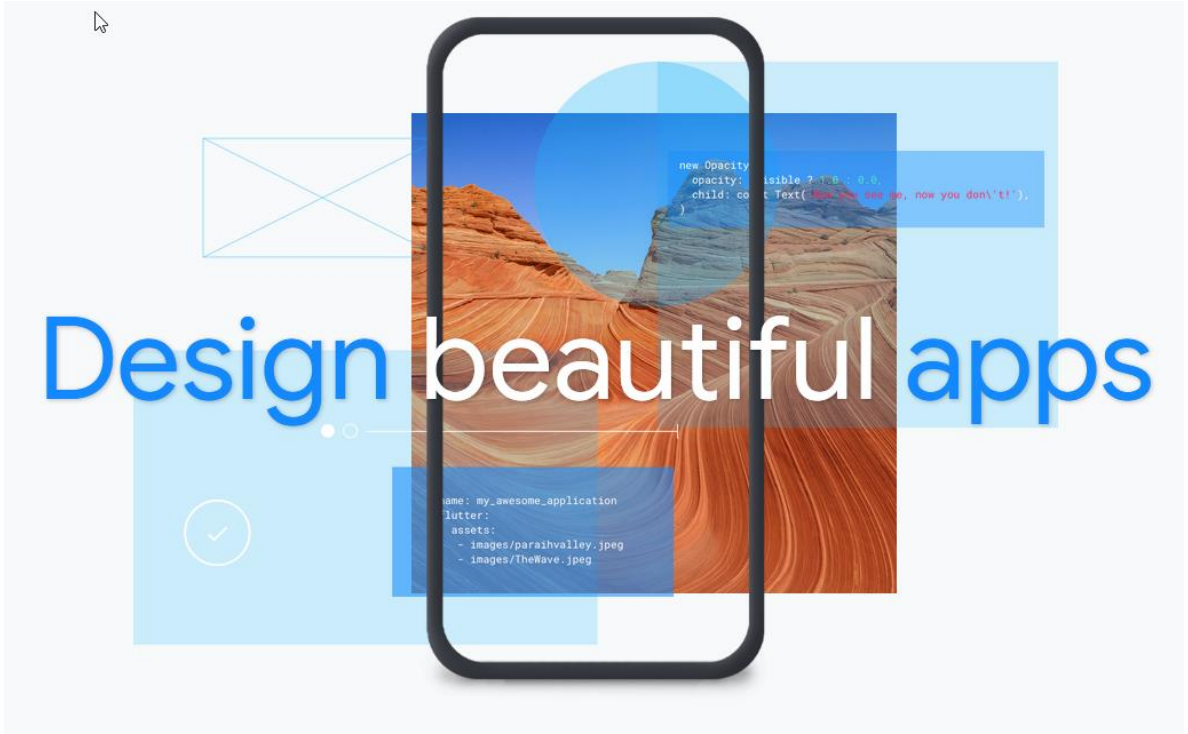
Transformando o futuro das pessoas
e as pessoas para o futuro.

#Senacfaz75

Senac

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.

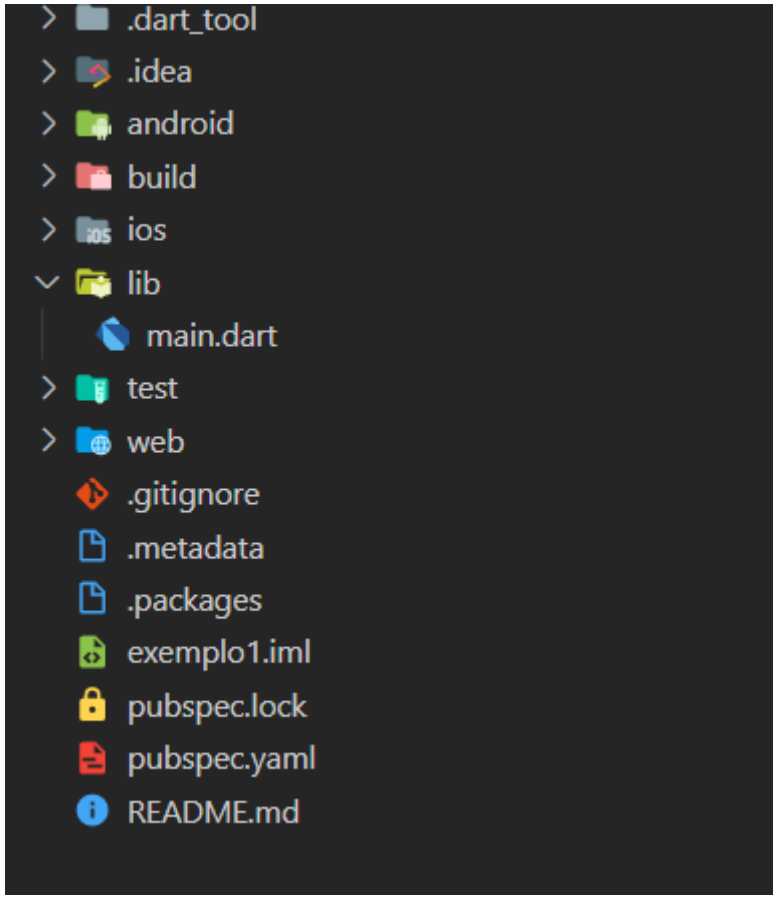
Características

- 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.

.metadadata: 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

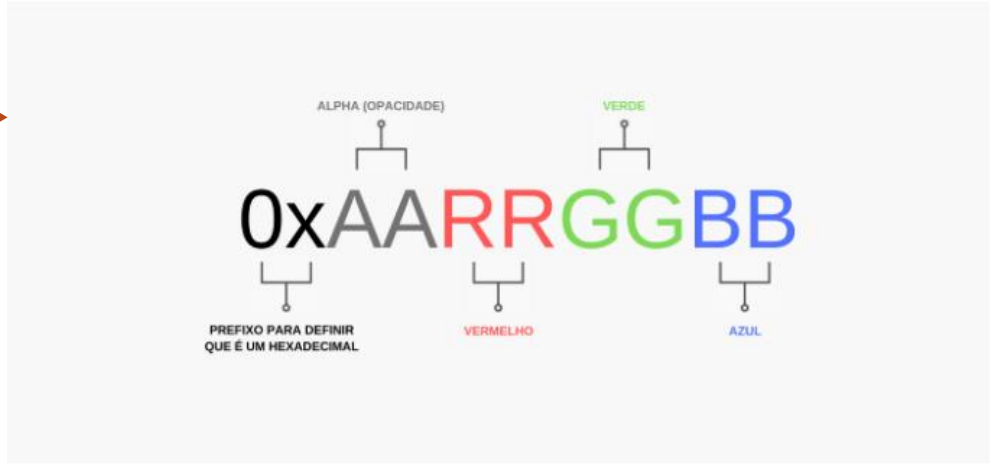
Title e home: Parâmetros nomeados de MaterialApp()

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

```dart
main01_primeiroPrograma.dart
lib > main01_primeiroPrograma.dart > ...
1  import 'package:flutter/material.dart';
2
   Run | Debug | Profile
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 > ...

```dart
//Inserindo um Widget Coluna
import 'package:flutter/material.dart';

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

```dart
//Formatação de texto
import 'package:flutter/material.dart';

Run | Debug | Profile
main() => runApp(MaterialApp(
  title: 'Sorteio MegaSena',
  //container é um envólucro para os conteúdos
  home: Container(
    color: Colors.white,

    //Definindo um filho pra o contaier
    child: Column(
      //Definindo um children para o Column
      children: [
        Text(
          'Esse é um exemplo de texto extenso para o widget',
          style: TextStyle(
            //Definindo as formatações do Texto
            fontSize: 40,
            fontStyle: FontStyle.normal,
            fontWeight: FontWeight.normal,

            //Espaçamento entre os caracteres
            letterSpacing: 2,

            //Retirar o sublinhado
            decoration: TextDecoration.none,

            //Aplicar na Fonte
            color: Colors.black
          ), // TextStyle
          ) // Text
      ],
    ), // Column
  ), // Container
)); // 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

```dart
lib > main.dart > main
import 'package:flutter/material.dart';
Run | Debug | Profile
main() => runApp(MaterialApp(
  //Removendo a faixa de debug
  debugShowCheckedModeBanner: false,
  title: 'Sorteio MegaSena',
  home: Container(
    //Defindo espaçamento internos
    //.all .fromLTRB .only
    //padding: EdgeInsets.all()

    padding: EdgeInsets.fromLTRB(20, 50, 20, 0),
    //Definindo os espaçamento externos
    //.all .fromLTRB .only
    //margin: EdgeInsets.all()

    margin: EdgeInsets.fromLTRB(20, 50, 20, 20),
    //Borda e cor do Container

    decoration: BoxDecoration(
      //Adiconando uma cor de backgound
      color: Colors.blueGrey,

