



# Componentes de Interface

Parte 2

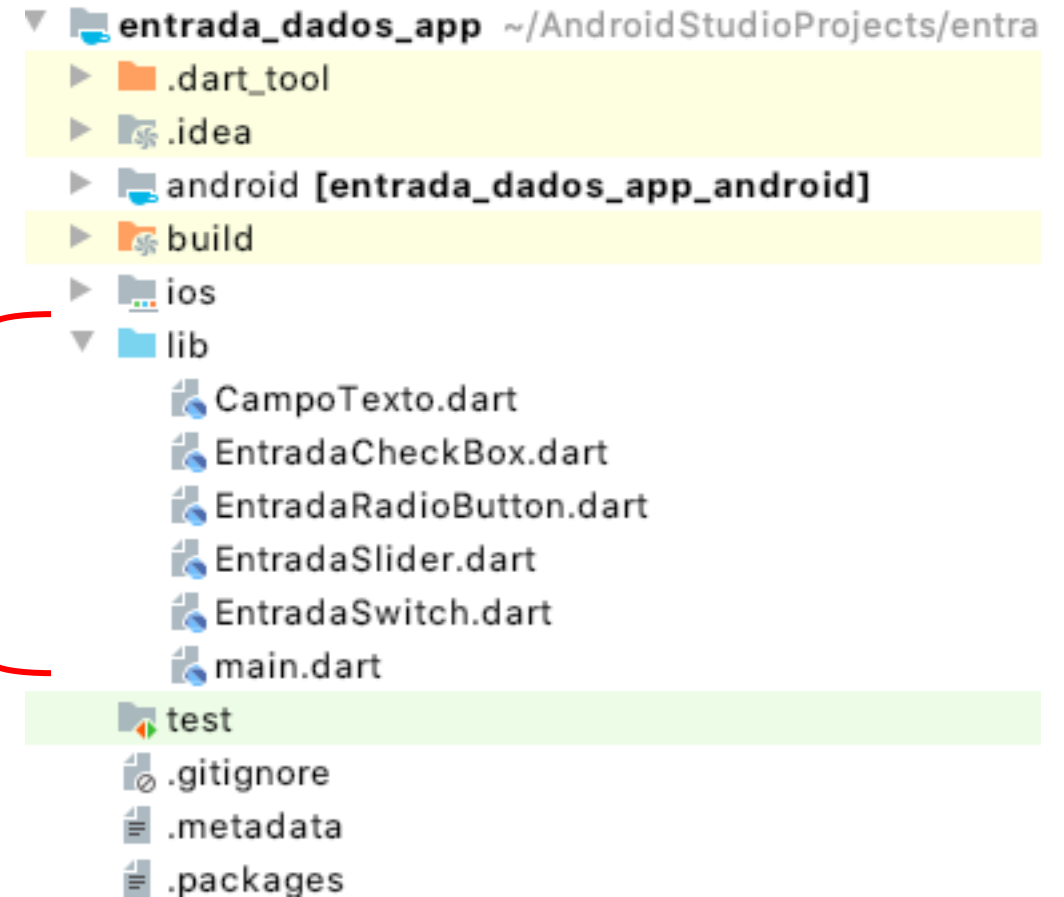
Prof. Ilo Rivero  
([ilo@pucminas.br](mailto:ilo@pucminas.br))

# O que vamos aprender nessa aula?

- Widgets Stateless e Stateful
- TextField
- Checkbox
- RadioButton
- Switch
- Slider

# O que vamos aprender nessa aula?

No nosso exemplo,  
vamos trabalhar com  
vários componentes de  
entrada, com diversos  
arquivos no nosso  
diretório lib



# Widgets Stateless e Stateful

- Ao escrever um App, normalmente, é necessário a criação de outros widgets, que são subclasses de Stateless ou Stateful.
- **Stateless:** caso seu widget não gerencie nenhum alteração de estado
- **Stateful:** caso seu widget gerencie alguma alteração de estado

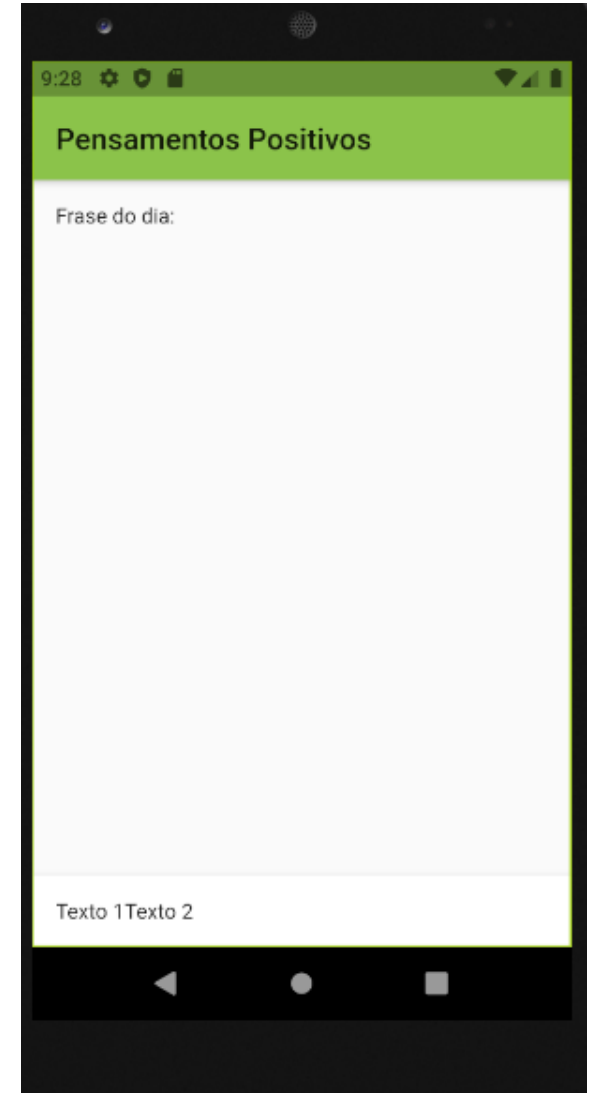
# Widgets Stateless e Stateful

- Exemplo: Considere que ao clicar em um botão você queira alterar um texto na tela do App, para isso, é necessário usar o Stateful. Por outro lado, caso você queira que alguns widgets na tela, que você não queira alterar valores, você deverá usar o Stateless

# Widgets Stateless

Exemplo:

```
class HomePage extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    var _titulo = "Pensamentos Positivos";  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(_titulo,  
          ), // Text  
      ), // AppBar  
      body: Padding(...), // Padding  
      bottomNavigationBar: BottomAppBar(...), // BottomAppBar  
    ); // Scaffold  
  }  
}
```



# Widgets Stateful

- Vamos agora usar o Stateful e nesse caso, vamos alterar o valor de uma variável. Observe que no caso do Stateful, são criadas duas classes, uma que estende de StatefulWidget, e outra que estende de State<HomeStateful>
- Observe que a primeira classe criada, é responsável por criar um estado inicial e a segunda classe é que contém o build

# Widgets Stateful

Exemplo:

```
class HomeStateful extends StatefulWidget {  
  @override  
  _HomeStatefulState createState() => _HomeStatefulState();  
}  
  
class _HomeStatefulState extends State<HomeStateful> {  
  @override  
  Widget build(BuildContext context) {  
    return Container();  
  }  
}
```



# Widgets Stateful

Exemplo:

```
class _HomeStatefulState extends State<HomeStateful> {  
  var _frase = "Seja feliz, hoje e sempre";  
  @override  
  Widget build(BuildContext context) {  
    print("chamado");  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Pensamento do dia",  
          ), // Text  
      ), // AppBar  
    );  
  }  
}
```

Criação de uma  
variável e inicializada  
com um valor

Continua...

# Widgets Stateful

## Exemplo:

```
body: Container(  
  child: Column(  
    children: <Widget>[  
      RaisedButton(  
        onPressed: () {  
          setState(() {  
            _frase = "Substitua pensamentos negativos por pensamentos positivos";  
          });  
        },  
        child: Text("Clique aqui"),  
      ), // RaisedButton  
      Padding(  
        padding: EdgeInsets.all(16), // EdgeInsets.all  
        child: Text("Frase: $_frase",
```

Nesse exemplo, ao clicar no botão, a variável `_frase` será alterada. Para alterar o valor, é necessário usar a função `setState()`

É apresentada na tela o valor da variável `_frase`, antes de clicar no botão, o valor apresentado, será o valor inicial da variável

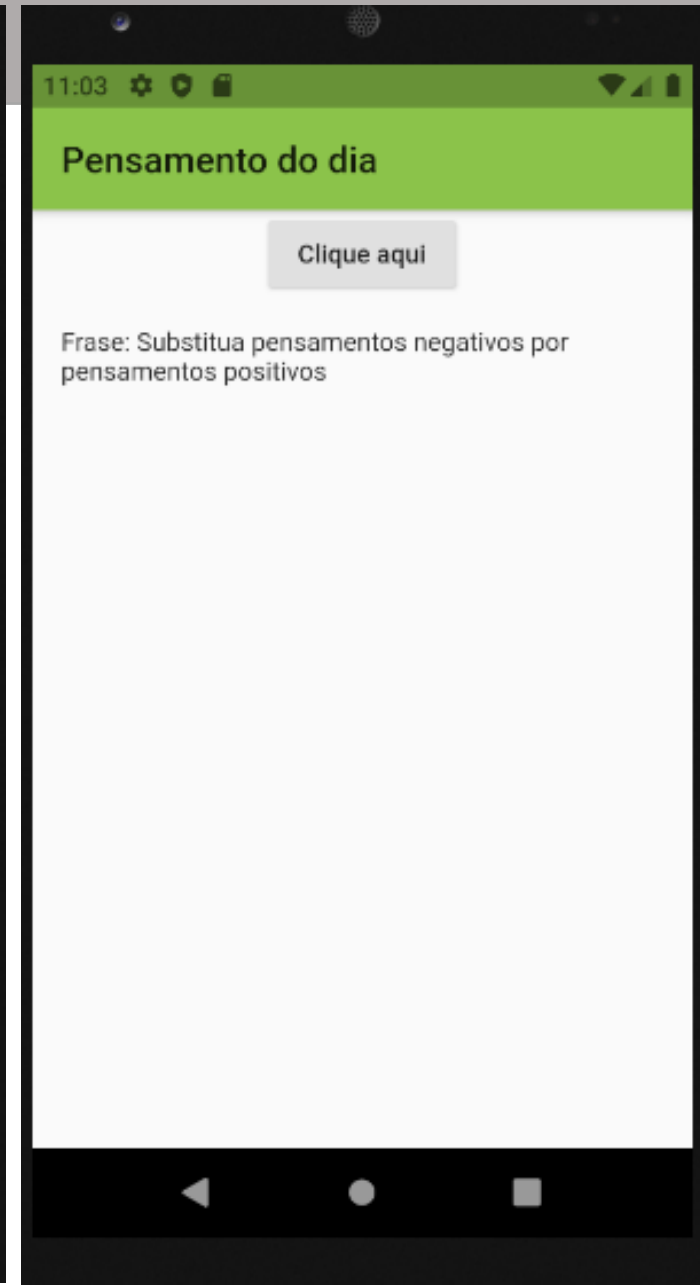
# Widgets Stateful

- A primeira imagem representa a tela em seu estado inicial, já a segunda imagem representa a tela após clicar no botão “clique aqui”, observe a frase que aparece em cada tela.

Tela1:



Tela2:



# Text Field

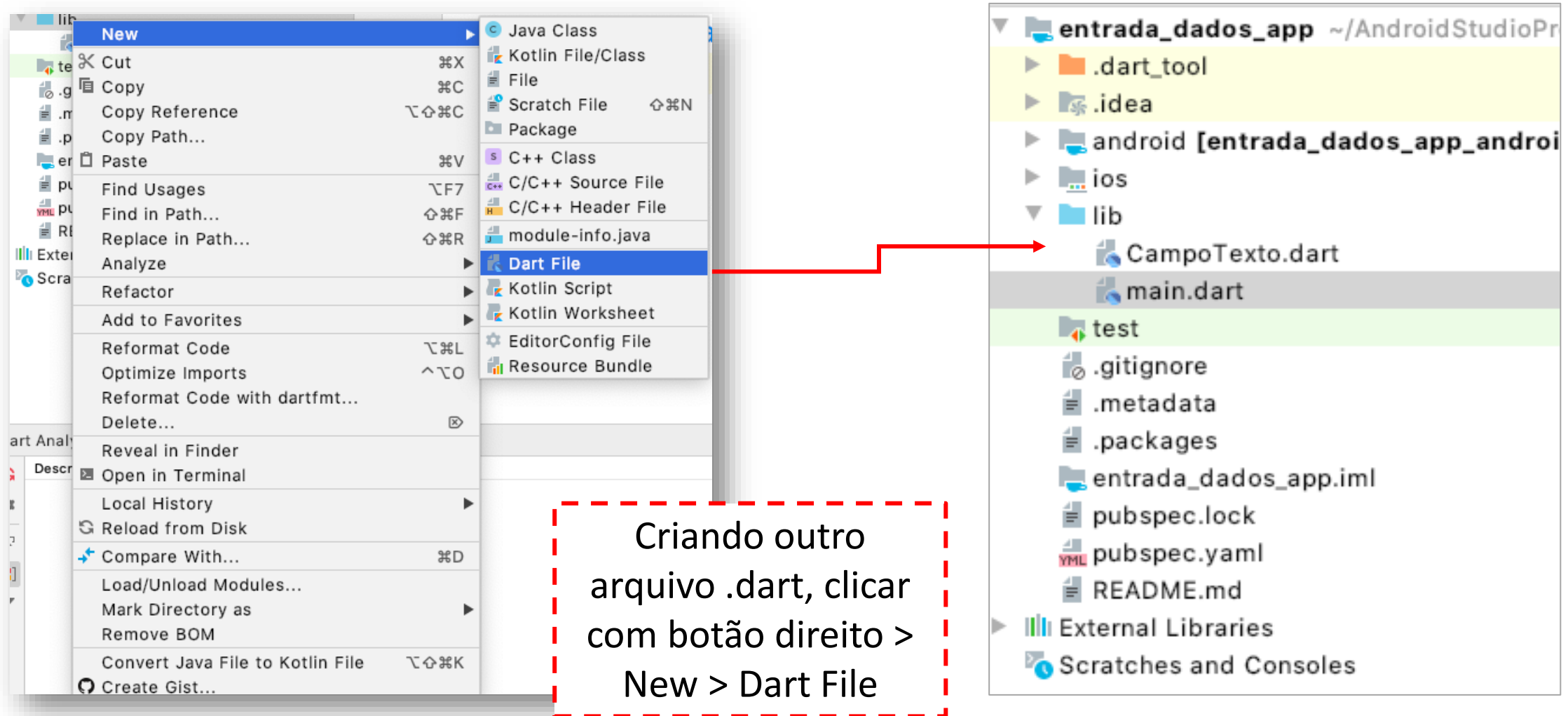
- Nesse exemplo, vamos criar outro arquivo .dart para incluir a classe que vamos fazer o controle da caixa de texto (TextField)

```
import 'package:entrada_dados_app/CampoTexto.dart';  
import 'package:flutter/material.dart';  
  
void main() {  
  runApp(MaterialApp(  
    home: CampoTexto(),  
  )); // MaterialApp  
}
```

