

Checkbox

O Widget Checkbox é um Widget de seleção representado por uma caixa selecionável. Quando está selecionada seu valor é true, e quando não está é false.

Esse Widget sempre deverá estar associado a uma variável de booleana, a qual utilizaremos para controlar o estado (Selecionado ou não) do Widget.

A screenshot of a code editor with a dark background and light-colored text. The code is written in Dart and demonstrates how to use the Checkbox widget within a Scaffold. It includes an AppBar with the title 'Aplicação exemplo' and a Container body with a margin. Inside the body, there is a Row widget containing a Checkbox and a Text widget. The Checkbox is initialized with a value of _mussarela and an onChanged callback that calls setState to update _mussarela. The Text widget displays 'Pizza de Mussarela'.

```
1 Scaffold(  
2   appBar: AppBar(  
3     title: Text("Aplicação exemplo"),  
4   ),  
5   body: Container(  
6     margin: EdgeInsets.all(16),  
7     child: Row(  
8       children: [  
9         Checkbox(  
10          value: _mussarela,  
11          onChanged: (selecao) {  
12            setState(() {  
13              _mussarela = selecao;  
14            });  
15          },  
16        ),  
17        Text(  
18          "Pizza de Mussarela",  
19        ),  
20      ],  
21    ),  
22  ),  
23);
```

Radio Button

O Widget Radio é muito parecido com o Widget Checkbox, se diferenciando pelo fato de que somente um valor pode ser selecionado.

Outro aspecto importante sobre esse Widget é o fato dele estar associado a um grupo, sendo que somente um elemento de um mesmo grupo pode ser selecionado, sendo que sempre que um elemento é selecionado, os demais serão desmarcados.

```

1 //Enumerador Nivel
2 enum Nivel { facil, normal, dificil }
3
4 // Variável nível
5 Nivel? _nivel;
6
7 Scaffold(
8   appBar: AppBar(
9     title: Text("Aplicação exemplo"),
10  ),
11  body: Container(
12    margin: EdgeInsets.all(16),
13    child: Row(
14      children: [
15        Radio<Nivel>(
16          value: Nivel.facil,
17          onChanged: (selecao) {
18            setState(() {
19              _nivel = selecao;
20            });
21          },
22          groupValue: _nivel,
23        ),
24        Text(
25          "Fácil",
26        ),
27      ],
28    ),
29  ),
30 );

```

SnackBar

SnackBar é um *Widget* do Flutter que nos permite exibir notificações no rodapé da aplicação, sendo essas automaticamente removidas após um determinado período de tempo.

content

A propriedade *content*, como o próprio nome já sugere, nos permite definir o conteúdo da mensagem a ser exibida.

Essa propriedade recebe um *Widget*, o que nos permite personalizar nossa mensagem adicionando ícones ou qualquer tipo de *Widget* que desejarmos.

A screenshot of a code editor with a dark background and light-colored text. The code is in Dart and defines a function named `exibirSnackBar` that takes a `BuildContext` parameter. Inside the function, a `SnackBar` widget is created with a `Row` of children. The first child is an `Icon` of a delete icon in white. The second child is a `Text` widget with the message "Remoção realizada com sucesso". The `SnackBar` is then shown using `ScaffoldMessenger.of(context).showSnackBar(snackBar);`.

```
1 void exibirSnackBar(BuildContext context) {  
2   // Criamos a SnackBar  
3   final snackBar = SnackBar(  
4     content: Row(  
5       children: [  
6         Icon(  
7           Icons.delete,  
8           color: Colors.white,  
9         ),  
10        Text("Remoção realizada com sucesso"),  
11      ],  
12    ),  
13  );  
14  // Exibimos a SnackBar  
15  ScaffoldMessenger.of(context).showSnackBar(snackBar);  
16 }
```

duration

A propriedade *duration*, como o próprio nome já sugere, define a duração da exibição do *SnackBar* na tela.

Essa propriedade recebe um objeto *Duration*, o qual nos permite definir um período de tempo em:

- **Dias**
- **Horas**
- **Microsegundos**
- **Milisegundos**
- **Minutos**
- **Segundos**

Para essa propriedade normalmente utilizamos a propriedade `segundos`. Sendo que se não for informada a duração padrão é de 4 segundos.

```

1 void exibirSnackBar(BuildContext context) {
2   // Criamos a SnackBar
3   final snackBar = SnackBar(
4     content: Row(
5       children: [
6         Icon(
7           Icons.delete,
8           color: Colors.white,
9         ),
10        Text("Remoção realizada com sucesso"),
11      ],
12    ),
13    duration: Duration(seconds: 6),
14  );
15  // Exibimos a SnackBar
16  ScaffoldMessenger.of(context).showSnackBar(snackBar);
17}

