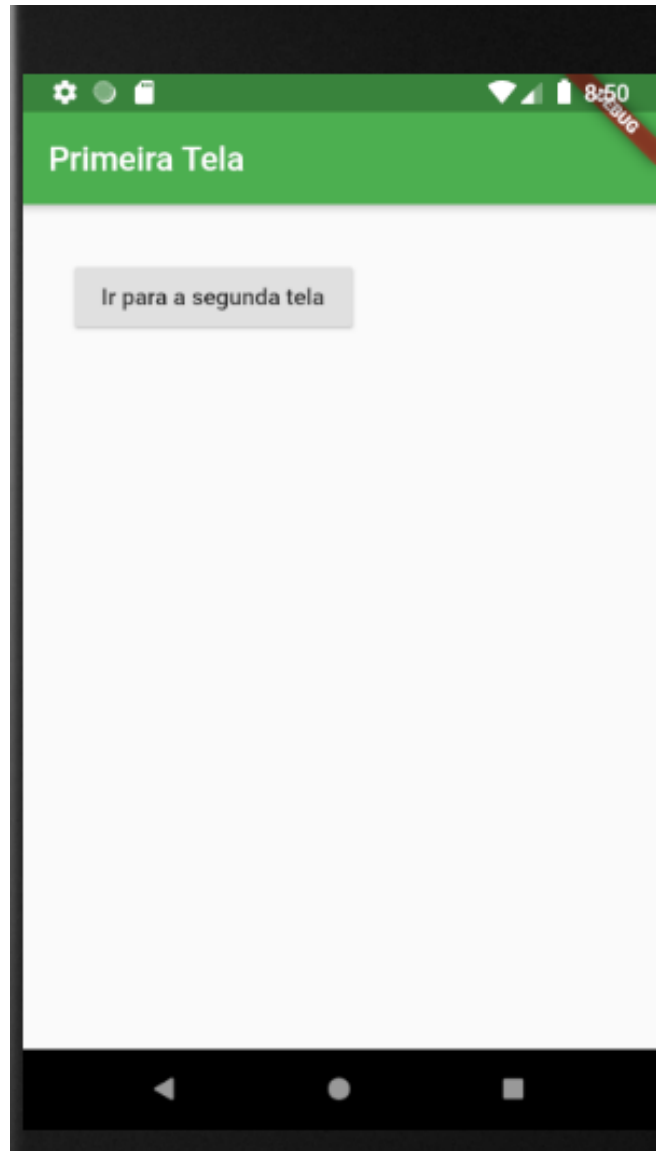
Navigator

- No exemplo, teremos duas telas (a primeira tela e a segunda), na primeira teremos um botão, que ao clicar abre a segunda tela



Navigator

- A segunda tela ficará sobreposta a primeira, portanto, a Primeira Tela não será fechada, mas ao voltar usando a setinha a Segunda Tela é fechada



Navigator

```
— RaisedButton(  
  — child: Text("Ir para a segunda tela"),  
    onPressed: () {  
      Navigator.push(context, route);  
    }  
  ), // RaisedButton
```

Context (contexto) é a tela na qual estamos. Como já temos no método build o context, basta colocar context mesmo aqui no push()

Definir a rota, existe mais de uma forma para fazer. Vamos usar o método mais fácil que é usando o MaterialPageRoute

Navigator

```
RaisedButton(  
  child: Text("Ir para a segunda tela"),  
  onPressed: () {  
    Navigator.push(  
      context,  
      MaterialPageRoute(  
        builder: (context) => SegundaTela()  
      ), // MaterialPageRoute  
    );  
  },  
), // RaisedButton
```

Usamos o `MaterialPageRoute`, que é uma classe que já tem tudo que precisamos para definir a rota

Indicamos qual é Tela para a qual devemos ir, de forma simples, indicando o nome da classe

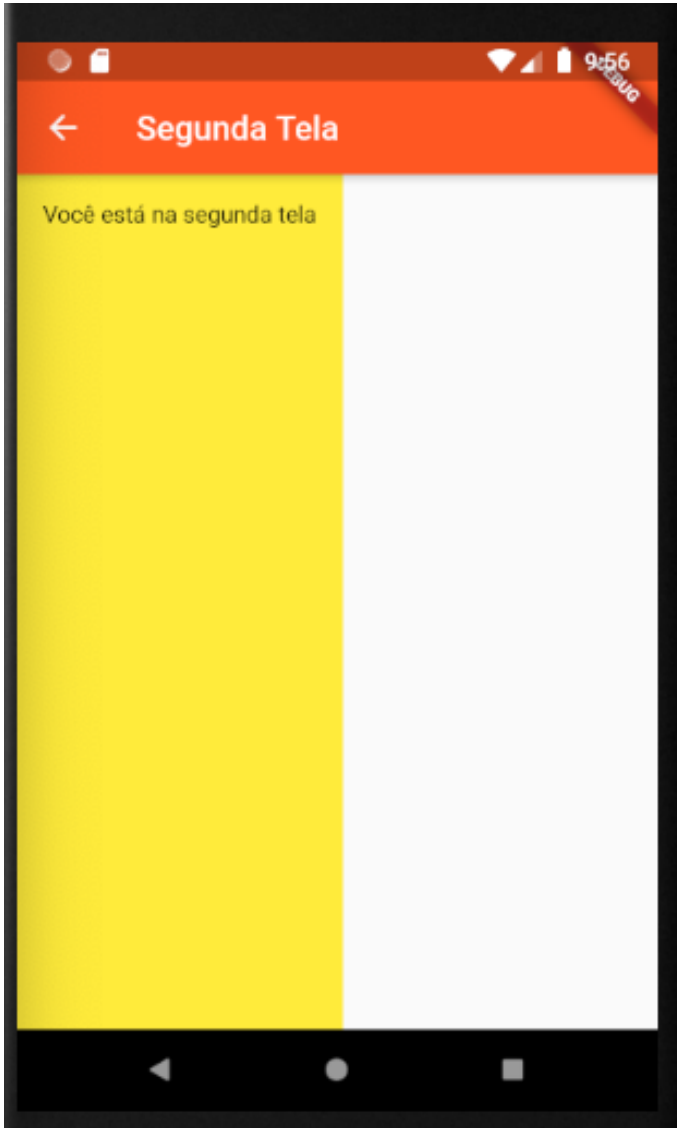
Parâmetro `builder`, com uma função anônima que precisa de um parâmetro que é o contexto

