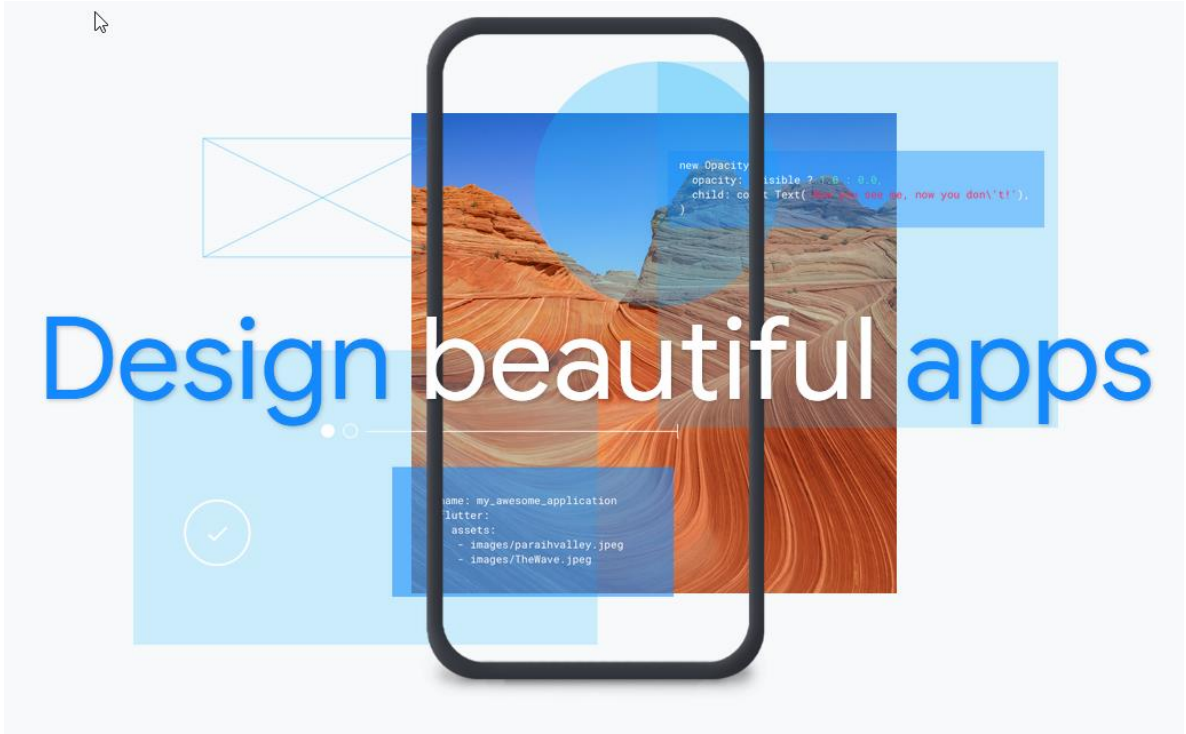
Transformando o futuro das pessoas
e as pessoas para o futuro.

#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.

