FLUTTER

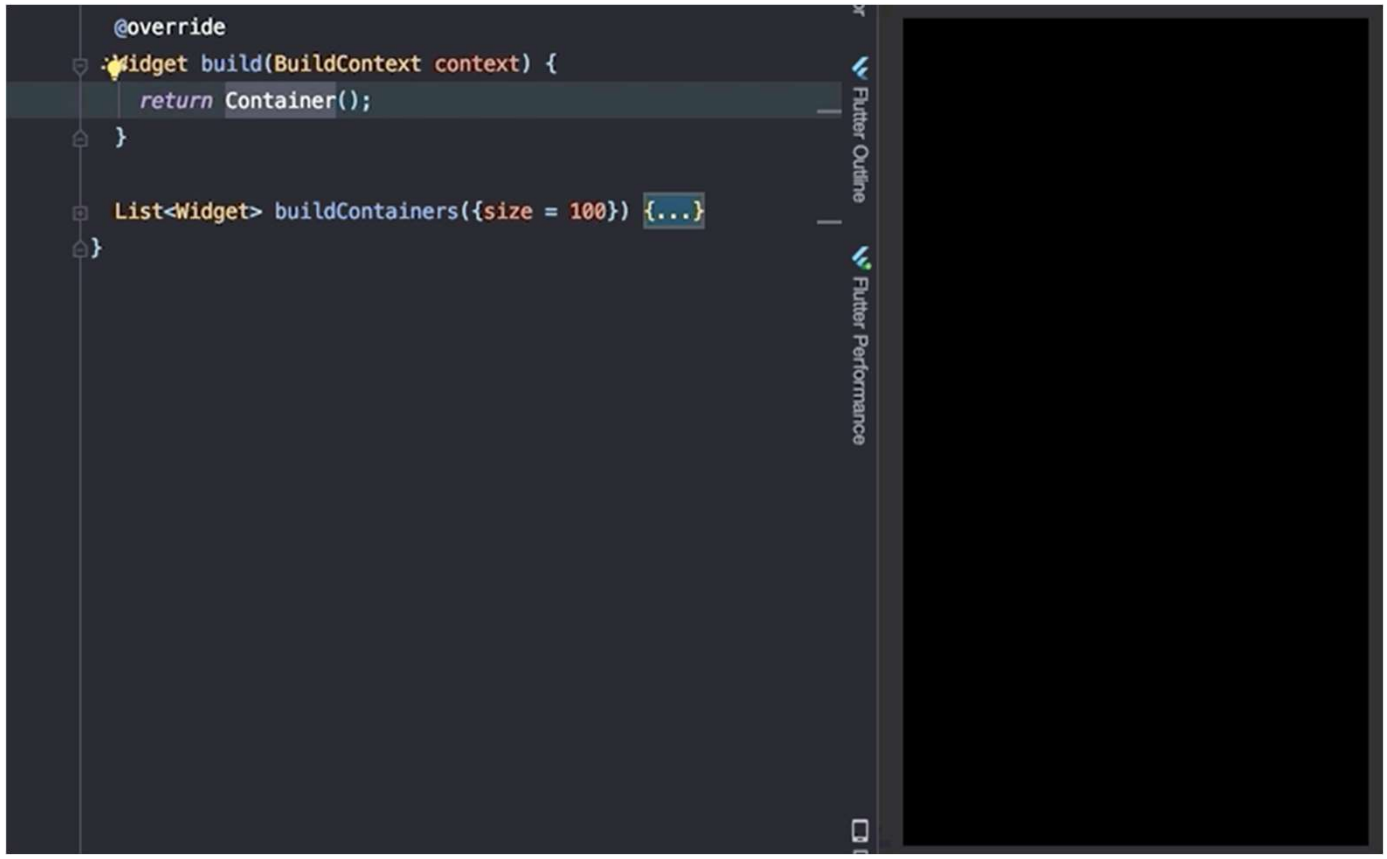
FLUTTER #6

FLUTTER WIDGETS PARTE 2

FLUTTER

Listview

O Widget **ListView** serve para mostrarmos listas e podermos utilizar o poder do scroll.
Ele é bem parecido com o **Column** ou a **Row**, porém precisa ter um tamanho delimitado, já que esse Widget irá tentar expandir ao máximo na tela para delimitar o início e fim do scroll.
O Flutter automaticamente configura as ações de chegar ao topo ou ao final da lista de acordo com o aparelho.



