

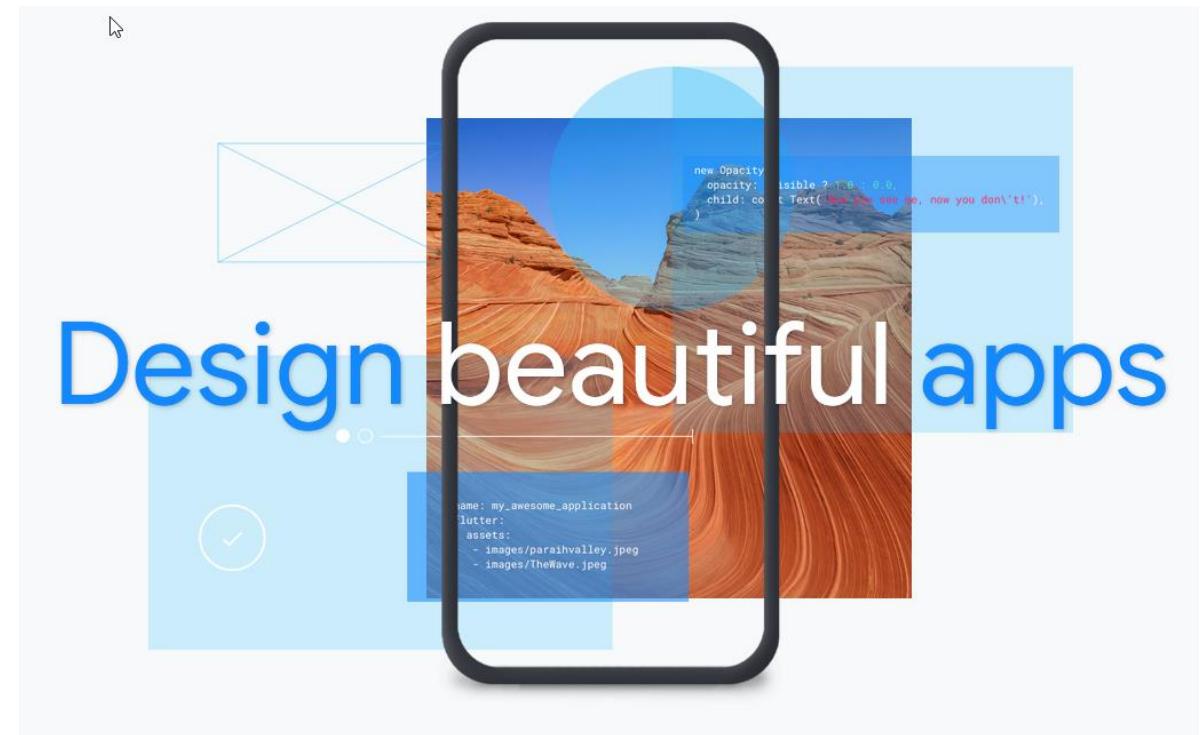


Transformando o futuro das pessoas
e as pessoas para o futuro.

#Senacfaz75



Desenvolvimento Mobile: Flutter



O que é o Flutter?

Flutter é um kit de desenvolvimento de interface de usuário (UI toolkit), de código aberto, criado pelo Google, que possibilita a criação de aplicativos compilados nativamente. Atualmente pode compilar para Android, iOS, Windows, Mac, Linux, Google *Fuchsia e Web.

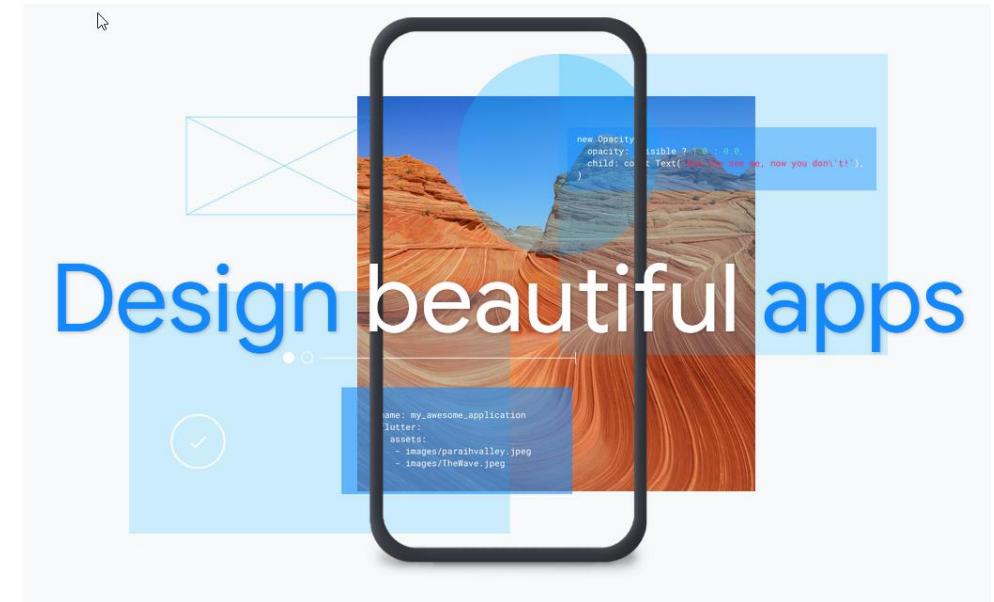
Características

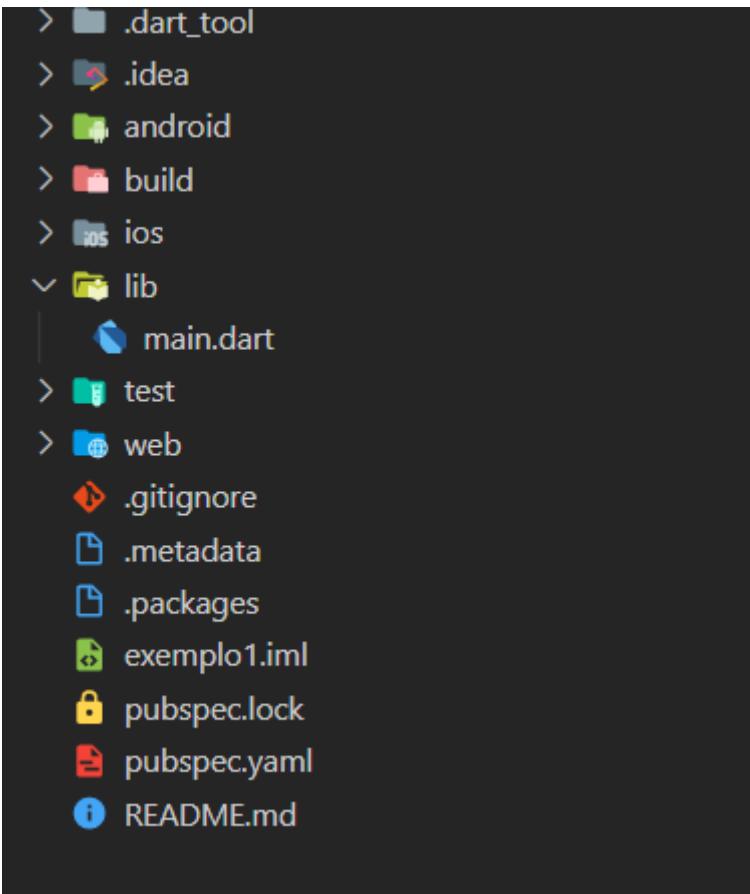
- tem como linguagem base o Dart
- É orientado a objetos
- Misto de Java, Javascript e C#
- Focada em Frontend
- Árvore de Widgets
- A tela inteira é um widget

Composição

SDK: Kit de desenvolvimento para compilar código fonte para código nativo

Widget: Componentes UI reutilizáveis desenvolvida do funções, classes e pacotes.





.dart_tool: Criado a partir de um pacote build runner, ficam os arquivos compilados da aplicação. (não mexer) está marcada pelo git como ignorada.

.idea: configuração do IntelliJ idea (não mexer).

Android: Todos arquivos relacionados a construção do projeto Android.

build: componentes de construção do projeto e compilados.

ios: pasta de projeto para o XCode.

lib: pasta onde programamos nosso projeto, criações de widgets, start do nosso projeto.

test: para que for trabalhar com testes, pode-se excluir essa pasta.

web: Pasta nova.

.gitignore: arquivos que serão ignorados no comit do git.

.metadatada: gerenciados pelo Flutter.

.packages: rotas de dependência.

.iml: criado para IntelliJ.

pubspec.lock e **pubspec.yaml:** Criados juntos para configuração do App, exemplo caminhos de assets, dependências.

Primeiro programa

Import: para importar bibliotecas e arquivos externos

Main(): Classe principal da aplicação Flutter

runApp(): Disparador do aplicativo

MaterialApp(): Classe de desenvolvimento visual do Flutter

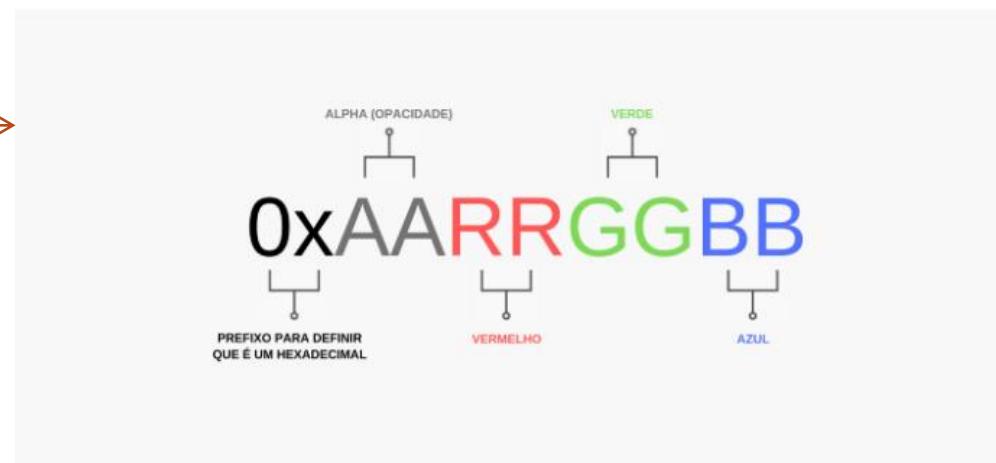
Title e home: Parâmetros nomeados de MaterialApp()

Container(): "Envelope" para um conteúdo da aplicação, lembra uma div.

Color(minúsculo): parâmetro nomeado para inserção de Cores

Colors(maiúsculo): Construtor para aplicação da cor

```
main01_primeiroPrograma.dart
lib > main01_primeiroProgramma.dart > ...
1 import 'package:flutter/material.dart';
2
3 main() => runApp(MaterialApp(
4   title: 'Sorteio da MegaSena',
5   home: Container(
6     color: Colors.white,
7   ), // Container
8 )); // MaterialApp
```



<https://api.flutter.dev/flutter/dart-ui/Color-class.html>

Inserindo Colunas

Import: para importar bibliotecas e arquivos externos

Main(): Classe principal da aplicação Flutter

runApp(): Disparador do aplicativo

MaterialApp(): Classe de desenvolvimento visual do Flutter

Title e home: Parâmetros nomeados de MaterialApp()

Container(): “Envelope” para um conteúdo da aplicação, lembra uma div.

Color(minúsculo): parâmetro nomeado para inserção de Cores

Colors(maiúsculo): Construtor para aplicação da cor

Child: Parâmetro filho de Container(), só recebe 1 widget

Column(): Widget para inserir colunas

Children: Array filho para vários Widget

Text(): Widget para inserir textos.

```
lib > main02_colunas.dart > ...
1 //Inserindo um Widget Coluna
2 import 'package:flutter/material.dart';
3
4 Run | Debug | Profile
5 main() => runApp(MaterialApp(
6   title: 'Sorteio da MegaSena',
7   home: Container(
8     color: Colors.grey,
9     child: Column(
10       //Uso um children para colocar vários widgets
11       children: [
12         Text('Texto 1'),
13         Text('Texto 2'),
14         Text('Texto 3')],
15     ), // Column
16   )), // Container // MaterialApp
```

Inserindo Linhas

Import: para importar bibliotecas e arquivos externos

Main(): Classe principal da aplicação Flutter

runApp(): Disparador do aplicativo

MaterialApp(): Classe de desenvolvimento visual do Flutter

Title e home: Parâmetros nomeados de MaterialApp()

Container(): “Envelope” para um conteúdo da aplicação, lembra uma div.

Color(minúsculo): parâmetro nomeado para inserção de Cores

Colors(maiúsculo): Construtor para aplicação da cor

Child: Parâmetro filho de Container(), só recebe 1 widget

Row(): Widget para inserir linhas

Children: Array filho para vários Widget

Text(): Widget para inserir textos.

```
lib > main03_linhas.dart > ...
1 //Inserindo um Widget Linhas
2 import 'package:flutter/material.dart';
3
4 Run | Debug | Profile
5 main() => runApp(MaterialApp(
6   title: 'Sorteio da MegaSena',
7   home: Container(
8     color: Colors.grey,
9     child: Row(
10       //Uso um children para colocar vários widgets
11       children: [
12         Text('Texto 1'),
13         Text('Texto 2'),
14         Text('Texto 3')],
15     ), // Row
16   )); // Container // MaterialApp
```

Formatando textos

- **style:** Definindo as formatações do Texto
- **fontSize:** Tamanho da fonte
- **fontStyle:** Estilo da fonte
- **fontWeight:** Peso da fonte (Negrito)
- **letterSpacing:** Espaçamento entre os caracteres
- **decoration:** sublinhados e tachados no texto
- **color:** Cor da fonte

```
lib > main04_FormatacaoTexto.dart > ...
1 //Formatação de texto
2 import 'package:flutter/material.dart';
3
4 Run | Debug | Profile
5 main() => runApp(MaterialApp(
6   title: 'Sorteio MegaSena',
7   //container é um envólucro para os conteúdos
8   home: Container(
9     color: Colors.white,
10
11   //Definindo um filho pra o container
12   child: Column(
13     //Definindo um children para o Column
14     children: [
15       Text(
16         'Esse é um exemplo de texto extenso para o widget',
17         style: TextStyle(
18           //Definindo as formatações do Texto
19           fontSize: 40,
20           fontStyle: FontStyle.normal,
21           fontWeight: FontWeight.normal,
22
23           //Espaçamento entre os caracteres
24           letterSpacing: 2,
25
26           //Retirar o sublinhado
27           decoration: TextDecoration.none,
28
29           //Aplicar na Fonte
30           color: Colors.black
31         ), // TextStyle
32       ],
33     ), // Column
34   ), // Container
35 )); // MaterialApp
```

Espaçamentos internos e externos (Padding e Margin)

padding: EdgeInsets

.all : Espaçamento para todos os lados (valor único)

.fromLTRB: Espaçamento para todos os lados (left, top, right, bottom)

.only: Espaçamento individual (left: 10, por exemplo)

margin: EdgeInsets

.all : Margem para todos os lados (valor único)

.fromLTRB: Margem para todos os lados (left, top, right, bottom)

.only: Espaçamento individual (left: 10, por exemplo)