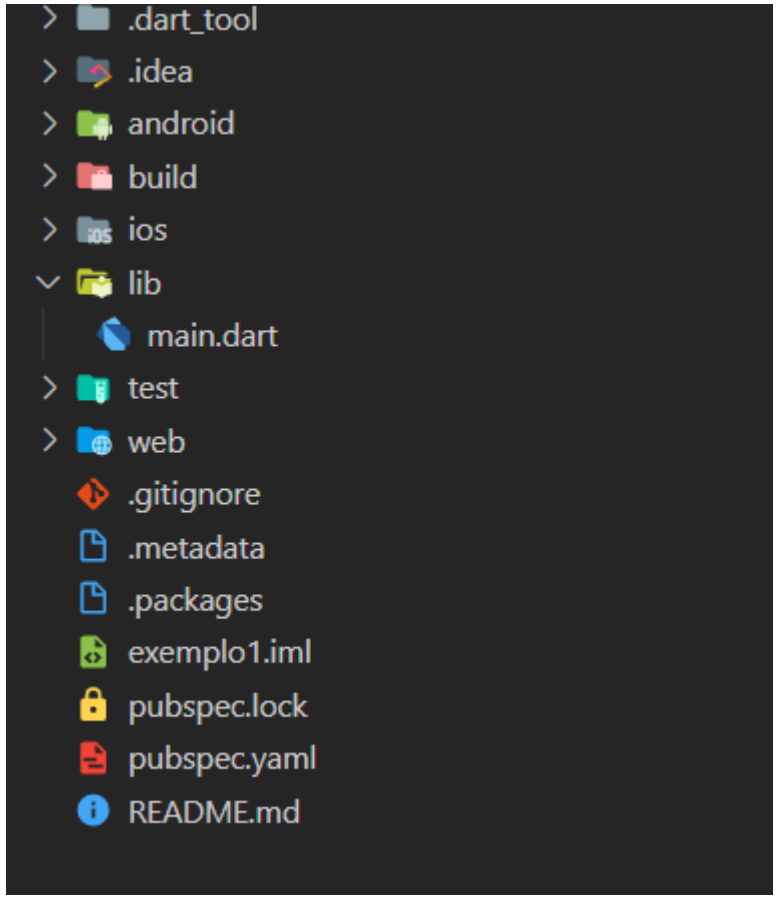
Características

- 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

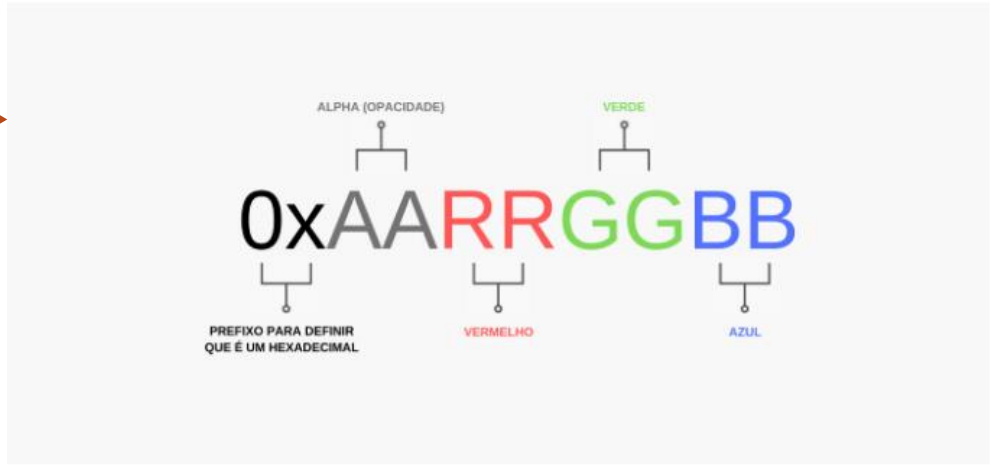
Title e home: Parâmetros nomeados de MaterialApp()

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 > ...

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

Run | Debug | Profile
main() => runApp(MaterialApp(
    title: 'Sorteio da MegaSena',
    home: Container(
      color: Colors.grey,
      child: Row(
        //Uso um children para colocar vários widgets
        children: [
          Text('Texto 1'),
          Text('Texto 2'),
          Text('Texto 3')],
      ), // Row
    ))); // 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

