

# **ANIMAÇÕES**

## **EM FLUTTER**



*FLUTTER OFERECE SUPORTE A ANIMAÇÕES PODEROSAS E FLEXÍVEIS PARA CRIAR INTERFACES DE USUÁRIO FLUIDAS E INTERATIVAS. EXISTEM VÁRIAS MANEIRAS DE REALIZAR ANIMAÇÕES EM FLUTTER, DESDE ANIMAÇÕES BÁSICAS ATÉ ANIMAÇÕES COMPLEXAS E PERSONALIZADAS.*

- Widgets de animação embutidos
- Widgets de transição
- Animations
- CurvedAnimation
- Animations com provider

# ANIMAÇÕES

## Widgets de animação embutidos

Flutter oferece uma variedade de widgets de animação embutidos que podem ser usados para animar propriedades de outros widgets. Alguns exemplos desses widgets são `AnimatedContainer`, `AnimatedOpacity`, `AnimatedPositioned`, `AnimatedBuilder`, entre outros. Esses widgets permitem animar propriedades como tamanho, opacidade, posição e muito mais. Eles são fáceis de usar e não requerem muito código adicional.

## Widgets de transição

Flutter também fornece widgets de transição que permitem animar a transição entre dois estados de interface do usuário. O mais conhecido é o `AnimatedSwitcher`, que permite animar a troca suave de um widget para outro. Outros widgets de transição incluem `Hero` (para animações entre duas rotas) e `SlideTransition` (para animações de deslize).

# ANIMAÇÕES

## Animations

A classe Animation é um conceito fundamental em Flutter para criar animações personalizadas. Ela define a propriedade que será animada e o intervalo de tempo da animação. A classe AnimationController é frequentemente usada em conjunto com Animation para controlar a animação. Você pode aplicar animações interpoladas, adicionar curvas de interpolação personalizadas e ouvir eventos durante a animação usando AnimationController e Animation.

## CurvedAnimation

A classe CurvedAnimation é usada para aplicar curvas de interpolação a uma animação. Ela permite que você especifique uma curva de animação personalizada, como uma curva elástica, uma curva de amortecimento ou qualquer outra curva desejada. Isso pode ajudar a dar uma sensação mais natural e suave às animações.

# ANIMAÇÕES

## **Animations com provider**

Se você estiver usando o padrão de gerenciamento de estado Provider, você pode combinar Animation e ChangeNotifier para criar animações reativas. O ChangeNotifier pode emitir notificações de mudança de estado e atualizar as animações correspondentes. Isso permite criar animações que respondem a alterações dinâmicas de estado em seu aplicativo.

# ANIMAÇÕES EM FLUTTER

Além dessas opções, existem bibliotecas de animação de terceiros disponíveis, como o `flare_flutter` para animações vetoriais e o Rive (anteriormente conhecido como Flare) para animações interativas. Essas bibliotecas oferecem recursos avançados e ferramentas para criar animações complexas e personalizadas.