BoxDecoration: Construtor para as bordas do box

Color: Aplicação de cores

Border: Definindo a borda

Width: Espessura

Color: Cor



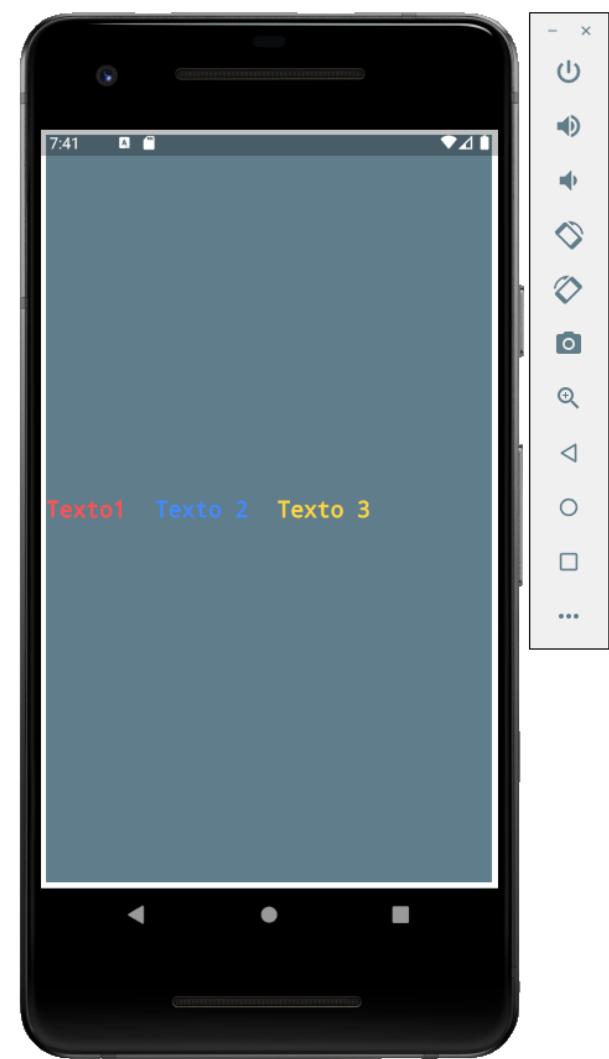
```
lib > main.dart > main
1 import 'package:flutter/material.dart';
Run | Debug | Profile
2 main() => runApp(MaterialApp(
3   //Removendo a faixa de debug
4   debugShowCheckedModeBanner: false,
5   title: 'Sorteio MegaSena',
6   home: Container(
7     //Defindo espaçamento internos
8     // .all .fromLTRB .only
9     //padding: EdgeInsets.all()
10
11    padding: EdgeInsets.fromLTRB(20, 50, 20, 0),
12    //Definindo os espaçamento externos
13    // .all .fromLTRB .only
14    //margin: EdgeInsets.all()
15
16    margin: EdgeInsets.fromLTRB(20, 50, 20, 20),
17    //Borda e cor do Container
18
19    decoration: BoxDecoration(
20      //Adiconando uma cor de background
21      color: Colors.blueGrey,
22
23      //Adicionando uma borda
24      border: Border.all(
25        width: 5,
26        color: Colors.amber
27      ), // Border.all
28    ), // BoxDecoration
29
```

```
30    //Adicionar um texto dentro do Container
31    child: Column(
32      children: [
33        Text(
34          'Texto exemplo para visualizar o preenchimento da tela do App',
35          textAlign: TextAlign.center,
36          style: TextStyle(
37            fontSize: 25,
38            fontWeight: FontWeight.normal,
39            decoration: TextDecoration.none,
40            color: Colors.white
41          ), // TextStyle
42        ) // Text
43      ],
44    ), // Column
45  ), // Container
46 ) // MaterialApp
47 );
```

Espaçamento no Widget

```
lib > main.dart > main
1 //Padding no Widget
2 import 'package:flutter/material.dart';
3
4 Run | Debug | Profile
5 main() => runApp(MaterialApp(
6     //Removendo a faixa de debug
7     debugShowCheckedModeBanner: false,
8     title: 'Padding no Widget',
9     home: Container(
10         decoration: BoxDecoration(
11             color: Colors.blueGrey,
12             border: Border.all(width: 5, color: Colors.white)), // BoxDecoration
13         child: Row(
14             children: [
15                 Text(
16                     'Texto1',
17                     style: TextStyle(
18                         fontSize: 20,
19                         decoration: TextDecoration.none,
20                         color: Colors.redAccent), // TextStyle
21                 ), // Text
22
23                 //Padding diretamente no widget
24                 Padding(→
25                     padding: EdgeInsets.all(25),
26                     child: Text(
27                         'Texto 2',
28                         style: TextStyle(
29                             fontSize: 20,
30                             decoration: TextDecoration.none,
31                             color: Colors.blueAccent
32
33                         ), // TextStyle
34                         ), // Text
35                 ), // Padding
36
37             ],
38         ), // Text
39     ], // Row
40     ), // Container
41 )); // MaterialApp
```

```
36
37         Text(
38             'Texto 3',
39             style: TextStyle(
40                 fontSize: 20,
41                 decoration: TextDecoration.none,
42                 color: Colors.amberAccent
43             ), // TextStyle
44         ), // Text
45     ], // Row
46     ), // Container
47 ); // MaterialApp
```



O construtor Padding() pode ser usado diretamente em outros widgets.

Há uma forma rápida para colocar um padding no widget, basta teclamos <CTRL> + <.›. Assim podemos envolver nosso widget em um Padding

Alinhamentos

Alinhamento Principal (mainAxisAlignment: MainAxisAlignment)

mainAxisAlignment: MainAxisAlignment.start: Alinhamento no início

mainAxisAlignment: MainAxisAlignment.end: Alinhamento no Fim

mainAxisAlignment: MainAxisAlignment.center: Alinhamento no centro

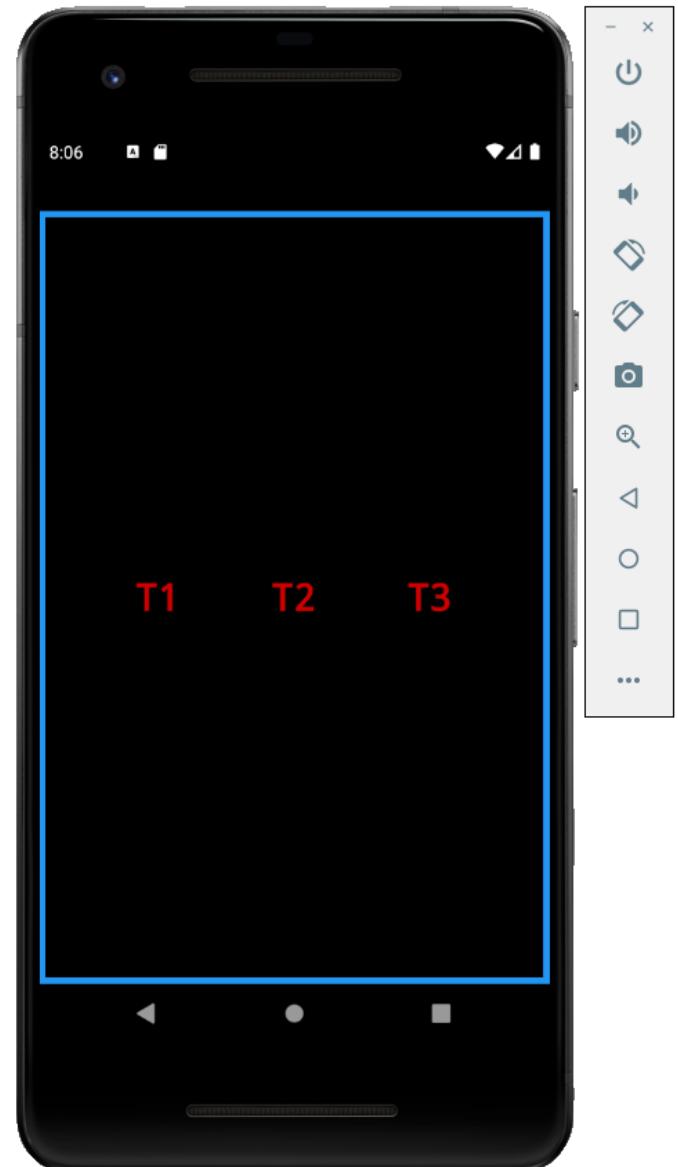
mainAxisAlignment: MainAxisAlignment.spaceEvenly: Alinhamento distribuído

Rebatendo o alinhamento do Eixo Principal (crossAxisAlignment)

crossAxisAlignment: CrossAxisAlignment.start: Alinhamento no início

crossAxisAlignment: CrossAxisAlignment.end: Alinhamento no Fim

crossAxisAlignment: CrossAxisAlignment.center: Alinhamento no centro



Alinhamentos

```
lib > main.dart > main
1 //Alinhamentos
2
3 import 'package:flutter/material.dart';
4
5 Run | Debug | Profile
6 main() => runApp(MaterialApp(
7   //removendo a faixa de debug
8   debugShowCheckedModeBanner: false,
9
10  title: 'Alinhamentos',
11  home: Container(
12    //definindo margens
13    margin: EdgeInsets.only(top: 60),
14
15    //defindo borda e cor do container
16    decoration: BoxDecoration(
17      border: Border.all(
18        width: 5,
19        color: Colors.blue
20      ) // Border.all
21    ), // BoxDecoration
22
23    //Criando um child
24    child: Row(
25      children: [
26
27        //Criando textos
28        Text(
29          'T1',
30          style: TextStyle(
31            decoration: TextDecoration.none,
32            fontSize: 30
33          ), // TextStyle
34        ), // Text
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
```

```
//Espaçando internamente o T2
Text(
  'T2',
  style: TextStyle(
    decoration: TextDecoration.none,
    fontSize: 30
  ), // TextStyle
), // Text

Text(
  'T3',
  style: TextStyle(
    decoration: TextDecoration.none,
    fontSize: 30
  ), // TextStyle
) // Text
],
//Alinhamento principal
//mainAxisAlignment: MainAxisAlignment.start .center .end .spaceEvenly
mainAxisAlignment: MainAxisAlignment.start,
//Configurar o eixo que cruza o alinhamento principal
//crossAxisAlignment: CrossAxisAlignment.start .end .center
crossAxisAlignment: CrossAxisAlignment.start,
), // Row
) // Container
)); // MaterialApp

```

Botões

TextButton():

Use botões de texto em barras de ferramentas, em diálogos ou em linha com outros conteúdos. Os botões de texto não possuem bordas visíveis e, portanto, devem confiar em sua posição em relação a outros conteúdos para contexto. Evite usar botões de texto onde eles se misturariam com outros conteúdos, por exemplo, no meio de listas.

ElevatedButton()

Use botões elevados para adicionar dimensão a layouts com botões, por exemplo, em longas listas de conteúdo ocupados ou em espaços amplos. Evite usar botões elevados em conteúdo já elevado, como diálogos ou cards.

OutlinedButton()

Os OutlinedButton são botões de ênfase média. Eles contêm ações que são importantes, mas não são a ação principal em um aplicativo.

```
lib > main08_buttons.dart > main
1   //Trabalhando com botões
2
3   import 'package:flutter/material.dart';
4
5   Run | Debug | Profile
6   main() => runApp(MaterialApp(
7       title: 'Trabalhando com botões',
8       home: Container(
9           //Margens
10          margin: EdgeInsets.only(top: 60),
11
12          //Espaçamento
13          padding: EdgeInsets.all(20),
14          color: Colors.blueGrey,
15
16          child: Column(
17              //Espaçamento principal
18              mainAxisAlignment: MainAxisAlignment.center,
19
20              children: [
21                  TextButton(
22                      style: TextButton.styleFrom(
23                          //borda arredondada
24                          shape: RoundedRectangleBorder(
25                              //controlar o arredondamento
26                              borderRadius: BorderRadius.circular(10)
27                          ), // RoundedRectangleBorder
28                          //cor de foreground
29                          primary: Colors.pinkAccent,
30
31                          //cor de background
32                          backgroundColor: Colors.white,
33
34                          //Espaçamento dentro do botão
35                          padding:
36                              EdgeInsets.symmetric(
37                                  horizontal: 32,
38                                  vertical: 32) // EdgeInsets.symmetric
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
onPressed: () {
    print('Executando...');
},
child: Text(
    'Meu Botão',
    style: TextStyle(
        fontSize: 20
    ), // TextStyle
) // Text
), // TextButton
//Forçar um espaçamento
const SizedBox(height: 20),
ElevatedButton(
style: ElevatedButton.styleFrom(
//cor Background
primary: Colors.amber,
//cor foreground
onPrimary: Colors.white,
),
onPressed: (){},
child: Text(
    'Botão 2',
    style: TextStyle(
        fontSize: 20
    ), // TextStyle
) // Text
), // ElevatedButton
const SizedBox(height: 20),
```

Botões

TextButton():

Use botões de texto em barras de ferramentas, em diálogos ou em linha com outros conteúdos. Os botões de texto não possuem bordas visíveis e, portanto, devem confiar em sua posição em relação a outros conteúdos para contexto. Evite usar botões de texto onde eles se misturariam com outros conteúdos, por exemplo, no meio de listas.

ElevatedButton()

Use botões elevados para adicionar dimensão a layouts com botões, por exemplo, em longas listas de conteúdo ocupados ou em espaços amplos. Evite usar botões elevados em conteúdo já elevado, como diálogos ou cards.

OutlinedButton()

Os OutlinedButton são botões de ênfase média. Eles contêm ações que são importantes, mas não são a ação principal em um aplicativo.

```
69          //Botão outline
70          OutlinedButton(
71              style: OutlinedButton.styleFrom(
72                  //Cor do foreground
73                  primary: Colors.black,
74
75                  //Cor da borda
76                  side: BorderSide(
77                      width: 1,
78                      color: Colors.orange
79                  ) // BorderSide
80
81              ),
82
83              onPressed: (){},
84              child: Text(
85                  'Botão 3',
86                  style: TextStyle(
87                      fontSize: 30
88                  ), // TextStyle
89                  ), // Text
90              ), // OutlinedButton
91
92              const SizedBox(height: 20),
93              //Botão com ícone
94              ElevatedButton.icon(
95                  style: ElevatedButton.styleFrom(
96                      side: BorderSide(
97                          width: 1,
98                          color: Colors.amber
99                      ), // BorderSide
100                     primary: Colors.red,
101                     onPrimary: Colors.amber
102                 ),
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129 ); // MaterialApp
```

```
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129 ); // MaterialApp
```

Botões

TextButton():

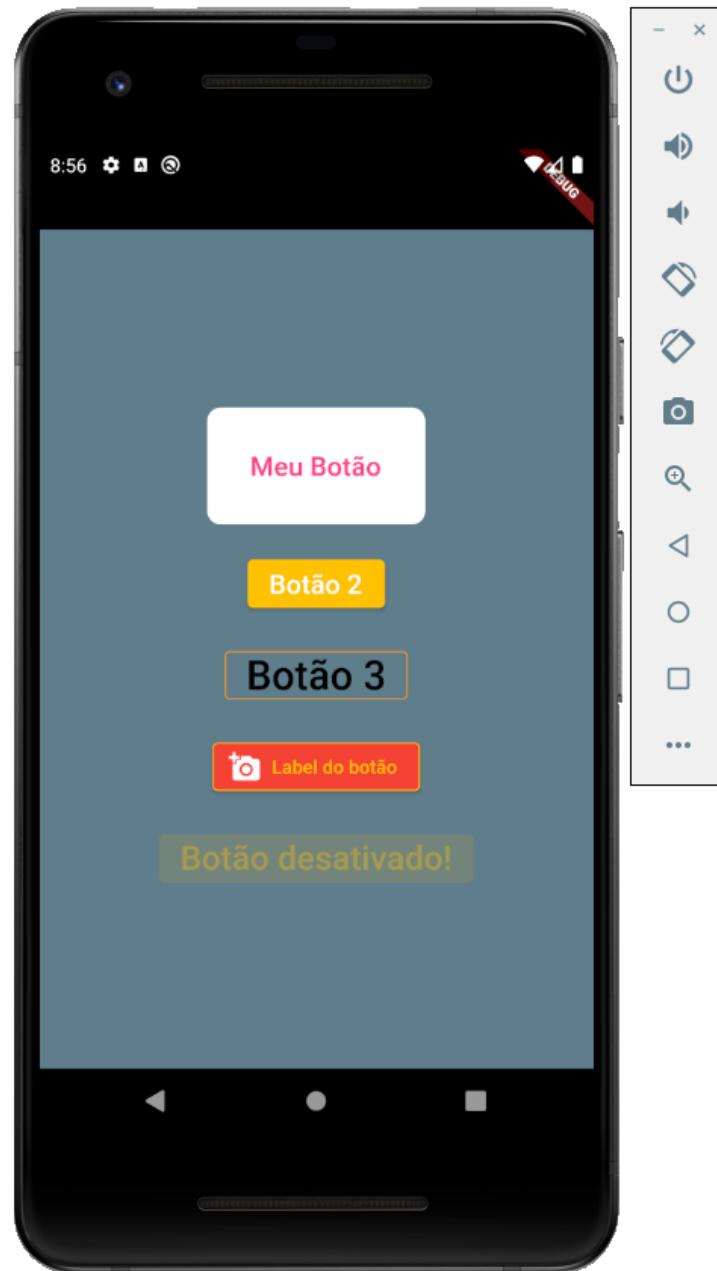
Use botões de texto em barras de ferramentas, em diálogos ou em linha com outros conteúdos. Os botões de texto não possuem bordas visíveis e, portanto, devem confiar em sua posição em relação a outros conteúdos para contexto. Evite usar botões de texto onde eles se misturariam com outros conteúdos, por exemplo, no meio de listas.