MainAxisAlignment

- Permite a movimentação de botões, textos, etc no eixo principal de uma linha ou de uma coluna
- É possível usar `MainAxisAlignment.start` (início), `MainAxisAlignment.center` (centro) e `MainAxisAlignment.end` (fim) da linha ou coluna, sendo o padrão o `start`

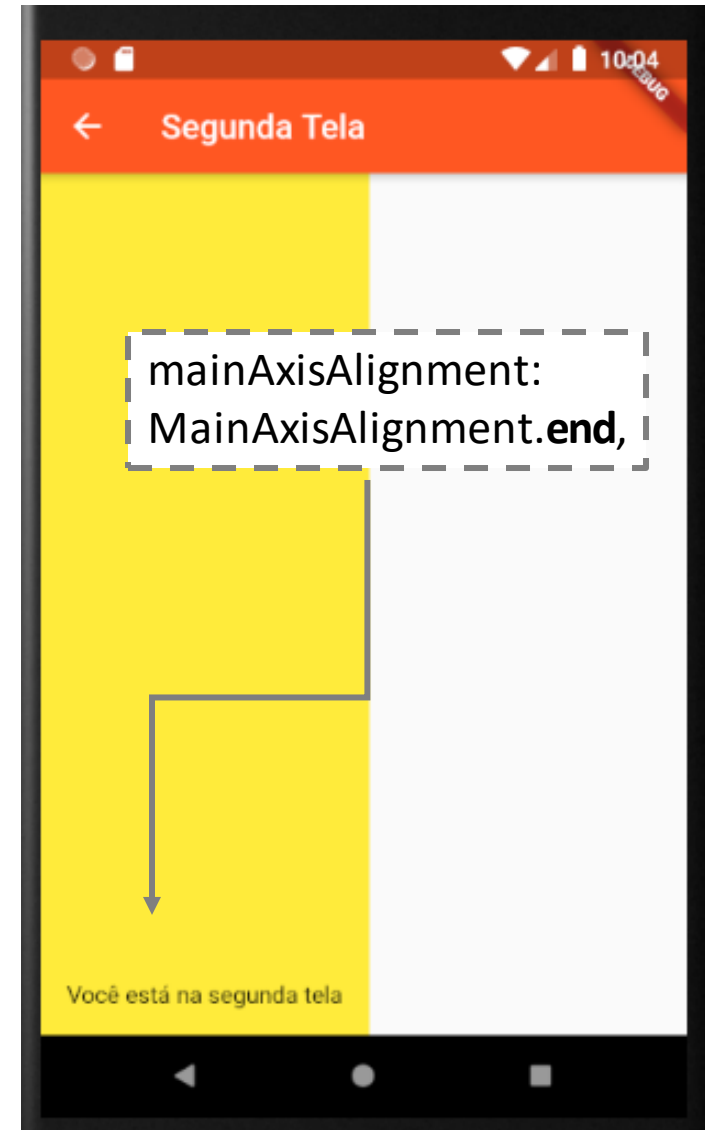
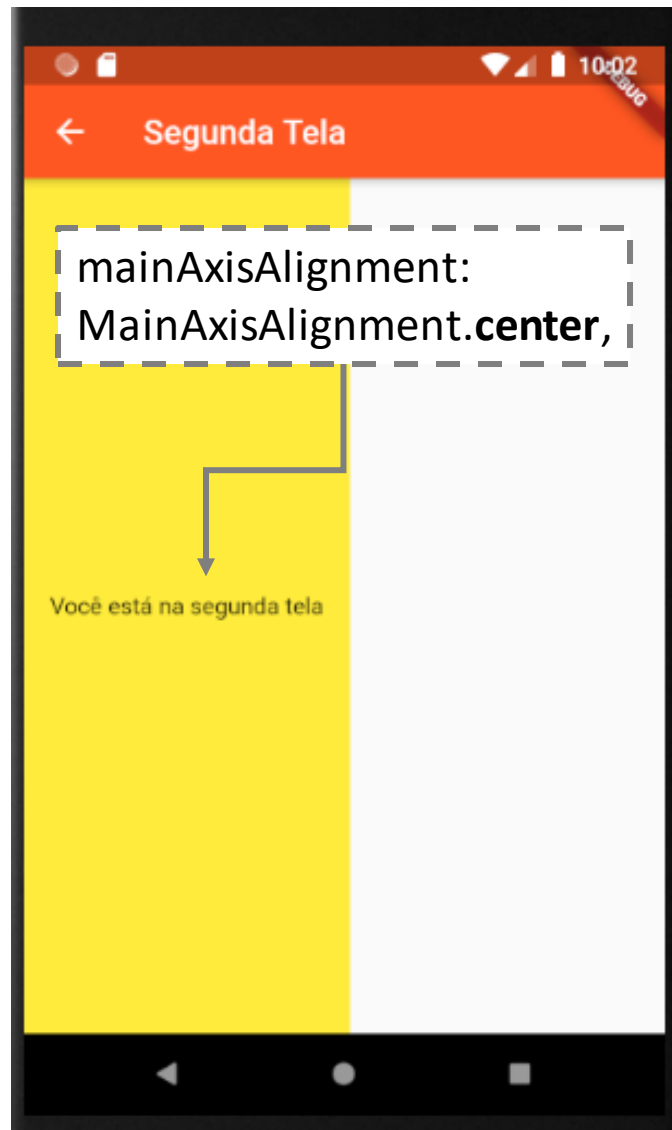
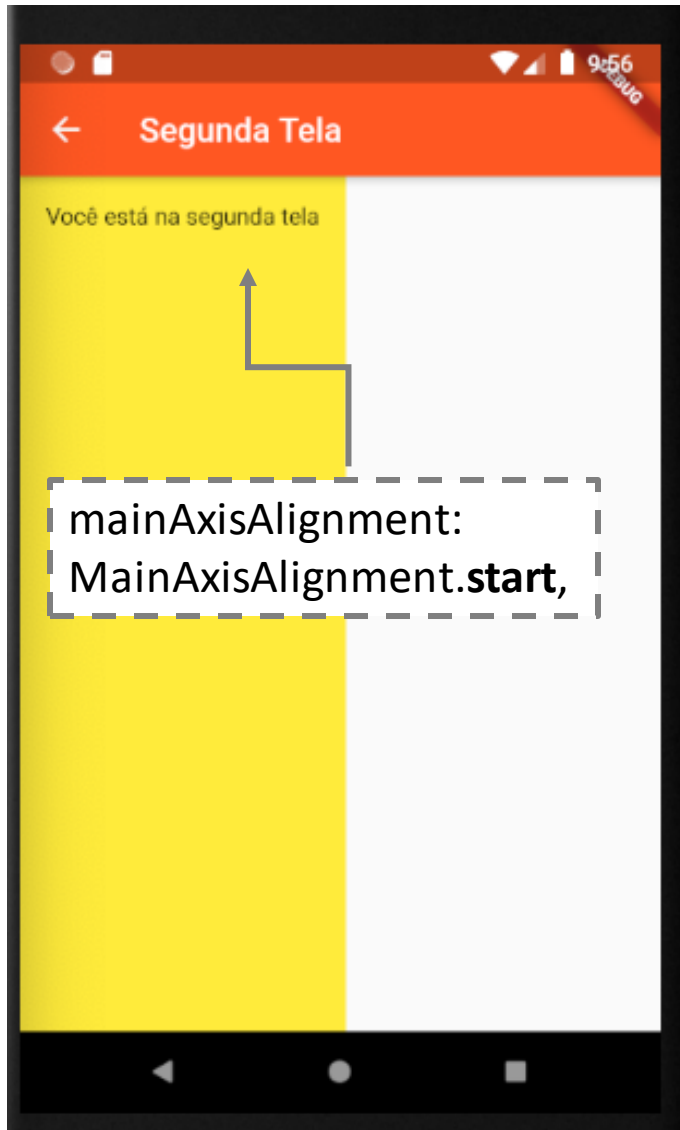