      //Adicionando uma borda
      border: Border.all(
        width: 5,
        color: Colors.amber
      ), // Border.all
    ), // BoxDecoration

    //Adicionar um texto dentro do Container
    child: Column(
      children: [
        Text(
          'Texto exemplo para visualizar o preenchimento da tela do App',
          textAlign: TextAlign.center,
          style: TextStyle(
            fontSize: 25,
            fontWeight: FontWeight.normal,
            decoration: TextDecoration.none,
            color: Colors.white
          ), // TextStyle
        ) // Text
      ],
    ), // Column
  ), // Container
) // MaterialApp
);
```

Espaçamento no Widget

```dart
//Padding no Widget
import 'package:flutter/material.dart';

Run | Debug | Profile
main() => runApp(MaterialApp(
    //Removendo a faixa de debug
    debugShowCheckedModeBanner: false,
    title: 'Padding no Widget',
    home: Container(
      decoration: BoxDecoration(
          color: Colors.blueGrey,
          border: Border.all(width: 5, color: Colors.white)), // BoxDecoration
      child: Row(
        children: [
          Text(
            'Texto1',
            style: TextStyle(
                fontSize: 20,
                decoration: TextDecoration.none,
                color: Colors.redAccent), // TextStyle
          ), // Text

          //Padding diretamente no widget
          Padding(
            padding: EdgeInsets.all(25),
            child: Text(
              'Texto 2',
              style: TextStyle(
                fontSize: 20,
                decoration: TextDecoration.none,
                color: Colors.blueAccent

              ), // TextStyle
              ), // Text
          ), // Padding

          Text(
            'Texto 3',
            style: TextStyle(
              fontSize: 20,
              decoration: TextDecoration.none,
              color: Colors.amberAccent
            ), // TextStyle
          ), // Text
        ],
      ), // Row
    ) // Container
)); // 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

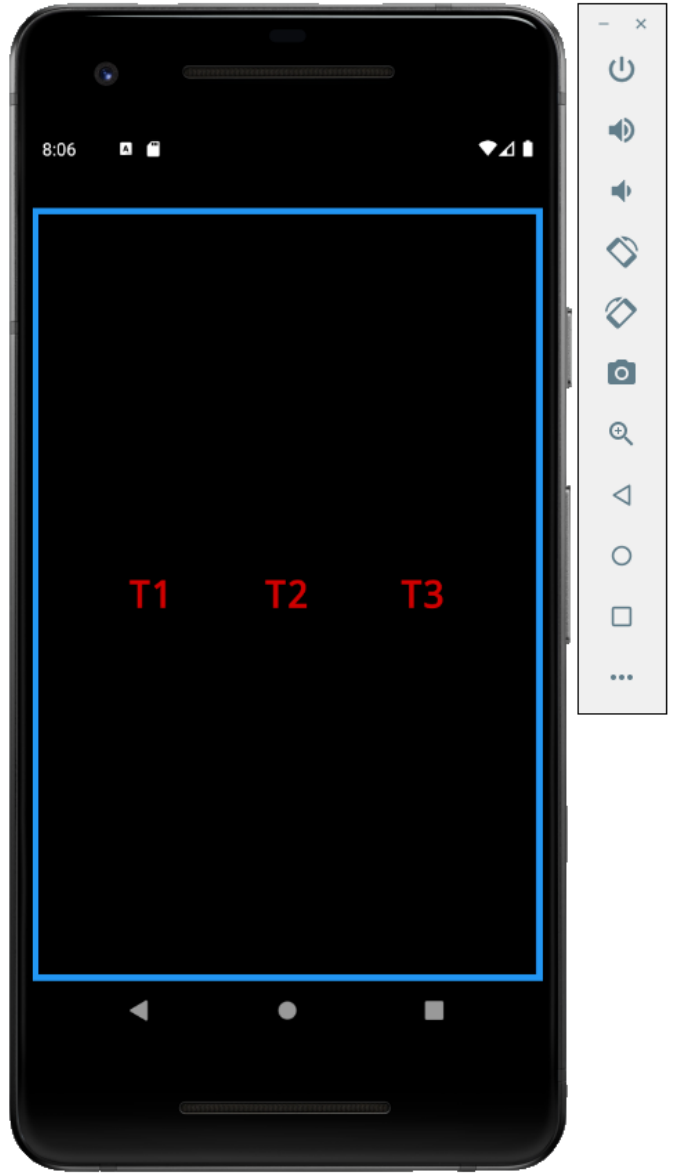
Alinhamentos

Alinhamento Principal (**mainAxisAlignment: MainAxisAlignment**)

mainAxisAlignment: MainAxisAlignment.start: Alinhamento no início
mainAxisAlignment: MainAxisAlignment.end: Alinhamento no Fim
mainAxisAlignment: MainAxisAlignment.center: Alinhamento no centro
mainAxisAlignment: MainAxisAlignment.spaceEnvely: 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

```dart
//Alinhamentos

import 'package:flutter/material.dart';

Run | Debug | Profile
main() => runApp(MaterialApp(
  //removendo a faixa de debug
  debugShowCheckedModeBanner: false,

  title: 'Alinhamentos',
  home: Container(
    //definindo margens
    margin: EdgeInsets.only(top: 60),

    //defindo borda e cor do container
    decoration: BoxDecoration(
      border: Border.all(
        width: 5,
        color: Colors.blue
      ) // Border.all
    ), // BoxDecoration

    //Criando um child
    child: Row(
      children: [

        //Criando textos
        Text(
          'T1',
          style: TextStyle(
            decoration: TextDecoration.none,
            fontSize: 30
          ), // TextStyle
        ), // Text

        //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
      ],

      //Alinhameto principal
      //mainAxisAlignment: MainAxisAlignment .start .center .end .spaceEnvely
      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_botoes.dart > main

```dart
//Trabalhando com botões

import 'package:flutter/material.dart';

Run | Debug | Profile
main() => runApp(MaterialApp(
      title: 'Trabalhando com botões',
      home: Container(
        //Margens
        margin: EdgeInsets.only(top: 60),

        //Espaçamento
        padding: EdgeInsets.all(20),
        color: Colors.blueGrey,

        child: Column(
          //Espaçamento principal
          mainAxisAlignment: MainAxisAlignment.center,

          children: [
            TextButton(
                style: TextButton.styleFrom(
                    //borda arredondada
                    shape: RoundedRectangleBorder(
                        //controlar o arredondamento
                        borderRadius: BorderRadius.circular(10)
                      ), // RoundedRectangleBorder
                    //cor de foreground
                    primary: Colors.pinkAccent,