ElevatedButton()

Use botões elevados para adicionar dimensão a layouts com botões, por exemplo, em longas listas de conteúdo ocupados ou em espaços amplos. Evite usar botões elevados em conteúdo já elevado, como diálogos ou cards.

OutlinedButton()

Os OutlinedButton são botões de ênfase média. Eles contêm ações que são importantes, mas não são a ação principal em um aplicativo.



Imagens

Os aplicativos Flutter podem incluir código e assets (às vezes chamados de recursos). Um asset é um arquivo que é empacotado e implantado com seu aplicativo e pode ser acessado no tempo de execução. Os tipos comuns de assets incluem dados estáticos (por exemplo, arquivos JSON), arquivos de configuração, ícones e imagens (JPEG, WebP, GIF, WebP / GIF animado, PNG, BMP e WBMP).

Especificando Assets

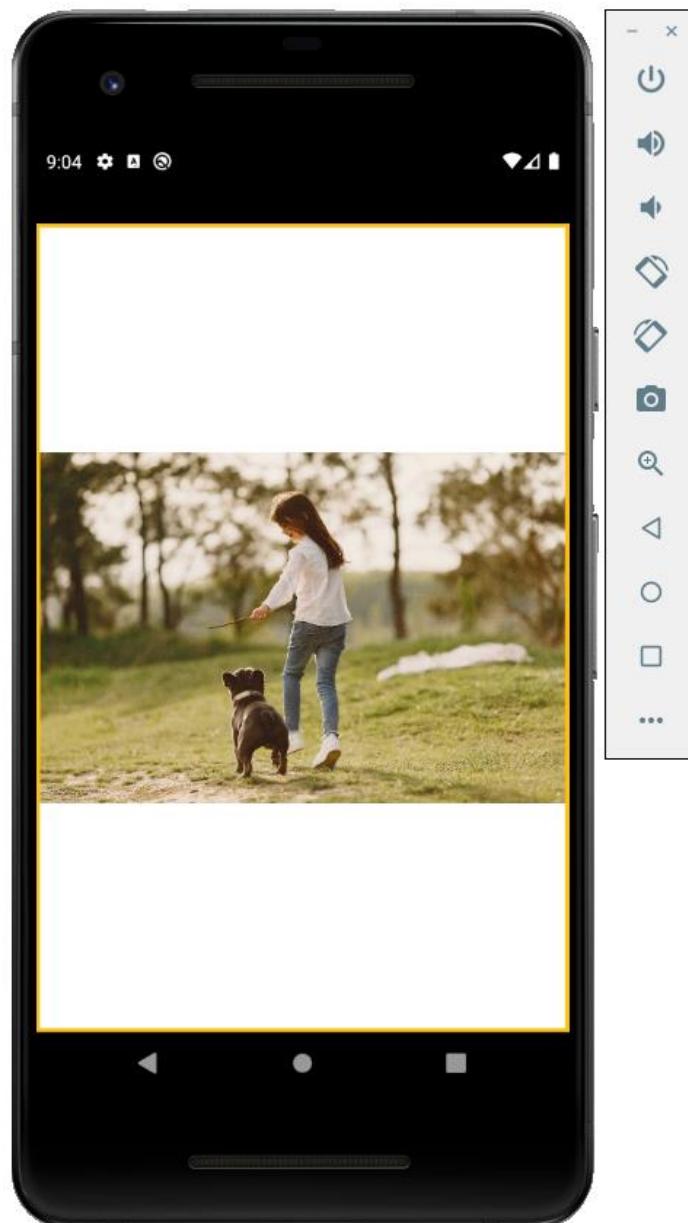
O Flutter usa o arquivo `pubspec.yaml`, localizado na raiz do seu projeto, para identificar os assets exigidos por um aplicativo.

```
! pubspec.yaml
46
47 # To add assets to your application, add an assets section, like this:
48 # assets:
49 #   - images/a_dot_burr.jpeg
50 #   - images/a_dot_ham.jpeg
51 assets:
52   - images/passeio.jpg
53
54 # An image asset can refer to one or more resolution-specific "variants", see
55 # https://flutter.dev/assets-and-images/#resolution-aware.
56
57 # For details regarding adding assets from package dependencies, see
58 # https://flutter.dev/assets-and-images/#from-packages
```

```
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 main() => runApp(MaterialApp(
5   debugShowCheckedModeBanner: false,
6   title: 'Trabalhando com imagens',
7   home: Container(
8     margin: EdgeInsets.only(top: 60),
9
10    decoration: BoxDecoration(
11      //background
12      color: Colors.white,
13
14      //definir bordas
15      border: Border.all(
16        width: 3,
17        color: Colors.amber
18      ) // Border.all
19    ), // BoxDecoration
20
21    //Um child para o widget Image
22    child: Image.asset(
23      //Caminho da imagem
24      'images/passeio.jpg',
25      //Definição de preenchimento da imagem
26      //cobra todo o espaçamento cortanto a imagem
27      //fit: BoxFit.cover,
28      //fit: BoxFit.contain //valor padrão
29      //fit: BoxFit.fill //Preenche com distorção
30      //fit: BoxFit.fitHeight, //preencher pela altura (Largura na proporção)
31      //fit: BoxFit.fitWidth, //preencher pela largura (altura na proporção)
32      //fit: BoxFit.none //Mantem a imagem original
33      fit: BoxFit.contain //Mantem a imagem original
34    ), // Image.asset
35  ), // Container
36)); // MaterialApp
```

Imagens

```
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MaterialApp(
4   debugShowCheckedModeBanner: false,
5   title: 'Trabalhando com imagens',
6   home: Container(
7     margin: EdgeInsets.only(top: 60),
8
9     decoration: BoxDecoration(
10       //background
11       color: Colors.white,
12
13       //definir bordas
14       border: Border.all(
15         width: 3,
16         color: Colors.amber
17       ) // Border.all
18     ), // BoxDecoration
19
20     //Um child para o widget Image
21     child: Image.asset(
22       //Caminho da imagem
23       'images/passeio.jpg',
24       //Definição de preenchimento da imagem
25       //cobra todo o espaçoamento cortanto a imagem
26       //fit: BoxFit.cover,
27       //fit: BoxFit.contain //valor padrão
28       //fit: BoxFit.fill //Preenche com distorção
29       //fit: BoxFit.fitHeight, //preencher pela altura (Largura na proporção)
30       //fit: BoxFit.fitWidth, //preencher pela largura (altura na proporção)
31       //fit: BoxFit.none //Mantem a imagem original
32       fit: BoxFit.contain //Mantem a imagem original
33     ), // Image.asset
34   ), // Container
35 )); // MaterialApp
```



Scaffold

Scaffold é uma classe de flutter que fornece muitos widgets ou podemos dizer APIs como Drawer, Snackbar, BottomNavigationBar, FloatingActionButton, AppBar etc. Scaffold irá expandir ou ocupar toda a tela do dispositivo. ... O Scaffold fornecerá uma estrutura para implementar o layout de Material Design básico do aplicativo.

```
lib > main10_Scaffold.dart > ...
1 import 'package:flutter/material.dart';
Run | Debug | Profile
2 main() => runApp(MaterialApp(
3   debugShowCheckedModeBanner: false,
4   home: Scaffold(
5     //Criando uma estrutura
6     //Divisão de 3 áreas: Título, corpo e rodapé
7     appBar: AppBar(
8       //Título
9       title: Text('Minha aplicação'),
10      backgroundColor: Colors.amber,
11    ), // AppBar
12    body: Padding(
13      padding: EdgeInsets.all(50),
14      child: Text(
15        'Conteúdo do App',
16        style: TextStyle(fontSize: 20),
17      ), // Text
18    ), // Padding
19    bottomNavigationBar: BottomAppBar(
20      //Aplicando uma cor
21      color: Colors.lightBlue,
22      child: Padding(
23        padding: EdgeInsets.all(20),
24        child: Row(
25          //Array de widgets
26          children: [
27            Text('Texto 1 bottom'),
28            Padding(
29              padding: const EdgeInsets.all(8.0),
30              child: Text(
31                'Texto 2 bottom',
32                style: TextStyle(fontSize: 20, color: Colors.white),
33              ), // Text
34            ), // Padding
35            Text('Texto 3 bottom')
36          ],
37        ), // Row
38      ), // Padding
39    ), // BottomAppBar
40  ), // Scaffold
41)); // MaterialApp
```



Entrada de Dados

Um campo de texto permite que o usuário insira texto, com teclado de hardware ou com teclado na tela.

O campo de texto chama o retorno de chamada `onChanged` sempre que o usuário altera o texto no campo. Se o usuário indicar que acabou de digitar no campo (por exemplo, pressionando um botão no teclado virtual), o campo de texto chama o retorno de chamada `onSubmitted` .

