ANIMAÇÕES EM FLUTTER

TRANSIÇÕES, LISTAGEM E PERSONALIZADAS

HTTPS://DOCS.FLUTTER.DEV/UI/WIDGETS/ANIMATION
HTTPS://DOCS.FLUTTER.DEV/UI/ANIMATIONS
HTTPS://PUB.DEV/PACKAGES/ANIMATIONS

ANIMAÇÕES EXPLÍCITAS VS IMPLÍCITAS

ANIMAÇÕES EXPLÍCITAS

- *As animações explícitas no Flutter são aquelas que você cria e controla manualmente.*
- *Elas permitem animar diferentes propriedades de um widget, como posição, tamanho, opacidade e rotação.*

ANIMAÇÕES IMPLÍCITAS

- *As animações implícitas no Flutter são animações pré-construídas que são aplicadas automaticamente a certos widgets*

ANIMAÇÕES IMPLÍCITAS

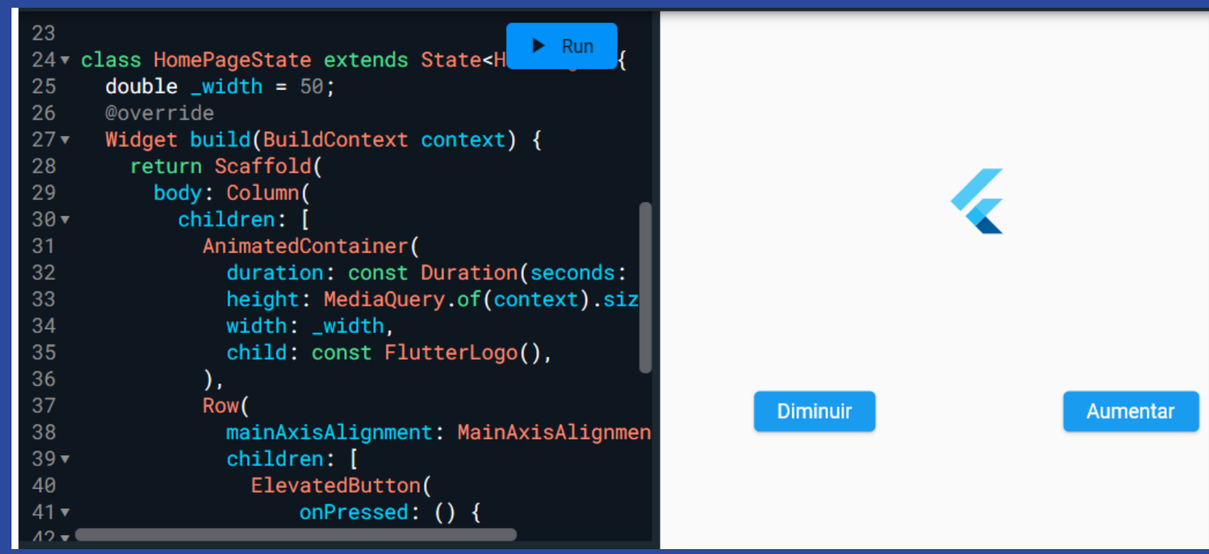
CURVES

- *Curve é usado para dizer como a animação irá se portar, ou seja, como ela irá ocorrer.*

```
AnimatedAlign(
  alignment: value,
  duration: Duration(milliseconds: 1000),
  curve: Curves.fastOutSlowIn,
  child: Container(
    height: 100,
    width: 100,
    color: Colors.green,
  ), // Container
), // AnimatedAlign
```

