



# Flutter Avançado - Aula 4

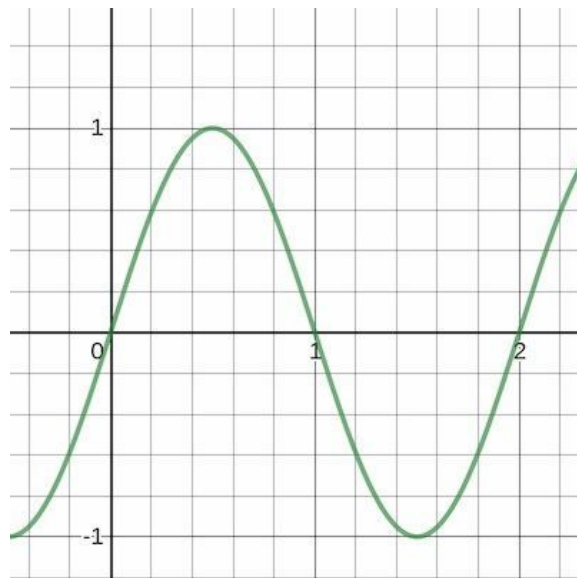
Exercícios 11 a 13



- Artigo: Top 8 Apps Built With Flutter Framework [Ver mais](#)



- Além de utilizarmos as curvas pré definidas no Flutter, podemos criar as nossas próprias
- Para isso, basta criar um novo objeto, herdando a classe Curve
- No nosso exemplo, vamos implementar uma curva Senóide
- A curva senóide descreve uma oscilação suave





```
class SineCurve extends Curve {  
    final double count;  
  
    SineCurve({this.count = 3});  
  
    @override  
    double transformInternal(double t) {  
        var val = sin(count * 2 * pi * t) * 0.5 + 0.5;  
  
        print(t);  
  
        return val; //f(x)  
    }  
}
```

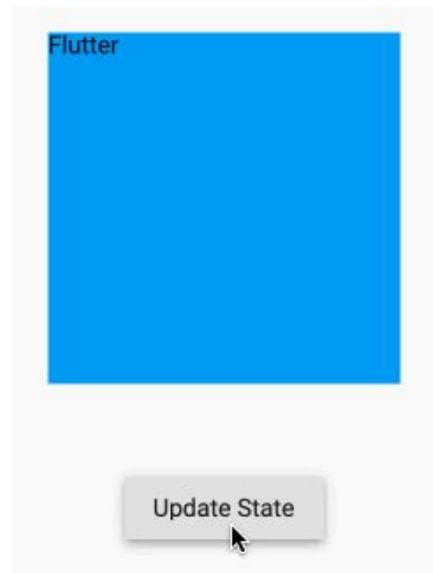


- Além de criarmos animações a partir do zero, o Flutter disponibiliza vários widgets com animações pré definidas
- A vantagem de utilizar animações implícitas está na simplicidade, pois não precisamos criar controllers ou utilizar os *Tickers*
- A desvantagem está em utilizar comportamentos já definidos





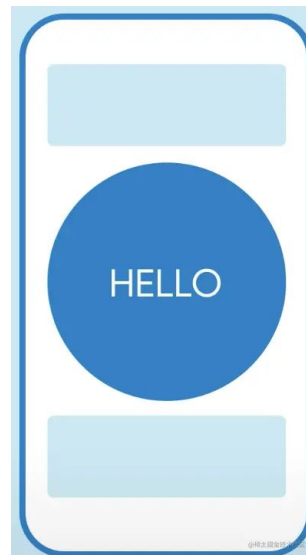
- O *AnimatedContainer* é um *Container* que se anima conforme os atributos do *Container*
- Ao alterar um atributo, a mudança visual é efetuada suavemente
- É necessário definir uma duração para a animação
- É possível determinar uma curva de animação



```
body: Center(  
  child: GestureDetector(  
    onTap: () {  
      setState(() {  
        this.vrSize = 50;  
        this.vrColorContainer = Colors.white24;  
      });  
    },  
    child: AnimatedContainer(  
      duration: Duration(seconds: 1),  
      height: this.vrSize,  
      width: this.vrSize,  
      curve: Curves.bounceInOut,  
      decoration: BoxDecoration(  
        borderRadius: BorderRadius.circular(10),  
        color: this.vrColorContainer,  
      ),  
    ),  
  ),  
)
```



- O *AnimatedCrossFade* efetua uma suave transição entre dois *widgets*
- Basta controlar qual widget deve ser exibido, usando a propriedade *crossFadeState*
- Devemos definir uma duração para a transição
- Para animações mais complexas, podemos definir uma curva de transição para cada *widget*







## Propriedades do *AnimatedCrossFade*

```
AnimatedCrossFade(  
  duration: Duration(milliseconds: 300),  
  firstChild: Container(),  
  secondChild: Container(),  
  firstCurve: Curves.bounceInOut,  
  secondCurve: Curves.decelerate,  
  crossFadeState: CrossFadeState.showFirst  
) ,
```