Para controlar o texto que é exibido no campo de texto, use o controlador. Por exemplo, para definir o valor inicial do campo de texto, use um controlador que já contenha algum texto. O controlador também pode controlar a seleção e a região de composição (e observar as mudanças no texto, seleção e região de composição).

```
lib > main.dart > ...
1 //Entrada de dados
2
3 import 'package:flutter/material.dart';
4 import 'package:flutter/services.dart';
5
6 Run | Debug | Profile
7 main() => runApp(MaterialApp(
8   home: Entrada(),
9 ));
10 //Criando um estado
11 class Entrada extends StatefulWidget {
12   @override
13   _EntradaState createState() => _EntradaState();
14 }
15
16 class _EntradaState extends State<Entrada> {
17   //Iniciar um controlador
18   TextEditingController _textEditingController = TextEditingController();
19
20   //Uma variável de saída na tela
21   String _resposta = 'Resultado';
22
23   @override
24   Widget build(BuildContext context) {
25     return Scaffold(
26       appBar: AppBar(
27         title: Text('Entrada de dados'),
28         backgroundColor: Colors.amber,
29       ), // AppBar
30       body: Column(
31         children: [
32           Padding(
33             padding: EdgeInsets.all(20),
34           ),
35         ],
36       ),
37     );
38   }
39
40   void _atualizarResposta() {
41     setState(() {
42       _resposta = _textEditingController.text;
43     });
44   }
45
46   void _limparResposta() {
47     setState(() {
48       _resposta = '';
49     });
50   }
51
52   void _deletarResposta() {
53     setState(() {
54       _resposta =
55       _resposta.substring(0, _resposta.length - 1);
56     });
57   }
58
59   void _adicionarResposta() {
60     setState(() {
61       _resposta += '!';
62     });
63   }
64
65   void _inverterResposta() {
66     setState(() {
67       _resposta =
68       _resposta
69       .split('')
70       .reversed
71       .join('');
72     });
73   }
74
75   void _limparTela() {
76     setState(() {
77       _resposta = '';
78     });
79   }
80
81   void _atualizarResposta(String novoValor) {
82     setState(() {
83       _resposta = novoValor;
84     });
85   }
86
87   void _limparTela() {
88     setState(() {
89       _resposta = '';
90     });
91   }
92
93   void _deletarResposta() {
94     setState(() {
95       _resposta =
96       _resposta
97       .substring(0, _resposta.length - 1);
98     });
99   }
100
101   void _adicionarResposta() {
102     setState(() {
103       _resposta += '!';
104     });
105   }
106
107   void _inverterResposta() {
108     setState(() {
109       _resposta =
110       _resposta
111       .split('')
112       .reversed
113       .join('');
114     });
115   }
116
117   void _limparTela() {
118     setState(() {
119       _resposta = '';
120     });
121   }
122
123   void _atualizarResposta(String novoValor) {
124     setState(() {
125       _resposta = novoValor;
126     });
127   }
128
129   void _limparTela() {
130     setState(() {
131       _resposta = '';
132     });
133   }
134
135   void _deletarResposta() {
136     setState(() {
137       _resposta =
138       _resposta
139       .substring(0, _resposta.length - 1);
140     });
141   }
142
143   void _adicionarResposta() {
144     setState(() {
145       _resposta += '!';
146     });
147   }
148
149   void _inverterResposta() {
150     setState(() {
151       _resposta =
152       _resposta
153       .split('')
154       .reversed
155       .join('');
156     });
157   }
158
159   void _limparTela() {
160     setState(() {
161       _resposta = '';
162     });
163   }
164
165   void _atualizarResposta(String novoValor) {
166     setState(() {
167       _resposta = novoValor;
168     });
169   }
170
171   void _limparTela() {
172     setState(() {
173       _resposta = '';
174     });
175   }
176
177   void _deletarResposta() {
178     setState(() {
179       _resposta =
180       _resposta
181       .substring(0, _resposta.length - 1);
182     });
183   }
184
185   void _adicionarResposta() {
186     setState(() {
187       _resposta += '!';
188     });
189   }
190
191   void _inverterResposta() {
192     setState(() {
193       _resposta =
194       _resposta
195       .split('')
196       .reversed
197       .join('');
198     });
199   }
200
201   void _limparTela() {
202     setState(() {
203       _resposta = '';
204     });
205   }
206
207   void _atualizarResposta(String novoValor) {
208     setState(() {
209       _resposta = novoValor;
210     });
211   }
212
213   void _limparTela() {
214     setState(() {
215       _resposta = '';
216     });
217   }
218
219   void _deletarResposta() {
220     setState(() {
221       _resposta =
222       _resposta
223       .substring(0, _resposta.length - 1);
224     });
225   }
226
227   void _adicionarResposta() {
228     setState(() {
229       _resposta += '!';
230     });
231   }
232
233   void _inverterResposta() {
234     setState(() {
235       _resposta =
236       _resposta
237       .split('')
238       .reversed
239       .join('');
240     });
241   }
242
243   void _limparTela() {
244     setState(() {
245       _resposta = '';
246     });
247   }
248
249   void _atualizarResposta(String novoValor) {
250     setState(() {
251       _resposta = novoValor;
252     });
253   }
254
255   void _limparTela() {
256     setState(() {
257       _resposta = '';
258     });
259   }
260
261   void _deletarResposta() {
262     setState(() {
263       _resposta =
264       _resposta
265       .substring(0, _resposta.length - 1);
266     });
267   }
268
269   void _adicionarResposta() {
270     setState(() {
271       _resposta += '!';
272     });
273   }
274
275   void _inverterResposta() {
276     setState(() {
277       _resposta =
278       _resposta
279       .split('')
280       .reversed
281       .join('');
282     });
283   }
284
285   void _limparTela() {
286     setState(() {
287       _resposta = '';
288     });
289   }
290
291   void _atualizarResposta(String novoValor) {
292     setState(() {
293       _resposta = novoValor;
294     });
295   }
296
297   void _limparTela() {
298     setState(() {
299       _resposta = '';
300     });
301   }
302
303   void _deletarResposta() {
304     setState(() {
305       _resposta =
306       _resposta
307       .substring(0, _resposta.length - 1);
308     });
309   }
310
311   void _adicionarResposta() {
312     setState(() {
313       _resposta += '!';
314     });
315   }
316
317   void _inverterResposta() {
318     setState(() {
319       _resposta =
320       _resposta
321       .split('')
322       .reversed
323       .join('');
324     });
325   }
326
327   void _limparTela() {
328     setState(() {
329       _resposta = '';
330     });
331   }
332
333   void _atualizarResposta(String novoValor) {
334     setState(() {
335       _resposta = novoValor;
336     });
337   }
338
339   void _limparTela() {
340     setState(() {
341       _resposta = '';
342     });
343   }
344
345   void _deletarResposta() {
346     setState(() {
347       _resposta =
348       _resposta
349       .substring(0, _resposta.length - 1);
350     });
351   }
352
353   void _adicionarResposta() {
354     setState(() {
355       _resposta += '!';
356     });
357   }
358
359   void _inverterResposta() {
360     setState(() {
361       _resposta =
362       _resposta
363       .split('')
364       .reversed
365       .join('');
366     });
367   }
368
369   void _limparTela() {
370     setState(() {
371       _resposta = '';
372     });
373   }
374
375   void _atualizarResposta(String novoValor) {
376     setState(() {
377       _resposta = novoValor;
378     });
379   }
380
381   void _limparTela() {
382     setState(() {
383       _resposta = '';
384     });
385   }
386
387   void _deletarResposta() {
388     setState(() {
389       _resposta =
390       _resposta
391       .substring(0, _resposta.length - 1);
392     });
393   }
394
395   void _adicionarResposta() {
396     setState(() {
397       _resposta += '!';
398     });
399   }
400
401   void _inverterResposta() {
402     setState(() {
403       _resposta =
404       _resposta
405       .split('')
406       .reversed
407       .join('');
408     });
409   }
410
411   void _limparTela() {
412     setState(() {
413       _resposta = '';
414     });
415   }
416
417   void _atualizarResposta(String novoValor) {
418     setState(() {
419       _resposta = novoValor;
420     });
421   }
422
423   void _limparTela() {
424     setState(() {
425       _resposta = '';
426     });
427   }
428
429   void _deletarResposta() {
430     setState(() {
431       _resposta =
432       _resposta
433       .substring(0, _resposta.length - 1);
434     });
435   }
436
437   void _adicionarResposta() {
438     setState(() {
439       _resposta += '!';
440     });
441   }
442
443   void _inverterResposta() {
444     setState(() {
445       _resposta =
446       _resposta
447       .split('')
448       .reversed
449       .join('');
450     });
451   }
452
453   void _limparTela() {
454     setState(() {
455       _resposta = '';
456     });
457   }
458
459   void _atualizarResposta(String novoValor) {
460     setState(() {
461       _resposta = novoValor;
462     });
463   }
464
465   void _limparTela() {
466     setState(() {
467       _resposta = '';
468     });
469   }
470
471   void _deletarResposta() {
472     setState(() {
473       _resposta =
474       _resposta
475       .substring(0, _resposta.length - 1);
476     });
477   }
478
479   void _adicionarResposta() {
480     setState(() {
481       _resposta += '!';
482     });
483   }
484
485   void _inverterResposta() {
486     setState(() {
487       _resposta =
488       _resposta
489       .split('')
490       .reversed
491       .join('');
492     });
493   }
494
495   void _limparTela() {
496     setState(() {
497       _resposta = '';
498     });
499   }
500
501   void _atualizarResposta(String novoValor) {
502     setState(() {
503       _resposta = novoValor;
504     });
505   }
506
507   void _limparTela() {
508     setState(() {
509       _resposta = '';
510     });
511   }
512
513   void _deletarResposta() {
514     setState(() {
515       _resposta =
516       _resposta
517       .substring(0, _resposta.length - 1);
518     });
519   }
520
521   void _adicionarResposta() {
522     setState(() {
523       _resposta += '!';
524     });
525   }
526
527   void _inverterResposta() {
528     setState(() {
529       _resposta =
530       _resposta
531       .split('')
532       .reversed
533       .join('');
534     });
535   }
536
537   void _limparTela() {
538     setState(() {
539       _resposta = '';
540     });
541   }
542
543   void _atualizarResposta(String novoValor) {
544     setState(() {
545       _resposta = novoValor;
546     });
547   }
548
549   void _limparTela() {
550     setState(() {
551       _resposta = '';
552     });
553   }
554
555   void _deletarResposta() {
556     setState(() {
557       _resposta =
558       _resposta
559       .substring(0, _resposta.length - 1);
560     });
561   }
562
563   void _adicionarResposta() {
564     setState(() {
565       _resposta += '!';
566     });
567   }
568
569   void _inverterResposta() {
570     setState(() {
571       _resposta =
572       _resposta
573       .split('')
574       .reversed
575       .join('');
576     });
577   }
578
579   void _limparTela() {
580     setState(() {
581       _resposta = '';
582     });
583   }
584
585   void _atualizarResposta(String novoValor) {
586     setState(() {
587       _resposta = novoValor;
588     });
589   }
590
591   void _limparTela() {
592     setState(() {
593       _resposta = '';
594     });
595   }
596
597   void _deletarResposta() {
598     setState(() {
599       _resposta =
600       _resposta
601       .substring(0, _resposta.length - 1);
602     });
603   }
604
605   void _adicionarResposta() {
606     setState(() {
607       _resposta += '!';
608     });
609   }
610
611   void _inverterResposta() {
612     setState(() {
613       _resposta =
614       _resposta
615       .split('')
616       .reversed
617       .join('');
618     });
619   }
620
621   void _limparTela() {
622     setState(() {
623       _resposta = '';
624     });
625   }
626
627   void _atualizarResposta(String novoValor) {
628     setState(() {
629       _resposta = novoValor;
630     });
631   }
632
633   void _limparTela() {
634     setState(() {
635       _resposta = '';
636     });
637   }
638
639   void _deletarResposta() {
640     setState(() {
641       _resposta =
642       _resposta
643       .substring(0, _resposta.length - 1);
644     });
645   }
646
647   void _adicionarResposta() {
648     setState(() {
649       _resposta += '!';
650     });
651   }
652
653   void _inverterResposta() {
654     setState(() {
655       _resposta =
656       _resposta
657       .split('')
658       .reversed
659       .join('');
660     });
661   }
662
663   void _limparTela() {
664     setState(() {
665       _resposta = '';
666     });
667   }
668
669   void _atualizarResposta(String novoValor) {
670     setState(() {
671       _resposta = novoValor;
672     });
673   }
674
675   void _limparTela() {
676     setState(() {
677       _resposta = '';
678     });
679   }
680
681   void _deletarResposta() {
682     setState(() {
683       _resposta =
684       _resposta
685       .substring(0, _resposta.length - 1);
686     });
687   }
688
689   void _adicionarResposta() {
690     setState(() {
691       _resposta += '!';
692     });
693   }
694
695   void _inverterResposta() {
696     setState(() {
697       _resposta =
698       _resposta
699       .split('')
700       .reversed
701       .join('');
702     });
703   }
704
705   void _limparTela() {
706     setState(() {
707       _resposta = '';
708     });
709   }
710
711   void _atualizarResposta(String novoValor) {
712     setState(() {
713       _resposta = novoValor;
714     });
715   }
716
717   void _limparTela() {
718     setState(() {
719       _resposta = '';
720     });
721   }
722
723   void _deletarResposta() {
724     setState(() {
725       _resposta =
726       _resposta
727       .substring(0, _resposta.length - 1);
728     });
729   }
730
731   void _adicionarResposta() {
732     setState(() {
733       _resposta += '!';
734     });
735   }
736
737   void _inverterResposta() {
738     setState(() {
739       _resposta =
740       _resposta
741       .split('')
742       .reversed
743       .join('');
744     });
745   }
746
747   void _limparTela() {
748     setState(() {
749       _resposta = '';
750     });
751   }
752
753   void _atualizarResposta(String novoValor) {
754     setState(() {
755       _resposta = novoValor;
756     });
757   }
758
759   void _limparTela() {
760     setState(() {
761       _resposta = '';
762     });
763   }
764
765   void _deletarResposta() {
766     setState(() {
767       _resposta =
768       _resposta
769       .substring(0, _resposta.length - 1);
770     });
771   }
772
773   void _adicionarResposta() {
774     setState(() {
775       _resposta += '!';
776     });
777   }
778
779   void _inverterResposta() {
780     setState(() {
781       _resposta =
782       _resposta
783       .split('')
784       .reversed
785       .join('');
786     });
787   }
788
789   void _limparTela() {
790     setState(() {
791       _resposta = '';
792     });
793   }
794
795   void _atualizarResposta(String novoValor) {
796     setState(() {
797       _resposta = novoValor;
798     });
799   }
800
801   void _limparTela() {
802     setState(() {
803       _resposta = '';
804     });
805   }
806
807   void _deletarResposta() {
808     setState(() {
809       _resposta =
810       _resposta
811       .substring(0, _resposta.length - 1);
812     });
813   }
814
815   void _adicionarResposta() {
816     setState(() {
817       _resposta += '!';
818     });
819   }
820
821   void _inverterResposta() {
822     setState(() {
823       _resposta =
824       _resposta
825       .split('')
826       .reversed
827       .join('');
828     });
829   }
830
831   void _limparTela() {
832     setState(() {
833       _resposta = '';
834     });
835   }
836
837   void _atualizarResposta(String novoValor) {
838     setState(() {
839       _resposta = novoValor;
840     });
841   }
842
843   void _limparTela() {
844     setState(() {
845       _resposta = '';
846     });
847   }
848
849   void _deletarResposta() {
850     setState(() {
851       _resposta =
852       _resposta
853       .substring(0, _resposta.length - 1);
854     });
855   }
856
857   void _adicionarResposta() {
858     setState(() {
859       _resposta += '!';
860     });
861   }
862
863   void _inverterResposta() {
864     setState(() {
865       _resposta =
866       _resposta
867       .split('')
868       .reversed
869       .join('');
870     });
871   }
872
873   void _limparTela() {
874     setState(() {
875       _resposta = '';
876     });
877   }
878
879   void _atualizarResposta(String novoValor) {
880     setState(() {
881       _resposta = novoValor;
882     });
883   }
884
885   void _limparTela() {
886     setState(() {
887       _resposta = '';
888     });
889   }
890
891   void _deletarResposta() {
892     setState(() {
893       _resposta =
894       _resposta
895       .substring(0, _resposta.length - 1);
896     });
897   }
898
899   void _adicionarResposta() {
900     setState(() {
901       _resposta += '!';
902     });
903   }
904
905   void _inverterResposta() {
906     setState(() {
907       _resposta =
908       _resposta
909       .split('')
910       .reversed
911       .join('');
912     });
913   }
914
915   void _limparTela() {
916     setState(() {
917       _resposta = '';
918     });
919   }
920
921   void _atualizarResposta(String novoValor) {
922     setState(() {
923       _resposta = novoValor;
924     });
925   }
926
927   void _limparTela() {
928     setState(() {
929       _resposta = '';
930     });
931   }
932
933   void _deletarResposta() {
934     setState(() {
935       _resposta =
936       _resposta
937       .substring(0, _resposta.length - 1);
938     });
939   }
940
941   void _adicionarResposta() {
942     setState(() {
943       _resposta += '!';
944     });
945   }
946
947   void _inverterResposta() {
948     setState(() {
949       _resposta =
950       _resposta
951       .split('')
952       .reversed
953       .join('');
954     });
955   }
956
957   void _limparTela() {
958     setState(() {
959       _resposta = '';
960     });
961   }
962
963   void _atualizarResposta(String novoValor) {
964     setState(() {
965       _resposta = novoValor;
966     });
967   }
968
969   void _limparTela() {
970     setState(() {
971       _resposta = '';
972     });
973   }
974
975   void _deletarResposta() {
976     setState(() {
977       _resposta =
978       _resposta
979       .substring(0, _resposta.length - 1);
980     });
981   }
982
983   void _adicionarResposta() {
984     setState(() {
985       _resposta += '!';
986     });
987   }
988
989   void _inverterResposta() {
990     setState(() {
991       _resposta =
992       _resposta
993       .split('')
994       .reversed
995       .join('');
996     });
997   }
998
999   void _limparTela() {
1000    setState(() {
1001      _resposta = '';
1002    });
1003  }
1004
1005  void _atualizarResposta(String novoValor) {
1006    setState(() {
1007      _resposta = novoValor;
1008    });
1009  }
1010
1011  void _limparTela() {
1012    setState(() {
1013      _resposta = '';
1014    });
1015  }
1016
1017  void _deletarResposta() {
1018    setState(() {
1019      _resposta =
1020      _resposta
1021      .substring(0, _resposta.length - 1);
1022    });
1023  }
1024
1025  void _adicionarResposta() {
1026    setState(() {
1027      _resposta += '!';
1028    });
1029  }
1030
1031  void _inverterResposta() {
1032    setState(() {
1033      _resposta =
1034      _resposta
1035      .split('')
1036      .reversed
1037      .join('');
1038    });
1039  }
1040
1041  void _limparTela() {
1042    setState(() {
1043      _resposta = '';
1044    });
1045  }
1046
1047  void _atualizarResposta(String novoValor) {
1048    setState(() {
1049      _resposta = novoValor;
1050    });
1051  }
1052
1053  void _limparTela() {
1054    setState(() {
1055      _resposta = '';
1056    });
1057  }
1058
1059  void _deletarResposta() {
1060    setState(() {
1061      _resposta =
1062      _resposta
1063      .substring(0, _resposta.length - 1);
1064    });
1065  }
1066
1067  void _adicionarResposta() {
1068    setState(() {
1069      _resposta += '!';
1070    });
1071  }
1072
1073  void _inverterResposta() {
1074    setState(() {
1075      _resposta =
1076      _resposta
1077      .split('')
1078      .reversed
1079      .join('');
1080    });
1081  }
1082
1083  void _limparTela() {
1084    setState(() {
1085      _resposta = '';
1086    });
1087  }
1088
1089  void _atualizarResposta(String novoValor) {
1090    setState(() {
1091      _resposta = novoValor;
1092    });
1093  }
1094
1095  void _limparTela() {
1096    setState(() {
1097      _resposta = '';
1098    });
1099  }
1100
1101  void _deletarResposta() {
1102    setState(() {
1103      _resposta =
1104      _resposta
1105      .substring(0, _resposta.length - 1);
1106    });
1107  }
1108
1109  void _adicionarResposta() {
1110    setState(() {
1111      _resposta += '!';
1112    });
1113  }
1114
1115  void _inverterResposta() {
1116    setState(() {
1117      _resposta =
1118      _resposta
1119      .split('')
1120      .reversed
1121      .join('');
1122    });
1123  }
1124
1125  void _limparTela() {
1126    setState(() {
1127      _resposta = '';
1128    });
1129  }
1130
1131  void _atualizarResposta(String novoValor) {
1132    setState(() {
1133      _resposta = novoValor;
1134    });
1135  }
1136
1137  void _limparTela() {
1138    setState(() {
1139      _resposta = '';
1140    });
1141  }
1142
1143  void _deletarResposta() {
1144    setState(() {
1145      _resposta =
1146      _resposta
1147      .substring(0, _resposta.length - 1);
1148    });
1149  }
1150
1151  void _adicionarResposta() {
1152    setState(() {
1153      _resposta += '!';
1154    });
1155  }
1156
1157  void _inverterResposta() {
1158    setState(() {
1159      _resposta =
1160      _resposta
1161      .split('')
1162      .reversed
1163      .join('');
1164    });
1165  }
1166
1167  void _limparTela() {
1168    setState(() {
1169      _resposta = '';
1170    });
1171  }
1172
1173  void _atualizarResposta(String novoValor) {
1174    setState(() {
1175      _resposta = novoValor;
1176    });
1177  }
1178
1179  void _limparTela() {
1180    setState(() {
1181      _resposta = '';
1182    });
1183  }
1184
1185  void _deletarResposta() {
1186    setState(() {
1187      _resposta =
1188      _resposta
1189      .substring(0, _resposta.length - 1);
1190    });
1191  }
1192
1193  void _adicionarResposta() {
1194    setState(() {
1195      _resposta += '!';
1196    });
1197  }
1198
1199  void _inverterResposta() {
1200    setState(() {
1201      _resposta =
1202      _resposta
1203      .split('')
1204      .reversed
1205      .join('');
1206    });
1207  }
1208
1209  void _limparTela() {
1210    setState(() {
1211      _resposta = '';
1212    });
1213  }
1214
1215  void _atualizarResposta(String novoValor) {
1216    setState(() {
1217      _resposta = novoValor;
1218    });
1219  }
1220
1221  void _limparTela() {
1222    setState(() {
1223      _resposta = '';
1224    });
1225  }
1226
1227  void _deletarResposta() {
1228    setState(() {
1229      _resposta =
1230      _resposta
1231      .substring(0, _resposta.length - 1);
1232    });
1233  }
1234
1235  void _adicionarResposta() {
1236    setState(() {
1237      _resposta += '!';
1238    });
1239  }
1240
1241  void _inverterResposta() {
1242    setState(() {
1243      _resposta =
1244      _resposta
1245      .split('')
1246      .reversed
1247      .join('');
1248    });
1249  }
1250
1251  void _limparTela() {
1252    setState(() {
1253      _resposta = '';
1254    });
1255  }
1256
1257  void _atualizarResposta(String novoValor) {
1258    setState(() {
1259      _resposta = novoValor;
1260    });
1261  }
1262
1263  void _limparTela() {
1264    setState(() {
1265      _resposta = '';
1266    });
1267  }
1268
1269  void _deletarResposta() {
1270    setState(() {
1271      _resposta =
1272      _resposta
1273      .substring(0, _resposta.length - 1);
1274    });
1275  }
1276
1277  void _adicionarResposta() {
1278    setState(() {
1279      _resposta += '!';
1280    });
1281  }
1282
1283  void _inverterResposta() {
1284    setState(() {
1285      _resposta =
1286      _resposta
1287      .split('')
1288      .reversed
1289      .join('');
1290    });
1291  }
1292
1293  void _limparTela() {
1294    setState(() {
1295      _resposta = '';
1296    });
1297  }
1298
1299  void _atualizarResposta(String novoValor) {
1300    setState(() {
1301      _resposta = novoValor;
1302    });
1303  }
1304
1305  void _limparTela() {
1306    setState(() {
1307      _resposta = '';
1308    });
1309  }
1310
1311  void _deletarResposta() {
1312    setState(() {
1313      _resposta =
1314      _resposta
1315      .substring(0, _resposta.length - 1);
1316    });
1317  }
1318
1319  void _adicionarResposta() {
1320    setState(() {
1321      _resposta += '!';
1322    });
1323  }
1324
1325  void _inverterResposta() {
1326    setState(() {
1327      _resposta =
1328      _resposta
1329      .split('')
1330      .reversed
1331      .join('');
1332    });
1333  }
1334
1335  void _limparTela() {
1336    setState(() {
1337      _resposta = '';
1338    });
1339  }
1340
1341  void _atualizarResposta(String novoValor) {
1342    setState(() {
1343      _resposta = novoValor;
1344    });
1345  }
1346
1347  void _limparTela() {
1348    setState(() {
1349      _resposta = '';
1350    });
1351  }
1352
1353  void _deletarResposta() {
1354    setState(() {
1355      _resposta =
1356      _resposta
1357      .substring(0, _resposta.length - 1);
1358    });
1359  }
1360
1361  void _adicionarResposta() {
1362    setState(() {
1363      _resposta += '!';
1364    });
1365  }
1366
1367  void _inverterResposta() {
1368    setState(() {
1369      _resposta =
1370      _resposta
1371      .split('')
1372      .reversed
1373      .join('');
1374    });
1375  }
1376
1377  void _limparTela() {
1378    setState(() {
1379      _resposta = '';
1380    });
1381  }
1382
1383  void _atualizarResposta(String novoValor) {
1384    setState(() {
1385      _resposta = novoValor;
1386    });
1387  }
1388
1389  void _limparTela() {
1390    setState(() {
1391      _resposta = '';
1392    });
1393  }
1394
1395  void _deletarResposta() {
1396    setState(() {
1397      _resposta =
1398      _resposta
1399
```