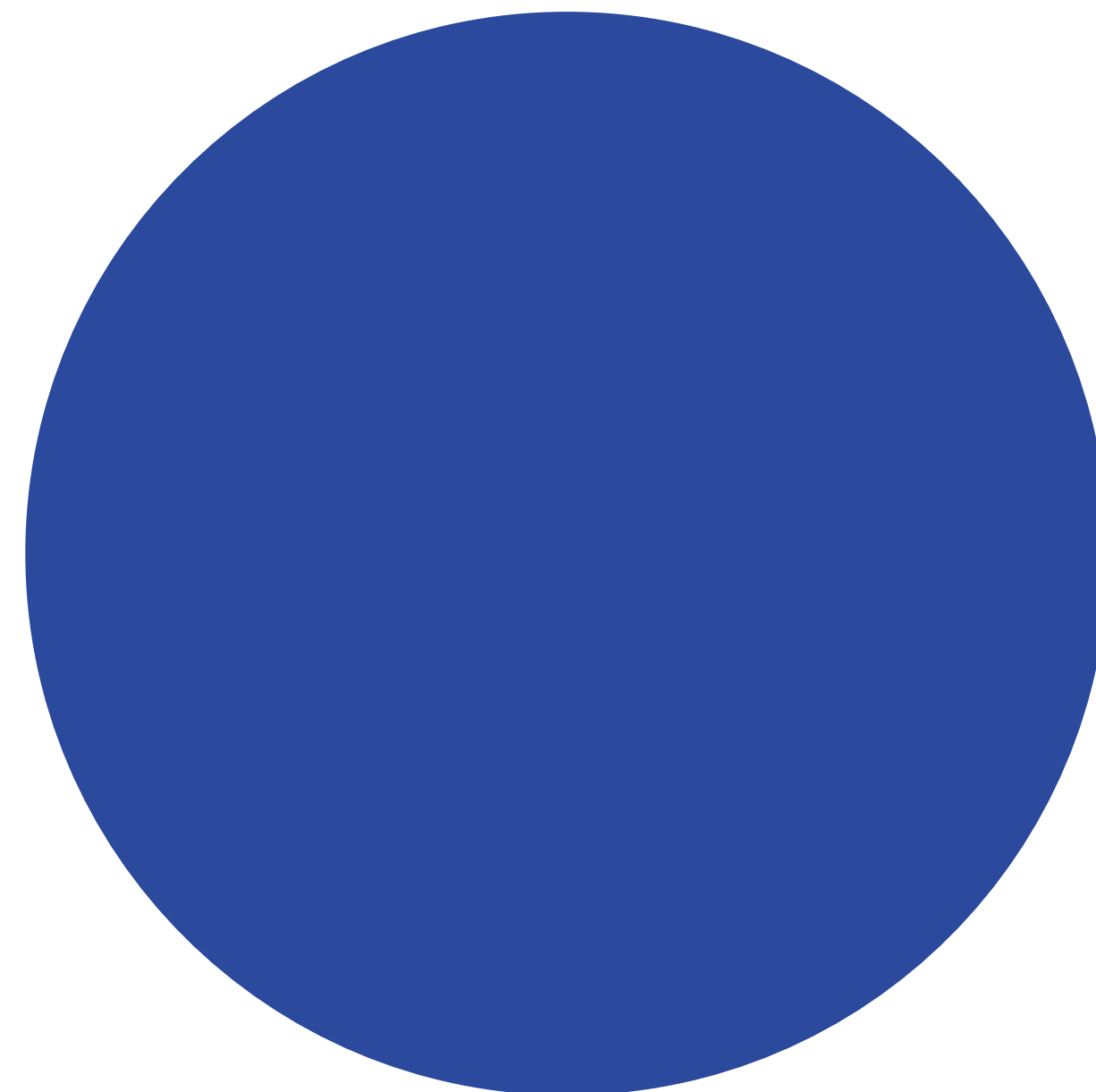
# ANIMAÇÕES EM FLUTTER

Independentemente da abordagem escolhida, é importante lembrar de otimizar as animações em Flutter para garantir um desempenho suave. Isso inclui o uso adequado de `setState`, `AnimatedBuilder`, `Tweens` e evitar atualizações desnecessárias de widgets durante as animações.

Explorar a documentação oficial do Flutter e procurar exemplos e tutoriais em animações em Flutter pode ser útil para obter mais informações e inspiração para suas animações



# **CRIAÇÃO DE WIDGETS PERSONALIZADO**





# CRIAÇÃO DE WIDGETS PERSONALIZADO

A criação de widgets personalizados em Flutter é uma das principais características da estrutura de desenvolvimento de aplicativos. Permite que você crie componentes reutilizáveis e personalizados para construir interfaces de usuário complexas.