                    //cor de background
                    backgroundColor: Colors.white,

                    //Espaçamento dentro do botão
                    padding:
                        EdgeInsets.symmetric(
                          horizontal: 32,
                          vertical: 32) // EdgeInsets.symmetric
                        ),
                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 Bckground
                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
73              //Cor do foreground
74              primary: Colors.black,
75
76              //Cor da borda
77              side: BorderSide(
78                width: 1,
79                color: Colors.orange
80              ) // BorderSide
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           onPressed: (){},
104           icon: Icon(
105             Icons.add_a_photo,
106             color: Colors.white,
107           ),  // Icon
108           label: Text('Label do botão')
109         ), // ElevatedButton.icon
110
111         const SizedBox(height: 20),
112         //Botão desativado
113         ElevatedButton(
114           style: ElevatedButton.styleFrom(
115             onSurface: Colors.amber
116           ),
117           onPressed: null,
118           child: Text(
119             'Botão desativado!',
120             style: TextStyle(
121               fontSize: 25
122             ), // TextStyle
123           ) // Text
124         ) // ElevatedButton
125       ],
126     ), // Column
127   ), // Container
128 )); // MaterialApp
129
```

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

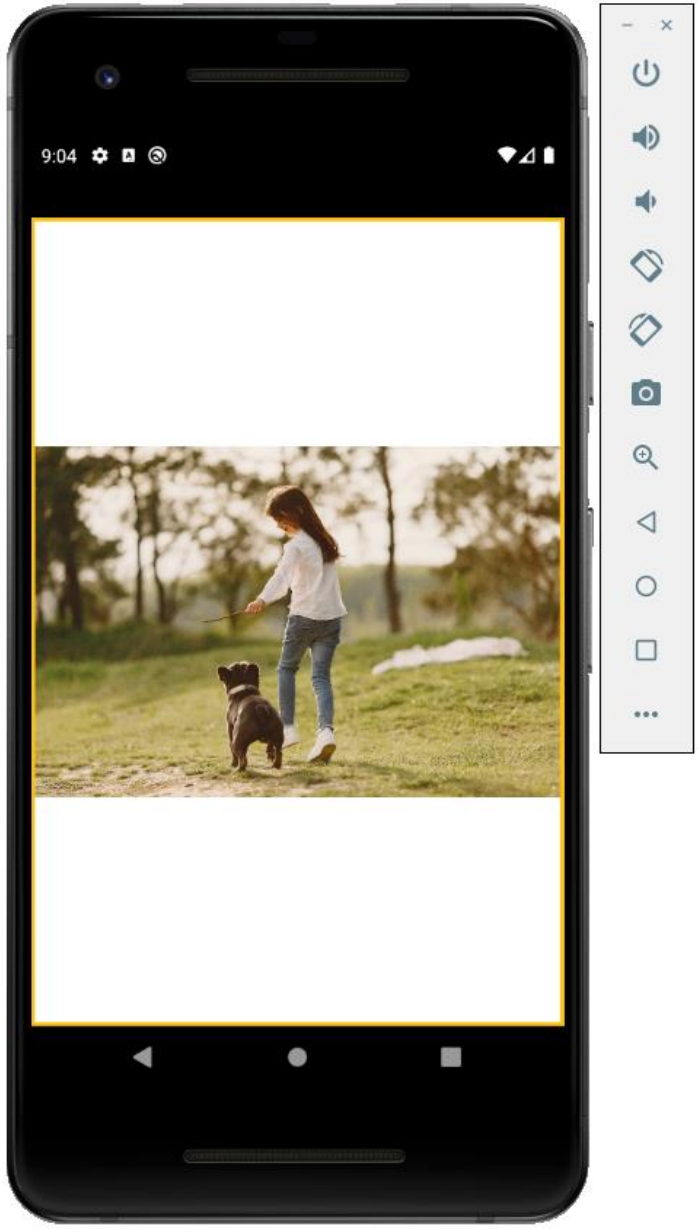
```
46
47  # To add assets to your application, add an assets section, like this:
48  # assets:
49  #   - images/a_dot_burr.jpeg
50  #   - images/a_dot_ham.jpeg
51  assets:
52    - images/passeio.jpg
53
54  # An image asset can refer to one or more resolution-specific "variants", see
55  # https://flutter.dev/assets-and-images/#resolution-aware.
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
    Run | Debug | Profile
3   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        //backgound
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      //cobre todo o espaçamento 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
```

Imagens

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

Run | Debug | Profile
main() => runApp(MaterialApp(
  debugShowCheckedModeBanner: false,
  title: 'Trabalhando com imagens',
  home: Container(
    margin: EdgeInsets.only(top: 60),

    decoration: BoxDecoration(
      //backgound
      color: Colors.white,

      //definir bordas
      border: Border.all(
        width: 3,
        color: Colors.amber
      ) // Border.all
    ), // BoxDecoration

    //Um child para o widget Image
    child: Image.asset(
      //Caminho da imagem
    'images/passeio.jpg',
    //Definição de preenchimento da imagem
    //cobre todo o espaçamento cortanto a imagem
    //fit: BoxFit.cover,
    //fit: BoxFit.contain //valor padrão
    //fit: BoxFit.fill //Preenche com distorção
    //fit: BoxFit.fitHeight, //preencher pela altura (Largura na proporção)
    //fit: BoxFit.fitWidth, //preencher pela largura (altura na proporção)
    //fit: BoxFit.none //Mantem a imagem original
    fit: BoxFit.contain //Mantem a imagem original
    ), // Image.asset
  ), // Container
)); // MaterialApp
```

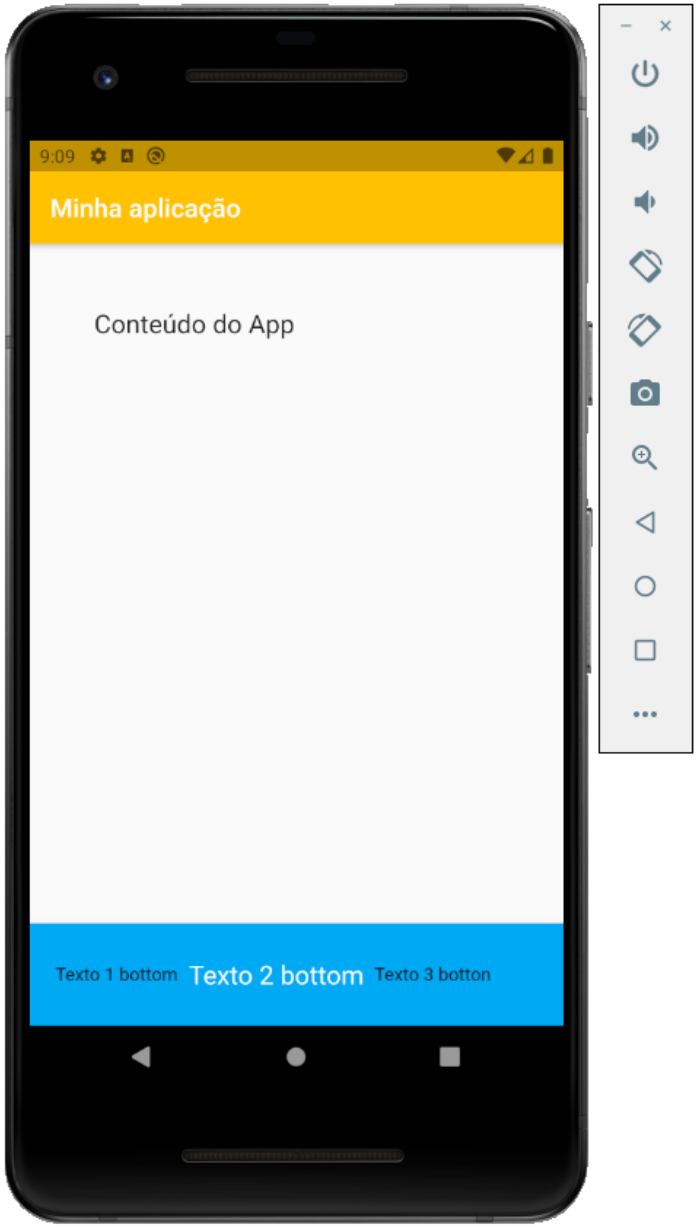


Scaffold

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 > ...

```dart
import 'package:flutter/material.dart';
Run | Debug | Profile
main() => runApp(MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        //Criando uma estrutura
        //Divisão de 3 áreas: Título, corpo e rodapé
        appBar: AppBar(
          //Título
          title: Text('Minha aplicação'),
          backgroundColor: Colors.amber,
        ), // AppBar
        body: Padding(
          padding: EdgeInsets.all(50),
          child: Text(
            'Conteúdo do App',
            style: TextStyle(fontSize: 20),
          ), // Text
        ), // Padding
        bottomNavigationBar: BottomAppBar(
          //Aplicando uma cor
          color: Colors.lightBlue,
          child: Padding(
            padding: EdgeInsets.all(20),
            child: Row(
              //Aray de widgets
              children: [
                Text('Texto 1 bottom'),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Text(
                    'Texto 2 bottom',
                    style: TextStyle(fontSize: 20, color: Colors.white),
                  ), // Text
                ), // Padding
                Text('Texto 3 botton')
              ],
            ), // Row
          ), // Padding
        ), // BottomAppBar
      ), // Scaffold
    )); // 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 > ...

```dart
//Entrada de dados

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

Run | Debug | Profile
main() => runApp(MaterialApp(
      home: Entrada(),
    )); // MaterialApp

//Criando um estado
class Entrada extends StatefulWidget {
  @override
  _EntradaState createState() => _EntradaState();
}

class _EntradaState extends State<Entrada> {
  //Iniciar um controlador
  TextEditingController _textEditingController = TextEditingController();

  //Uma variável de saída na tela
  String _resposta = 'Resultado';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Entrada de dados'),
        backgroundColor: Colors.amber,
      ), // AppBar
      body: Column(
        children: [
          Padding(
            padding: EdgeInsets.all(20),
```

```dart
          //Entrada de dados
          child: TextField(
            //tipos: Text, number, emailAddress, datetime
            keyboardType: TextInputType.text,

            decoration: InputDecoration(labelText: 'Digite alguma coisa'),

            //Habilitando e desabilitando um campo
            enabled: true,

            //Estabelecendo a quantidade de caracteres
            maxLength: 10,

            //Controlando o comportamento do Maxlegth
            //enforced: libera a digitação
            //none: bloqueia a digitacao acima do permitido
            maxLengthEnforcement: MaxLengthEnforcement.enforced,

            //Formatando o texto
            style: TextStyle(
              fontSize: 25,
              color: Colors.red,
              decoration: TextDecoration.none
              ), // TextStyle

            //Escondendo o texto digitado (true/false)
            obscureText: false,
            //Capturando o texto
            controller: _textEditingController, //Variável para controle
          ), // TextField

        ), // Padding
        ElevatedButton(
          onPressed: () {
            setState(() {
              _resposta = _textEditingController.text;
            });
          },
```

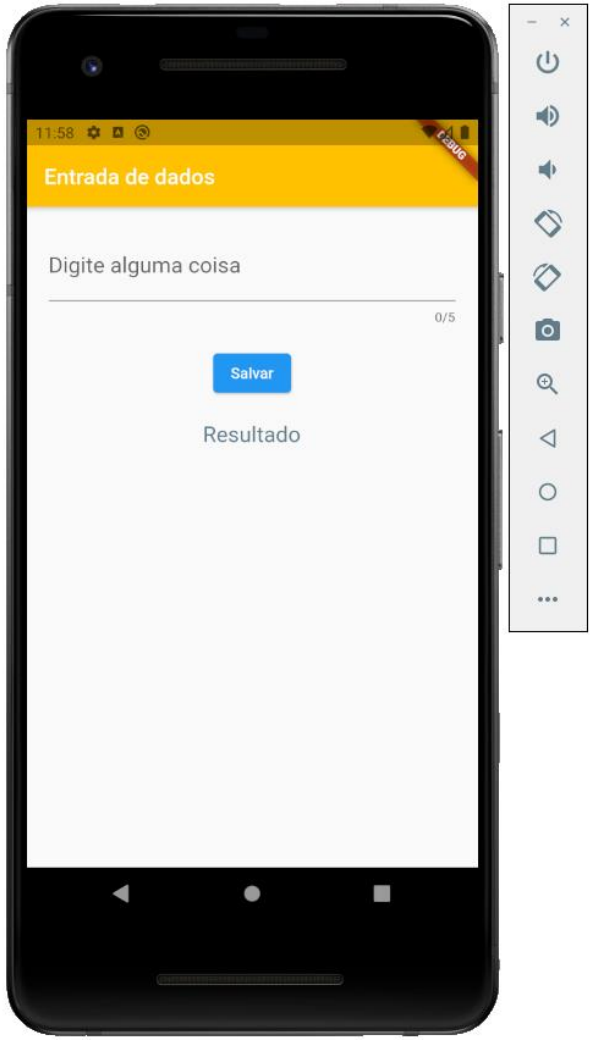
Entrada de Dados

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  }
```

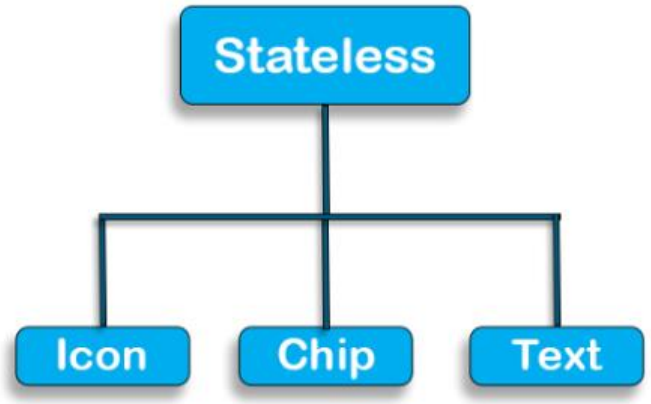


Stateless

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.

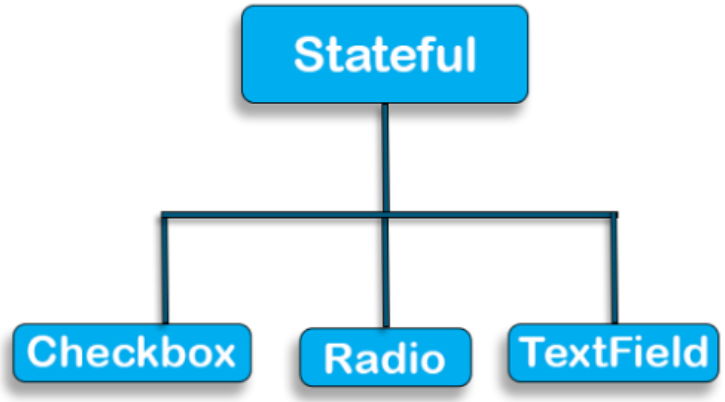
Exemplos de widgets sem estado são os seguintes:



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 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 :

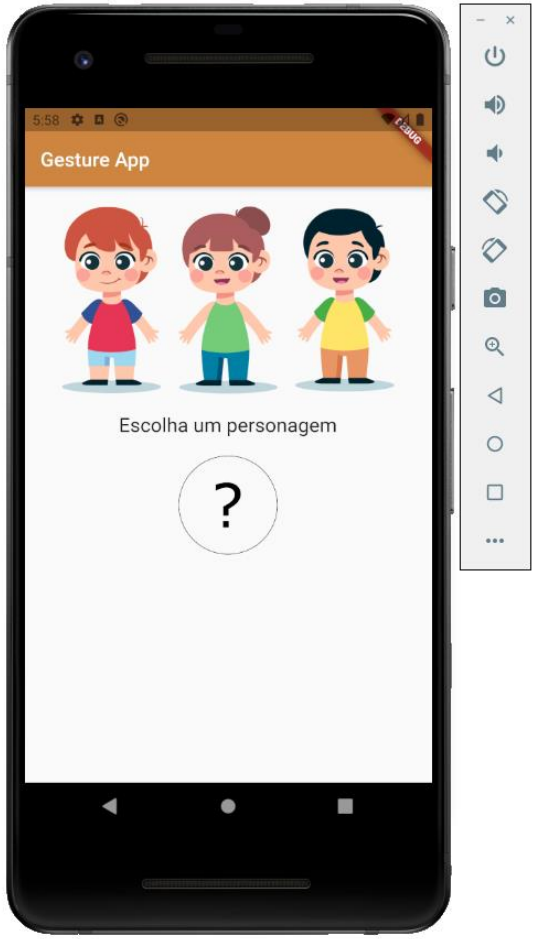
Observações:
Enviar a pasta lib, a pasta images
o arquivo .yaml compactados para o email do professor.

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,




```dart
//Gesture Detector
//Detectando toque na tela

import 'package:flutter/material.dart';

Run | Debug | Profile
main() => runApp(MaterialApp(
      home: Gesture(),
    )); // MaterialApp

class Gesture extends StatefulWidget {
  @override
  _GestureState createState() => _GestureState();
}
class _GestureState extends State<Gesture> {

  String _personagem = '';
  var _imagem = AssetImage('images/imagem.png');

  personagemEscolhido(String personagem) {
    if (personagem == 'joao') {
      _personagem = 'Você escolheu o João!';
      _imagem = AssetImage('images/jo.png');
    } else if(personagem == 'juliana') {
      _personagem = 'Você escolheu a Juliana!';
      _imagem = AssetImage('images/ju.png');
    } else {
      _personagem = 'Você escolheu o José!';
      _imagem = AssetImage('images/ze.png');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Gesture App'),
        backgroundColor: Color(0xffcd853f),
      ), // AppBar
      body: Padding(
        padding: EdgeInsets.all(20),
        //Primeiro uma column
        child: Column(
          children: [
            //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,

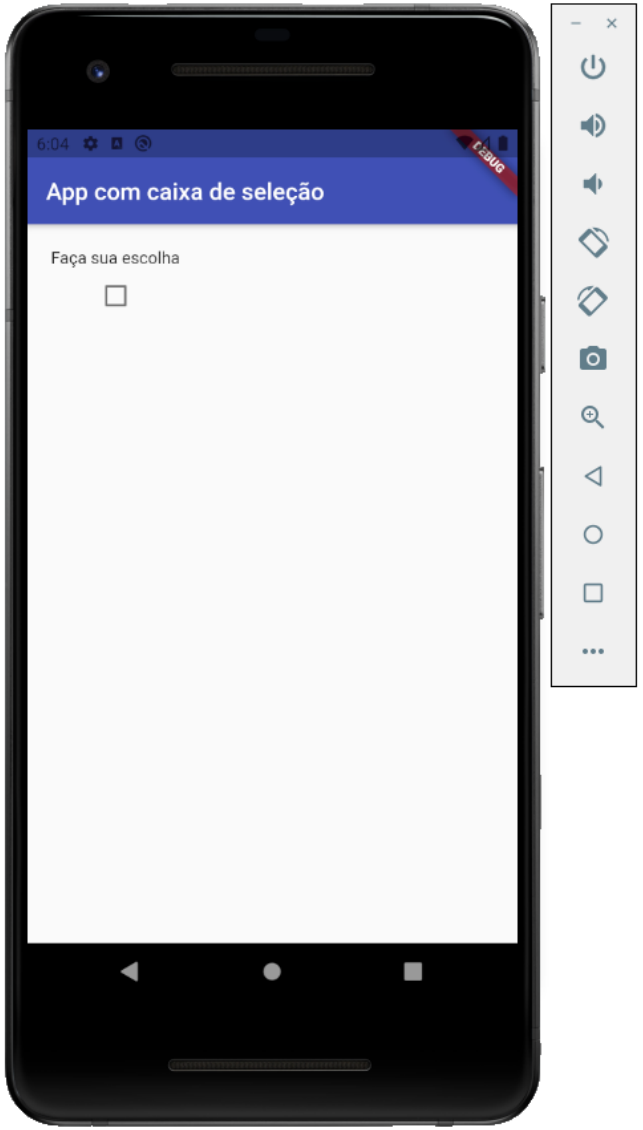
```
45              Row(
46                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
47                children: [
48                  //Detectando o toque na imagem
49                  GestureDetector(
50                    //Evento de 1 toque na tela
51                    onTap: () {
52                      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                   Padding(
112                     padding: const EdgeInsets.only(top: 20,
113                     ), // EdgeInsets.only
114                     child: Text(
115                       _personagem,
116                       style: TextStyle(
117                         fontSize: 30,
118                         color: Colors.orange,
119                       ), // TextStyle
120                     ), // Text
121                   ) // Padding
122                 ],
123               ), // Column
124             ) // Padding
125           ],
126         ), // Column
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.