Entrada de Dados

Um campo de texto permite que o usuário insira texto, com teclado de hardware ou com teclado na tela.

O campo de texto chama o retorno de chamada `onChanged` sempre que o usuário altera o texto no campo. Se o usuário indicar que acabou de digitar no campo (por exemplo, pressionando um botão no teclado virtual), o campo de texto chama o retorno de chamada `onSubmitted`.

Para controlar o texto que é exibido no campo de texto, use o controlador. Por exemplo, para definir o valor inicial do campo de texto, use um controlador que já contenha algum texto. O controlador também pode controlar a seleção e a região de composição (e observar as mudanças no texto, seleção e região de composição).

```
73         child: Text('Salvar')
74     ), // ElevatedButton
75     Padding(
76         padding: const EdgeInsets.only(top: 20),
77         child: Text(_resposta, style:
78             TextStyle(
79                 decoration: TextDecoration.none
80             ) // TextStyle
81         ), // Text
82         ) // Padding
83     ],
84     ), // Column
85 ); // Scaffold
86 }
87 }
```



Stateless

Widgets sem estado não requerem estado mutável, ou seja, é imutável . Em palavras simples, Widgets sem estado não podem mudar seu estado durante o tempo de execução do aplicativo, o que significa que os widgets não podem ser redesenhados enquanto o aplicativo está em ação.

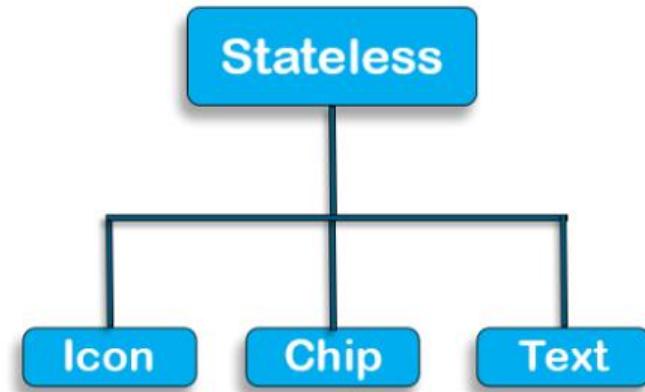
No widget sem estado, o método “ build ” pode ser chamado apenas UMA VEZ enquanto o aplicativo está em ação, que é responsável por desenhar os widgets na tela do dispositivo.

Stateful

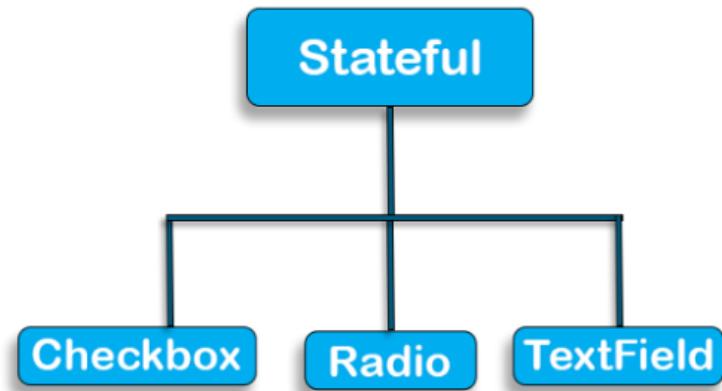
Widgets com estado têm um estado mutável, ou seja, são mutáveis e podem ser desenhados várias vezes durante seu tempo de vida.

Eles são os widgets que podem mudar de estado várias vezes e podem ser redesenhados na tela qualquer número de vezes enquanto o aplicativo está em ação.

Exemplos de widgets sem estado são os seguintes:



Exemplos de widgets com estado são os seguintes:

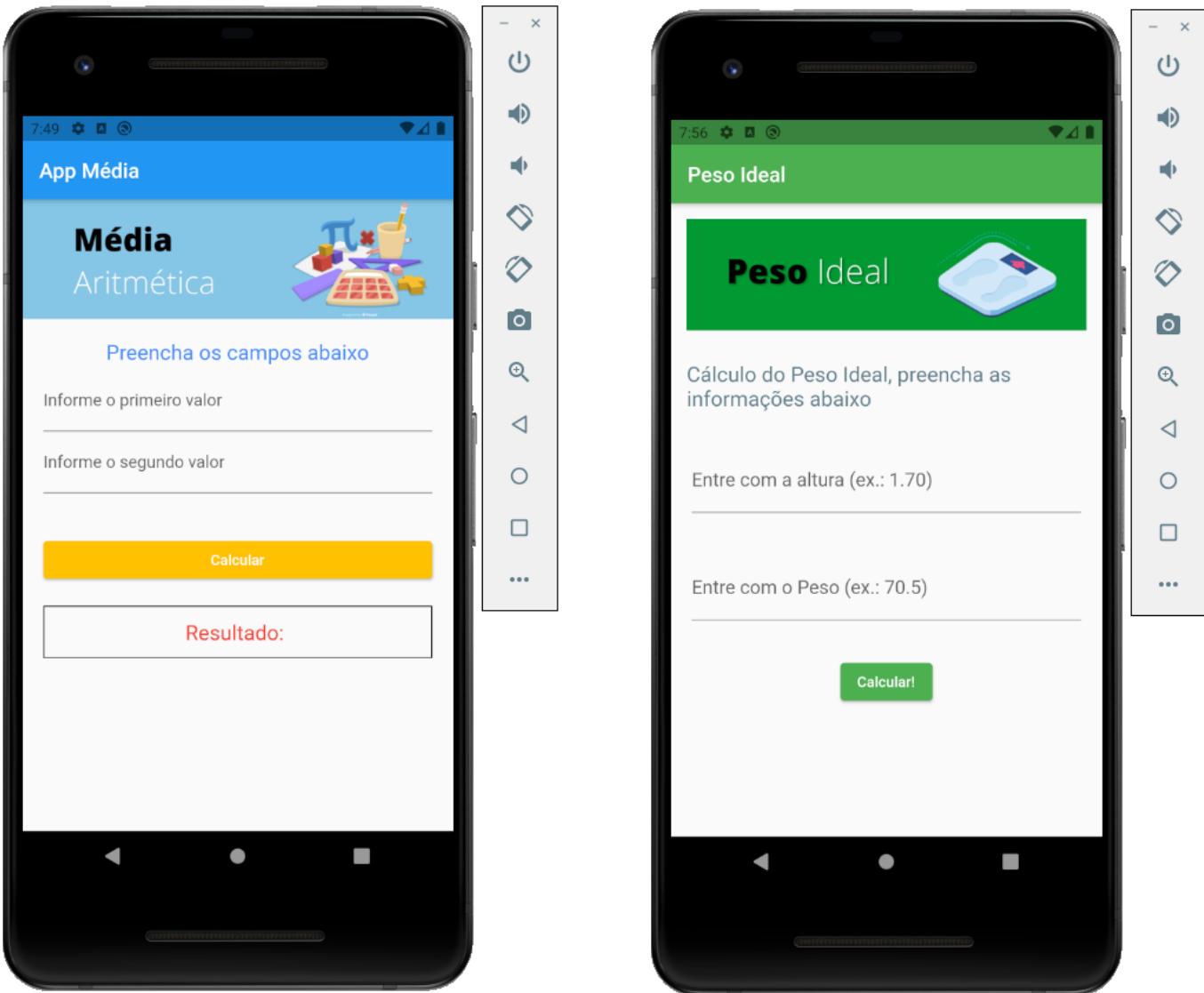


Atividade 1 Flutter

Partindo do pressuposto de que vocês receberam um job de um cliente e baseando-se no design ao lado, fornecido por ele, desenvolva as aplicações :

Observações:

Enviar a pasta lib, a pasta images o arquivo .yaml compactados para o email do professor.



Gesture Detector()

Se você precisar detectar que tipo de gesto está acontecendo em um widget, a maneira mais fácil é envolver o widget como filho de GestureDetector.

Ele pode ter várias funções de retorno de chamada, cada uma para diferentes tipos de gestos. Ele também suporta vários tipos de gestos, como toque, toque secundário, toque longo, arrasto vertical / horizontal, panorâmica e escala.

Usaremos o parâmetro onTap(), mas fique a vontade para aprender as outras funcionalidades dessa classe,



```
lib > main.dart > _GestureState > build
1 //Gesture Detector
2 //Detectando toque na tela
3
4 import 'package:flutter/material.dart';
5
6 Run | Debug | Profile
7 main() => runApp(MaterialApp(
8   home: Gesture(),
9 )); // MaterialApp
10
11 class Gesture extends StatefulWidget {
12   @override
13   _GestureState createState() => _GestureState();
14 }
15
16 class _GestureState extends State<Gesture> {
17
18   String _personagem = '';
19   var _imagem = AssetImage('images/imagem.png');
20
21   personagemEscolhido(String personagem) {
22     if (personagem == 'joao') {
23       _personagem = 'Você escolheu o João!';
24       _imagem = AssetImage('images/jo.png');
25     } else if (personagem == 'juliana') {
26       _personagem = 'Você escolheu a Juliana!';
27       _imagem = AssetImage('images/ju.png');
28     } else {
29       _personagem = 'Você escolheu o José!';
30       _imagem = AssetImage('images/ze.png');
31     }
32   }
33
34   @override
35   Widget build(BuildContext context) {
36     return Scaffold(
37       appBar: AppBar(
38         title: Text('Gesture App'),
39         backgroundColor: Color(0xffcd853f),
40       ), // AppBar
41       body: Padding(
42         padding: EdgeInsets.all(20),
43         //Primeiro uma column
44         child: Column(
45           children: [
46             //Agora uma Row
```

Gesture Detector()

Se você precisar detectar que tipo de gesto está acontecendo em um widget, a maneira mais fácil é envolver o widget como filho de GestureDetector.

Ele pode ter várias funções de retorno de chamada, cada uma para diferentes tipos de gestos. Ele também suporta vários tipos de gestos, como toque, toque secundário, toque longo, arrasto vertical / horizontal, panorâmica e escala.