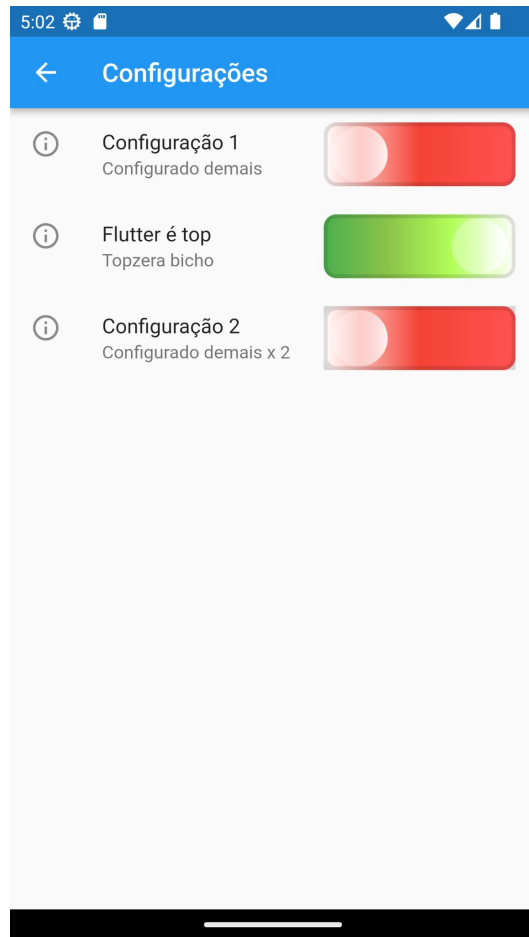
## Controlando a exibição do *widget* com estados

```
crossFadeState: (this._showFirst ? CrossFadeState.showFirst : CrossFadeState.showSecond),
```



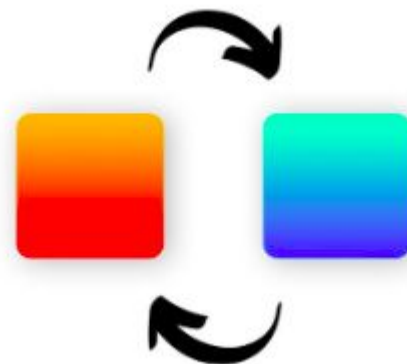
- Vamos criar um widget que ao ser pressionado, liga e desliga
- A transição entre o estado de ligado e desligado deve ser suave
- Esse widget deve possuir duas propriedades:
  1. Seu estado atual
  2. Um evento, para notificar uma mudança de estado\*

\*Vimos a criação de eventos na 1ª aula do curso de Flutter Básico





- Efetua uma transição de diferentes estilos em um mesmo *widget*
- É necessário definir um Tween (intervalo) de estilos
- Também precisaremos definir um controller





## Definindo um *DecorationTween*

```
final DecorationTween decorationTween = DecorationTween(  
    begin: BoxDecoration(color: Colors.white,  
  
    ),  
    end: BoxDecoration(  
        color: Colors.yellow,  
        borderRadius: BorderRadius.circular(15),  
        border: Border.all(  
            color: Colors.yellowAccent,  
            width: 15,  
            style: BorderStyle.solid)));
```



## Criando o DecoratedBoxTransition

```
DecoratedBoxTransition(  
  decoration: this.decorationTween.animate(this.controller),  
  child: Container(  
    height: double.infinity,  
    width: double.infinity,  
    alignment: Alignment.center,  
    child: Image(  
      image: NetworkImage("https://images"),  
      height: 500,  
    )  
  ),  
)
```



- Muito semelhante ao *AnimatedCrossFade*, efetuando uma transição entre dois *widgets*
- A principal diferença entre este e o *AnimatedCrossFade*, é que no *AnimatedSwitcher*, temos mais controle da transação entre os *widgets*

```
AnimatedSwitcher(  
  transitionBuilder: (child, animation) {  
    return SizeTransition(  
      sizeFactor: animation,  
      child: child,  
      axis: Axis.horizontal,  
      axisAlignment: -1,  
    );  
  },  
  duration: Duration(milliseconds: 300),  
  child: (this.isOn  
    ? this._getSwitch(  
      [Colors.white, Colors.yellowAccent, Colors.yellow],  
      Alignment.topLeft)  
    : this._getSwitch([  
      Colors.green,  
      Colors.lightGreen,  
      Colors.lightGreenAccent,  
      Colors.white  
    ], Alignment.topRight)),  
))
```







- Ao trocar de telas, caso o Hero estiver sendo usado em ambas as telas, um efeito de transição é efetuado
- Para que o efeito funcione, a propriedade *tag* deve ser igual nas duas telas

```
ListTile(  
  leading: const Hero(  
    tag: 'hero-rectangle',  
    child: BoxWidget(size: Size(50.0, 50.0)),  
  ),  
  onTap: () => _gotoDetailsPage(context),  
  title: const Text(  
    'Tap on the icon to view hero animation transition.',  
  ),  
)
```



- Vamos um app de notícias
- O app deve listar as notícias e ao clicar sobre uma, deve abrir uma página de detalhes da notícia
- A transição até a nova tela deve ser feita utilizando Hero