# CRIAÇÃO DE WIDGETS PERSONALIZADO

1 - Crie uma nova classe que herde de StatelessWidget ou StatefulWidget. A escolha entre essas duas classes depende se o seu widget precisará gerenciar um estado interno mutável.

```
class MeuWidget extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    // Implemente o método build para retornar a representação visual  
    return Container(  
      // Defina a aparência e o layout do seu widget  
    );  
  }  
}
```

# **CRIAÇÃO DE WIDGETS PERSONALIZADO**

2 - Dentro do método build, retorne um widget que representa a aparência e o layout do seu widget personalizado. Pode ser um widget de layout, como Container, Column ou Row, ou qualquer outro widget específico que você precise.

# CRIAÇÃO DE WIDGETS PERSONALIZADO

3 - Se necessário, você pode definir propriedades personalizadas para o seu widget, permitindo que os usuários o personalizem e forneçam dados. Para fazer isso, adicione parâmetros ao construtor da sua classe de widget e use esses valores dentro do método build.

```
class MeuWidget extends StatelessWidget {  
  final String texto;  
  
  MeuWidget({required this.texto});  
  
  @override  
  Widget build(BuildContext context) {  
    return Text(texto);  
  }  
}
```


# CRIAÇÃO DE WIDGETS PERSONALIZADO

4. Use o seu widget personalizado em qualquer lugar do seu aplicativo Flutter. Basta instanciar a classe do seu widget, fornecendo os valores necessários e incorporando-o à árvore de widgets do Flutter.



```
MeuWidget(  
  texto: 'Olá, mundo!',  
)
```

# **CRIAÇÃO DE WIDGETS PERSONALIZADO**

Essas são apenas as etapas básicas para criar um widget personalizado em Flutter. Você pode expandir a funcionalidade do seu widget personalizado adicionando mais propriedades, métodos e interatividade, dependendo dos requisitos do seu aplicativo.



# **USO DE PLUGINS PARA ACESSO A RECURSOS DO DISPOSITIVO, COMO CÂMERA E GEOLOCALIZAÇÃO**



# USO DE PLUGINS PARA ACESSO A RECURSOS DO DISPOSITIVO

Para acessar recursos do dispositivo, como câmera e geolocalização em Flutter, você pode utilizar plugins específicos que fornecem interfaces para interagir com esses recursos nativos. O Flutter possui uma ampla variedade de plugins disponíveis na comunidade que facilitam o acesso aos recursos do dispositivo.



# EXEMPLOS DE PLUGINS PARA CÂMERA E GEOLOCALIZAÇÃO

1

## PARA ACESSO À CÂMERA:

- *'camera': Este plugin permite capturar fotos e vídeos usando a câmera do dispositivo. Ele fornece uma API fácil de usar para iniciar a câmera, tirar fotos, gravar vídeos e muito mais.*
- *'image\_picker': Este plugin permite selecionar imagens da galeria do dispositivo ou capturar uma foto usando a câmera. Ele oferece suporte a seleção de imagens e vídeos.*

2

## PARA ACESSO À GEOLOCALIZAÇÃO:

- *'geolocator': Este plugin permite obter a localização atual do dispositivo, incluindo coordenadas de latitude e longitude, velocidade, altitude, entre outros. Ele oferece recursos para rastrear as mudanças de localização em tempo real.*
- *'location': Este plugin também permite obter a localização atual do dispositivo e fornece recursos para rastrear as mudanças de localização. Além disso, oferece suporte a recursos avançados, como geocodificação reversa e estimativa de distância.*

# PARA USAR ESTES PLUGINS, SIGA ESTAS ETAPAS BÁSICAS:

1 - Adicione a dependência do plugin ao arquivo pubspec.yaml do seu projeto:

```
dependencies:  
  camera: ^x.x.x  
  image_picker: ^x.x.x  
  geolocator: ^x.x.x  
  location: ^x.x.x
```

2 - Execute o comando flutter 'pub get' no terminal para baixar as dependências atualizadas.