No home vamos chamar a classe CampoTexto( ) que será criada em outro arquivo chamada CampoTexto.dart

No arquivo main.dart será necessário incluir um import para o(s) outro(s) arquivos .dart para poder usar o que estão no(s) outro(s) arquivos

# Text Field



# Text Field

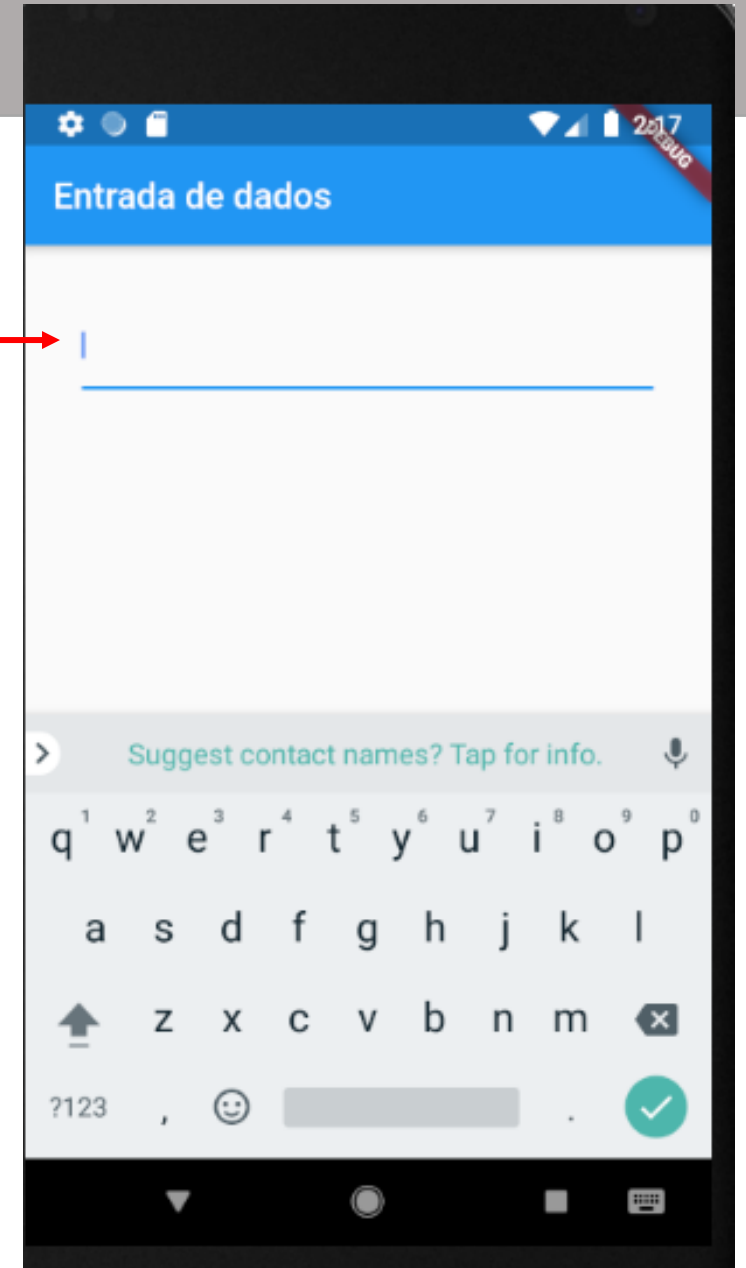
```
import 'package:flutter/material.dart';  
class CampoTexto extends StatefulWidget {  
  @override  
  _CampoTextoState createState() => _CampoTextoState();  
}  
  
class _CampoTextoState extends State<CampoTexto> {  
  @override  
  Widget build(BuildContext context) {  
    return Container();  
  }  
}
```

Nossa classe deverá estender do widget Stateful, pois dessa forma é possível alterar o valor da caixa de texto

# Text Field

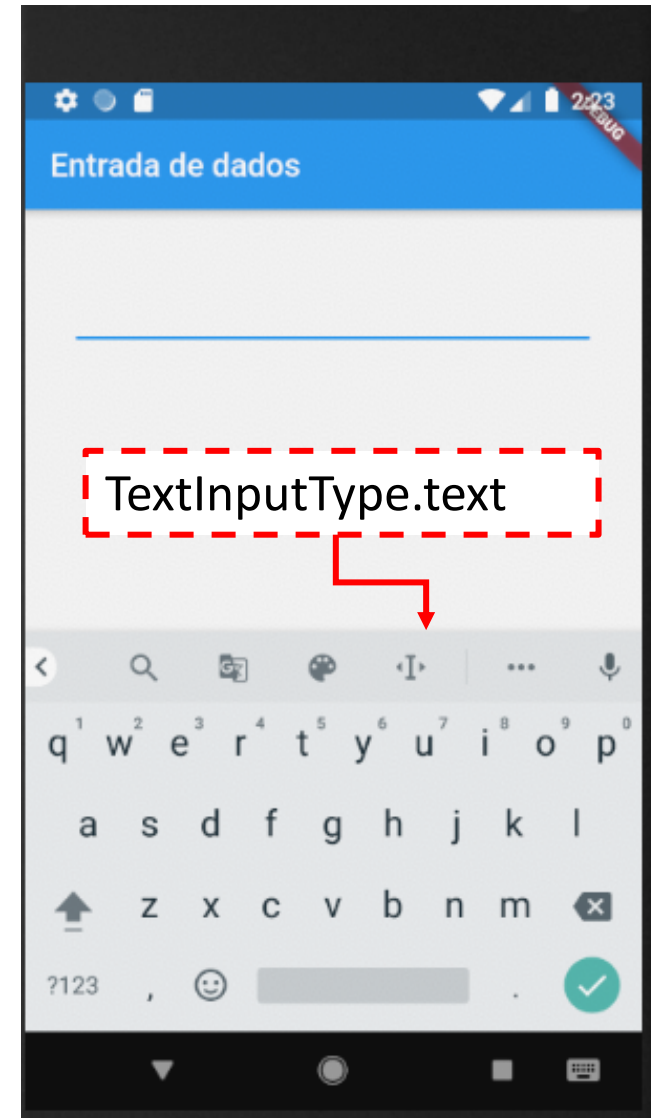
```
class _CampoTextoState extends State<CampoTexto> {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Entrada de dados"),  
      ), // AppBar  
      body: Column(  
        children: <Widget>[  
          Padding(  
            padding: EdgeInsets.all(32),  
            child: TextField(  
            ), // TextField  
          ) // Padding  
        ], // <Widget>[]  
      ), // Column  
    ); // Scaffold  
  }  
}
```

Observe que só de incluir o Text Field e rodar, já aparece na tela do App esse campo para digitar. E ao clicar já abre o teclado como na imagem



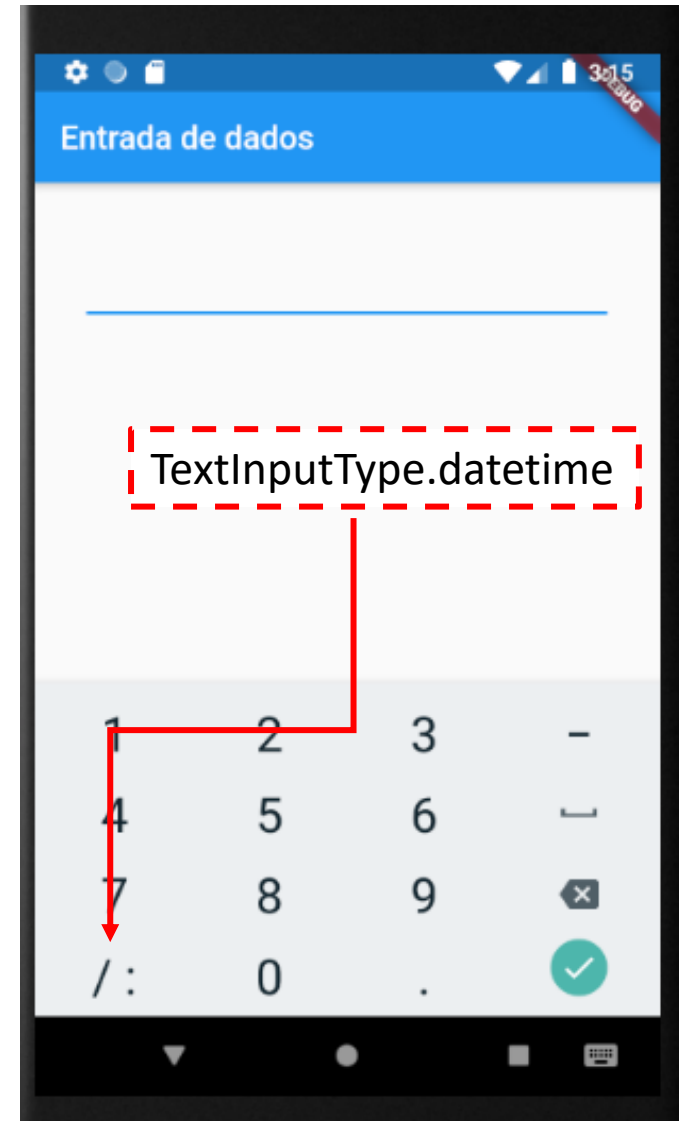
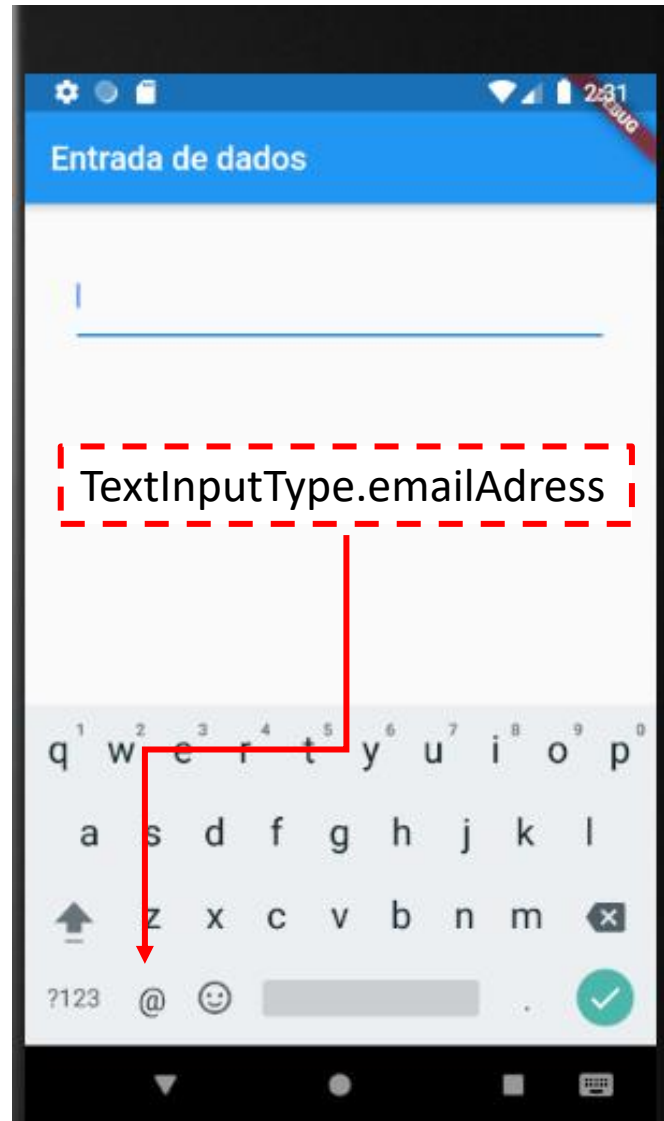
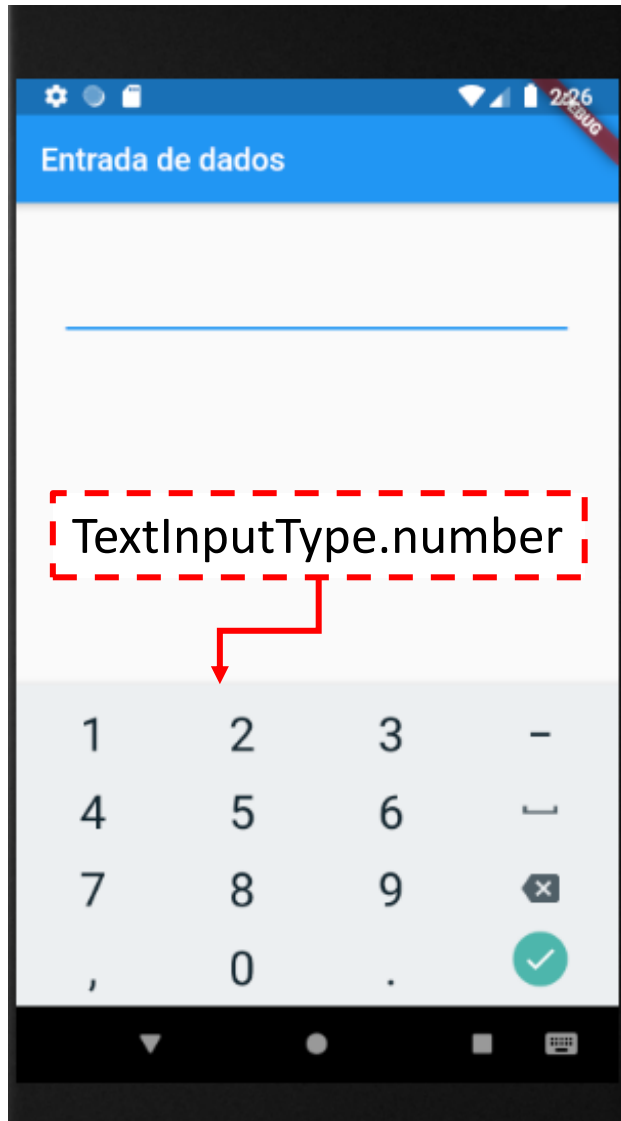
# Text Field