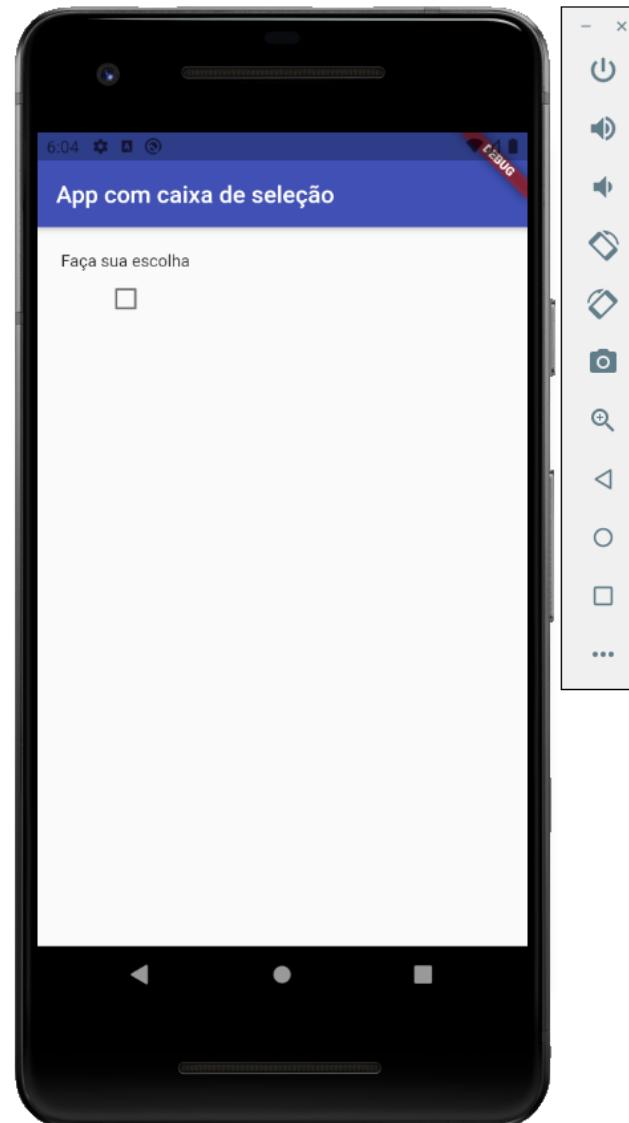
Usaremos o parâmetro onTap(), mas fique a vontade para aprender as outras funcionalidades dessa classe,

```
45   Row(
46     mainAxisSize: MainAxisSize.spaceEvenly,
47     children: [
48       //Detectando o toque na imagem
49       GestureDetector(
50         //Evento de 1 toque na tela
51         onTap: () {
52           setState(() {
53             personagemEscolhido('joao');
54           });
55         },
56         //Inserção da imagem
57         child: Image.asset(
58           'images/Joao.png',
59           width: 100,
60         ), // Image.asset
61       ), // GestureDetector
62       GestureDetector(
63         onTap: () {
64           setState(() {
65             personagemEscolhido('juliana');
66           });
67         },
68         child: Image.asset(
69           'images/Juliana.png',
70           width: 100,
71         ), // Image.asset
72       ), // GestureDetector
73       GestureDetector(
74         onTap: () {
75           setState(() {
76             personagemEscolhido('jose');
77           });
78         },
79         child: Image.asset(
80           'images/Jose.png',
81           width: 100,
82         ), // Image.asset
83       ), // GestureDetector
84     ],
85   ), // Row
86   //mais uma Column
87   Padding(
88     padding: const EdgeInsets.only(
89       top: 20,
90     ), // EdgeInsets.only
91     child: Column(
92       children: [
93         Text(
94           'Escolha um personagem',
95           style: TextStyle(
96             fontSize: 20,
97           ), // TextStyle
98         ), // Text
99         Padding(
100           padding: EdgeInsets.only(top: 20, bottom: 20),
101         ), // EdgeInsets.only
102         child: Column(
103           children: [
104             Image(
105               image: _imagem,
106               width: 100,
107             ), // Image
108           ],
109         ), // Column
110       ), // Padding
111     ),
112   ), // Padding
113   Padding(
114     padding: const EdgeInsets.only(top: 20),
115   ), // EdgeInsets.only
116   child: Text(
117     _personagem,
118     style: TextStyle(
119       fontSize: 30,
120       color: Colors.orange,
121     ), // TextStyle
122   ), // Text
123   Padding(
124     padding: const EdgeInsets.only(
125       top: 20,
126     ), // EdgeInsets.only
127   ), // Padding
128 ); // Scaffold
129 }
130 }
```

Checkbox()

A caixa de seleção não mantém nenhum estado. Em vez disso, quando o estado da caixa de seleção muda, o widget chama o retorno de chamada onChanged.

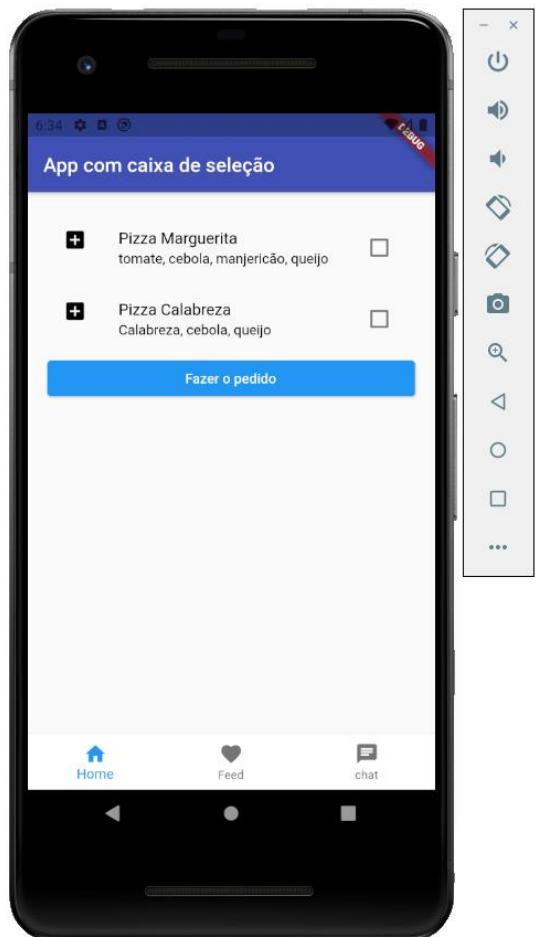
A maioria dos widgets que usam uma caixa de seleção ouvirá o retorno de chamada onChanged e reconstruirá a caixa de seleção com um novo valor para atualizar sua aparência.



```
lib > main14_checkbox_1.dart > ...
1 import 'package:flutter/material.dart';
Run | Debug | Profile
2 main() => runApp(MaterialApp(
3   |   |   home: MeuCheckbox(),
4   |   | ); // MaterialApp
5
6 class MeuCheckbox extends StatefulWidget {
7   @override
8   _MeuCheckboxState createState() => _MeuCheckboxState();
9 }
10
11 class _MeuCheckboxState extends State<MeuCheckbox> {
12   bool _selecaoCheckbox = false;
13
14   @override
15   Widget build(BuildContext context) {
16     return Scaffold(
17       appBar: AppBar(
18         title: Text('App com caixa de seleção'),
19         backgroundColor: Colors.indigo,
20       ), // AppBar
21       body: Padding(
22         padding: const EdgeInsets.all(20.0),
23         child: Column(
24           children: [
25             Text('Faça sua escolha'),
26             Checkbox(
27               //valor inicial do checkbox
28               value: _selecaoCheckbox, //desmarcado
29               onChanged: (bool? valor) {
30                 setState(() {
31                   _selecaoCheckbox = valor!;
32                 });
33               } // Checkbox
34             ],
35           ), // Column
36         ), // Padding
37       ); // Scaffold
38     }
39 }
```

CheckboxListTile()

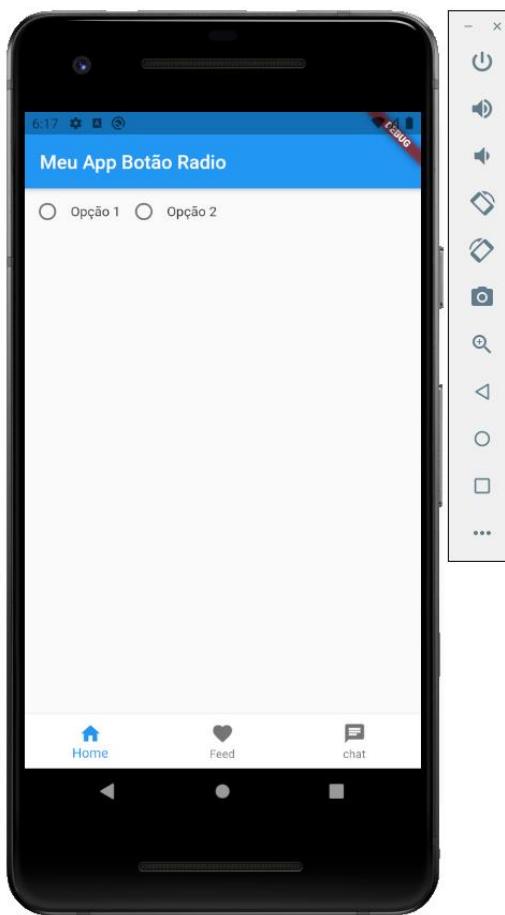
CheckboxListTile é um widget que combina uma caixa de seleção e um bloco de lista. Ele permite que você crie uma caixa de seleção junto com o título, subtítulo e ícone sem a necessidade de criar um widget separado para cada parte.



```
1 import 'package:flutter/material.dart';
Run | Debug | Profile
2 main() => runApp(MaterialApp(
3   home: MeuCheckbox(),
4 )); // MaterialApp
5
6 class MeuCheckbox extends StatefulWidget {
7   @override
8   _MeuCheckboxState createState() => _MeuCheckboxState();
9 }
10
11 class _MeuCheckboxState extends State<MeuCheckbox> {
12   bool _pizzaMarguerita = false;
13   bool _pizzaCalabresa = false;
14
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       appBar: AppBar(
19         title: Text('App com caixa de seleção'),
20         backgroundColor: Colors.indigo,
21       ), // AppBar
22       body: Padding(
23         padding: const EdgeInsets.all(20.0),
24         child: Column(
25           //Para esticar o conteúdo
26           crossAxisAlignment: CrossAxisAlignment.stretch,
27           children: [
28
29             CheckboxListTile(
30               title: Text('Pizza Marguerita'),
31               subtitle: Text('tomate, cebola, manjericão, queijo'),
32               //Cor do checkbox
33               activeColor: Colors.black,
34               //cor do texto
35               selected: true,
36               //definindo um ícone
37               secondary: Icon(Icons.add_box),
38               value: _pizzaMarguerita,
39               onChanged: (bool? valor) {
40                 setState(() {
41                   _pizzaMarguerita = valor!;
42                 });
43               }, // CheckboxListTile
44
45             CheckboxListTile(
46               title: Text('Pizza Calabresa'),
47               subtitle: Text('Calabresa, cebola, queijo'),
48               //Cor do checkbox
49               activeColor: Colors.black,
50               //cor do texto
51               selected: true,
52               //definindo um ícone
53               secondary: Icon(Icons.add_box),
54               value: _pizzaCalabresa,
55               onChanged: (bool? valor) {
56                 setState(() {
57                   _pizzaCalabresa = valor!;
58                 });
59               }, // CheckboxListTile
60               ElevatedButton(
61                 onPressed: () {
62                   print('Pedido:');
63                   print('Pizza Marguerita: ${_pizzaMarguerita}');
64                   print('PizzaCalabresa: ${_pizzaCalabresa}');
65                 },
66                 child: Text('Fazer o pedido'),
67               ), // ElevatedButton
68             ), // Column
69           ], // Padding
70           bottomNavigationBar: BottomNavigationBar(
71             backgroundColor: Colors.white,
72             currentIndex: 0,
73             items: [
74               BottomNavigationBarItem(
75                 icon: Icon(Icons.home),
76                 label: 'Home',
77               ), // BottomNavigationBarItem
78               BottomNavigationBarItem(
79                 icon: Icon(Icons.favorite),
80                 label: 'Feed',
81               ), // BottomNavigationBarItem
82               BottomNavigationBarItem(
83                 icon: Icon(Icons.chat),
84                 label: 'chat',
85               ), // BottomNavigationBarItem
86             ], // BottomNavigationBar
87           ), // Scaffold
88         }
89 }
```

Radio()

Usado para selecionar entre vários valores mutuamente exclusivos. Quando um botão de opção em um grupo é selecionado, os outros botões de opção no grupo deixam de ser selecionados. Os valores são do tipo True/False.

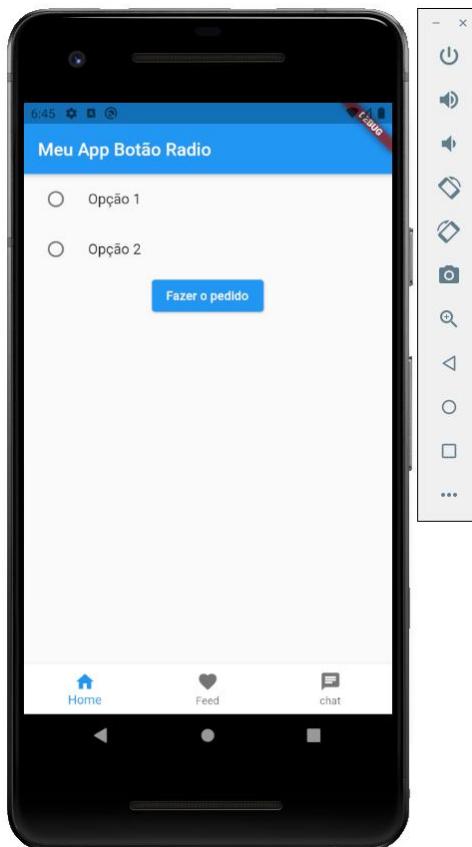


```
1 import 'package:flutter/material.dart';
2 Run | Debug | Profile
3 void main() => runApp(MaterialApp(
4   home: BotaoRadio(),
5 )); // MaterialApp
6
7 class BotaoRadio extends StatefulWidget {
8   @override
9   _BotaoRadioState createState() => _BotaoRadioState();
10
11 class _BotaoRadioState extends State<BotaoRadio> {
12   int? _opcao;
13
14   @override
15   Widget build(BuildContext context) {
16     return Scaffold(
17       appBar: AppBar(
18         title: Text('Meu App Botão Radio'),
19         backgroundColor: Colors.blue,
20     ), // AppBar
21     body: Container(
22       child: Row(
23         children: [
24           Radio(
25             value: 1,
26             groupValue: _opcao,
27             onChanged: (int? selecao) {
28               print('Seleção: $selecao');
29               setState(() {
30                 _opcao = selecao;
31               });
32             }, // Radio
33             Text('Opção 1'),
34             Radio(
35               value: 2,
36               groupValue: _opcao,
37               onChanged: (int? selecao) {
38                 print('Seleção: $selecao');
39                 setState(() {
40                   _opcao = selecao;
41                 });
42               }, // Radio
43               Text('Opção 2')
44             ],
45           ), // Row
46         ], // Container
47       bottomNavigationBar: BottomNavigationBar(
48         backgroundColor: Colors.white,
49         currentIndex: 0,
50         items: [
51           BottomNavigationBarItem(
52             icon: Icon(Icons.home),
53             label: 'Home',
54         ), // BottomNavigationBarItem
55         BottomNavigationBarItem(
56             icon: Icon(Icons.favorite),
57             label: 'Feed',
58         ), // BottomNavigationBarItem
59         BottomNavigationBarItem(
60             icon: Icon(Icons.chat),
61             label: 'chat',
62         ), // BottomNavigationBarItem
63     ], // BottomNavigationBar
64   ); // Scaffold
65 }
66 }
```

RadioListTile()

O RadioListTile nada mais é que um ListTile contendo um botão de rádio por padrão. Todo o bloco da lista é interativo: tocar em qualquer lugar do bloco seleciona o botão de rádio.