MainAxisAlignment

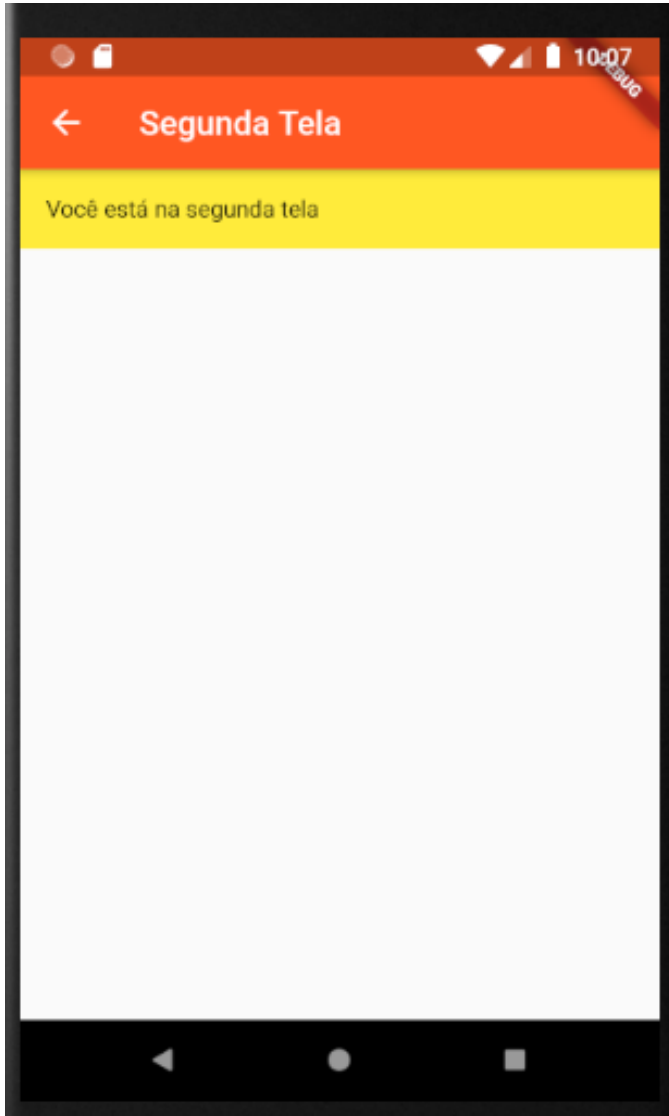


```
body: Container(  
  color: Colors.yellow,  
  padding: EdgeInsets.all(16),  
  child: Column(  
    mainAxisAlignment: MainAxisAlignment.start,  
    children: <Widget>[  
      Text("Você está na segunda tela")  
    ], // <Widget>[]  
  ), // Column  
) , // Container
```

MainAxisAlignment

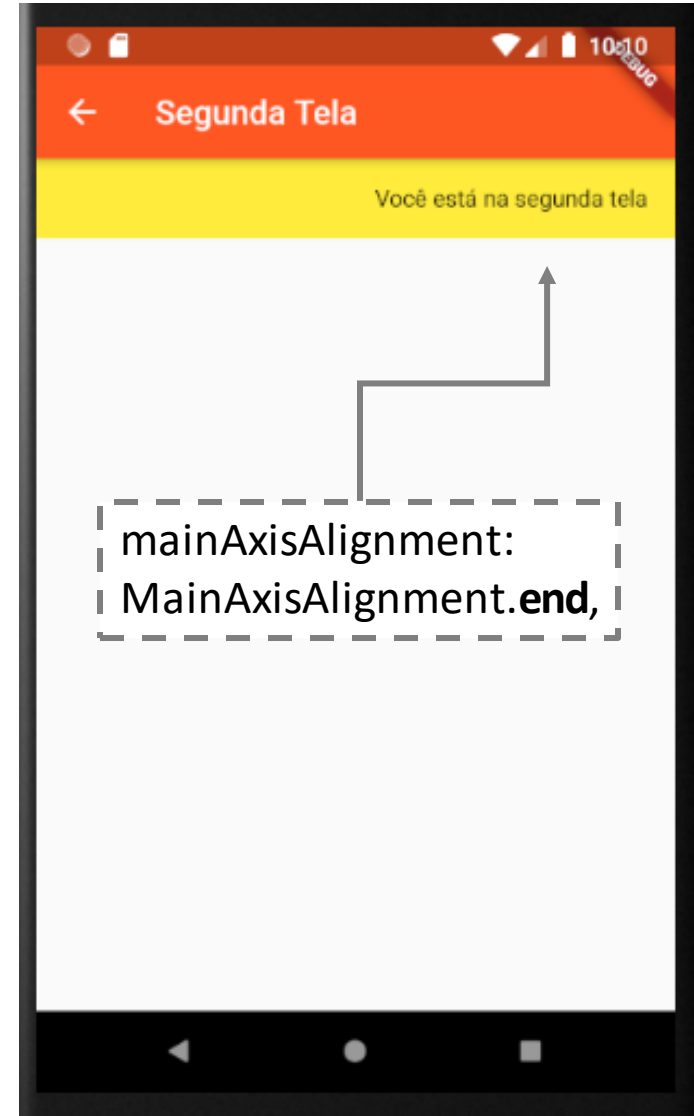
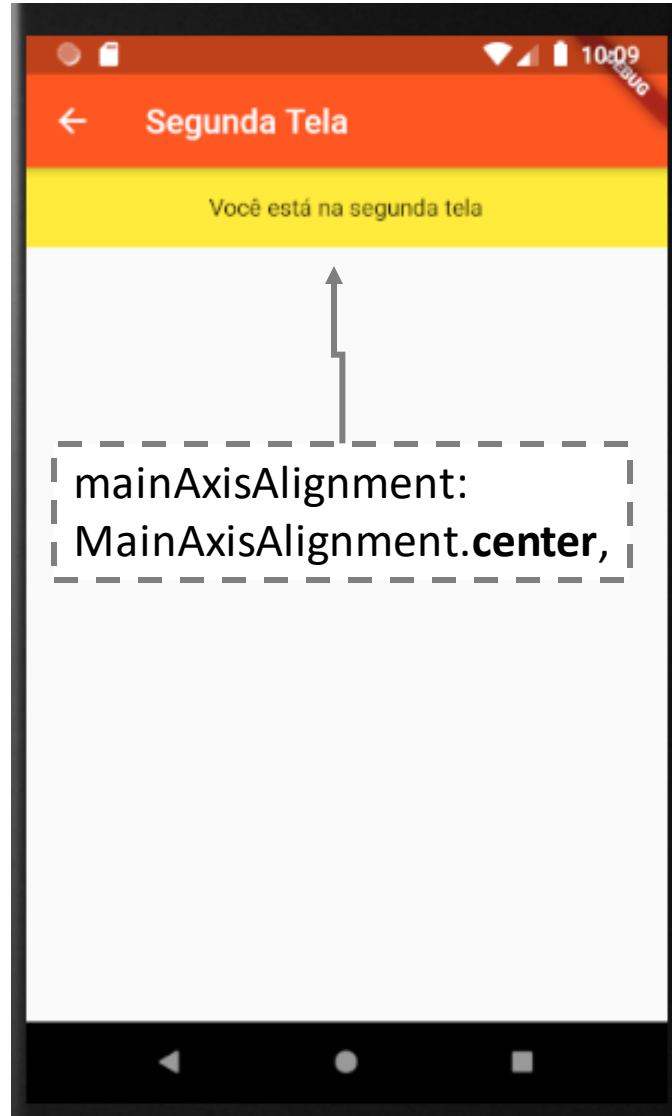
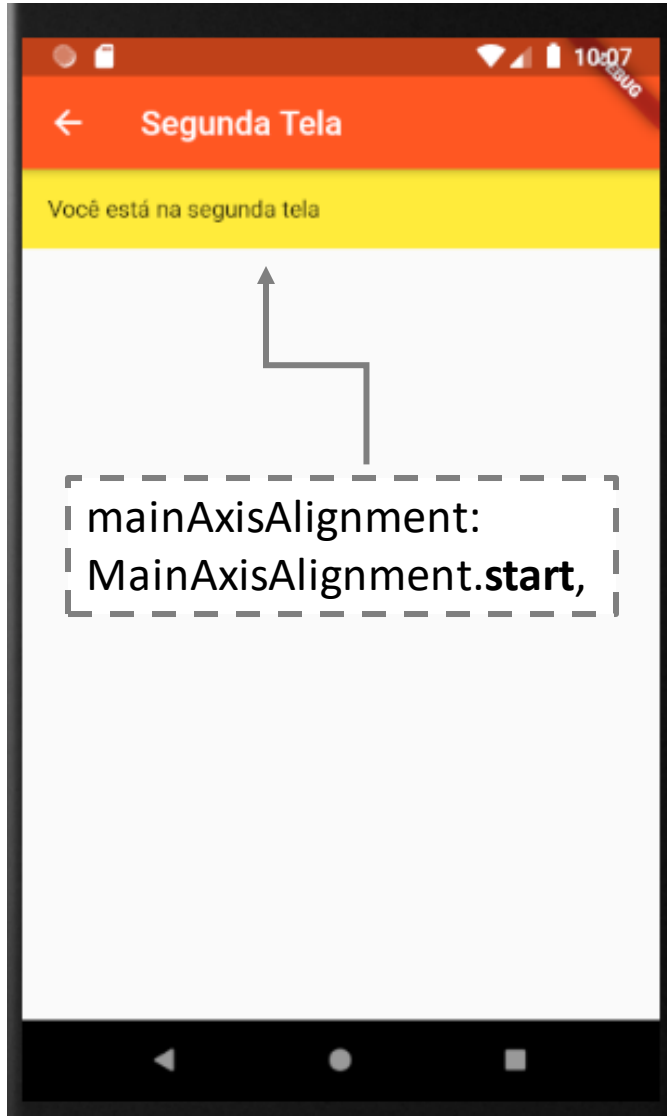


MainAxisAlignment



```
body: Container(  
  color: Colors.yellow,  
  padding: EdgeInsets.all(16),  
  child: Row(  
    mainAxisAlignment: MainAxisAlignment.start,  
    children: <Widget>[  
      Text("Você está na segunda tela")  
    ], // <Widget>[]  
  ), // Row  
) , // Container
```

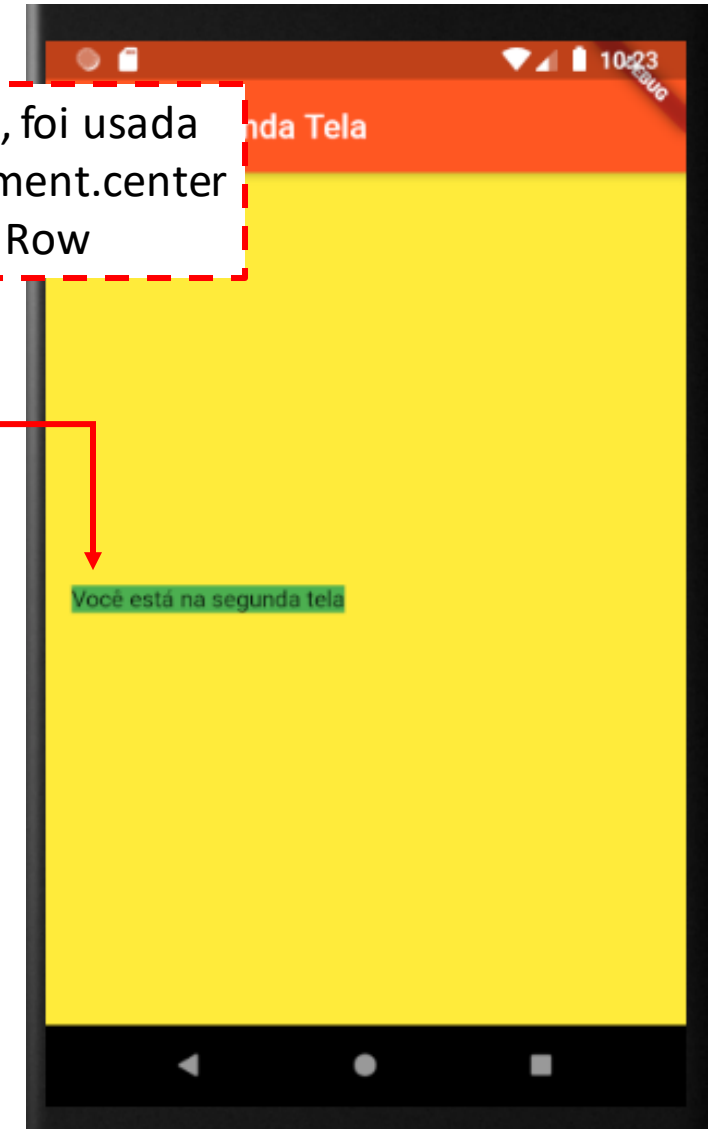
MainAxisAlignment



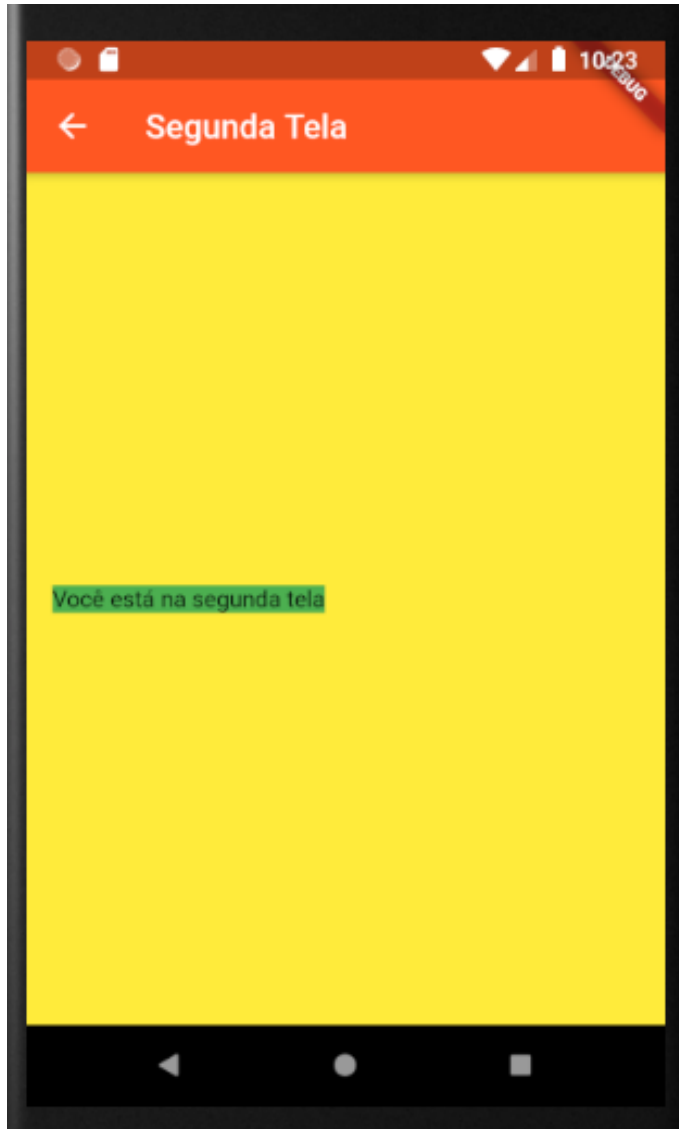
CrossAxisAlignment

- Permite a movimentação de botões, textos, etc no eixo cruzado de uma linha ou de uma coluna
- O padrão do CrossAxisAlignment é centro (center)
- É importante lembrar de estender a largura (no caso da coluna) ou altura (no caso da linha) do container

Nesse exemplo, foi usada
`CrossAxisAlignment.center`
dentro de uma Row



CrossAxisAlignment

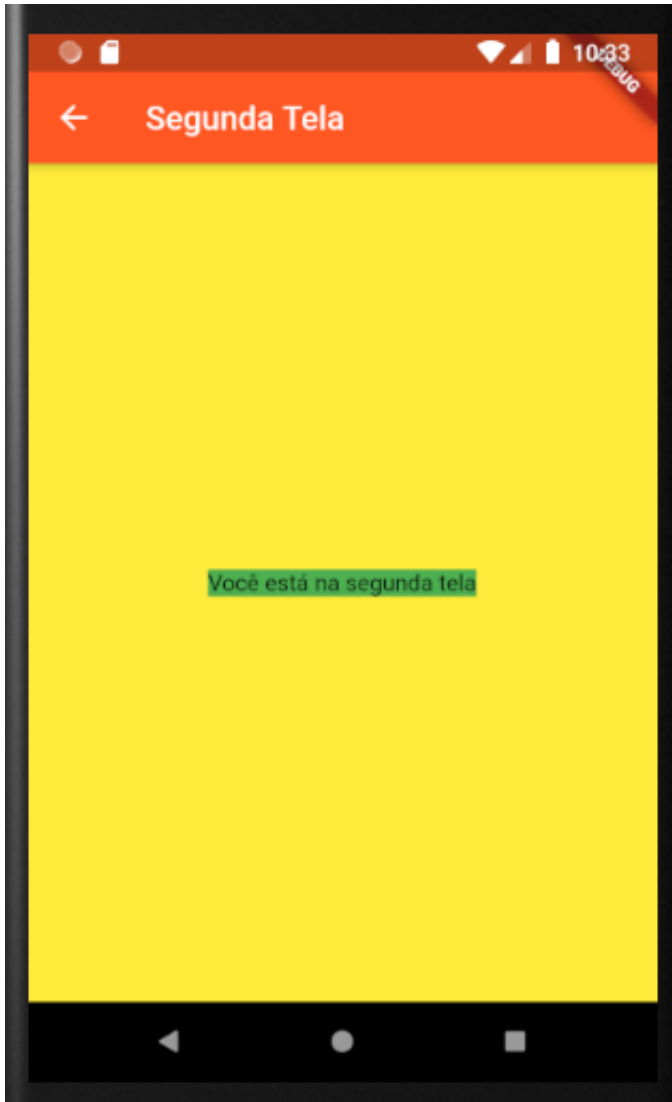


```
body: Container(  
  height: double.infinity,  
  color: Colors.yellow,  
  padding: EdgeInsets.all(16),  
  child: Row(  
    crossAxisAlignment: CrossAxisAlignment.center,  
    children: <Widget>[  
      Text("Você está na segunda tela",  
        style: TextStyle(  
          backgroundColor: Colors.green,  
        ),) // TextStyle, Text  
    ], // <Widget>[]  
  ), // Row  
) // Container
```

Dessa forma, a altura do container será "infinita", ou seja, considerando a tela toda.