- No TextField vamos fazer algumas definições, como:  
keyboardType (tipo de teclado):
- keyboardType:  
TextInputType.text, para abrir um teclado para texto, como na imagem apresentada





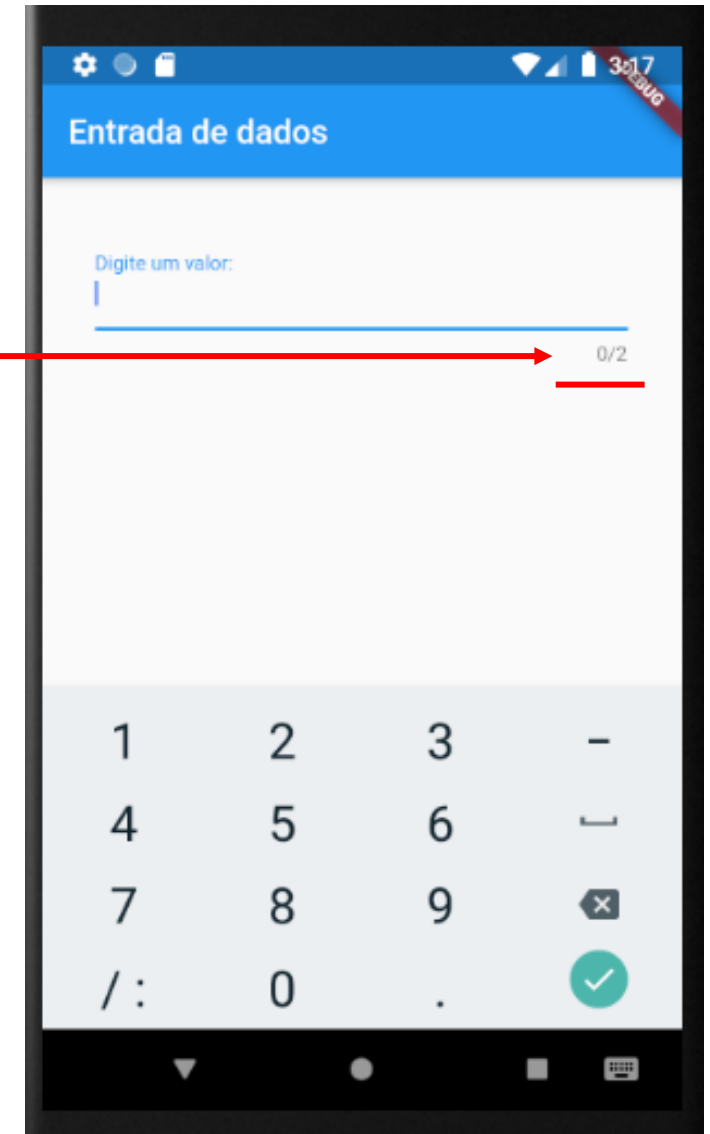
# Text Field - keyboardType



# Text Field

```
- child: TextField(  
  keyboardType: TextInputType.datetime,  
  decoration: InputDecoration(  
    labelText: "Digite um valor: "  
  ), // InputDecoration  
  enabled: true,  
  maxLength: 2,   
  maxLengthEnforced: true,  
), // TextField
```

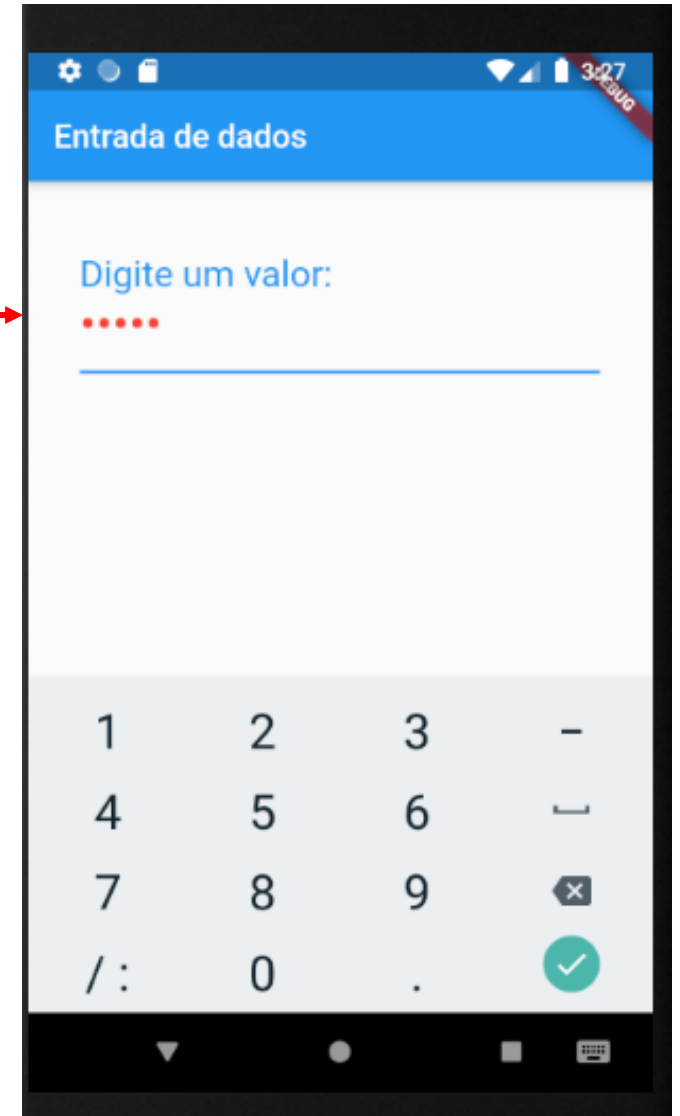
Não permite digitar  
mais do que o valor  
máximo definido  
pelo maxLength



# Text Field

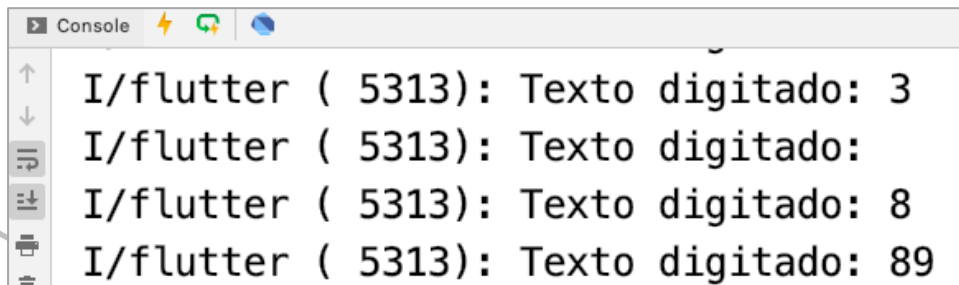
- É possível formatar o texto do TextField usando o TextStyle, aumentando o tamanho da fonte, mudando a cor do texto, etc.
- Além disso, é possível esconder o texto digitado, por exemplo, em senhas, usando o obscureText: **true**,

obscureText  
igual a true



# Text Field - Recuperar o valor digitado

- Para recuperar o valor digitado, vamos usar o parâmetro **onChanged** com função anônima
- Ele será chamado sempre que houver alguma alteração no campo, observe o console:



The screenshot shows the Flutter console with the following log messages:

```
I/flutter ( 5313): Texto digitado: 3
I/flutter ( 5313): Texto digitado:
I/flutter ( 5313): Texto digitado: 8
I/flutter ( 5313): Texto digitado: 89
```

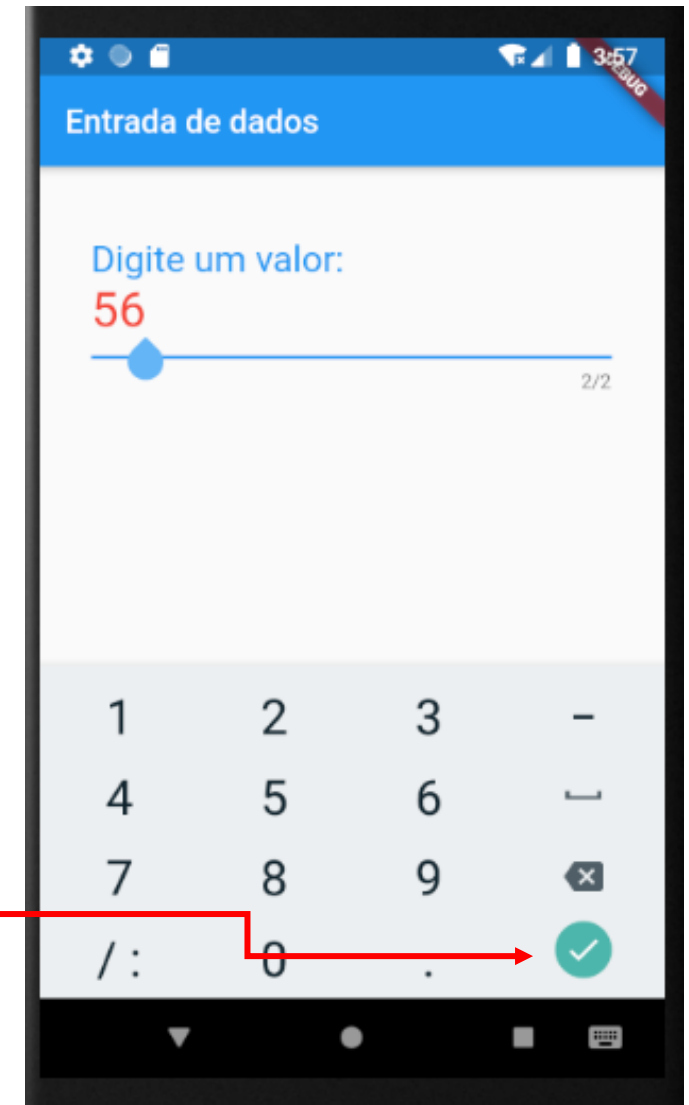
## Exemplo:

```
child: TextField(
  keyboardType: TextInputType.datetime,
  decoration: InputDecoration(
    labelText: "Digite um valor: "
  ), // InputDecoration
  style: TextStyle(
    fontSize: 30,
    color: Colors.red,
  ), // TextStyle
  onChanged: (String texto){
    print("Texto digitado: " + texto);
  },
), // TextField
```

# Text Field - Recuperar o valor digitado

- Para recuperar o valor digitado, vamos usar o parâmetro **onSubmitted** com função anônima
- Ele será chamado após confirmar a alteração do valor na caixa de texto, observe no próximo slide o console

onSubmitted  
recupera o valor  
após confirmação  
nesse botão

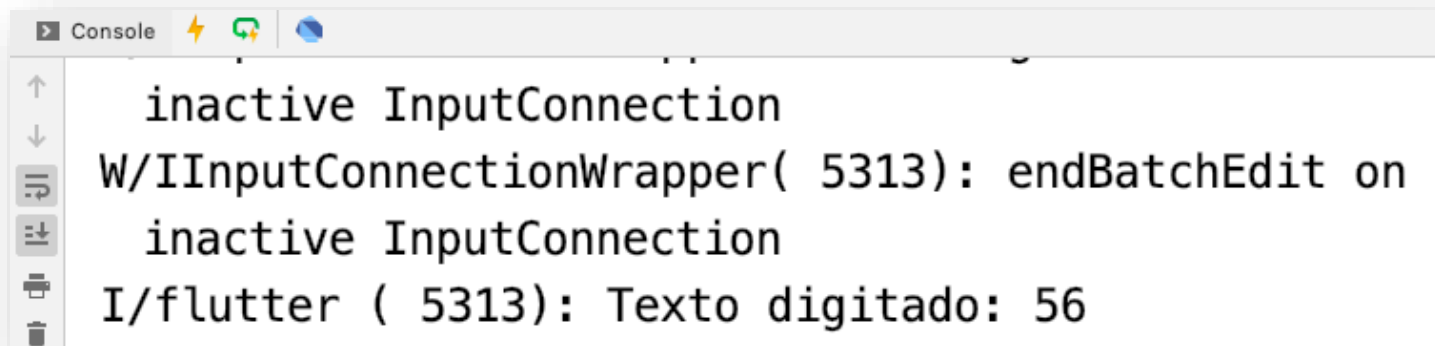


# Text Field - Recuperar o valor digitado

```
onSubmitted: (String texto){  
  print("Texto digitado: " + texto);  
},
```

onSubmitted  
recupera o valor  
após confirmação  
nesse botão

## Console:



The screenshot shows the Flutter console with the following output:

```
inactive InputConnection  
W/IInputConnectionWrapper( 5313): endBatchEdit on  
inactive InputConnection  
I/flutter ( 5313): Texto digitado: 56
```

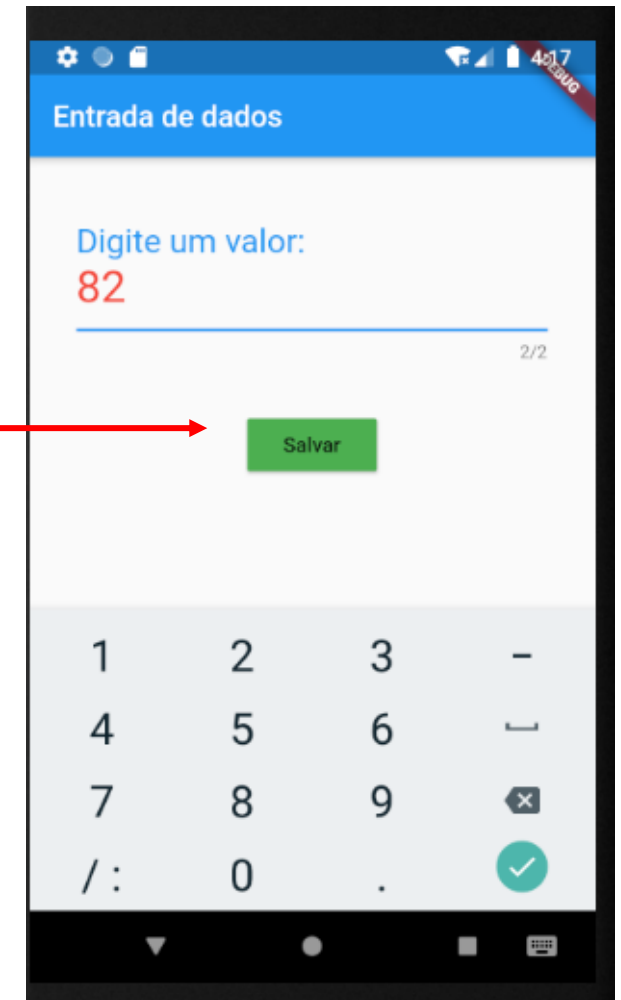
A red arrow points from the text box on the right to the console output.