```

backgroundColor

A propriedade *backgroundColor*, como o próprio nome já sugere, define a cor de fundo do *SnackBar* na tela. Caso essa propriedade não for informada, a cor padrão será utilizada, sendo essa preto. E se porventura desejarmos modificar esse padrão, basta informar um objeto *Color* como parâmetro desta propriedade.

```

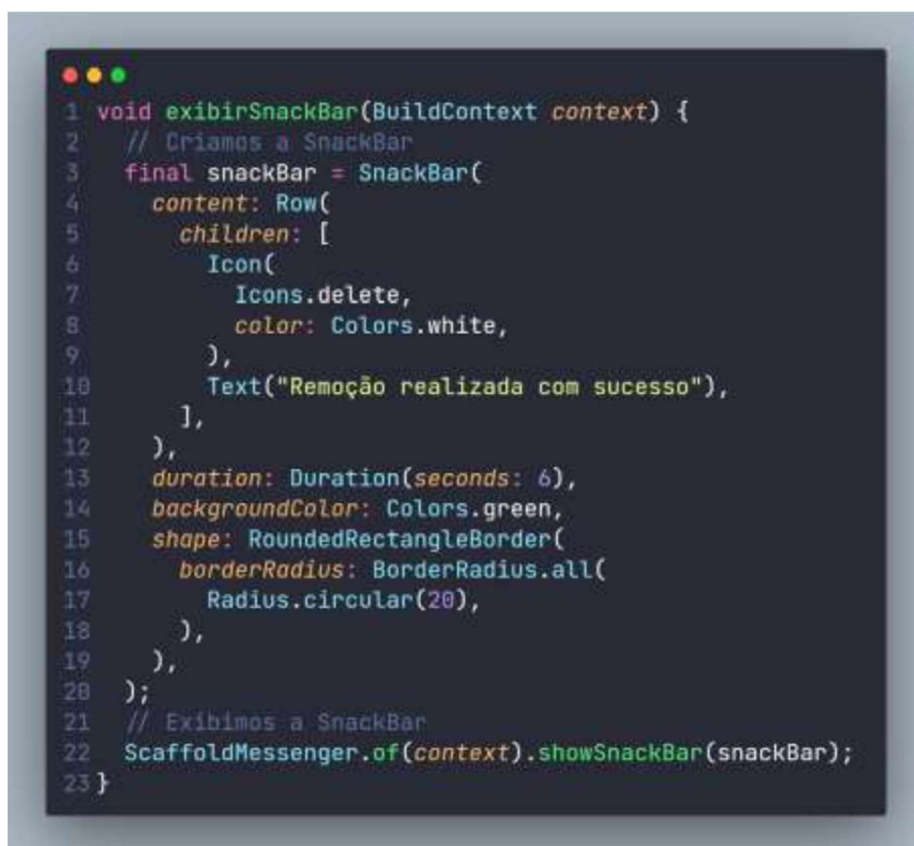
1 void exibirSnackBar(BuildContext context) {
2   // Criamos a SnackBar
3   final snackBar = SnackBar(
4     content: Row(
5       children: [
6         Icon(
7           Icons.delete,
8           color: Colors.white,
9         ),
10        Text("Remoção realizada com sucesso"),
11      ],
12    ),
13    duration: Duration(seconds: 6),
14    backgroundColor: Colors.green,
15  );
16  // Exibimos a SnackBar
17  ScaffoldMessenger.of(context).showSnackBar(snackBar);
18}

```

shape

A propriedade *shape* nos permite definir o arredondamento das bordas da *SnackBar*. Essa propriedade recebe um objeto *ShapeBorder* ou um descendente dessa classe. Normalmente utilizamos um objeto *RoundedRectangleBorder*, mas também é possível utilizar um objeto *ContinuousRectangleBorder* ou *CircleBorder*.

Caso deseje arredondar as bordas do seu *SnackBar* é recomendado que defina a propriedade *behavior* para *floating*.



```
1 void exibirSnackBar(BuildContext context) {
2   // Criamos a SnackBar
3   final snackBar = SnackBar(
4     content: Row(
5       children: [
6         Icon(
7           Icons.delete,
8           color: Colors.white,
9         ),
10        Text("Remoção realizada com sucesso"),
11      ],
12    ),
13    duration: Duration(seconds: 6),
14    backgroundColor: Colors.green,
15    shape: RoundedRectangleBorder(
16      borderRadius: BorderRadius.all(
17        Radius.circular(20),
18      ),
19    ),
20  );
21  // Exibimos a SnackBar
22  ScaffoldMessenger.of(context).showSnackBar(snackBar);
23 }
```

behavior

A propriedade *behavior* define onde um *SnackBar* deve aparecer dentro de um *Scaffold* e como sua localização deve ser ajustada quando o *Scaffold* também inclui um *FloatingActionButton* ou *BottomNavigationBar*. Sendo que seu valor padrão é *fixed*.

```

1 void exibirSnackBar(BuildContext context) {
2   // Criamos a SnackBar
3   final snackBar = SnackBar(
4     content: Row(
5       children: [
6         Icon(
7           Icons.delete,
8           color: Colors.white,
9         ),
10        Text("Remoção realizada com sucesso"),
11      ],
12    ),
13    duration: Duration(seconds: 6),
14    backgroundColor: Colors.green,
15    shape: RoundedRectangleBorder(
16      borderRadius: BorderRadius.all(
17        Radius.circular(20),
18      ),
19    ),
20    behavior: SnackBarBehavior.floating,
21  );
22  // Exibimos a SnackBar
23  ScaffoldMessenger.of(context).showSnackBar(snackBar);
24}

```

action

A propriedade *action* define um botão de ação para ser inserido juntamente a *SnackBar*. Normalmente esse botão é utilizado para desfazer uma ação.

```

1 void exibirSnackBar(BuildContext context) {
2   // Criamos a SnackBar
3   final snackBar = SnackBar(
4     content: Text("Remoção realizada com sucesso"),
5     action: SnackBarAction(
6       label: "Desfazer",
7       onPressed: () {
8         print("Desfez a ação");
9       },
10    ),
11  );
12  // Exibimos a SnackBar
13  ScaffoldMessenger.of(context).showSnackBar(snackBar);
14}

```