3 - Importe o pacote do plugin no arquivo Dart onde você deseja utilizar os recursos:

```
import 'package:camera/camera.dart';  
import 'package:image_picker/image_picker.dart';  
import 'package:geolocator/geolocator.dart';  
import 'package:location/location.dart';
```

# 4

*SIGA A DOCUMENTAÇÃO DO PLUGIN  
ESPECÍFICO PARA APRENDER COMO USAR AS  
CLASSES, MÉTODOS E EVENTOS  
DISPONIBILIZADOS PELO PLUGIN. CADA  
PLUGIN TEM SUA PRÓPRIA API E CONJUNTO  
DE INSTRUÇÕES PARA USAR OS RECURSOS  
DO DISPOSITIVO.*

# USO DE PLUGINS PARA ACESSO A RECURSOS DO DISPOSITIVO

É importante observar que diferentes plugins podem ter requisitos específicos, como configurações no arquivo 'AndroidManifest.xml' ou 'Info.plist', dependendo da plataforma. Verifique a documentação do plugin para obter instruções detalhadas sobre essas configurações específicas.

Sempre que usar plugins, certifique-se de estar atualizado com as últimas versões e leia a documentação oficial para entender como usá-los corretamente.

# **TESTES EM FLUTTER**

**TESTES UNITÁRIOS, TESTES DE  
WIDGET, TESTES DE INTEGRAÇÃO**

# EM FLUTTER, VOCÊ PODE REALIZAR TESTES PARA GARANTIR A QUALIDADE DO SEU APLICATIVO.

## OS PRINCIPAIS TIPOS DE TESTES SÃO:

- Testes Unitários:  
Verifique se unidades individuais de código estão funcionando corretamente.
- Testes de Widget:  
Testam a interação e o comportamento de widgets individuais ou grupos pequenos.
- Testes de Integração:  
Testam a interação e o comportamento de vários componentes do aplicativo trabalhando juntos.

# TESTES EM FLUTTER

Você pode usar a biblioteca `'flutter_test'` para escrever testes em Flutter. Os testes podem ser executados no terminal ou no ambiente de desenvolvimento integrado. É importante escrever testes automatizados para garantir a estabilidade e a consistência do aplicativo.

# FONTES

- [HTTPS://DEVELOPERS-BR.GOOGLEBLOG.COM/2020/02/CONCEITOS-BASICOS-DO-FLUTTER-PARA.HTML](https://developers-br.googleblog.com/2020/02/conceitos-basicos-do-flutter-para.html)
- [HTTPS://DOCS.FLUTTER.DEV/UI/ANIMATIONS](https://docs.flutter.dev/ui/animations)
- [HTTPS://ACERVOLIMA.COM/FLUTTER-WIDGETS-PERSONALIZADOS/](https://acervolima.com/flutter-widgets-personalizados/)
- [HTTPS://WWW.ALURA.COM.BR/ARTIGOS/CONHECENDO-O-FLUTTER-E-UMA-VISAO-DO-DESENVOLVIMENTO-MOBILE](https://www.alura.com.br/artigos/conhecendo-o-flutter-e-uma-visao-do-desenvolvimento-mobile)
- [HTTPS://WWW.DIGITALOCEAN.COM/COMMUNITY/TUTORIALS/FLUTTER-GEOLOCATOR-PLUGIN](https://www.digitalocean.com/community/tutorials/flutter-geolocator-plugin)
- [HTTPS://WWW.ALURA.COM.BR/CONTEUDO/TESTES-WIDGETS-FLUTTER](https://www.alura.com.br/conteudo/testes-widgets-flutter)
- [HTTPS://BLOG.FLUTTERANDO.COM.BR/TESTE-DE-INTEGRA%C3%A7%C3%A3O-NO-FLUTTER-28556106784B](https://blog.flutterando.com.br/teste-de-integra%C3%A7%C3%A3o-no-flutter-28556106784b)