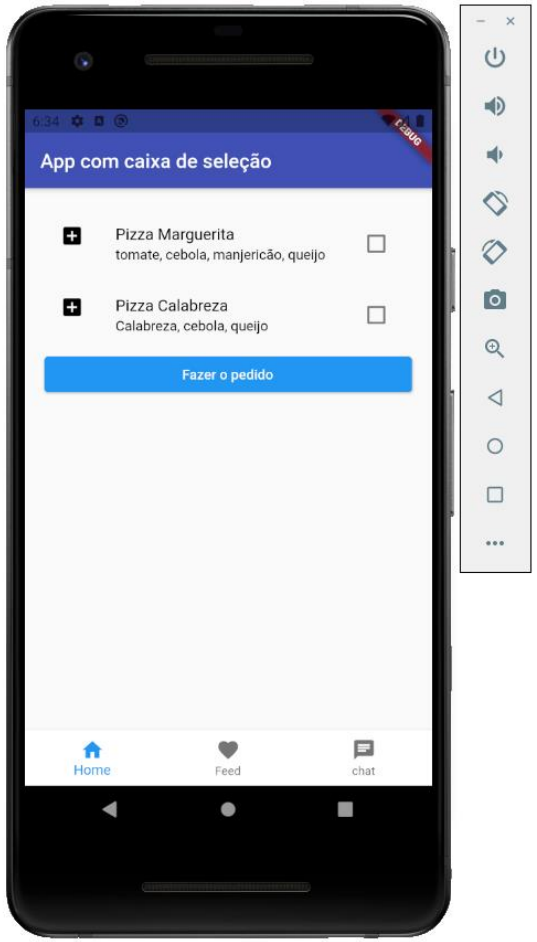
A maioria dos widgets que usam uma caixa de seleção ouvirá o retorno de chamada onChanged e reconstruirá a caixa de seleção com um novo valor para atualizar sua aparência.




```dart
lib > main14_checkbox_1.dart > ...
import 'package:flutter/material.dart';
Run | Debug | Profile
main() => runApp(MaterialApp(
      home: MeuCheckbox(),
    )); // MaterialApp

class MeuCheckbox extends StatefulWidget {
  @override
  _MeuCheckboxState createState() => _MeuCheckboxState();
}

class _MeuCheckboxState extends State<MeuCheckbox> {
  bool _selecaoCheckbox = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('App com caixa de seleção'),
        backgroundColor: Colors.indigo,
      ), // AppBar
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            Text('Faça sua escolha'),
            Checkbox(
                //valor inicial do checkbox
                value: _selecaoCheckbox, //desmarcado
                onChanged: (bool? valor) {
                  setState(() {
                    _selecaoCheckbox = valor!;
                  });
                }) // Checkbox
          ],
        ), // Column
      ), // Padding
    ); // Scaffold
  }
}
```

CheckboxListTile()

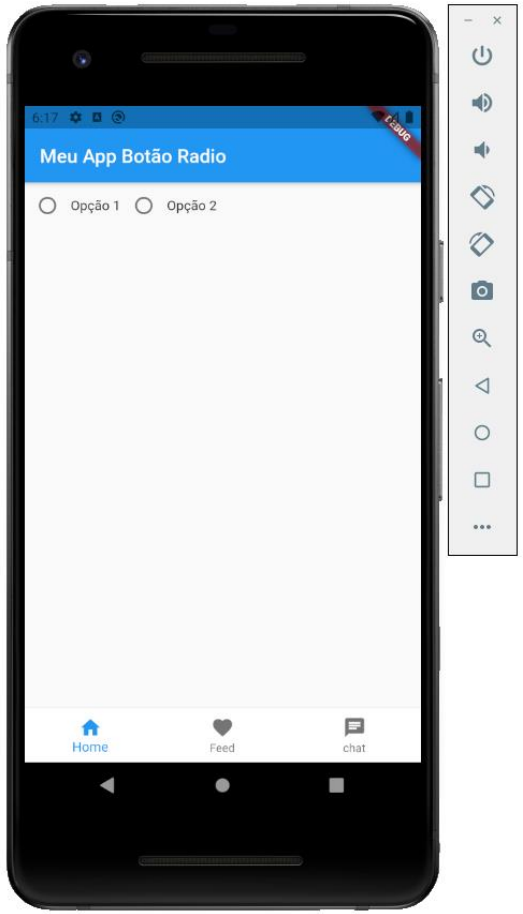
CheckboxListTile é um widget que combina uma caixa de seleção e um bloco de lista. Ele permite que você crie uma caixa de seleção junto com o título, subtítulo e ícone sem a necessidade de criar um widget separado para cada parte.




```dart
import 'package:flutter/material.dart';
Run | Debug | Profile
main() => runApp(MaterialApp(
      home: MeuCheckbox(),
    )); // MaterialApp

class MeuCheckbox extends StatefulWidget {
  @override
  _MeuCheckboxState createState() => _MeuCheckboxState();
}

class _MeuCheckboxState extends State<MeuCheckbox> {
  bool _pizzaMarguerita = false;
  bool _pizzaCalabreza = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('App com caixa de seleção'),
        backgroundColor: Colors.indigo,
      ), // AppBar
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          //Para esticar o conteúdo
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [

            CheckboxListTile(
                title: Text('Pizza Marguerita'),
                subtitle: Text('tomate, cebola, manjericão, queijo'),
                //Cor do checkbox
                activeColor: Colors.black,
                //cor do texto
                selected: true,
                //definindo um ícone
                secondary: Icon(Icons.add_box),
                value: _pizzaMarguerita,
                onChanged: (bool? valor) {
                  setState(() {
                    _pizzaMarguerita = valor!;
                  });
                }), // CheckboxListTile
            CheckboxListTile(
                title: Text('Pizza Calabreza'),
                subtitle: Text('Calabreza, cebola, queijo'),
                //Cor do checkbox
                activeColor: Colors.black,
                //cor do texto
                selected: true,
                //definindo um ícone
                secondary: Icon(Icons.add_box),
                value: _pizzaCalabreza,
                onChanged: (bool? valor) {
                  setState(() {
                    _pizzaCalabreza = valor!;
                  });
                }), // CheckboxListTile
            ElevatedButton(
              onPressed: () {
                print('Pedido:');
                print('Pizza Marguerita: $_pizzaMarguerita');
                print('PizzaCalabreza: $_pizzaCalabreza');
              },
              child: Text('Fazer o pedido'),
            ) // ElevatedButton
          ],
        ), // Column
      ), // Padding
      bottomNavigationBar: BottomNavigationBar(
          backgroundColor: Colors.white,
          currentIndex: 0,
          items: [
            BottomNavigationBarItem(
              icon: Icon(Icons.home),
              label: 'Home',
            ), // BottomNavigationBarItem
            BottomNavigationBarItem(
              icon: Icon(Icons.favorite),
              label: 'Feed',
            ), // BottomNavigationBarItem
            BottomNavigationBarItem(
              icon: Icon(Icons.chat),
              label: 'chat',
            ) // BottomNavigationBarItem
          ]), // BottomNavigationBar
    ); // Scaffold
  }
}
```

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.

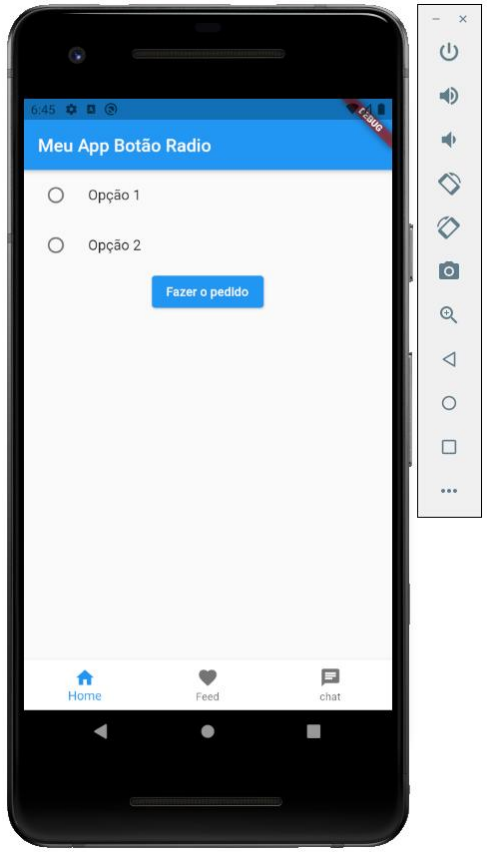



```dart
import 'package:flutter/material.dart';
Run | Debug | Profile
void main() => runApp(MaterialApp(
      home: BotaoRadio(),
    )); // MaterialApp

class BotaoRadio extends StatefulWidget {
  @override
  _BotaoRadioState createState() => _BotaoRadioState();
}

class _BotaoRadioState extends State<BotaoRadio> {
  int? _opcao;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Meu App Botão Radio'),
          backgroundColor: Colors.blue,
        ), // AppBar
        body: Container(
          child: Row(
            children: [
              Radio(
                  value: 1,
                  groupValue: _opcao,
                  onChanged: (int? selecao) {
                    print('Seleção: $selecao');
                    setState(() {
                      _opcao = selecao;
                    });
                  }), // Radio
              Text('Opção 1'),
              Radio(
                  value: 2,
                  groupValue: _opcao,
                  onChanged: (int? selecao) {
                    print('seleção: $selecao');
                    setState(() {
                      _opcao = selecao;
                    });
                  }), // Radio
              Text('Opção 2')
            ],
          ), // Row
        ), // Container
        bottomNavigationBar: BottomNavigationBar(
        backgroundColor: Colors.white,
        currentIndex: 0,
        items: [
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.favorite),
            label: 'Feed',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.chat),
            label: 'chat',
          ) // BottomNavigationBarItem
        ]), // BottomNavigationBar
      ); // Scaffold
  }
}
```

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.



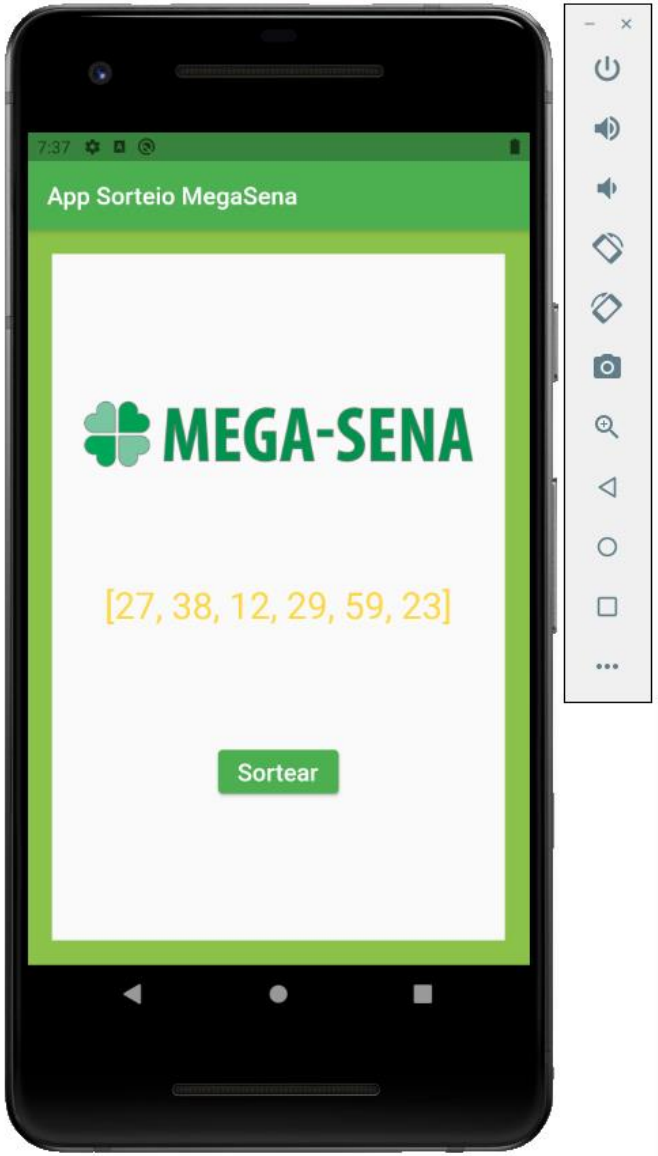

```dart
import 'package:flutter/material.dart';
Run | Debug | Profile
void main() => runApp(MaterialApp(
      home: BotaoRadio(),
    )); // MaterialApp

class BotaoRadio extends StatefulWidget {
  @override
  _BotaoRadioState createState() => _BotaoRadioState();
}

class _BotaoRadioState extends State<BotaoRadio> {
  //Variável para capturar o valor do botão
  int? _opcao;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Meu App Botão Radio'),
        backgroundColor: Colors.blue,
      ), // AppBar
      body: Container(
        child: Column(
          children: [
            RadioListTile(
                title: Text('Opção 1'),
                value: 1,
                groupValue: _opcao,
                onChanged: (int? selecao) {
                  setState(() {
                    _opcao = selecao;
                  });
                }), // RadioListTile
            RadioListTile(
                title: Text('Opção 2'),
                value: 2,
                groupValue: _opcao,
                onChanged: (int? selecao) {
                  setState(() {
                    _opcao = selecao;
                  });
                }), // RadioListTile
```

```dart
                ElevatedButton(
                    onPressed: (){
                      print('Seleção:');
                      print('Opção escolinha: $_opcao');
                    },
                    child: Text('Fazer o pedido')
                ) // ElevatedButton
            ],
          ), // Column
        ), // Container
        bottomNavigationBar: BottomNavigationBar(
            backgroundColor: Colors.white,
            currentIndex: 0,
            items: [
              BottomNavigationBarItem(
                icon: Icon(Icons.home),
                label: 'Home',
              ), // BottomNavigationBarItem
              BottomNavigationBarItem(
                icon: Icon(Icons.favorite),
                label: 'Feed',
              ), // BottomNavigationBarItem
              BottomNavigationBarItem(
                icon: Icon(Icons.chat),
                label: 'chat',
              ) // BottomNavigationBarItem
            ]), // BottomNavigationBar
      ); // Scaffold
    }
  }
```

Atividade 2 Flutter

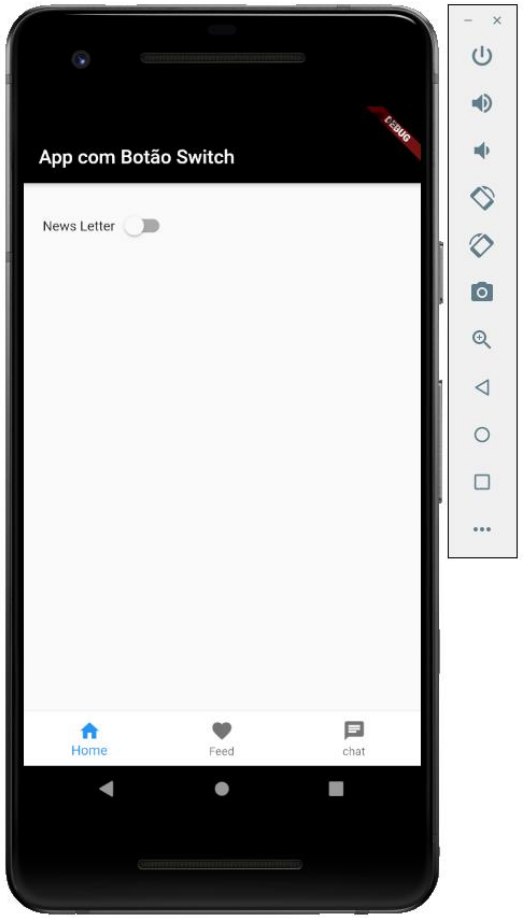
Partindo do pressuposto de que vocês receberam um job de um cliente e baseando-se no design ao lado, fornecido por ele, desenvolva as aplicações :

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.

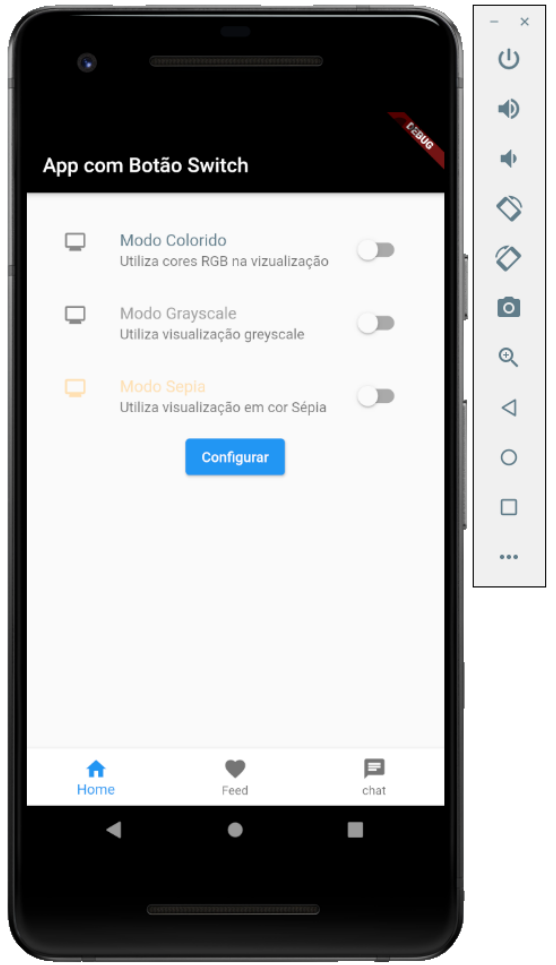



```dart
import 'package:flutter/material.dart';

Run | Debug | Profile
main() => runApp(MaterialApp(
      home: AppSwitch(),
    )); // MaterialApp

class AppSwitch extends StatefulWidget {
  @override
  _AppSwitchState createState() => _AppSwitchState();
}

class _AppSwitchState extends State<AppSwitch> {
  //Definindo uma variável para o estado do switch
  bool _selecaoEstado = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('App com Botão Switch'),
        backgroundColor: Colors.black,
      ), // AppBar
      body: Container(
        child: Padding(
          padding: const EdgeInsets.all(20.0),
          child: Row(
            children: [
              Text('News Letter'),
              //Cria um botão Switch
              Switch(
                  value: _selecaoEstado,
                  onChanged: (bool estado) {
                    setState(() {
                      _selecaoEstado = estado;
                    });
                  }), // Switch
            ],
          ), // Row
        ), // Padding
      ), // Container
      bottomNavigationBar: BottomNavigationBar(
          backgroundColor: Colors.white,
          currentIndex: 0,
          items: [
            BottomNavigationBarItem(
              icon: Icon(Icons.home),
              label: 'Home',
            ), // BottomNavigationBarItem
            BottomNavigationBarItem(
              icon: Icon(Icons.favorite),
              label: 'Feed',
            ), // BottomNavigationBarItem
            BottomNavigationBarItem(
              icon: Icon(Icons.chat),
              label: 'chat',
            ) // BottomNavigationBarItem
          ]), // BottomNavigationBar
    ); // Scaffold
  }
}
```

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.

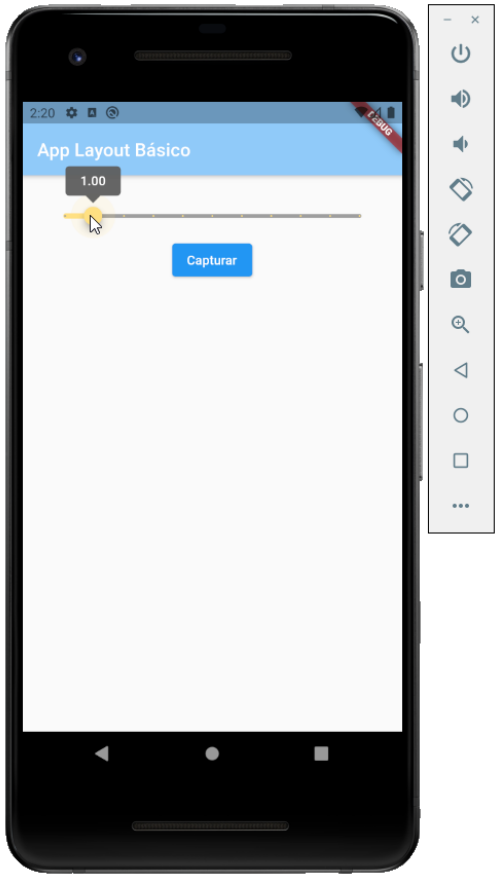
```dart
import 'package:flutter/material.dart';

Run | Debug | Profile
main() => runApp(MaterialApp(
      home: AppSwitch(),
    )); // MaterialApp

class AppSwitch extends StatefulWidget {
  @override
  _AppSwitchState createState() => _AppSwitchState();
}

class _AppSwitchState extends State<AppSwitch> {
  //Definindo uma variável para o estado do switch
  bool _modoCor = false;
  bool _modoTomCinza = false;
  bool _modoSepia = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('App com Botão Switch'),
        backgroundColor: Colors.black,
      ), // AppBar
      body: Container(
        child: Padding(
          padding: const EdgeInsets.all(20.0),
          child: Column(
            children: [
              //Cria um botão Switch
              SwitchListTile(
                  title: Text(
                    'Modo Colorido',
                    style: TextStyle(color: Colors.blueGrey),
                  ), // Text
                  subtitle: Text('Utiliza cores RGB na vizualização'),
                  activeColor: Colors.blueGrey,
                  secondary: Icon(Icons.monitor),
                  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: () {
                    print('Modo Cor: $_modoCor');
                    print('Modo Tons de Cinza: $_modoTomCinza');
                    print('Modo Modo Sépia: $_modoSepia');
                  },
                  child: Text('Configurar')) // ElevatedButton
            ],
          ), // Column
        ), // Padding
      ), // 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.

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

Run | Debug | Profile
main() => runApp(MaterialApp(
      home: AppSlider(),
    )); // MaterialApp

class AppSlider extends StatefulWidget {
  @override
  _AppSliderState createState() => _AppSliderState();
}

class _AppSliderState extends State<AppSlider> {
  double _valorSlider = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('App Layout Básico'),
        backgroundColor: Colors.blue.shade200,
      ), // AppBar
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Container(
          child: Column(
            children: [
              Slider(
                  value: _valorSlider,
                  min: 0,
                  max: 10,
                  //Definindo um label no slider
                  label: '${_valorSlider.toStringAsFixed(2)}',
                  divisions: 10,
                  activeColor: Colors.amber.shade200,
                  inactiveColor: Colors.grey,
                  onChanged: (double valorAlterado) {
                    setState(() {
                      _valorSlider = valorAlterado;
                    });
                  }), // Slider
              ElevatedButton(
                onPressed: (){
                  print(
                    'Valor selecionado: ${_valorSlider.toStringAsFixed(2)}');
                }, child: Text('Capturar')) // ElevatedButton
            ],
          ), // Column
        ), // Container
      ), // Padding
    ); // Scaffold
  }
}
```



Navegação

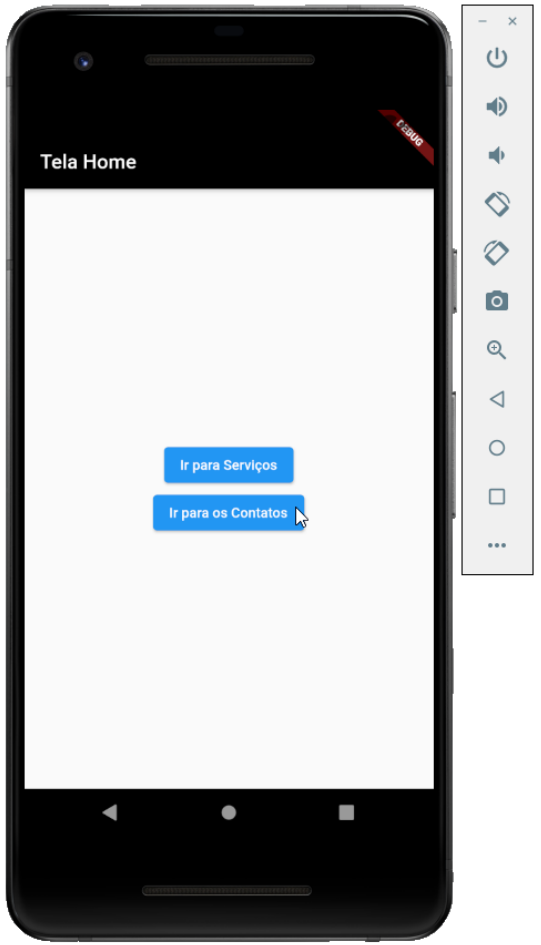
- Navigator(): Classe responsável pela Navegação
- push(): Abre uma tela
- pop(): Fecha a tela corrente
- Parâmetro context: Contexto da Aplicação, em qual tela está, já foi definido logo no começo da aplicação
- Parâmetro route: A rota de destino definida com MaterialPageRoute().
- Parâmetro builder: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