ANIMATION WIDGETS

- *AnimatedContainer*
- *SlideTransition*
- *AnimatedCrossFade*
- *AnimatedList*



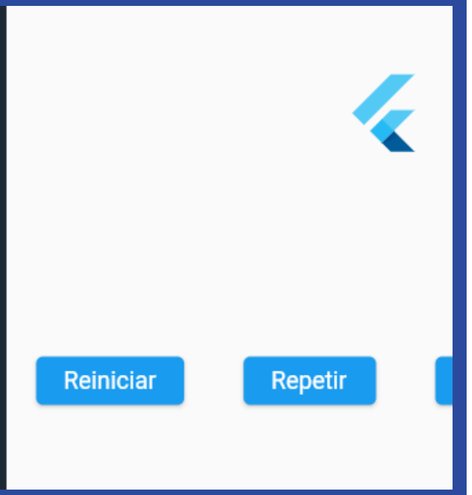
ANIMAÇÕES EXPLÍCITAS

VANTAGENS

- *É o mais poderoso tipo de animação com Flutter. Você tem o total controle sobre ela: iniciar, parar, repetir, reverter... não importa o "movimento".*
- *São úteis para criar animações complexas e personalizadas.*

ANIMATION CONTROLLERS

```
24  class HomePageState extends State<HomePage>
25      with SingleTickerProviderStateMixin {
26
27    late Animation<double> animation;
28    late AnimationController controller;
29
30    @override
31    void initState() {
32      super.initState();
33      controller =
34          AnimationController(duration: const Dur
35      animation = Tween<double>(begin: 50, end: 4
36      animation.addListener(() {
37        setState(() {});
38      });
```



WIDGETS PERSONALIZADOS

WIDGET PERSONALIZADOS

SOBRE

- *Os widgets são os blocos de construção fundamentais no Flutter, permitindo criar interfaces de usuário flexíveis e reutilizáveis.*

- *O Flutter oferece a flexibilidade de criar widgets personalizados para atender às necessidades específicas do seu aplicativo. Aqui estão alguns pontos importantes sobre a criação de widgets personalizados:*

TIPOS

- *Herança de Widgets*
  - *Criação um widget personalizado estendendo uma classe de widget existente.*
- *Composição de Widgets*
  - *Compor widgets existentes para criar um widget personalizado.*
- *Propriedades Personalizadas*
  - *Os widgets personalizados podem ter propriedades personalizadas que permitem a configuração e personalização do widget.*
- *Reutilização de Widgets*
  - *A criação de widgets personalizados promove a reutilização de código, tornando mais fácil manter e atualizar seu aplicativo*

RECURSOS NATIVOS

CÂMERA, GEOLOCALIZAÇÃO, ENTRE OUTROS....

RECURSOS NATIVOS

SOBRE

- *O Flutter possui uma vasta biblioteca de plugins que permitem acessar os recursos do dispositivo, como câmera, geolocalização, sensor de acelerômetro, armazenamento local e muito mais.*

- *Esses plugins facilitam a integração do Flutter com os recursos nativos do dispositivo, permitindo que você crie aplicativos mais poderosos e interativos.*

PLUGINS

- *Câmera*
  - *https://pub.dev/packages/camera*

- *Geolocalização*
  - *https://pub.dev/packages/location*

- *Armazenamento Local*
  - *https://pub.dev/packages/sqflite*

- *Sensor de Acelerômetro*
  - *https://pub.dev/packages/sensors_plus*

TESTES EM FLUTTER

TESTES UNITÁRIOS, WIDGETS, INTEGRAÇÃO

TESTES

UNITÁRIO

Os testes unitários são usados para testar funções e métodos individuais do código. Eles garantem que cada unidade de código funcione corretamente e produza os resultados esperados

WIDGET

Os testes de widget são usados para verificar o comportamento e a aparência visual dos widgets em resposta a diferentes interações. Eles ajudam a garantir que os widgets sejam renderizados corretamente e que suas interações com o usuário funcionem conforme o esperado.

INTEGRAÇÃO

Os testes de integração são usados para verificar a integração correta entre diferentes partes do aplicativo. Eles testam a interação entre componentes, como widgets, serviços e API externas, para garantir que tudo funcione em conjunto de maneira adequada.