As propriedades value, groupValue, onChanged e activeColor deste widget são idênticas às propriedades com nomes semelhantes no widget Radio.



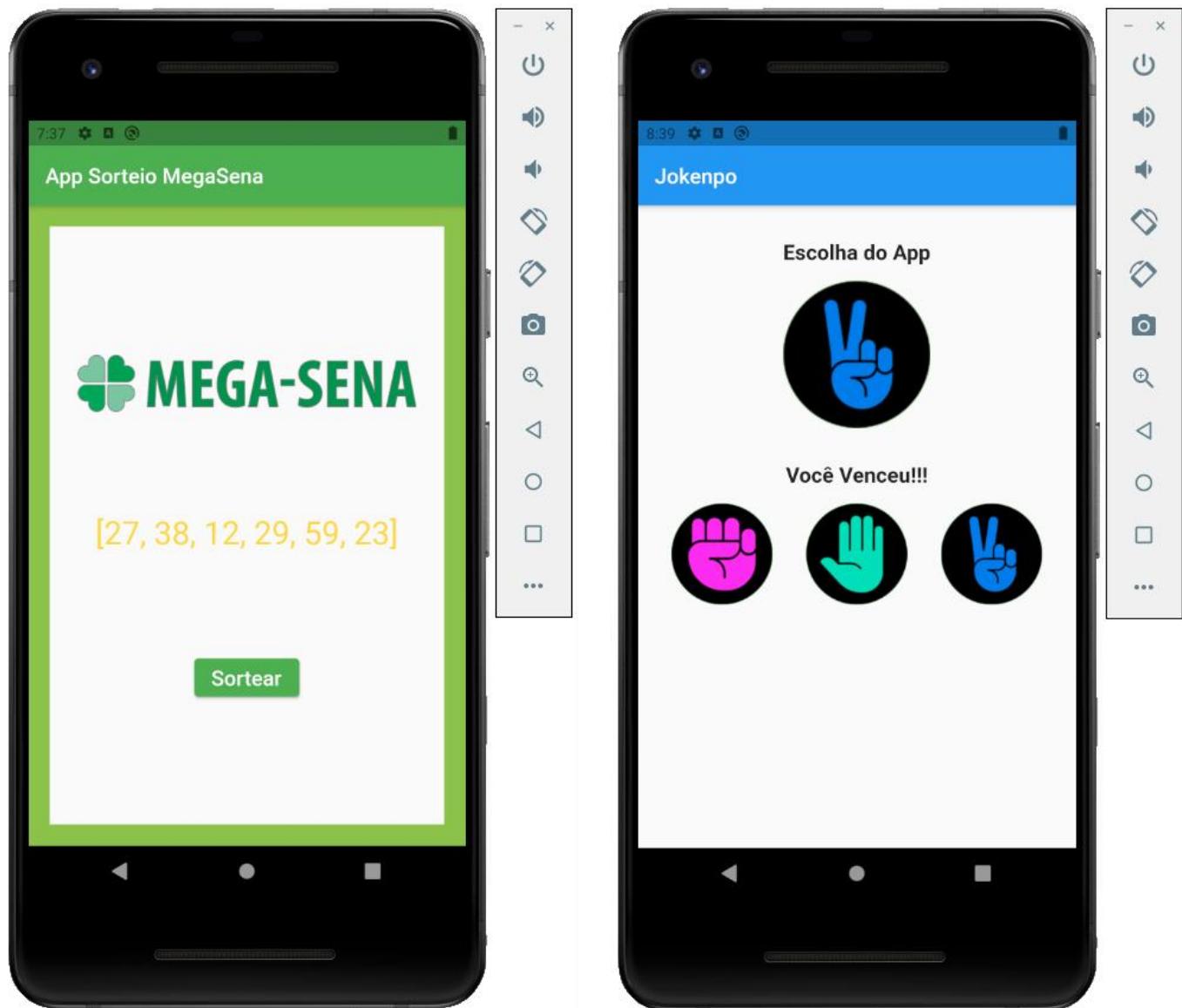
```
1 import 'package:flutter/material.dart';
Run | Debug | Profile
2 void main() => runApp(MaterialApp(
3   home: BotaoRadio(),
4 )); // MaterialApp
5
6 class BotaoRadio extends StatefulWidget {
7   @override
8   _BotaoRadioState createState() => _BotaoRadioState();
9 }
10
11 class _BotaoRadioState extends State<BotaoRadio> {
12   //Variável para capturar o valor do botão
13   int? _opcao;
14
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       appBar: AppBar(
19         title: Text('Meu App Botão Radio'),
20         backgroundColor: Colors.blue,
21       ), // AppBar
22       body: Container(
23         child: Column(
24           children: [
25             RadioListTile(
26               title: Text('Opção 1'),
27               value: 1,
28               groupValue: _opcao,
29               onChanged: (int? selecao) {
30                 setState(() {
31                   _opcao = selecao;
32                 });
33               },
34             ), // RadioListTile
35             RadioListTile(
36               title: Text('Opção 2'),
37               value: 2,
38               groupValue: _opcao,
39               onChanged: (int? selecao) {
40                 setState(() {
41                   _opcao = selecao;
42                 });
43             }, // RadioListTile
44           ],
45         ),
46       ),
47     );
48   }
49 }
50
51 ElevatedButton(
52   onPressed: () {
53     print('Seleção:');
54     print('Opção escolhida: $_opcao');
55   },
56   child: Text('Fazer o pedido')
57 ) // ElevatedButton
58 ],
59 ), // Column
60 ), // Container
61 bottomNavigationBar: BottomNavigationBar(
62   backgroundColor: Colors.white,
63   currentIndex: 0,
64   items: [
65     BottomNavigationBarItem(
66       icon: Icon(Icons.home),
67       label: 'Home',
68     ), // BottomNavigationBarItem
69     BottomNavigationBarItem(
70       icon: Icon(Icons.favorite),
71       label: 'Feed',
72     ), // BottomNavigationBarItem
73     BottomNavigationBarItem(
74       icon: Icon(Icons.chat),
75       label: 'chat',
76     ), // BottomNavigationBarItem
77   ],
78 ), // BottomNavigationBar
79 ); // Scaffold
80 }
```

Atividade 2 Flutter

Partindo do pressuposto de que vocês receberam um job de um cliente e baseando-se no design ao lado, fornecido por ele, desenvolva as aplicações :

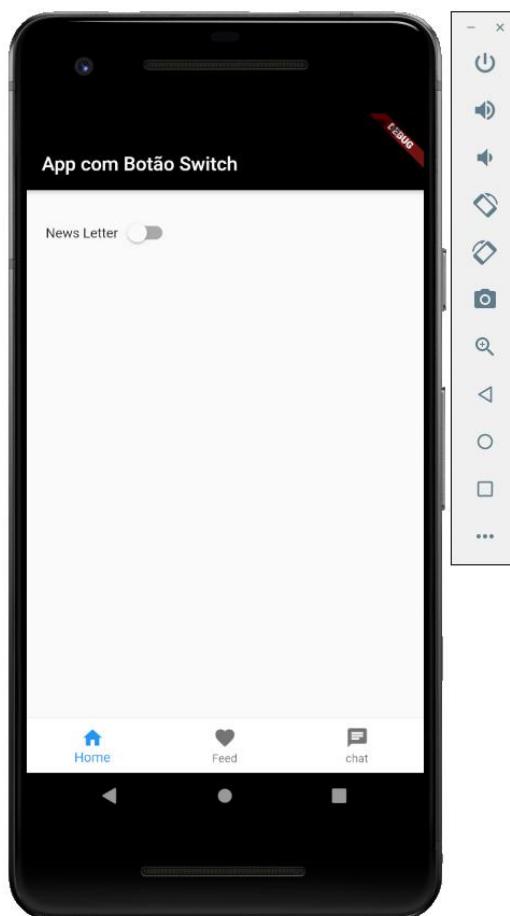
Observações:

- Os números da mega não pode se repetir
- Enviar a pasta lib, a pasta images
- o arquivo .yaml compactados para o email do professor.



Switch()

Um Switch é usado para alternar uma configuração entre on / off, que é verdadeiro / falso, respectivamente . Quando a chave está ligada, o valor retornado pela propriedade Switch onChanged é verdadeiro, enquanto a chave está desligada, a propriedade onChanged retorna falso.



```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 main() => runApp(MaterialApp(
5   home: AppSwitch(),
6 )); // MaterialApp
7
8 class AppSwitch extends StatefulWidget {
9   @override
10  _AppSwitchState createState() => _AppSwitchState();
11
12 class _AppSwitchState extends State<AppSwitch> {
13   //Definindo uma variável para o estado do switch
14   bool _selecaoEstado = false;
15
16   @override
17   Widget build(BuildContext context) {
18     return Scaffold(
19       appBar: AppBar(
20         title: Text('App com Botão Switch'),
21         backgroundColor: Colors.black,
22       ), // AppBar
23       body: Container(
24         child: Padding(
25           padding: const EdgeInsets.all(20.0),
26           child: Row(
27             children: [
28               Text('News Letter'),
29               //Cria um botão Switch
30               Switch(
31                 value: _selecaoEstado,
32                 onChanged: (bool estado) {
33                   setState(() {
34                     _selecaoEstado = estado;
35                   });
36                 }, // Switch
37               ],
38             ), // Row
39           ), // Padding
40         ), // Container
41         bottomNavigationBar: BottomNavigationBar(
42           backgroundColor: Colors.white,
43           currentIndex: 0,
44           items: [
45             BottomNavigationBarItem(
46               icon: Icon(Icons.home),
47               label: 'Home',
48             ), // BottomNavigationBarItem
49             BottomNavigationBarItem(
50               icon: Icon(Icons.favorite),
51               label: 'Feed',
52             ), // BottomNavigationBarItem
53             BottomNavigationBarItem(
54               icon: Icon(Icons.chat),
55               label: 'chat',
56             ) // BottomNavigationBarItem
57           ], // BottomNavigationBar
58         ); // Scaffold
59       }
60     }
```

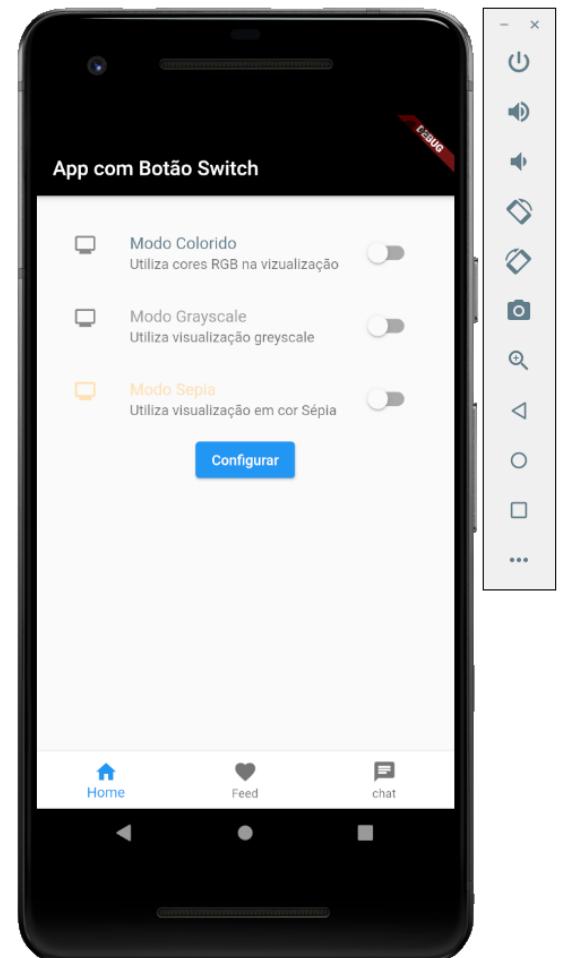
SwitchListTile()

Um SwitchListTile é usado para alternar uma configuração entre on / off, que é verdadeiro / falso, respectivamente . Quando a chave está ligada, o valor retornado pela propriedade Switch onChanged é verdadeiro, enquanto a chave está desligada, a propriedade onChanged retorna falso. A diferença é a criação do switch em lista.

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 main() => runApp(MaterialApp(
5   home: AppSwitch(),
6 )); // MaterialApp
7
8 class AppSwitch extends StatefulWidget {
9   @override
10  _AppSwitchState createState() => _AppSwitchState();
11 }
12
13 class _AppSwitchState extends State<AppSwitch> {
14   //Definindo uma variável para o estado do switch
15   bool _modoCor = false;
16   bool _modoTomCinza = false;
17   bool _modoSepia = false;
18
19   @override
20   Widget build(BuildContext context) {
21     return Scaffold(
22       appBar: AppBar(
23         title: Text('App com Botão Switch'),
24         backgroundColor: Colors.black,
25       ), // AppBar
26       body: Container(
27         child: Padding(
28           padding: const EdgeInsets.all(20.0),
29           child: Column(
30             children: [
31               //Cria um botão Switch
32               SwitchListTile(
33                 title: Text(
34                   'Modo Colorido',
35                   style: TextStyle(color: Colors.blueGrey),
36                 ), // Text
37                 subtitle: Text('Utiliza cores RGB na visualização'),
38                 activeColor: Colors.blueGrey,
39                 secondary: Icon(Icons.monitor),
40               ),
41             ],
42           ),
43         ),
44       ),
45     );
46   }
47 }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
```

```
value: _modoCor,
onChanged: (bool estado1) {
  setState(() {
    _modoCor = estado1;
  });
}, // SwitchListTile
SwitchListTile(
  title: Text(
    'Modo Grayscale',
    style: TextStyle(color: Colors.grey),
  ), // Text
  subtitle: Text('Utiliza visualização greyscale'),
  activeColor: Colors.grey,
  secondary: Icon(Icons.monitor),
  value: _modoTomCinza,
  onChanged: (bool estado1) {
    setState(() {
      _modoTomCinza = estado1;
    });
  }, // SwitchListTile
SwitchListTile(
  title: Text(
    'Modo Sepia',
    style: TextStyle(color: Colors.orange.shade100),
  ), // Text
  subtitle: Text('Utiliza visualização em cor Sépia'),
  activeColor: Colors.orange.shade100,
  secondary: Icon(
    Icons.monitor,
    color: Colors.orange.shade100,
  ), // Icon
  value: _modoSepia,
  onChanged: (bool estado1) {
    setState(() {
      _modoSepia = estado1;
    });
  }, // SwitchListTile
ElevatedButton(
  onPressed: () {
```

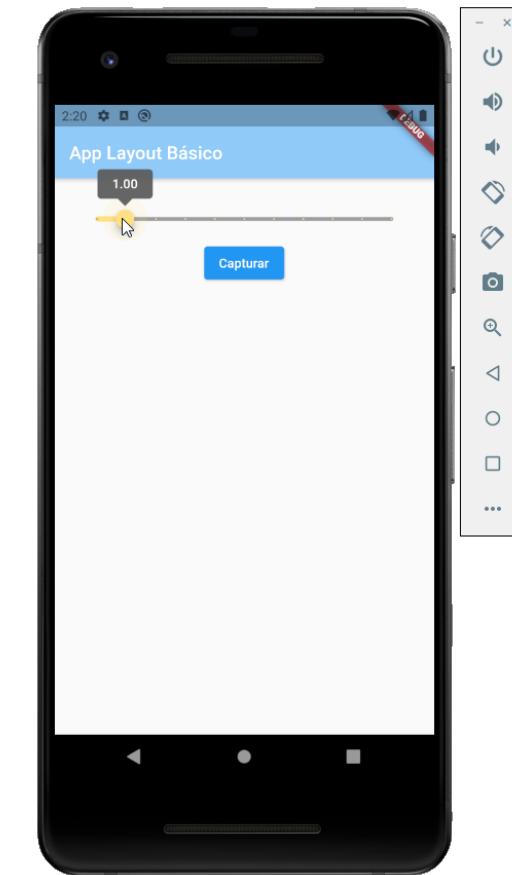
```
78   print('Modo Cor: ${_modoCor}');
79   print('Modo Tons de Cinza: ${_modoTomCinza}');
80   print('Modo Modo Sépia: ${_modoSepia}');
81 },
82 ],
83 ), // Column
84 ), // Padding
85 ), // Container
```



Slider()

Um controle deslizante pode ser usado para selecionar um conjunto de valores contínuo ou discreto. O padrão é usar uma faixa contínua de valores de mínimo a máximo.