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

Run | Debug | Profile
main() => runApp(MaterialApp(
      home: Home(),
    )); // MaterialApp

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}
//////////////Tela Home
class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tela Home'),
        backgroundColor: Colors.black,
      ), // AppBar
      body: Center(
        child: Container(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              ElevatedButton(
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => Servicos(),
                    ), // MaterialPageRoute
                  );
                },
                child: Text('Ir para Serviços'),
              ), // ElevatedButton
              ElevatedButton(
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => Contato(),
                    ), // MaterialPageRoute
                  );
                },
                child: Text('Ir para os Contatos'),
              ) // ElevatedButton
            ],
          ), // Column
        ), // Container
      ), // Center
    ); // Scaffold
  }
}
//////////////Tela Serviços
class Servicos extends StatefulWidget {
  @override
  _ServicosState createState() => _ServicosState();
}

class _ServicosState extends State<Servicos> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tela de Servicos'),
        backgroundColor: Colors.blue,
      ), // AppBar
      body: Center(
        child: Container(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'Tela de Serviços',
                style: TextStyle(fontSize: 20),
              ) // Text
            ],
```

Navegação

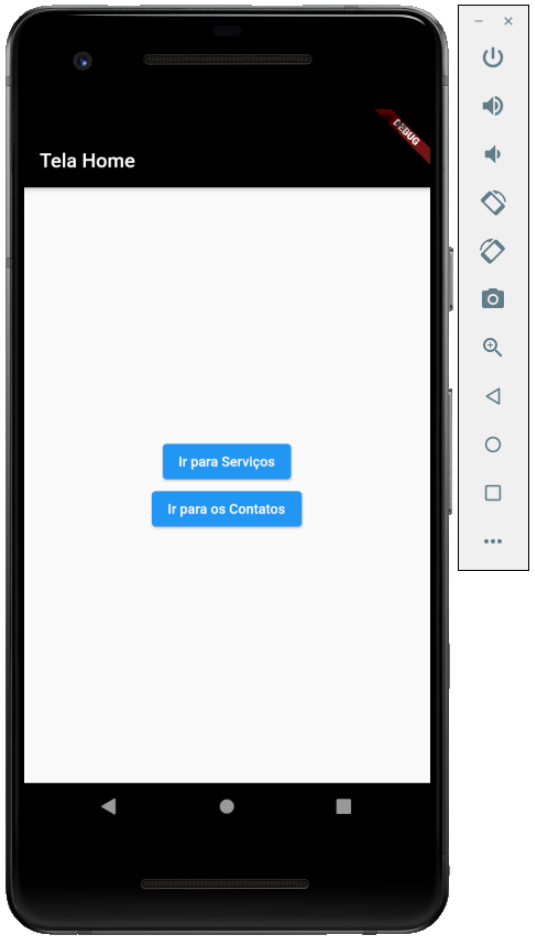
- Navigator(): Classe responsável pela Navegação
- push(): Abre uma tela
- pop(): Fecha a tela corrente
- Parâmetro context: Contexto da Aplicação, em qual tela está, já foi definido logo no começo da aplicação
- Parâmetro route: A rota de destino definida com MaterialPageRoute().
- Parâmetro builder: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

```dart
            ), // Column
          ), // Container
        ), // Center
      ); // Scaffold
    }
  }

  //////////////Tela Contato
  class Contato extends StatefulWidget {
    @override
    _ContatoState createState() => _ContatoState();
  }

  class _ContatoState extends State<Contato> {
    @override
    Widget build(BuildContext context) {
      return Scaffold(
        appBar: AppBar(
          title: Text('Serviços'),
          backgroundColor: Colors.amber,
        ), // AppBar
        body: Center(
          child: Container(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                ElevatedButton(
                  onPressed: () {
                    Navigator.pop(context);
                  },
                  child: Text('Voltar para a Home'),
                ) // ElevatedButton
              ],
            ), // Column
          ), // Container
        ), // Center
      ); // Scaffold
    }
  }
```



Passando valores entre telas

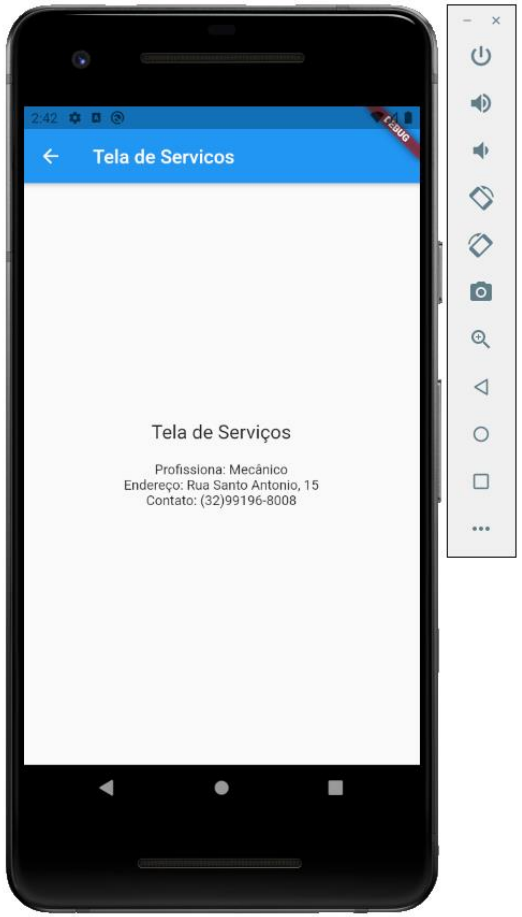
- Ainda utilizando o Navigator(), vamos enviar dados de uma tela para outra. Para isso criamos construtores para definir os atributos das informações.




```dart
import 'package:flutter/material.dart';
Run | Debug | Profile
main() => runApp(MaterialApp(
      home: Home(),
    )); // MaterialApp

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}
//////////////Tela Home
class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tela Home'),
        backgroundColor: Colors.black,
      ), // AppBar
      body: Center(
        child: Container(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              ElevatedButton(
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => Servicos(
                        //Dados montados no construtor da classe Serviços
                        nome: 'Mecânico',
                        endereco: 'Rua Santo Antonio, 15',
                        contato: '(32)99196-8008',
                      ), // Servicos
                    ), // MaterialPageRoute
                  );
                },
                child: Text('Ir para Serviços'),
              ), // ElevatedButton
              ElevatedButton(
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => Contato(),
                    ), // MaterialPageRoute
                  );
                },
                child: Text('Ir para os Contatos'),
              ) // ElevatedButton
            ],
          ), // Column
        ), // Container
      ), // Center
    ); // Scaffold
  }
}

//////////////Tela Serviços
class Servicos extends StatefulWidget {
  final String? nome;
  final String? endereco;
  final String? contato;

  //Criando um construtor para passa os valores
  Servicos({this.nome, this.endereco, this.contato});

  @override
  _ServicosState createState() => _ServicosState();
}
class _ServicosState extends State<Servicos> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tela de Servicos'),
        backgroundColor: Colors.blue,
```

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.




```dart
      ), // AppBar
      body: Center(
        child: Container(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'Tela de Serviços',
                style: TextStyle(fontSize: 20),
              ), // Text
              Padding(
                padding: EdgeInsets.all(20),
                child: Column(
                  children: [
                    //O objeto Widget recupera os valores
                    Text('Profissiona: ${widget.nome}'),
                    Text('Endereço: ${widget.endereco}'),
                    Text('Contato: ${widget.contato}'),
                  ],
                ), // Column
              ) // Padding
            ],
          ), // Column
        ), // Container
      ), // Center
    ); // Scaffold
  }
}
//////////////Tela Contato
class Contato extends StatefulWidget {
  @override
  _ContatoState createState() => _ContatoState();
}

class _ContatoState extends State<Contato> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```dart
      appBar: AppBar(
        title: Text('Serviços'),
        backgroundColor: Colors.amber,
      ), // AppBar
      body: Center(
        child: Container(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              ElevatedButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: Text('Voltar para a Home'),
              ) // ElevatedButton
            ],
          ), // Column
        ), // Container
      ), // Center
    ); // Scaffold
  }
}

```

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.