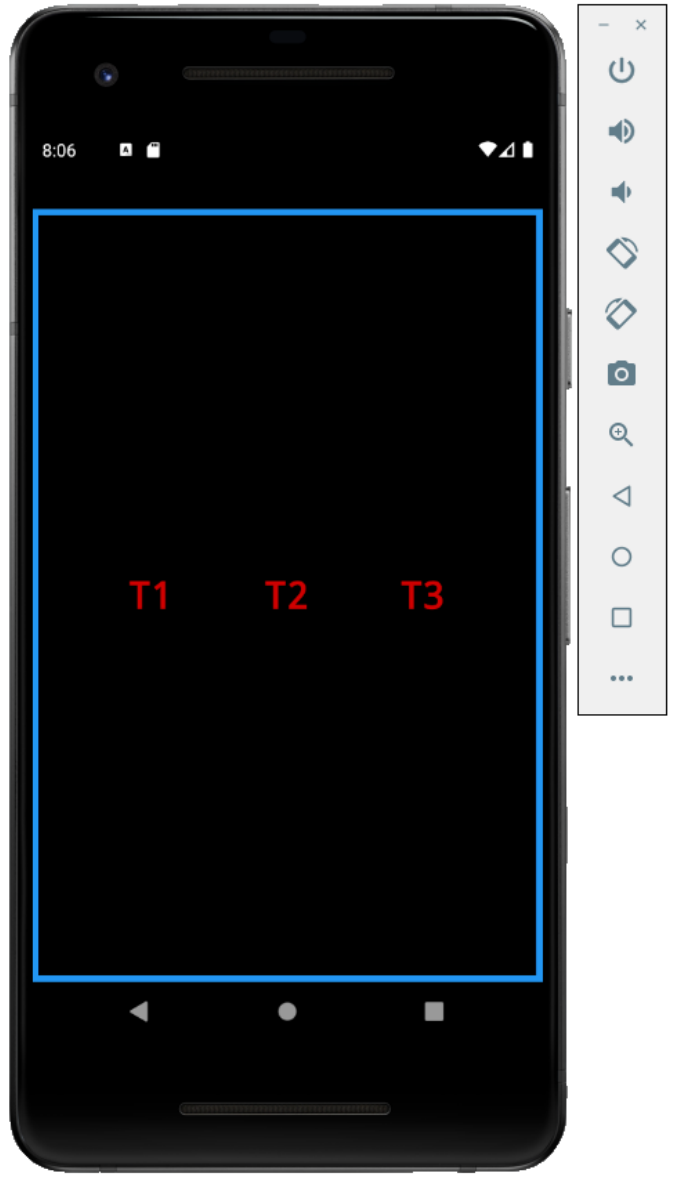
Alinhamentos

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

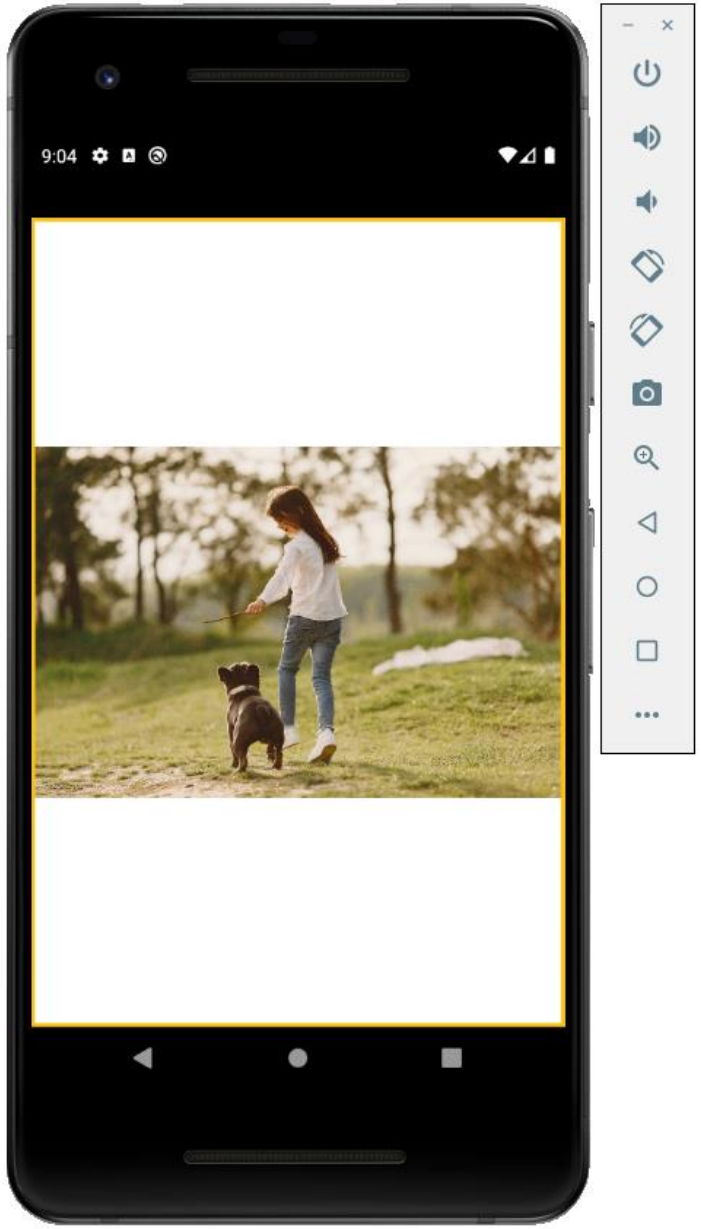
```dart
import 'package:flutter/material.dart';

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
```