```
1 import 'package:flutter/material.dart';
Run | Debug | Profile
2 main() => runApp(MaterialApp(
3   |   home: AppSlider(),
4   |   )); // MaterialApp
5
6 class AppSlider extends StatefulWidget {
7   @override
8   _AppSliderState createState() => _AppSliderState();
9 }
10
11 class _AppSliderState extends State<AppSlider> {
12   double _valorSlider = 0;
13
14   @override
15   Widget build(BuildContext context) {
16     return Scaffold(
17       appBar: AppBar(
18         title: Text('App Layout Básico'),
19         backgroundColor: Colors.blue.shade200,
20       ), // AppBar
21       body: Padding(
22         padding: const EdgeInsets.all(20.0),
23         child: Container(
24           child: Column(
25             children: [
26               Slider(
27                 value: _valorSlider,
28                 min: 0,
29                 max: 10,
30                 //Definindo um label no slider
31                 label: '${_valorSlider.toStringAsFixed(2)}',
32                 divisions: 10,
33                 activeColor: Colors.amber.shade200,
34                 inactiveColor: Colors.grey,
35                 onChanged: (double valorAlterado) {
36                   setState(() {
37                     _valorSlider = valorAlterado;
38                   });
39               }, // Slider
40               ElevatedButton(
41                 onPressed: () {
42                   print(
43                     'Valor selecionado: ${_valorSlider.toStringAsFixed(2)}');
44                 }, child: Text('Capturar')) // ElevatedButton
45             ],
46           ), // Column
47         ), // Container
48       ), // Padding
49     ); // Scaffold
50 }
51 }
```



Navegação

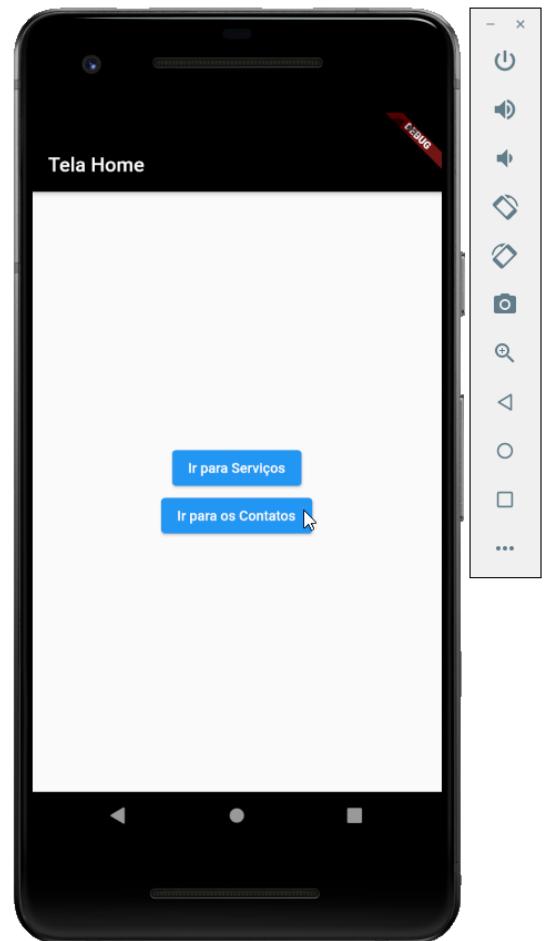
- Navigator(): Classe responsável pela Navegação
 - push(): Abre uma tela
 - pop(): Fecha a tela corrente
 - Parâmetro context: Contexto da Aplicação, em qual tela está, já foi definido logo no começo da aplicação
 - Parâmetro route: A rota de destino definida com MaterialPageRoute().
 - Parâmetro builder: Usa uma função anônima para chamar a tela de destino. Utiliza context como argumento

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 main() => runApp(MaterialApp(
5   |   home: Home(),
6   )); // MaterialApp
7
8 class Home extends StatefulWidget {
9   @override
10  _HomeState createState() => _HomeState();
11 }
12 ///////////////Tela Home
13 class _HomeState extends State<Home> {
14   @override
15   Widget build(BuildContext context) {
16     return Scaffold(
17       appBar: AppBar(
18         title: Text('Tela Home'),
19         backgroundColor: Colors.black,
20       ), // AppBar
21       body: Center(
22         child: Container(
23           child: Column(
24             mainAxisAlignment: MainAxisAlignment.center,
25             children: [
26               ElevatedButton(
27                 onPressed: () {
28                   Navigator.push(
29                     context,
30                     MaterialPageRoute(
31                       builder: (context) => Servicos(),
32                     ), // MaterialPageRoute
33                 );
34               },
35               child: Text('Ir para Serviços'),
36             ), // ElevatedButton
37             ElevatedButton(
38               onPressed: () {
39                 Navigator.push(
40                   context,
41                   MaterialPageRoute(
42                     builder: (context) => Contato(),
43                   ), // MaterialPageRoute
44                 );
45               },
46               child: Text('Ir para os Contatos'),
47             ), // ElevatedButton
48           ],
49         ), // Column
50       ), // Container
51     ); // Scaffold
52   }
53 }
54 ///////////////Tela Serviços
55 class Servicos extends StatefulWidget {
56   @override
57   _ServicosState createState() => _ServicosState();
58 }
59
60 class _ServicosState extends State<Servicos> {
61   @override
62   Widget build(BuildContext context) {
63     return Scaffold(
64       appBar: AppBar(
65         title: Text('Tela de Serviços'),
66         backgroundColor: Colors.blue,
67       ), // AppBar
68       body: Center(
69         child: Container(
70           child: Column(
71             mainAxisAlignment: MainAxisAlignment.center,
72             children: [
73               Text(
74                 'Tela de Serviços',
75                 style: TextStyle(fontSize: 20),
76               ), // Text
77             ],
78           ),
79         ),
80       ), // Container
81     ); // Scaffold
82   }
83 }
```

Navegação

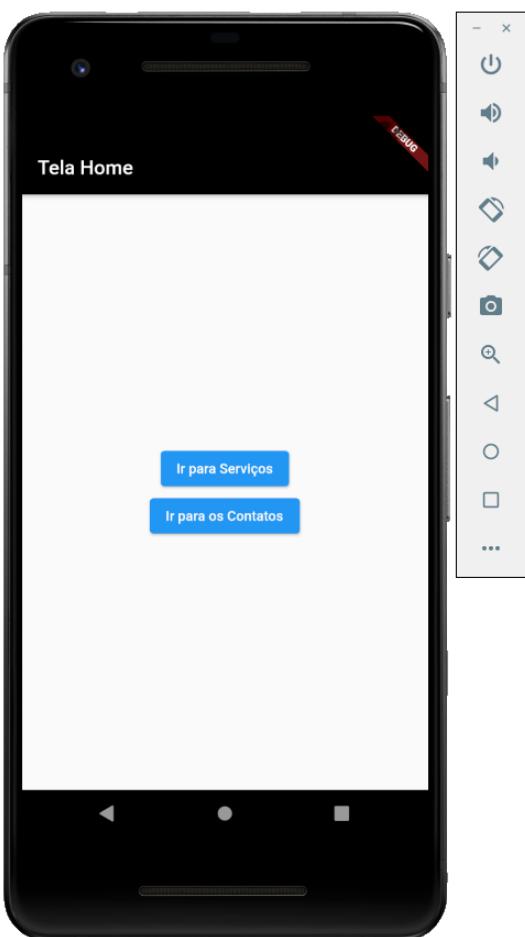
- Navigator(): Classe responsável pela Navegação
- push(): Abre uma tela
- pop(): Fecha a tela corrente
- Parâmetro context: Contexto da Aplicação, em qual tela está, já foi definido logo no começo da aplicação
- Parâmetro route: A rota de destino definida com MaterialPageRoute().
- Parâmetro builder: Usa uma função anônima para chamar a tela de destino. Utiliza context como argumento

```
78     ), // Column
79     ], // Container
80   ), // Center
81 ); // Scaffold
82 }
83 }
84
85 ///////////////Tela Contato
86 class Contato extends StatefulWidget {
87   @override
88   _ContatoState createState() => _ContatoState();
89 }
90
91 class _ContatoState extends State<Contato> {
92   @override
93   Widget build(BuildContext context) {
94     return Scaffold(
95       appBar: AppBar(
96         title: Text('Serviços'),
97         backgroundColor: Colors.amber,
98       ), // AppBar
99       body: Center(
100         child: Container(
101           child: Column(
102             mainAxisAlignment: MainAxisAlignment.center,
103             children: [
104               ElevatedButton(
105                 onPressed: () {
106                   Navigator.pop(context);
107                 },
108                 child: Text('Voltar para a Home'),
109               ) // ElevatedButton
110             ],
111           ), // Column
112         ), // Container
113       ), // Center
114     ); // Scaffold
115   }
116 }
```



Passando valores entre telas

- Ainda utilizando o Navigator(), vamos enviar dados de uma tela para outra. Para isso criamos construtores para definir os atributos das informações.

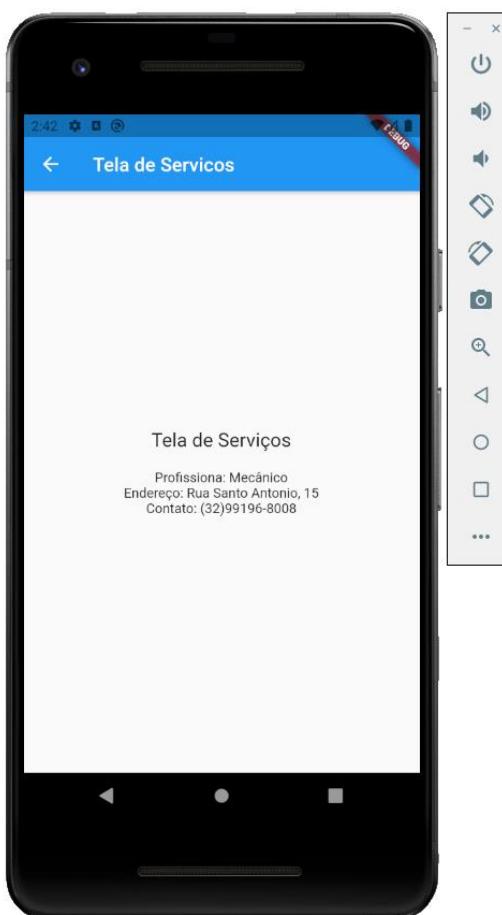


```
1 import 'package:flutter/material.dart';
Run | Debug | Profile
2 main() => runApp(MaterialApp(
3   |   home: Home(),
4   )); // MaterialApp
5
6 class Home extends StatefulWidget {
7   @override
8   _HomeState createState() => _HomeState();
9 }
10 ///////////////Tela Home
11 class _HomeState extends State<Home> {
12   @override
13   Widget build(BuildContext context) {
14     return Scaffold(
15       appBar: AppBar(
16         title: Text('Tela Home'),
17         backgroundColor: Colors.black,
18       ), // AppBar
19       body: Center(
20         child: Container(
21           child: Column(
22             mainAxisAlignment: MainAxisAlignment.center,
23             children: [
24               ElevatedButton(
25                 onPressed: () {
26                   Navigator.push(
27                     context,
28                     MaterialPageRoute(
29                       builder: (context) => Servicos(
30                         //Dados montados no construtor da classe Servicos
31                         nome: 'Mecânico',
32                         endereco: 'Rua Santo Antonio, 15',
33                         contato: '(32)99196-8008',
34                         ), // Servicos
35                         ), // MaterialPageRoute
36               );
37             ],
38           ),
39           child: Text('Ir para Serviços'),
40         ),
41       ),
42     );
43   }
44 }
45
46 ///////////////Tela Serviços
47 class Servicos extends StatefulWidget {
48   final String? nome;
49   final String? endereco;
50   final String? contato;
51
52   //Criando um construtor para passa os valores
53   Servicos({this.nome, this.endereco, this.contato});
54
55   @override
56   _ServicosState createState() => _ServicosState();
57 }
58
59 class _ServicosState extends State<Servicos> {
60   @override
61   Widget build(BuildContext context) {
62     return Scaffold(
63       appBar: AppBar(
64         title: Text('Tela de Servicos'),
65         backgroundColor: Colors.blue,
66       ),
67     );
68   }
69 }
```

```
70 }
71
72 class _ServicosState extends State<Servicos> {
73   @override
74   Widget build(BuildContext context) {
75     return Scaffold(
76       appBar: AppBar(
77         title: Text('Tela de Servicos'),
78         backgroundColor: Colors.blue,
79       ),
80     );
81   }
82 }
```

Passando valores entre telas

- Ainda utilizando o Navigator(), vamos enviar dados de uma tela para outra. Para isso criamos construtores para definir os atributos das informações.



```
78     ), // AppBar
79     body: Center(
80       child: Container(
81         child: Column(
82           mainAxisAlignment: MainAxisAlignment.center,
83           children: [
84             Text(
85               'Tela de Serviços',
86               style: TextStyle(fontSize: 20),
87             ), // Text
88             Padding(
89               padding: EdgeInsets.all(20),
90               child: Column(
91                 children: [
92                   // O objeto Widget recupera os valores
93                   Text('Profissão: ${widget.nome}'),
94                   Text('Endereço: ${widget.endereco}'),
95                   Text('Contato: ${widget.contato}'),
96                 ],
97               ), // Column
98             ) // Padding
99           ],
100         ), // Column
101       ), // Container
102     ), // Center
103   ); // Scaffold
104 }
105 }
106 ///////////////Tela Contato
107 class Contato extends StatefulWidget {
108   @override
109   _ContatoState createState() => _ContatoState();
110 }
111
112 class _ContatoState extends State<Contato> {
113   @override
114   Widget build(BuildContext context) {
115     return Scaffold(
```

```
116     appBar: AppBar(
117       title: Text('Serviços'),
118       backgroundColor: Colors.amber,
119     ), // AppBar
120     body: Center(
121       child: Container(
122         child: Column(
123           mainAxisAlignment: MainAxisAlignment.center,
124           children: [
125             ElevatedButton(
126               onPressed: () {
127                 Navigator.pop(context);
128               },
129               child: Text('Voltar para a Home'),
130             ) // ElevatedButton
131           ],
132         ), // Column
133       ), // Container
134     ), // Center
135   ); // Scaffold
136 }
137 }
138 }
```

Atividade 3 Flutter e Inkscape

- App 1: construa um app com no mínimo 4 telas para apresentar os destaques da Olimpíada de Tóquio.
- App2: construa um app para apresentar 4 personagens da Disney.
- Inkscape: vetorize a logo das olimpíadas, a Fonte Disney e o desenho do Mickey.