Ficou no centro porém no canto, pois por padrão o MainAxisAlignment é start

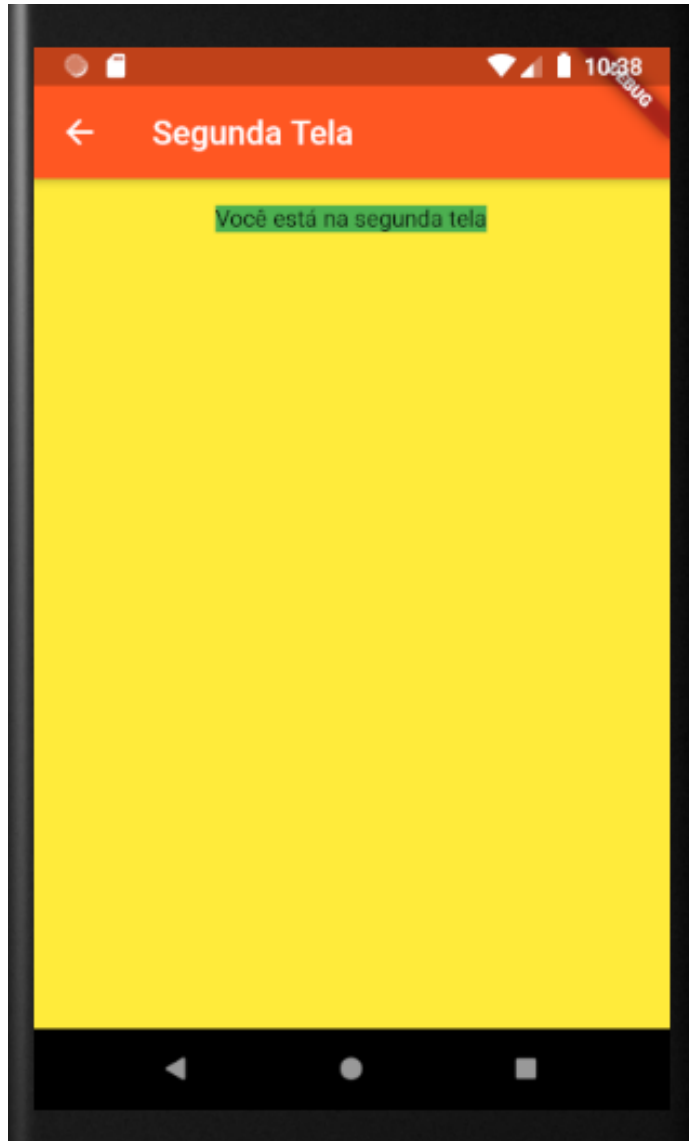
CrossAxisAlignment



```
body: Container(  
  height: double.infinity,  
  color: Colors.yellow,  
  padding: EdgeInsets.all(16),  
  child: Row(  
    mainAxisAlignment: MainAxisAlignment.center,  
    crossAxisAlignment: CrossAxisAlignment.center,  
    children: <Widget>[  
      Text("Você está na segunda tela",  
        style: TextStyle(  
          backgroundColor: Colors.green,  
        ),) // TextStyle, Text  
    ], // <Widget>[]  
  ), // Row  
) // Container
```