UNITÁRIO

1 CONFIGURAÇÃO

No início de um teste unitário, você pode configurar o ambiente e fornecer os dados necessários para o teste. Isso pode envolver a criação de objetos, a definição de valores iniciais e a preparação de qualquer outra condição necessária.

2 EXECUÇÃO

A etapa de execução do teste unitário envolve a chamada da função ou método que você deseja testar, passando os parâmetros adequados. O objetivo é executar a unidade de código e obter o resultado esperado.

3 VERIFICAÇÃO

Comparar o resultado obtido com o resultado esperado usando asserções. As asserções são declarações que verificam se uma determinada condição é verdadeira. Se o resultado obtido não corresponder ao esperado, o teste falhará e um erro será relatado.

```dart
import 'package:counter_app/counter.dart';
import 'package:test/test.dart';

void main() {
  group('Counter', () {
    test('value should start at 0', () {
      expect(Counter().value, 0);
    });

    test('value should be incremented', () {
      final counter = Counter();

      counter.increment();

      expect(counter.value, 1);
    });

    test('value should be decremented', () {
      final counter = Counter();

      counter.decrement();

      expect(counter.value, -1);
    });
  });
}
```

HTTPS://DOCS.FLUTTER.DEV/COOKBOOK/TESTING/UNIT/INTRODUCTION

WIDGET

1 TESTE DE COMPORTAMENTO

Os testes de widget garantem que os widgets se comportem corretamente em diferentes situações. Isso pode envolver testar a aparência visual, as transições, a interação com o usuário e a manipulação de estados

2 WIDGETTESTER

Os testes de widget garantem que os widgets se comportem corretamente em diferentes situações. Isso pode envolver testar a aparência visual, as transições, a interação com o usuário e a manipulação de estados

3 CENÁRIOS DE TESTE

Durante os testes de widget, você pode criar diferentes cenários para verificar o comportamento dos widgets. Isso pode incluir diferentes entradas do usuário, diferentes estados de aplicativo ou diferentes disposições de widgets.

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

void main() {
  // Define a test. The TestWidgets function also provides a WidgetTester
  // to work with. The WidgetTester allows building and interacting
  // with widgets in the test environment.
  testWidgets('MyWidget has a title and message', (tester) async {
    // Create the widget by telling the tester to build it.
    await tester.pumpWidget(const MyWidget(title: 'T', message: 'M'));

    // Create the Finders.
    final titleFinder = find.text('T');
    final messageFinder = find.text('M');

    // Use the `findsOneWidget` matcher provided by flutter_test to
    // verify that the Text widgets appear exactly once in the widget tree.
    expect(titleFinder, findsOneWidget);
    expect(messageFinder, findsOneWidget);
  });
}
```

HTTPS://DOCS.FLUTTER.DEV/COOKBOOK/TESTING/WIDGET/INTRODUCTION

INTEGRAÇÃO

1 SOBRE

Os testes de integração no Flutter são usados para verificar a integração correta entre diferentes partes do aplicativo. Eles testam a interação entre componentes, como widgets, serviços e APIs externas, para garantir que tudo funcione em conjunto de maneira adequada.

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized(); // NEW

  testWidgets('failing test example', (tester) async {
    expect(2 + 2, equals(5));
  });
}
```

2 INTERAÇÃO ENTRE COMPONENTES

Os testes de integração garantem que os diferentes componentes do aplicativo se comuniquem corretamente e compartilhem informações conforme o esperado. Isso pode envolver a verificação de chamadas de função, troca de dados, manipulação de eventos e fluxo de informações

HTTPS://DOCS.FLUTTER.DEV/COOKBOOK/TESTING/WIDGET/INTRODUCTION

CONCLUSÃO

- ANIMAÇÕES
- WIDGETS PERSONALIZADOS
- RECURSOS NATIVOS
- TESTES UNITÁRIOS, WIDGET, INTEGRAÇÃO