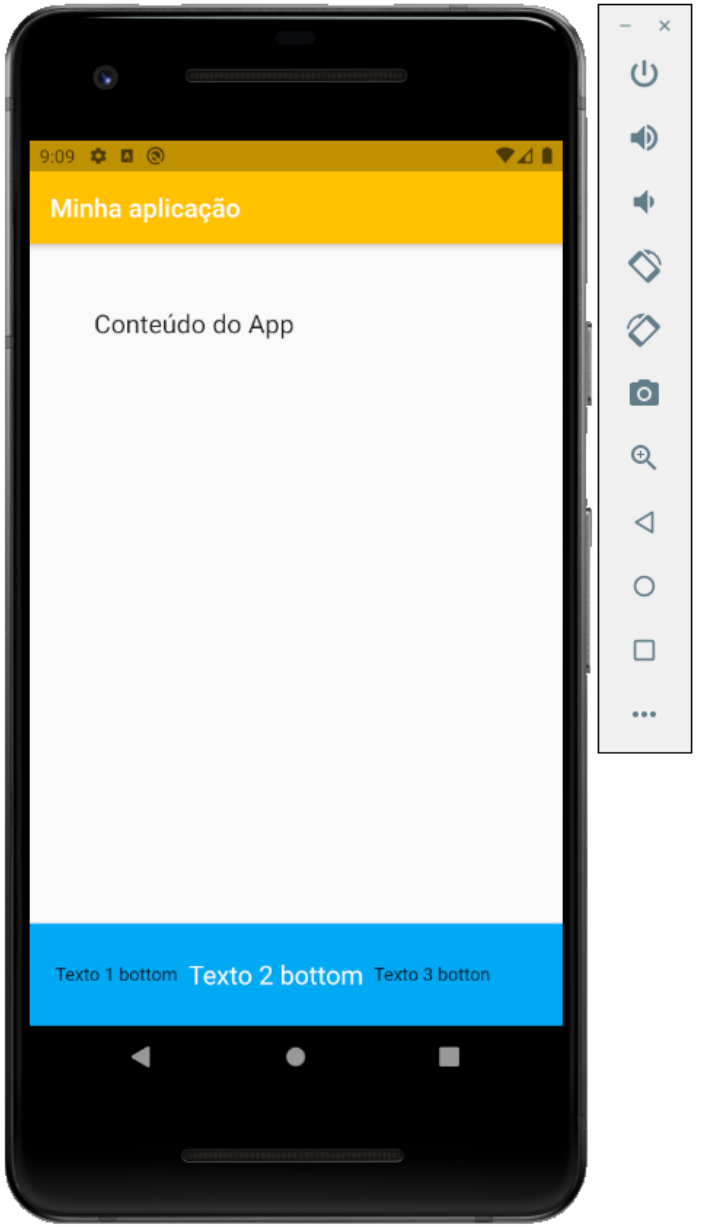


Scaffold

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
```

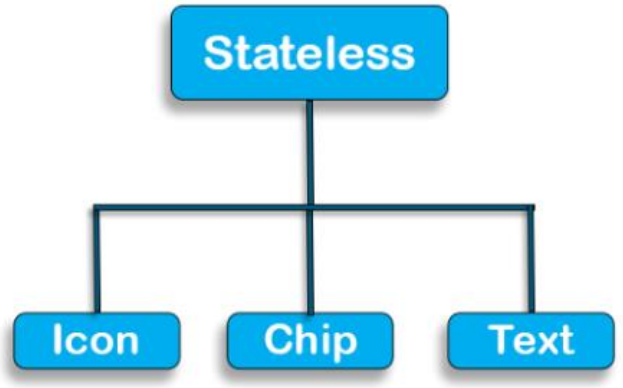


Stateless

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.

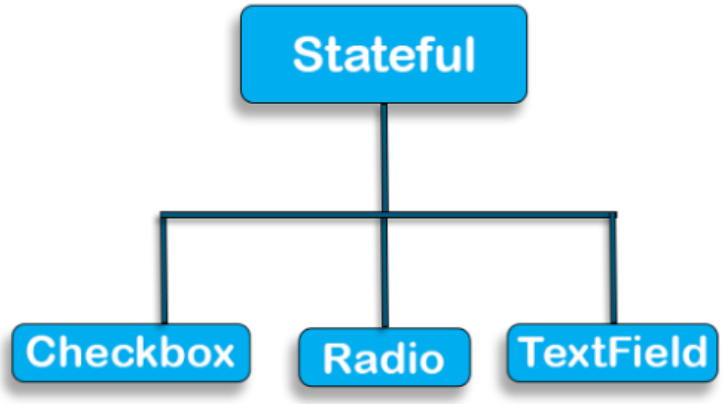
Exemplos de widgets sem estado são os seguintes:



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.

Siga o Senac em Minas nas Redes Sociais: