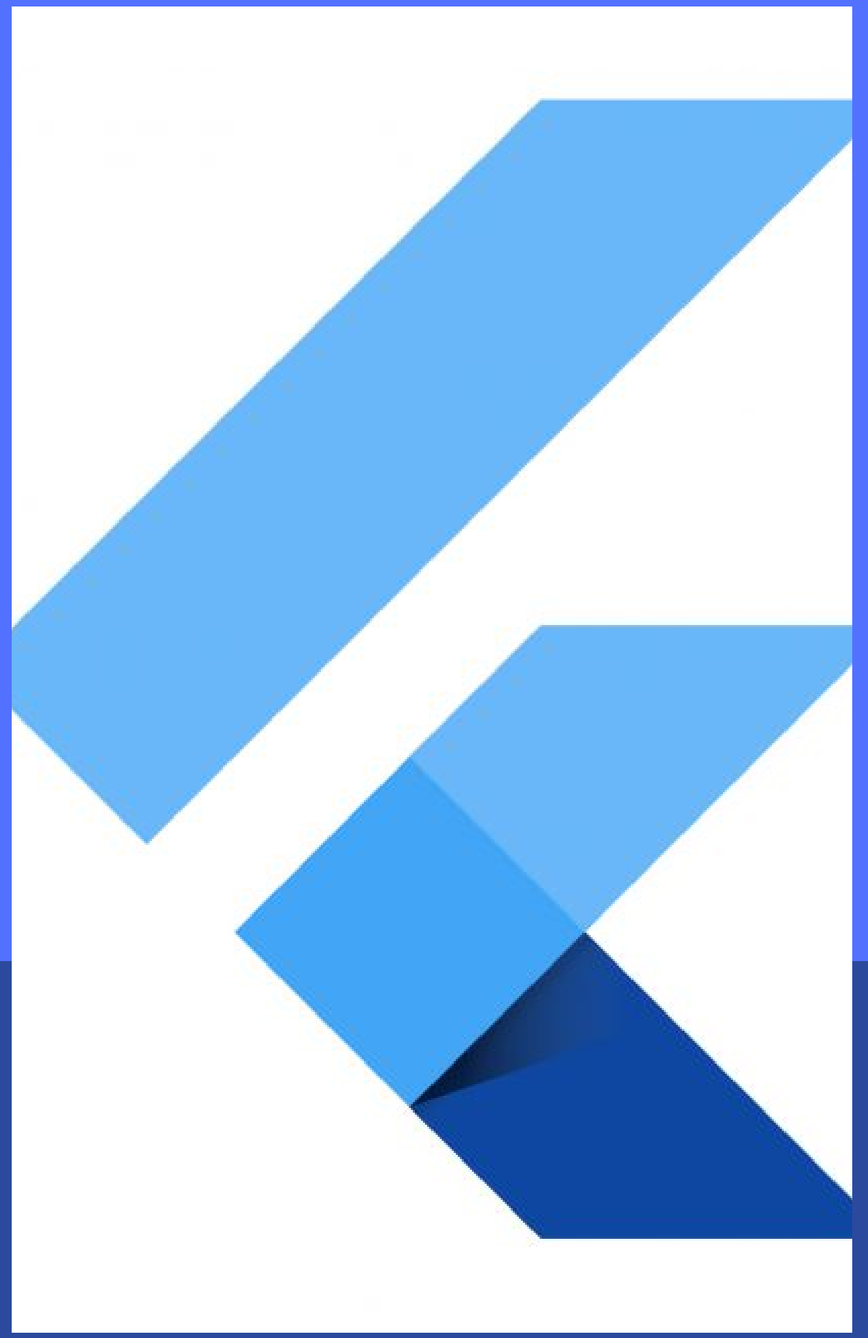
WIDGETS EM FLUTTER

O QUE SÃO WIDGETS E COMO FUNCIONAM EM FLUTTER

- Widgets são os blocos de construção básicos de uma interface de usuário em Flutter.
- Tudo em Flutter é um widget, incluindo elementos visuais, layouts, estilos e até mesmo o aplicativo em si.
- Os widgets são declarativos e reativos, o que significa que a interface de usuário é atualizada automaticamente sempre que ocorrem mudanças nos widgets.



WIDGETS BÁSICOS

- **Container:** Um widget versátil usado para criar layouts, aplicar margens, preenchimentos e decorações aos elementos.
- **Row:**Um widget que organiza seus filhos horizontalmente em uma linha.
- **Column:**Um widget que organiza seus filhos verticalmente em uma coluna.
- **Text:**Um widget usado para exibir texto com estilo.
- **Image:**Um widget usado para exibir imagens em um aplicativo Flutter.

- E muitos outros widgets disponíveis no Flutter para diferentes finalidades.

COMPOSIÇÃO DE WIDGETS PARA CRIAR LAYOUTS

1. Em Flutter, os layouts são criados pela composição de vários widgets.

2. Os widgets são aninhados uns dentro dos outros para criar a hierarquia visual do aplicativo.

3. A composição flexível de widgets permite a criação de layouts complexos e personalizados.

EXEMPLO DE COMPOSIÇÃO DE WIDGETS

- O código cria um **Container** contendo um **Row** com dois filhos: um **Icon** representando uma estrela e um **Text** exibindo a mensagem "Avaliação: 4.5". O **Row** organiza esses dois filhos em uma linha horizontal dentro do **Container**

```
Container(
  child: Row(
    children: [
      Icon(Icons.star),
      Text('Avaliação: 4.5'),
    ],
  ),
);
```

MÉTODO CONSTRUTOR

- Um método construtor, como o próprio nome já diz, é responsável pela criação do objeto daquela classe, iniciando com valores seus atributos ou realizando outras funções que possam vir a ser necessárias.

- Em Flutter, classes também têm construtores que permitem personalizar a criação de objetos

TIPOS DE CONSTRUTORES EM FLUTTER

- Construtor padrão: é fornecido automaticamente pelo Flutter se nenhum construtor personalizado for definido

- Construtor nomeado: permite criar construtores adicionais com nomes diferentes para atender a necessidades específicas

EXEMPLO DE METÓDO CONSTRUTOR PADRÃO

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

class ExemploClasse {
  String nome;
  int idade;

  // Construtor padrão
  ExemploClasse({
    this.nome = '',
    this.idade = 0,
  });
}

void main() {
  // Criando uma instância da classe ExemploClasse
  ExemploClasse exemplo = ExemploClasse();

  // Atribuindo valores aos atributos da instância
  exemplo.nome = 'João';
  exemplo.idade = 25;

  // Imprimindo os valores
  print('Nome: ${exemplo.nome}');
  print('Idade: ${exemplo.idade}');
}
```

EXEMPLO DE METÓDO CONSTRUTOR PADRÃO

- Neste exemplo, a classe **ExemploClasse** possui dois atributos: **nome** e **idade**.
- O construtor padrão é definido entre chaves **{}** e utiliza parâmetros nomeados para permitir a atribuição dos valores dos atributos no momento da criação de uma instância da classe.
- O construtor padrão define valores padrão vazios para os atributos caso nenhum valor seja fornecido.
- No método **main**, criamos uma instância da classe **ExemploClasse** chamada **exemplo** e atribuímos valores aos atributos **nome** e **idade**. Em seguida, imprimimos os valores na saída.

EXEMPLO DE METÓDO CONSTRUTOR NOMEADO

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

class ExemploClasse {
  String nome;
  int idade;

  // Construtor nomeado
  ExemploClasse.nomeado({String nome = '', int idade = 0}) {
    this.nome = nome;
    this.idade = idade;
  }
}

void main() {
  // Criando uma instância da classe ExemploClasse usando o construtor nomeado
  ExemploClasse exemplo = ExemploClasse.nomeado(nome: 'Maria', idade: 30);

  // Imprimindo os valores
  print('Nome: ${exemplo.nome}');
  print('Idade: ${exemplo.idade}');
}
```

EXEMPLO DE METÓDO CONSTRUTOR NOMEADO

- Neste exemplo, a classe **ExemploClasse** possui dois atributos: **nome** e **idade**. O construtor nomeado **ExemploClasse.nomeado** é definido utilizando a sintaxe **{}** para parâmetros nomeados. Ele permite a atribuição dos valores dos atributos no momento da criação de uma instância da classe.
- No método **main**, criamos uma instância da classe **ExemploClasse** chamada **exemplo** utilizando o construtor nomeado **ExemploClasse.nomeado**. Fornecemos os valores dos parâmetros **nome** e **idade** ao criar a instância.
- Em seguida, imprimimos os valores dos atributos **nome** e **idade** na saída.

GESTÃO DE EVENTOS EM WIDGETS

- Em Flutter, os widgets podem responder a eventos, como toques, gestos e interações do usuário.
- Cada widget tem métodos específicos para lidar com eventos.
- Alguns exemplos de eventos incluem onTap, onLongPress, onDoubleTap, entre outros.

EXEMPLO DE GESTAO DE EVENTOS

1. Importamos as bibliotecas necessárias.
2. Definimos uma classe **MyApp** que é o ponto de entrada do aplicativo.
3. **MyApp** retorna um **MaterialApp** que configura o aplicativo com um título e a página inicial.
4. A página inicial é definida como um **Scaffold** com um **AppBar** e um **body** centralizado.
5. No corpo (**body**), temos um único **ElevatedButton** com um texto "Clique aqui".
6. Quando o botão é clicado (**onPressed**), uma função anônima é chamada para exibir a mensagem "Botão clicado!" no console.

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Eventos em Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Gestão de Eventos'),
        ),
        body: Center(
          child: ElevatedButton(
            onPressed: () {
              print('Botão clicado!');
            },
            child: Text('Clique aqui'),
          ),
        ),
      ),
    );
  }
}
```

CONCLUSÃO

OS WIDGETS SÃO OS BLOCOS DE CONSTRUÇÃO FUNDAMENTAIS EM FLUTTER PARA CRIAR INTERFACES DE USUÁRIO.

IMPORTÂNCIA DO USO DE CONSTRUTORES EM FLUTTER PARA CRIAR E INICIALIZAR OBJETOS DE FORMA EFICIENTE

A GESTÃO DE EVENTOS EM WIDGETS PERMITE A INTERAÇÃO DO USUÁRIO EM SEU APLICATIVO FLUTTER.

REFERÊNCIAS

- https://devsimoes.medium.com/widgets-do-flutter-básico-8d5b4fcd62dd
- https://docs.flutter.dev/reference/widgets