Agora o texto ficou no centro tanto em termos de eixo cruzado, como também de eixo principal

CrossAxisAlignment

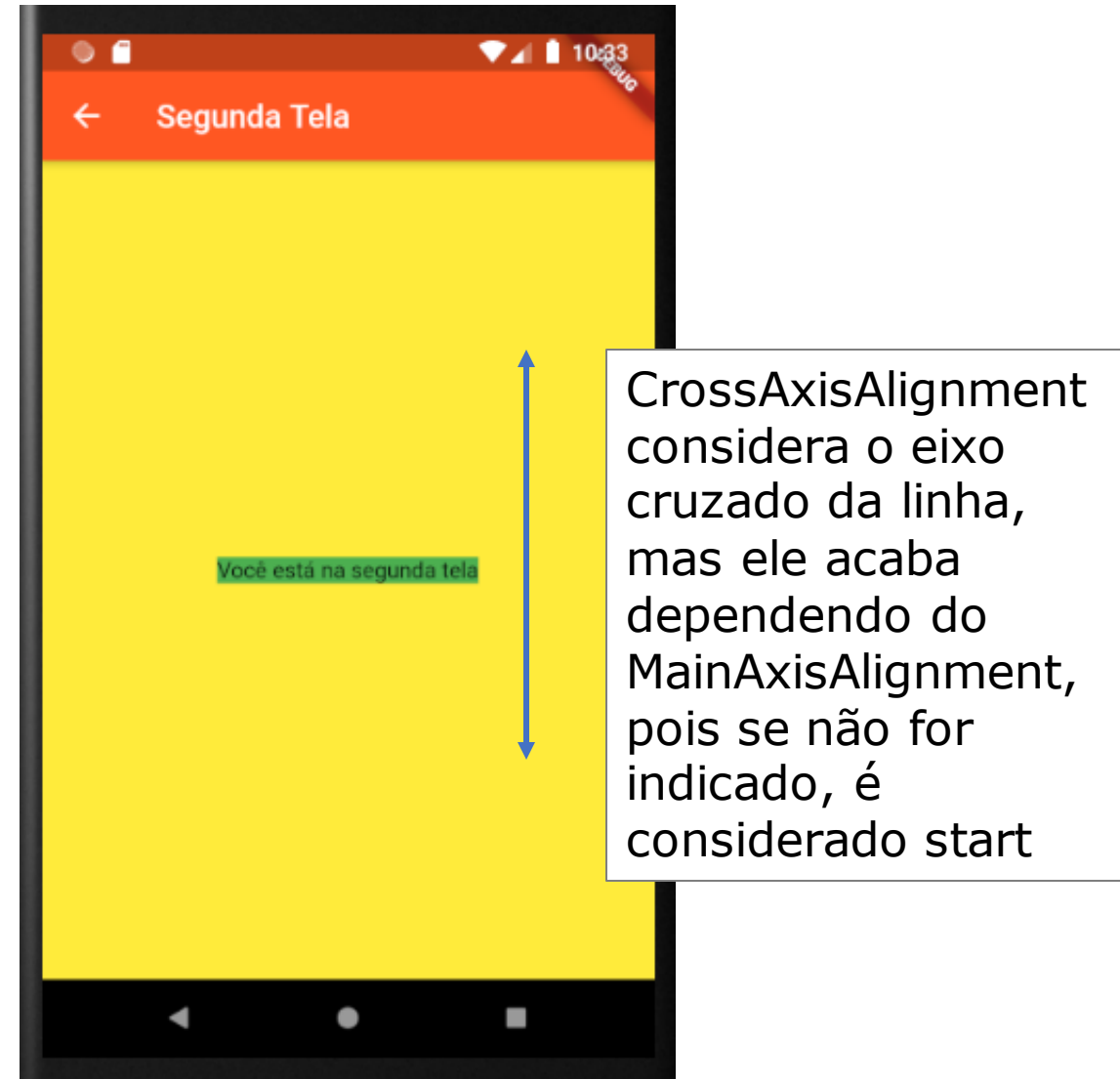
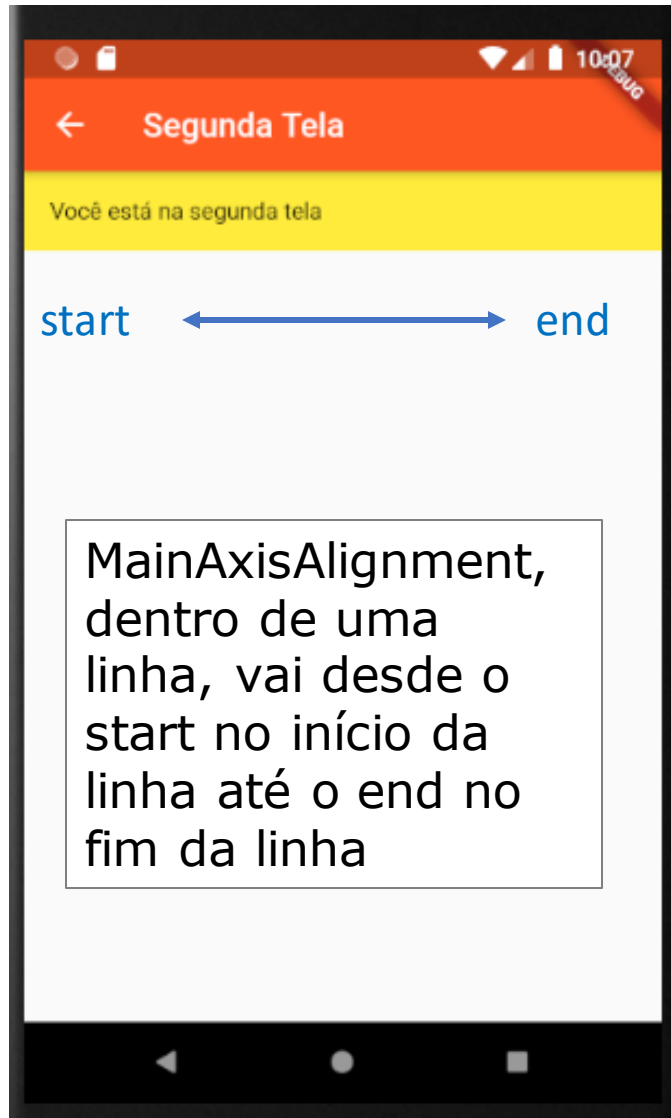


```
body: Container(  
  width: double.infinity,  
  color: Colors.yellow,  
  padding: EdgeInsets.all(16),  
  child: Column(  
    crossAxisAlignment: CrossAxisAlignment.center,  
    children: <Widget>[  
      Text("Você está na segunda tela",  
        style: TextStyle(  
          backgroundColor: Colors.green,  
        ),) // TextStyle, Text  
    ], // <Widget>[]  
  ), // Column  
) // Container
```

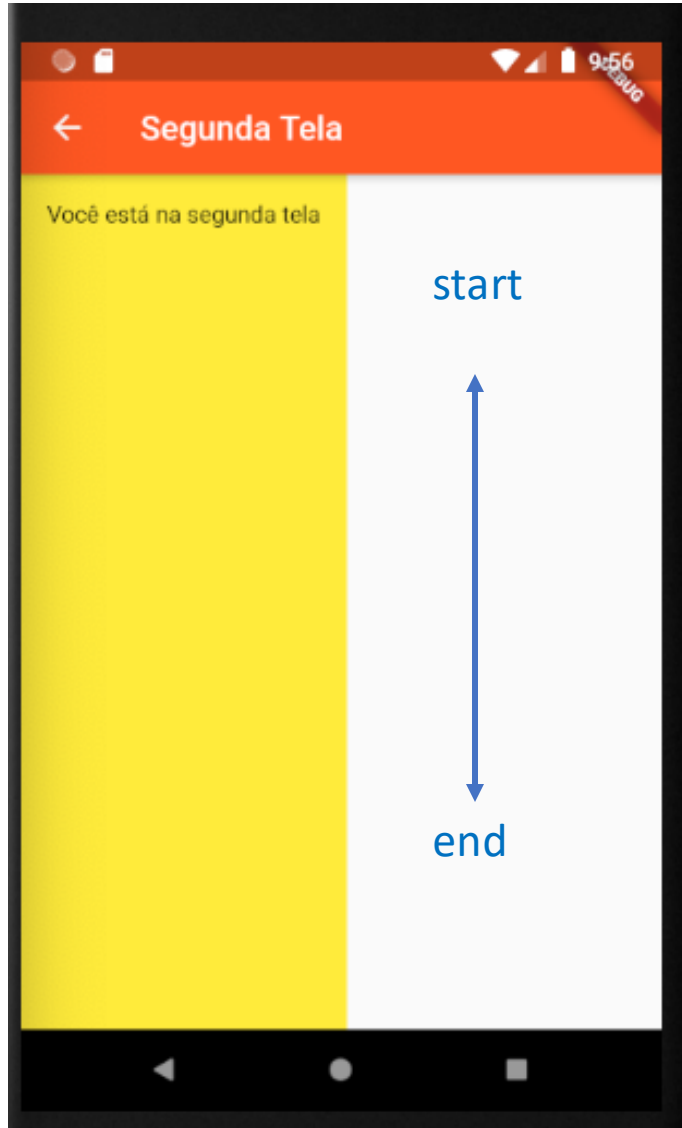
Dessa forma, a largura do container será "infinita", ou seja, considerando a tela toda.

Ficou no centro porém no encima, pois por padrão o MainAxisAlignment é start, e agora estamos considerando a coluna

MainAxisAlignment X CrossAxisAlignment - Row



MainAxisAlignment X CrossAxisAlignment - Column



Referências Bibliográficas

- Curso da Udemy – Flutter Essencial do professor Ricardo Lecheta.
- Curso da Udemy - Desenvolvimento Android e IOS com Flutter 2020 – Crie 15 Apps do professor Jamilton Damasceno.
- Site Flutter – flutter.dev