# Text Field - Recuperar o valor digitado

- Podemos recuperar também um valor ao clicar em um botão, para isso, é necessário criar o botão, usar a função onPressed e um controlador

Após clicar no botão Salvar que será recuperado o valor

Tela:



```
inactive InputConnection
W/IInputConnectionWrapper( 5313): endBatchEdit on
inactive InputConnection
I/flutter ( 5313): Texto digitado: 82
```

# Text Field - Recuperar o valor digitado

```
class _CampoTextoState extends State<CampoTexto> {  
  → TextEditingController _textEditingController = TextEditingController();  
  @override  
  Widget build(BuildContext context) {
```

```
    child: TextField(  
      keyboardType: TextInputType.datetime,  
      decoration: InputDecoration(  
        labelText: "Digite um valor: "  
      ), // InputDecoration  
      style: TextStyle(  
        fontSize: 30,  
        color: Colors.red,  
      ), // TextStyle  
      → controller: _textEditingController, //  
    ), // TextField
```

```
    ElevatedButton( ←  
      child: Text("Salvar"),  
      style: ElevatedButton.styleFrom(  
        primary: Colors.green,  
      ),  
      → onPressed: () {  
        print("Texto digitado: " + _textEditingController.text);  
      },  
    ), // ElevatedButton
```



# Checkbox

- Como no exemplo anterior, vamos criar outro arquivo para incluir uma classe para usar o componente Checkbox

```
import 'package:flutter/material.dart';  
import 'EntradaCheckBox.dart';  
  
void main() {  
  runApp(MaterialApp(  
    home: EntradaCheckBox(),  
  )); // MaterialApp  
}
```

Vamos precisar fazer um import para o arquivo EntradaCheckBox.dart

No home vamos chamar a classe EntradaCheckBox() que será criada em outro arquivo chamada EntradaCheckBox.dart

# Checkbox

- Para uso do Checkbox são necessárias algumas configurações
- É preciso definir os dois parâmetros que estão como @required: value (estado inicial, true ou false) e onChanged

Checkbox.dart:

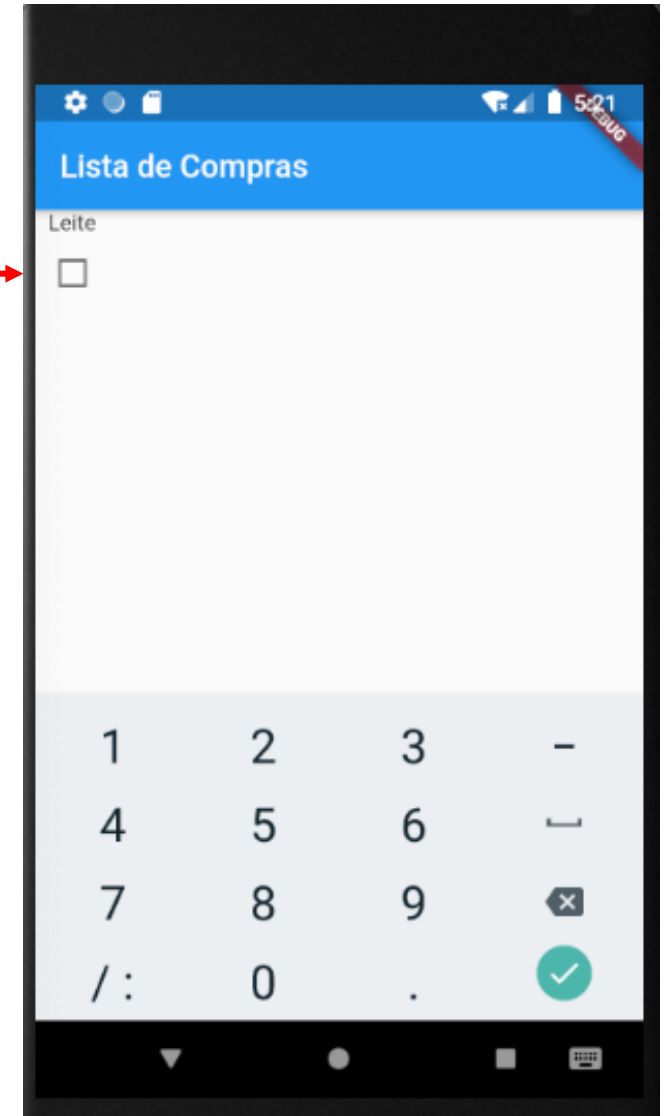
```
const Checkbox({  
  Key key,  
  @required this.value,  
  this.tristate = false,  
  @required this.onChanged,  
  this.activeColor,  
  this.checkColor,  
  this.focusColor,  
  this.hoverColor,  
  this.materialTapTargetSize,
```

# Checkbox

## Exemplo:

```
body: Container(  
  child: Column(  
    children: <Widget>[  
      Text("Leite"),  
      Checkbox(  
        value: false,  
        onChanged: (bool valor){  
          },  
        ), // Checkbox  
    ], // <Widget>[]  
  ), // Column  
, // Container
```

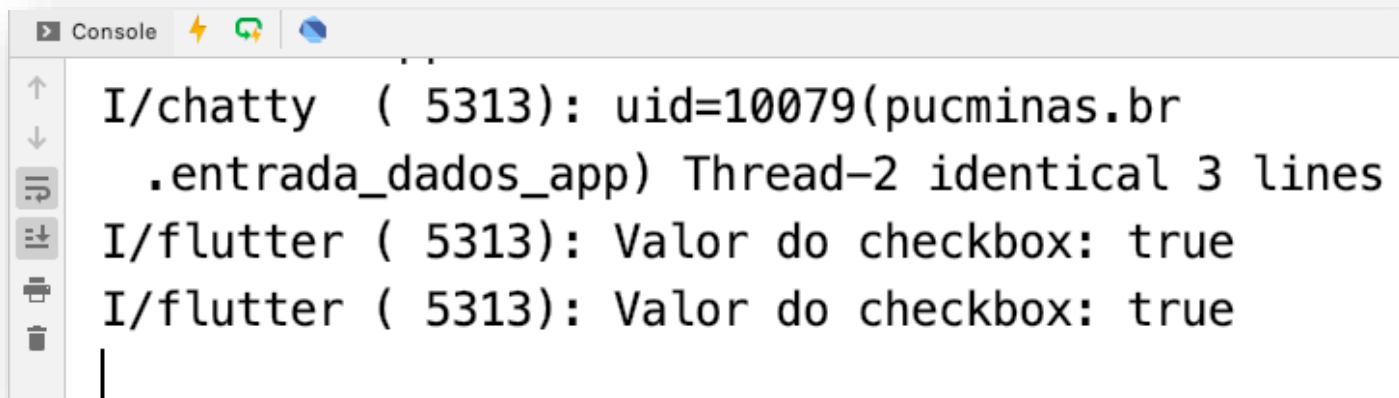
Value do  
Checkbox define  
se a caixinha vem  
ou não marcada



# Checkbox

## Exemplo:

```
Checkbox(  
  value: false,  
  onChanged: (bool valor){  
    print("Valor do checkbox: "+valor.toString());  
  },  
) , // Checkbox
```



The screenshot shows a Flutter console window with the following log output:

```
I/chatty ( 5313): uid=10079(pucminas.br  
.entrada_dados_app) Thread-2 identical 3 lines  
I/flutter ( 5313): Valor do checkbox: true  
I/flutter ( 5313): Valor do checkbox: true
```

Red arrows point from the text box on the right to the log output in the console.

Ao clicar no Checkbox, será apresentado no console o valor, no entanto, ele é booleano. Lembrando que o onChanged pega o estado que está sendo alterado, além disso, observe que o checkbox não fica selecionado/marcado

# Checkbox

- Para manter o checkbox como marcado ao selecionar, vamos precisar incluir o **setState()** dentro do `onChanged`, além de criar uma variável booleana para receber o valor `true` dentro desse método

## Exemplo:

```
Checkbox(  
  value: _seleccionado,  
  onChanged: (bool valor){  
    print("Valor do checkbox: "+valor.toString());  
    setState(() {  
      _seleccionado = valor;  
    });  
  },  
) // Checkbox
```

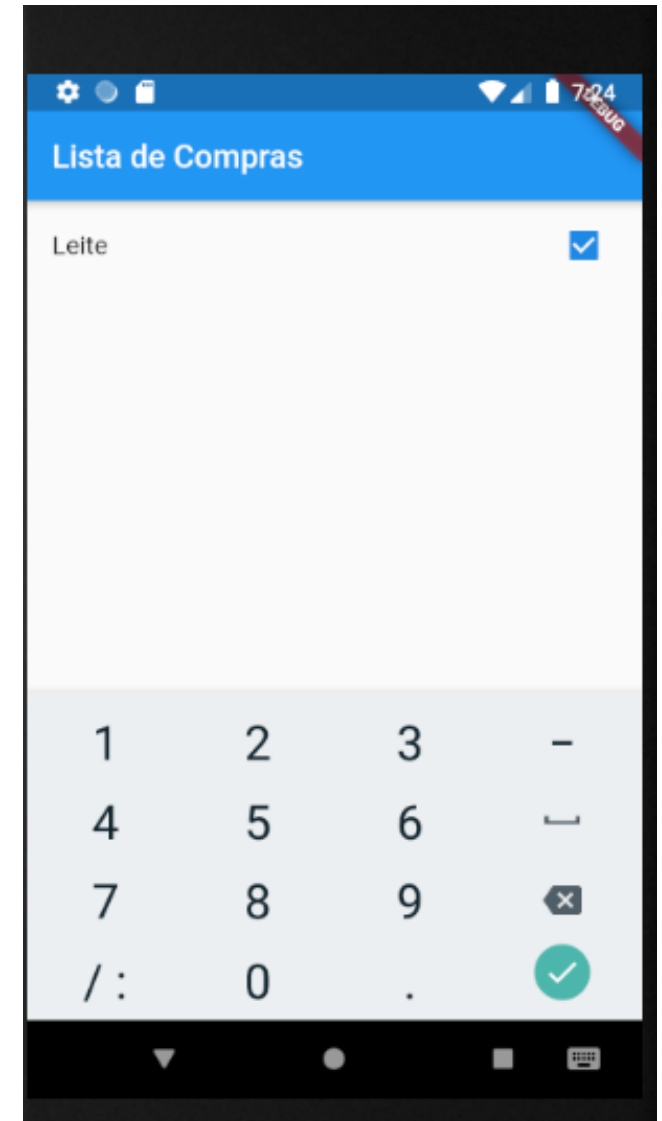
A variável **\_seleccionado** deve ser criada e inicializada com o valor `false`, como a seguir:

```
bool _seleccionado = false;
```

# Checkbox - CheckboxListTile

Tela:

- Outro método para definir um checkbox é usando a classe `CheckboxListTile`
- O **`CheckboxListTile`** possui mais configurações e, por exemplo, conseguimos definir um title direto nele
- Observe ao lado como fica o checkbox com o texto de um lado e a “caixinha” do outro



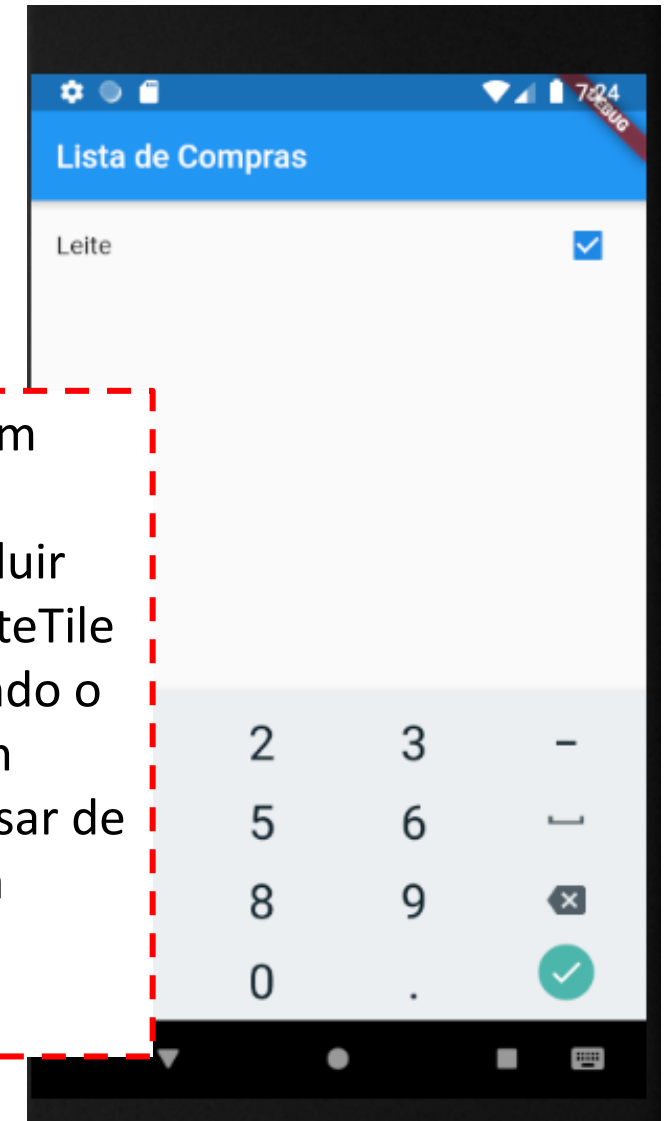
# Checkbox

Tela:

Exemplo:

```
CheckboxListTile(  
  title: Text("Leite"),  
  value: _selecionado,  
  onChanged: (bool valor){  
    setState(() {  
      _selecionado = valor;  
    });  
  },  
  // CheckboxListTile
```

Para incluir mais um item na lista de comprar, basta incluir outro CheckboxListTile como esse, alterando o valor do Text. Além disso, vamos precisar de outra variável para controlar o valor selecionado



# Checkbox - CheckboxListTile

Exemplo:

```
CheckboxListTile(  
  title: Text("Leite"),  
  secondary: Icon(Icons.add_box),  
  value: _leite,  
  onChanged: (bool valor){  
    setState(() {  
      _leite = valor;  
    });  
  }  
) , // CheckboxListTile
```

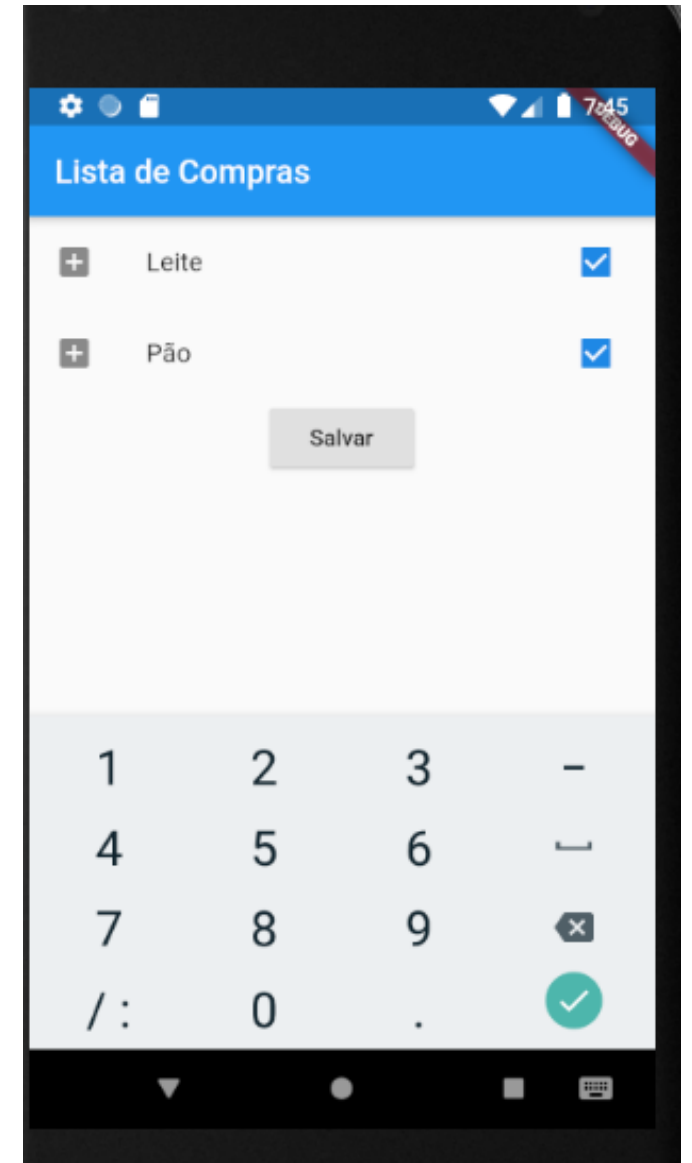
```
CheckboxListTile(  
  title: Text("Pão"),  
  secondary: Icon(Icons.add_box),  
  value: _pao,  
  onChanged: (bool valor){  
    setState(() {  
      _pao = valor;  
    });  
  }  
) , // CheckboxListTile
```



# Checkbox - CheckboxListTile

Tela:

- Agora vamos incluir um botão para obter qual ou quais checkbox foram marcados
- Vamos usar o **ElevatedButton** "Salvar" e ao selecionar um dos dois checkbox vamos imprimir no console o valor da variável que identifica se foi ou não selecionado, observe o console no slide a seguir

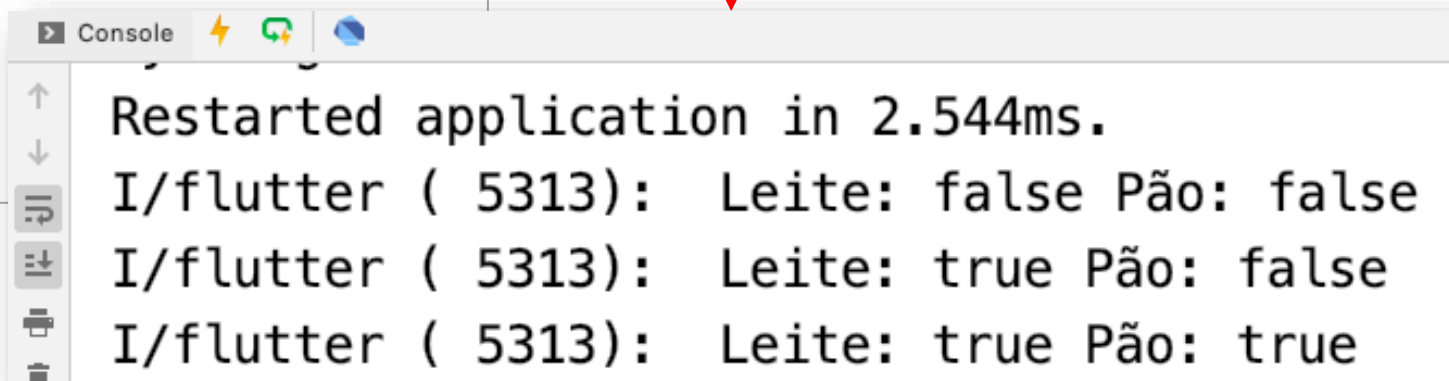


# Checkbox - CheckboxListTile

Exemplo:

```
 RaisedButton(  
  child: Text("Salvar"),  
  onPressed: (){  
    print(  
      " Leite: " + _leite.toString() +  
      " Pão: " + _pao.toString()  
    );  
  }  
) , // RaisedButton
```

Ao executar o App, observe que os dois checkbox estão como false, ou seja, não estão selecionados. Depois, foi selecionado o primeiro. Por fim, foram selecionados os dois.



```
 Console ⚡ ↺ 🌐  
↑  
↓  
I/flutter ( 5313): Leite: false Pão: false  
I/flutter ( 5313): Leite: true Pão: false  
I/flutter ( 5313): Leite: true Pão: true
```

# RadioButton - Radio

- **CheckBox** permite que o usuário selecione mais de uma opção dentre a lista apresentada, podendo até selecionar todos os itens
- Já **RadioButton**, permite que seja selecionada apenas uma opção dentre os itens/opções
- Para esse exemplo, também vamos criar um arquivo separado, `EntradaRadioButton.dart`, e como nos exemplos anteriores, vamos precisar incluir na main o import para esse arquivo, além de alterar o home para executar a classe criada para teste do `RadioButton`

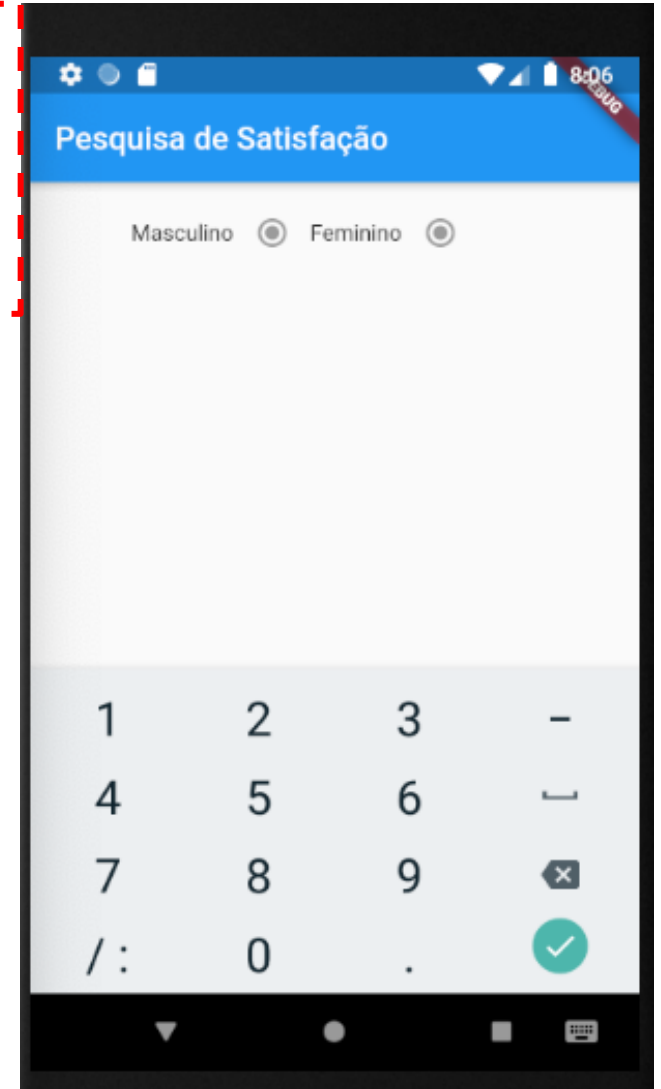
# RadioButton - Radio

Tela:

```
child: Row(  
  children: <Widget>[  
    Padding(  
      padding: EdgeInsets.all(32),  
    ), // Padding  
    Text("Masculino"),  
    Radio(  
      value: null,  
      groupValue: null,  
      onChanged: null  
    ), // Radio  
    Text("Feminino"),  
    Radio(  
      value: null,  
      groupValue: null,  
      onChanged: null  
    ), // Radio  
  ],  
),
```

Observe que o Radio foi inserido dentro de um Row, ou seja, linha, por isso, cada texto fica um do lado do outro na mesma linha.

Value: vai definir o valor para o Radio selecionado  
groupValue: define qual dos itens foi selecionado  
onChanged: identifica qual opção sofreu alteração



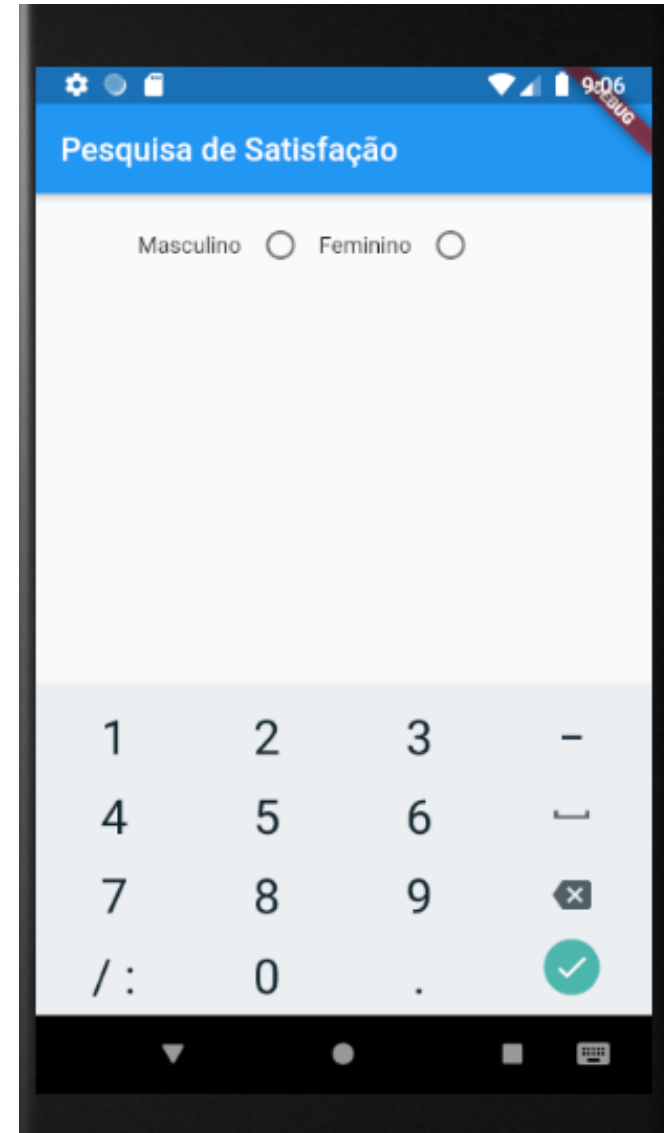
# RadioButton - Radio

Tela:

Exemplo:

```
Text("Masculino"),
Radio(
    value: "m",
    groupValue: _selecionado, |
    onChanged: (String e){
        print("Resultado: " + e);
    }
), // Radio
Text("Feminino"),
Radio(
    value: "f",
    groupValue: _selecionado,
    onChanged: (String e){
        print("Resultado: " + e);
    }
), // Radio
```

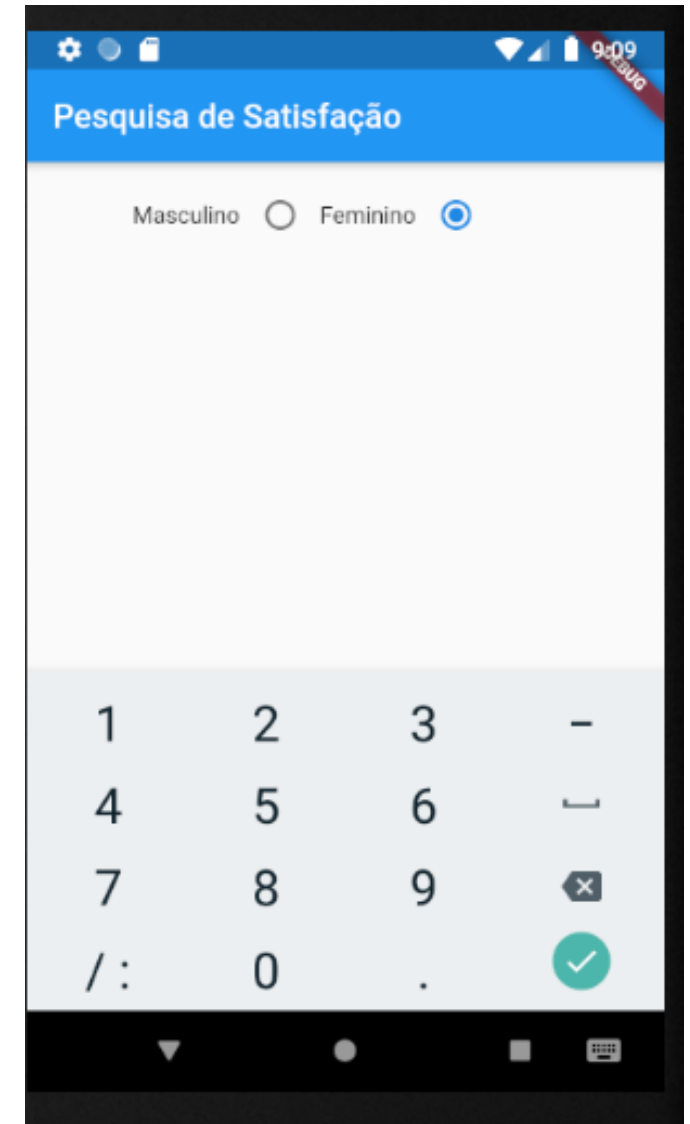
A variável \_selecionado foi definida como String e observe que a mesma é usada nos dois Radio, já que apenas é possível selecionar um item



# RadioButton - Radio

- Para marcar a opção que está sendo selecionado, usaremos como no CheckBox o `setState()` dentro do `onChanged` da mesma forma setando a variável `_selecionado` com o valor escolhido

Tela:

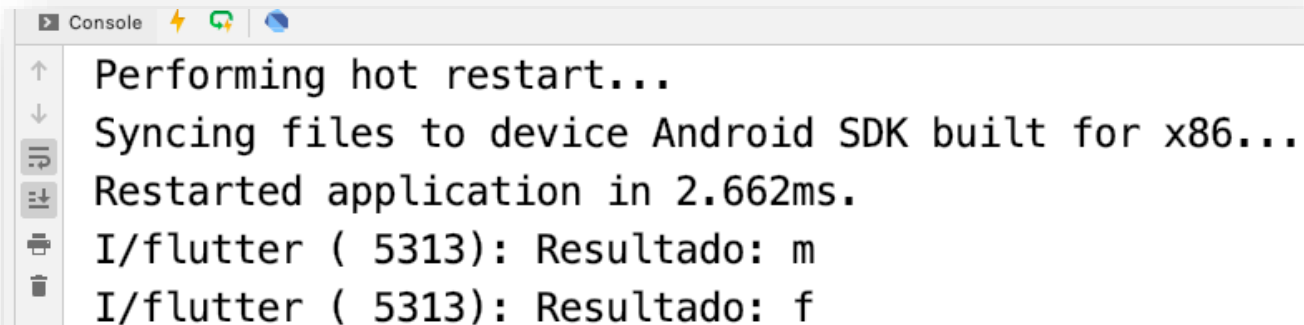


# RadioButton - Radio

Exemplo:

```
-Text("Masculino"),  
-Radio(  
  value: "m",  
  groupValue: _selecionado,  
  onChanged: (String escolha){  
    print("Resultado: " + escolha);  
    setState(() {  
      _selecionado = escolha;  
    });  
  })  
), // Radio
```

```
-Radio(  
  value: "f",  
  groupValue: _selecionado,  
  onChanged: (String escolha){  
    print("Resultado: " + escolha);  
    setState(() {  
      _selecionado = escolha;  
    });  
  })  
, // Radio
```

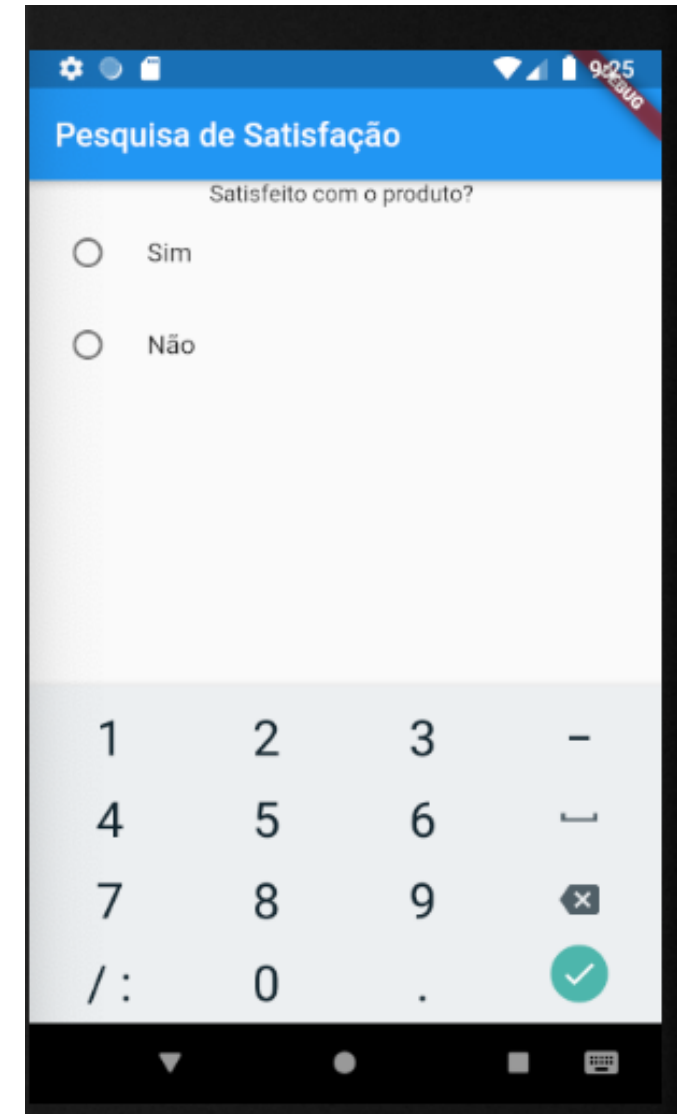


```
Console ⚡ 🔁 🌐  
↑  
↓  
⌕  
⌕  
⌕  
Performing hot restart...  
Syncing files to device Android SDK built for x86...  
Restarted application in 2.662ms.  
I/flutter ( 5313): Resultado: m  
I/flutter ( 5313): Resultado: f
```

# RadioButton - RadioListTile

- Podemos criar botões Radio usando também o **RadioListTile**, e o funcionamento é o mesmo, porém com mais configurações nele mesmo, além disso, a disposição dos itens é um debaixo do outro

Tela:





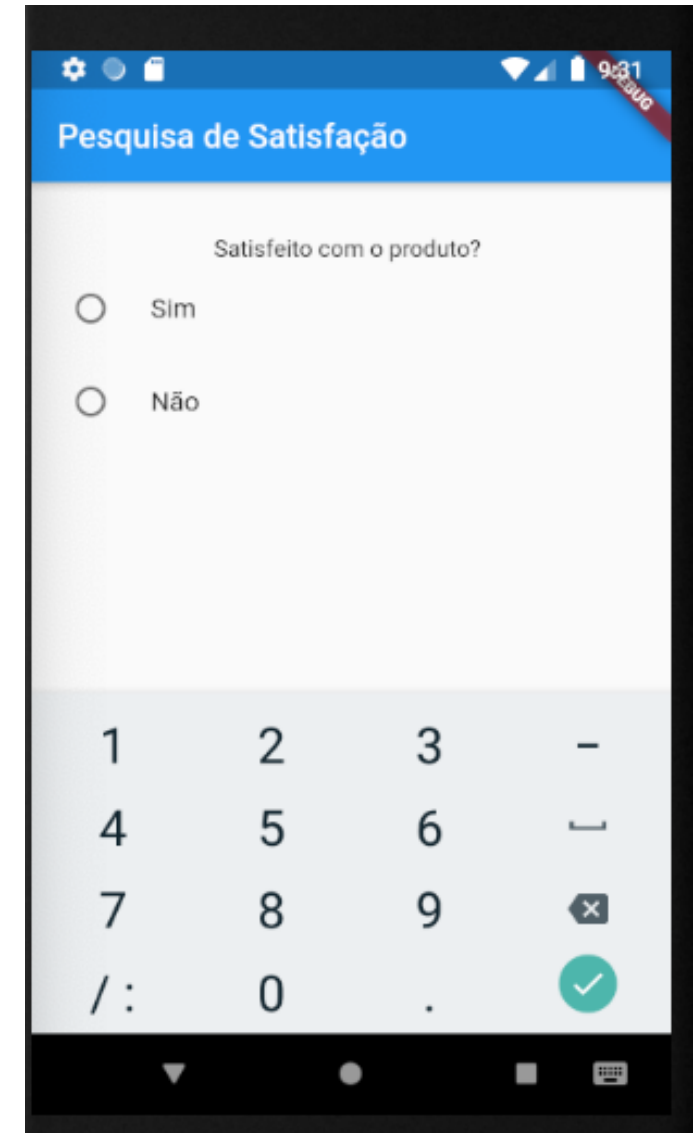
# RadioButton - RadioListTile

Tela:

Exemplo:

```
child: Column(  
  children: <Widget>[  
    Text("Satisfeito com o produto? "),  
    RadioListTile(  
      title: Text("Sim"),  
      value: "s",  
      groupValue: _selecionado,  
      onChanged: (String escolha){  
        setState(() {  
          _selecionado = escolha;  
        });  
      },  
    ), // RadioListTile  
    RadioListTile(  

```



# RadioButton - RadioListTile

Tela:

Exemplo:

```
ElevatedButton(  
  child: Text("Salvar"),  
  onPressed: () {  
    print("Item selecionado: " + _selecionado);  
  },  
  // ElevatedButton
```

```
Console  
↑  
↓  
Performing hot restart...  
Syncing files to device Android SDK built for x86...  
Restarted application in 2.089ms.  
I/flutter ( 5313): Item selecionado: sim
```

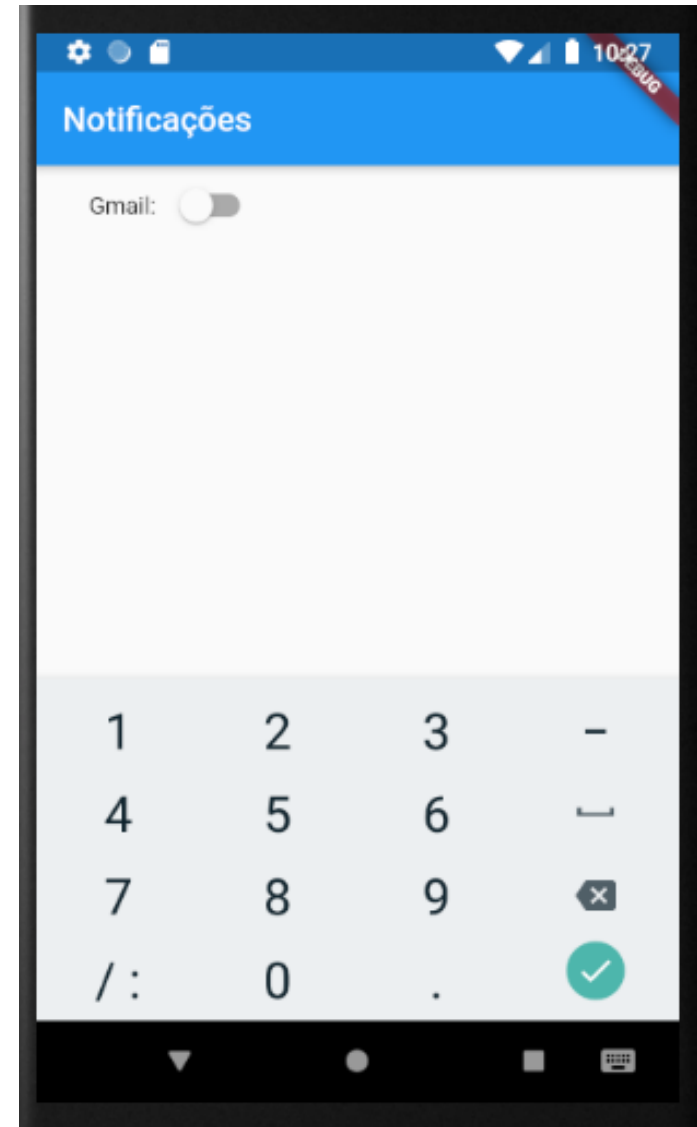
Selecione o item e  
clique no botão Salvar



# Switch

Tela:

- O uso do **Switch** é semelhante aos outros componentes de entrada de dados já vistos, sendo assim, será necessário inicializar o seu valor, além de realizar o controle de sua alteração com o **onChanged()**



# Switch

- Observe que é necessário usar o **setState()** para marcar a seleção do item como true ou false e na interface conseguirmos identificar o(s) item(s) selecionado(s)
- Além disso, a variável `_selecionado` foi declarada anteriormente e inicializada como false

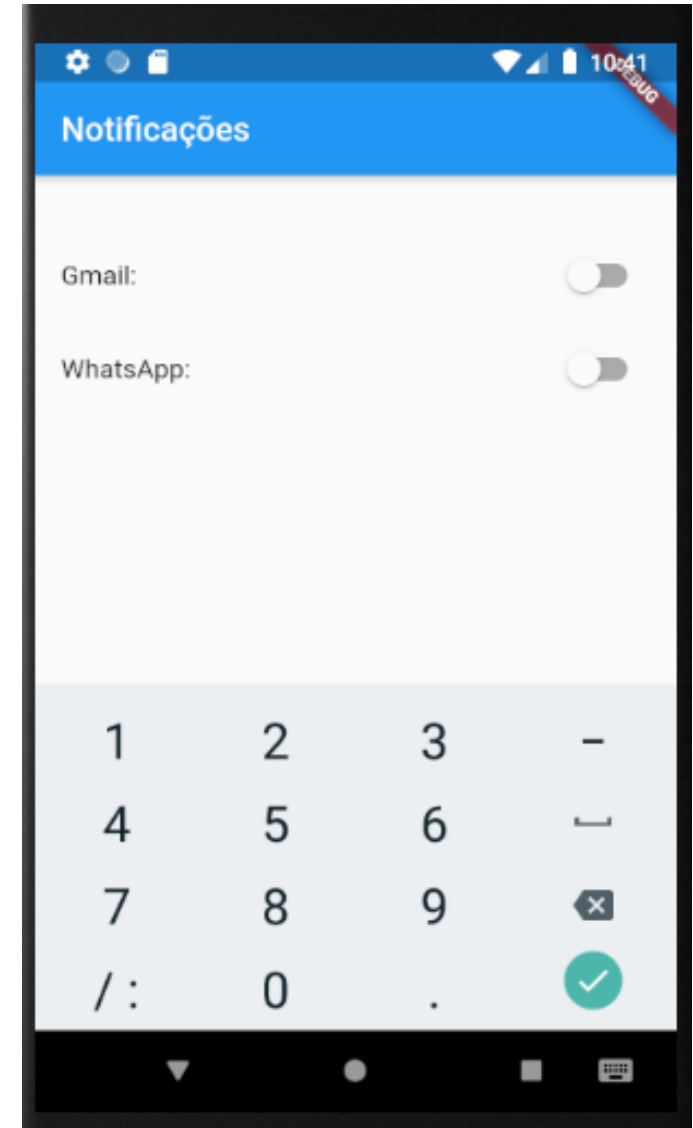
Exemplo:

```
child: Row(  
  children: <Widget>[  
    Padding(  
      padding: EdgeInsets.all(16),  
    ), // Padding  
    Text("Gmail: "),  
    Switch(  
      value: _selecionado,  
      onChanged: (bool valor){  
        setState(() {  
          _selecionado = valor;  
        });  
      }  
    ), // Switch  
  ], // <Widget>[]  
) , // Row
```

# Switch - SwitchListTile

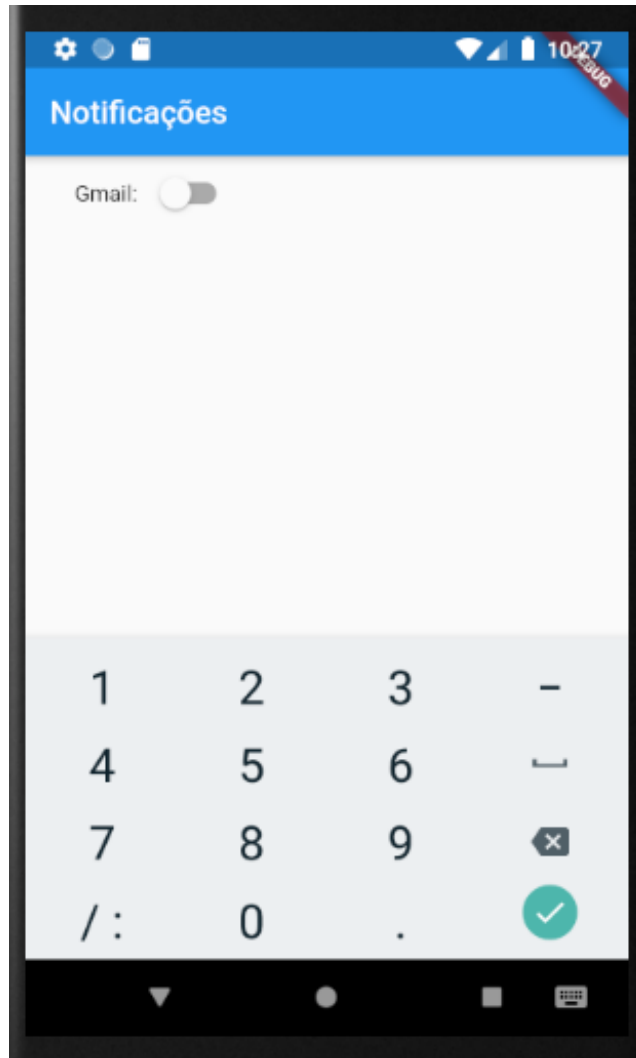
Tela:

- Também podemos incluir um Switch com o componente **SwitchListTile**, que o uso é semelhante aos outros componentes já vistos
- Como SwitchListTile é uma listagem de itens, deve ser incluído dentro de um Column

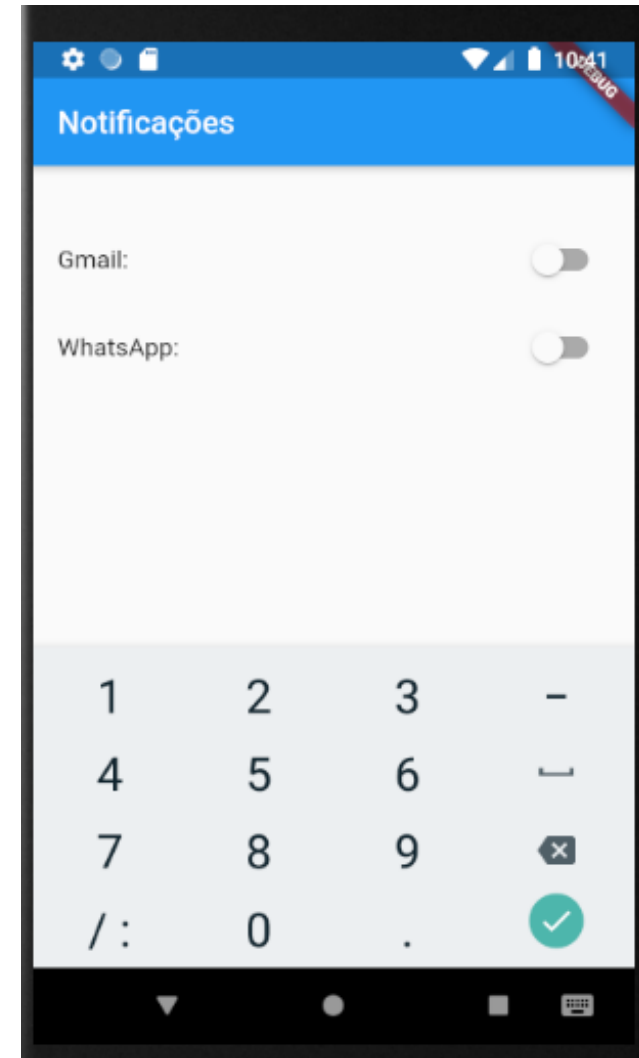


# Switch - Comparação

Tela do Switch():



Tela do SwitchListTile():



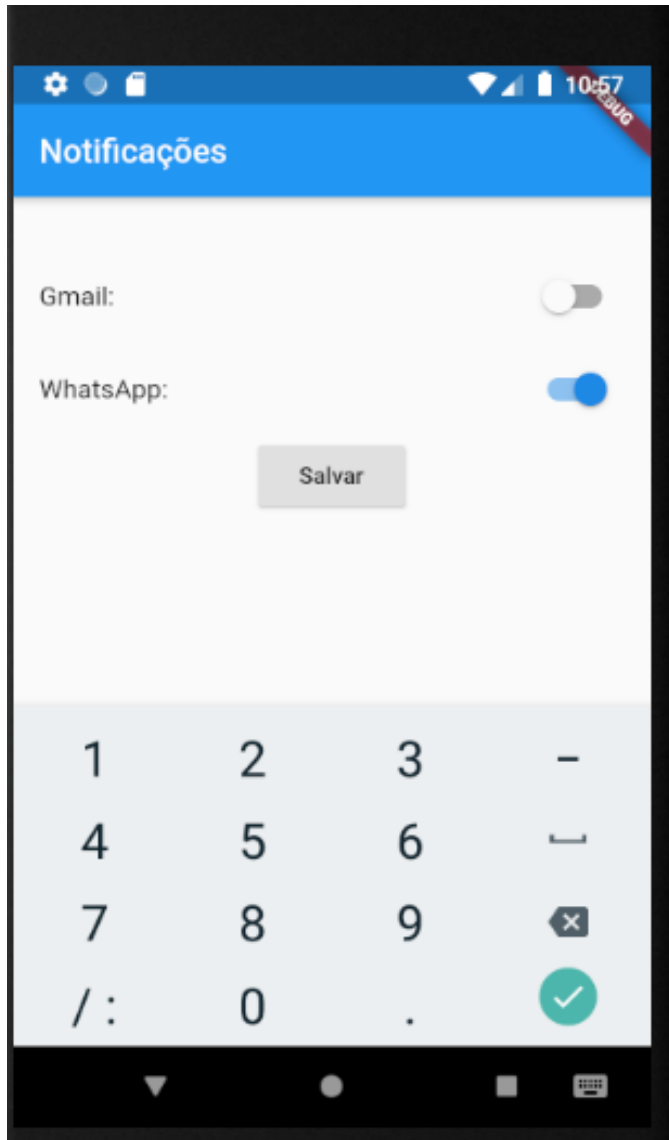
# Switch - SwitchListTile

Exemplo:

```
SwitchListTile(  
  title: Text("Gmail: "),  
  value: _gmail,  
  onChanged: (bool valor) {  
    setState(() {  
      _gmail = valor;  
    });  
  },  
), // SwitchListTile
```

```
SwitchListTile(  
  title: Text("WhatsApp: "),  
  value: _whatsapp,  
  onChanged: (bool valor) {  
    setState(() {  
      _whatsapp = valor;  
    });  
  },  
), // SwitchListTile
```

# Switch - SwitchListTile



```
ElevatedButton(  
  child: Text("Salvar"),  
  onPressed: () {  
    if (_gmail)  
      print("Escolha: ativar Gmail");  
    else  
      print("Escolha: desativar Gmail");  
    if (_whatsapp)  
      print("Escolha: ativar WhatsApp");  
    else  
      print("Escolha: desativar WhatsApp");  
  }  
) , // ElevatedButton
```

```
Console  
Performing hot restart...  
Syncing files to device Android SDK built for x86...  
Restarted application in 2.025ms.  
I/flutter ( 5313): Escolha: desativar Gmail  
I/flutter ( 5313): Escolha: ativar WhatsApp
```

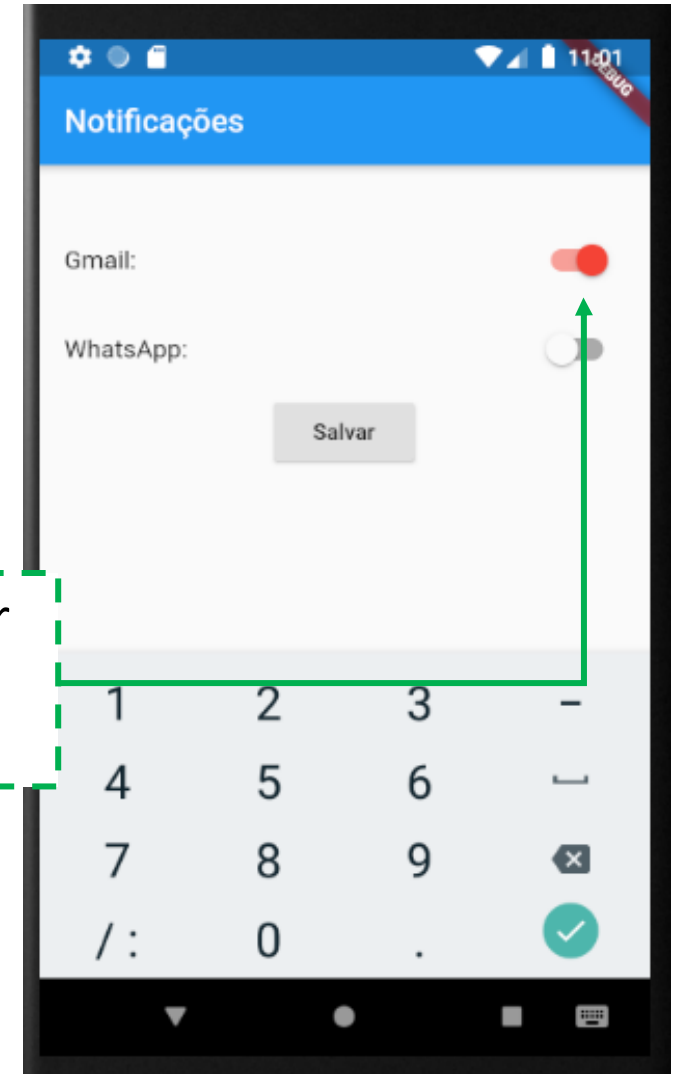


# Switch - SwitchListTile

Exemplo:

```
SwitchListTile(  
  activeColor: Colors.red,  
  title: Text("Gmail: "),  
  value: _gmail,  
  onChanged: (bool valor) {  
    setState(() {  
      _gmail = valor;  
    });  
  },  
), // SwitchListTile
```

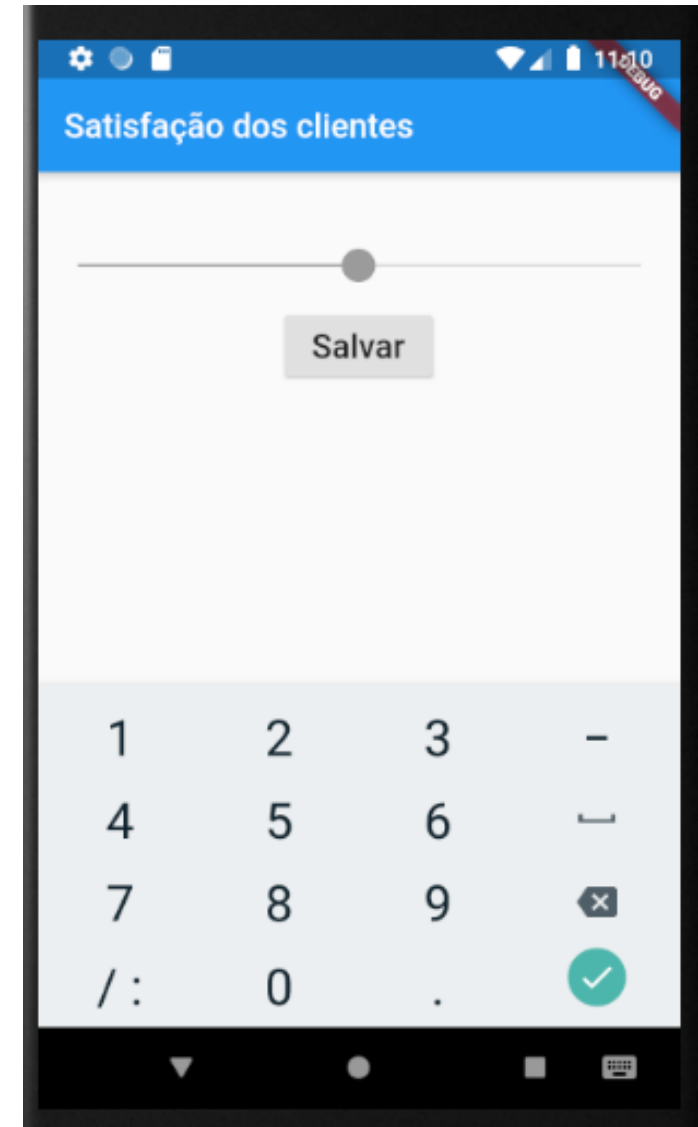
Ao selecionar, a cor  
é alterada para  
vermelho



# Slider

Tela:

- O **slider** permite ao usuário selecionar um valor dentro de um conjunto de valores contínuo, dado o valor mínimo e o valor máximo
- Usa valores reais, portanto, double
- Assim como os outros componentes possui como parâmetro o value e o onChanged



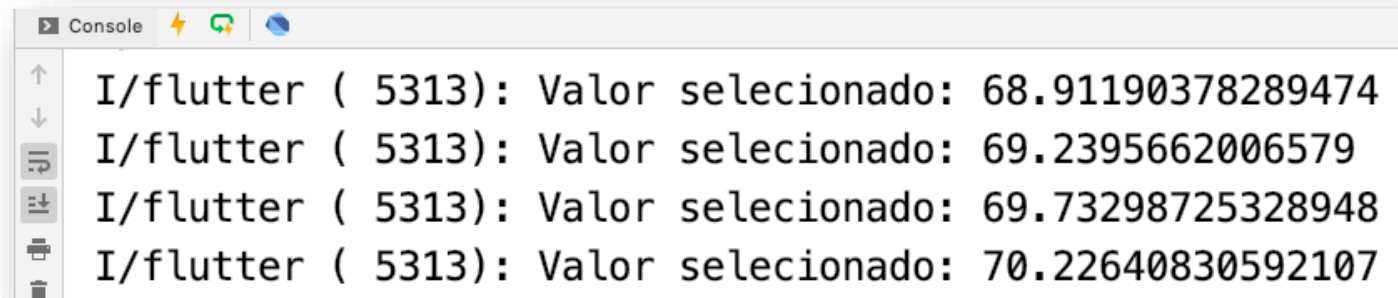
# Slider

## Exemplo:

```
Slider(  
  value: 50, //definir o valor inicial  
  min:0,  
  max:100,  
  onChanged: (double valor){  
    print("Valor selecionado: "+valor.toString());  
  },  
  // Slider
```

Value: define o valor inicial  
apresentado na tela  
Min: valor de início (mínimo)  
Max: valor final (máximo)

Observe que ao  
selecionar os  
valores são reais



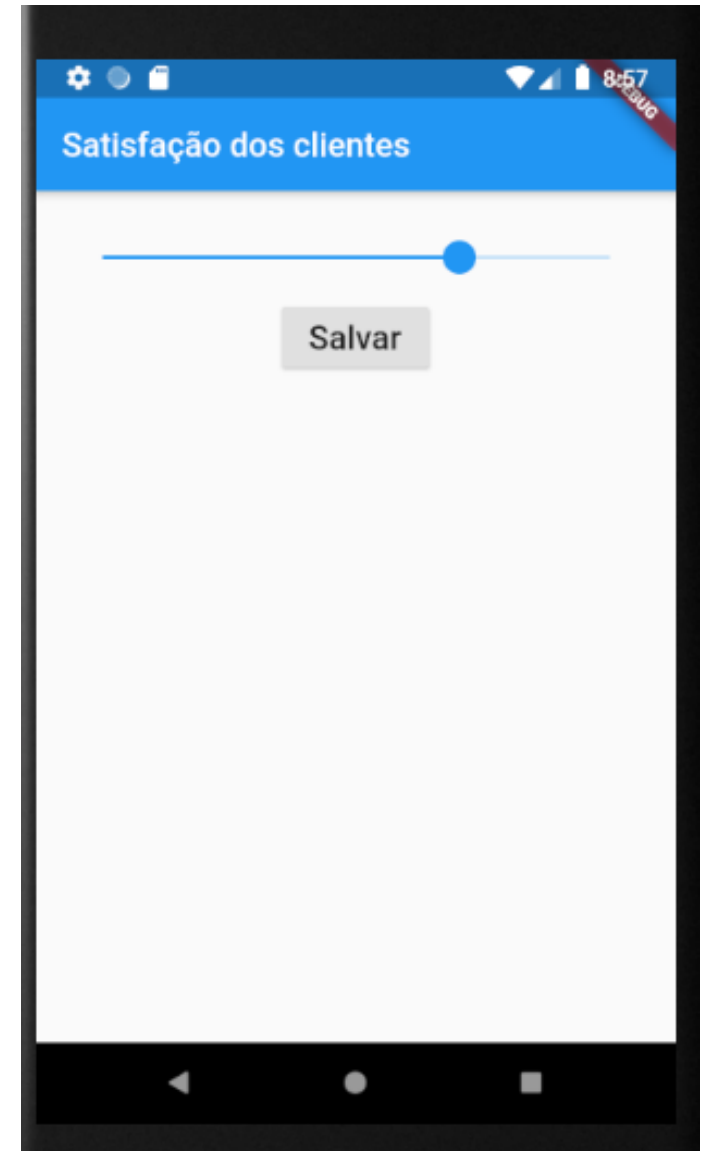
The screenshot shows a Flutter console window with the following log messages:

```
I/flutter ( 5313): Valor selecionado: 68.91190378289474  
I/flutter ( 5313): Valor selecionado: 69.2395662006579  
I/flutter ( 5313): Valor selecionado: 69.73298725328948  
I/flutter ( 5313): Valor selecionado: 70.22640830592107
```

# Slider

Tela:

- Para que o seletor mude é necessário fazer um controle do value
- É necessário criar um variável, inicializar ela com o valor de início e agora value recebe essa variável, além disso, usar o **setState()** para receber o valor selecionado



# Slider

Tela:

Exemplo:

```
Slider(  
  value: valor,  
  min:0,  
  max:100,  
  label: "seleção",  
  divisions: 10,  
  onChanged: (double novoValor){  
    setState(() {  
      valor = novoValor;  
    });  
    print("Valor selecionado: "+valor.toString());  
  }  
) , // Slider
```

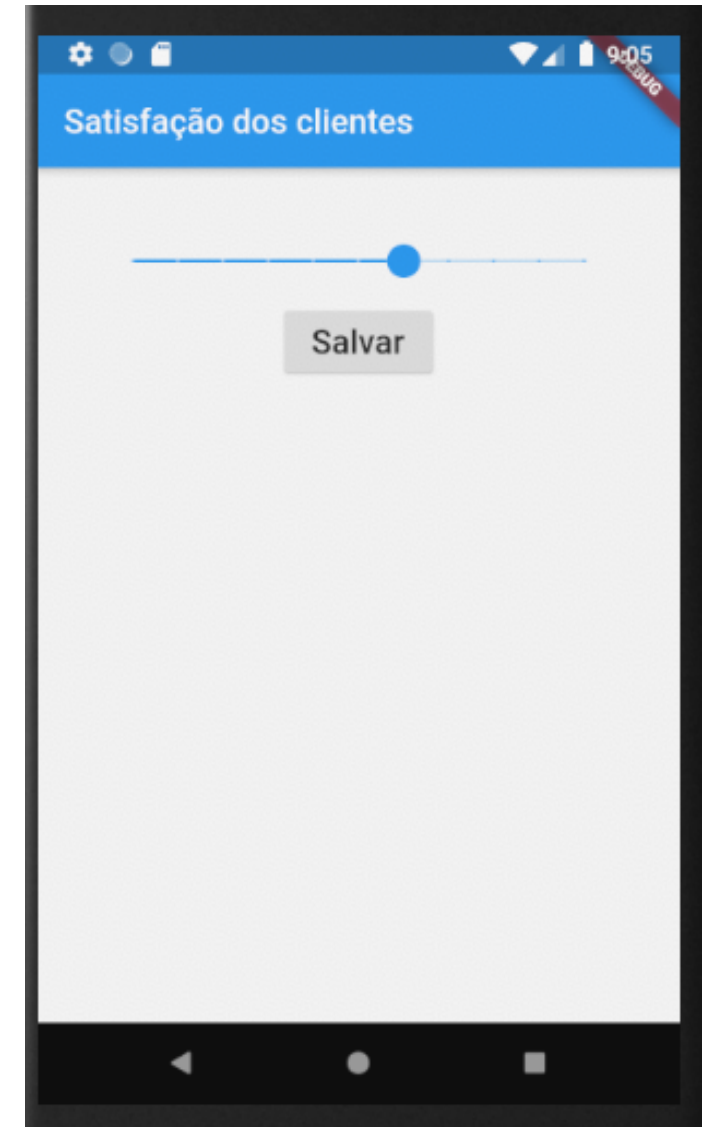


# Slider

Tela:

- Com uso do **divisions** só conseguimos selecionar os valores marcados, observe o console a seguir:

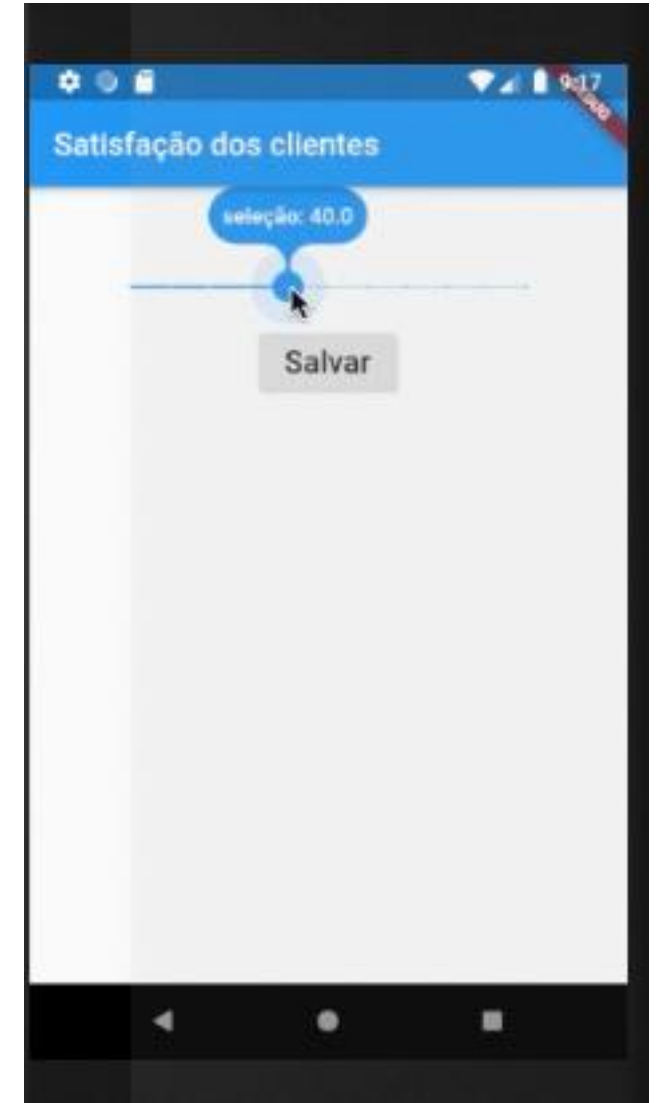
```
Console
identical 1 line
I/flutter ( 8047): Valor selecionado: 60.0
I/flutter ( 8047): Valor selecionado: 50.0
I/flutter ( 8047): Valor selecionado: 40.0
```



# Slider

Tela:

- Usando **labels dinâmicos** para apresentar o valor selecionado juntamente com um texto, para isso, será necessário a criação de uma variável e alterar o valor dentro de `setState()` e, nesse caso, podemos concatenar texto com valor



# Slider

Exemplo:

```
Slider(  
    value: valor, //definir o valor inicial  
    min:0,  
    max:100,  
    label: label, //label dinamico  
    divisions: 10, //define as divisoes entre o minimo e o maximo  
    onChanged: (double novoValor){  
        setState(() {  
            valor = novoValor;  
            label = "seleção: " + novoValor.toString();  
        });  
        print("Valor selecionado: "+valor.toString());  
    }  
), // Slider
```



# Slider

- Assim, como nos exemplos anteriores, usando outros componentes, ao clicar no botão **Salvar**, o valor selecionado será capturado e mostrado no console

Exemplo:

```
ElevatedButton(  
  child: Text("Salvar",  
    style: TextStyle(  
      fontSize: 20  
    ), // TextStyle  
  ), // Text  
  onPressed: () {  
    print("Valor salvo: "+valor.toString());  
  }  
), // ElevatedButton
```

# 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
  - <https://api.flutter.dev/flutter/material/CheckboxListTile-class.html>
  - <https://api.flutter.dev/flutter/material/Slider-class.html>