Listview

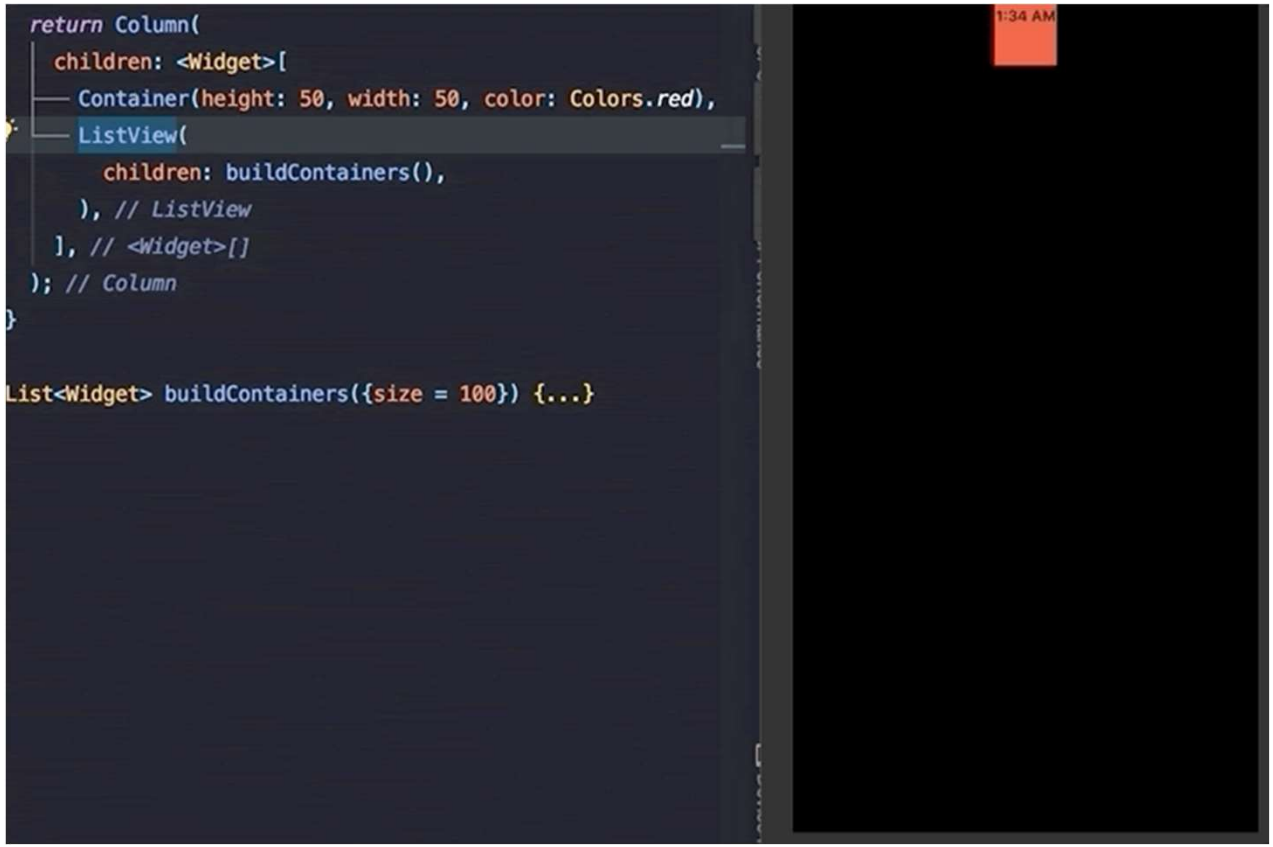
Podemos utilizar a **ListView** na horizontal, fazendo um efeito parecido com um carousel.

```
@override
Widget build(BuildContext context) {
  return ListView(
    children: buildContainers(),
  );
}

List<Widget> buildContainers({size = 100}) {...}
}
```

Expanded

Para conseguir uma **ListView** dentro de uma **Column**, precisamos de um Widget chamado **Expanded**, que expande ao máximo que puder e passa o limite para seu Child fazendo o seguinte efeito:

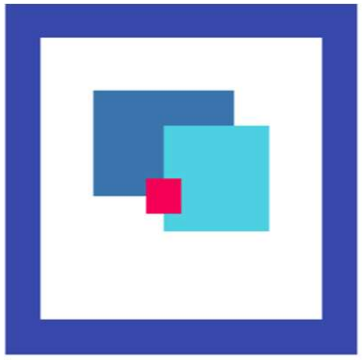


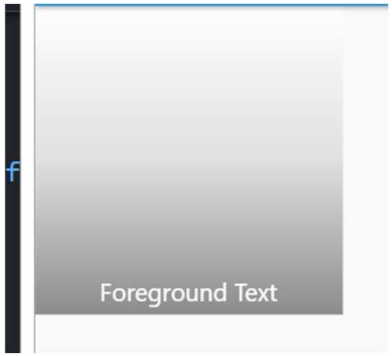
Expanded

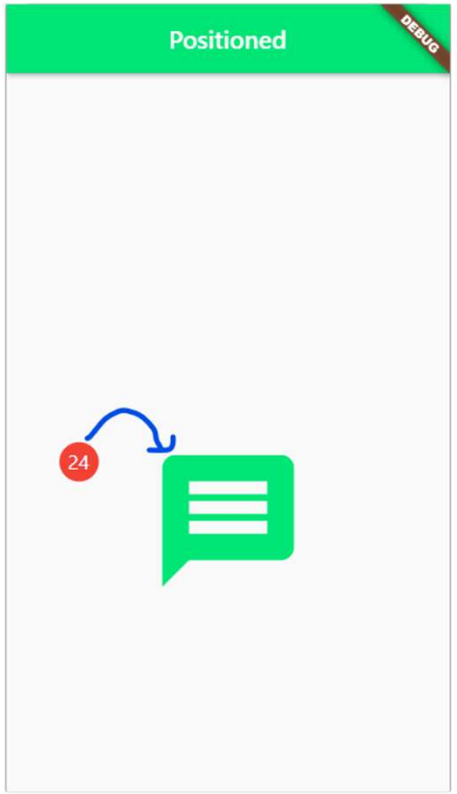
```
Row(
 crossAxisAlignment: CrossAxisAlignment.center,
 children: [
  Expanded(
   child: Image.asset('images/pic1.jpg'),
  ),
  Expanded(
   child: Image.asset('images/pic2.jpg'),
  ),
  Expanded(
   child: Image.asset('images/pic3.jpg'),
  ),
 ],
);
```

Stack

Usando uma **stack** você pode posicionar widgets uns sobre os outros





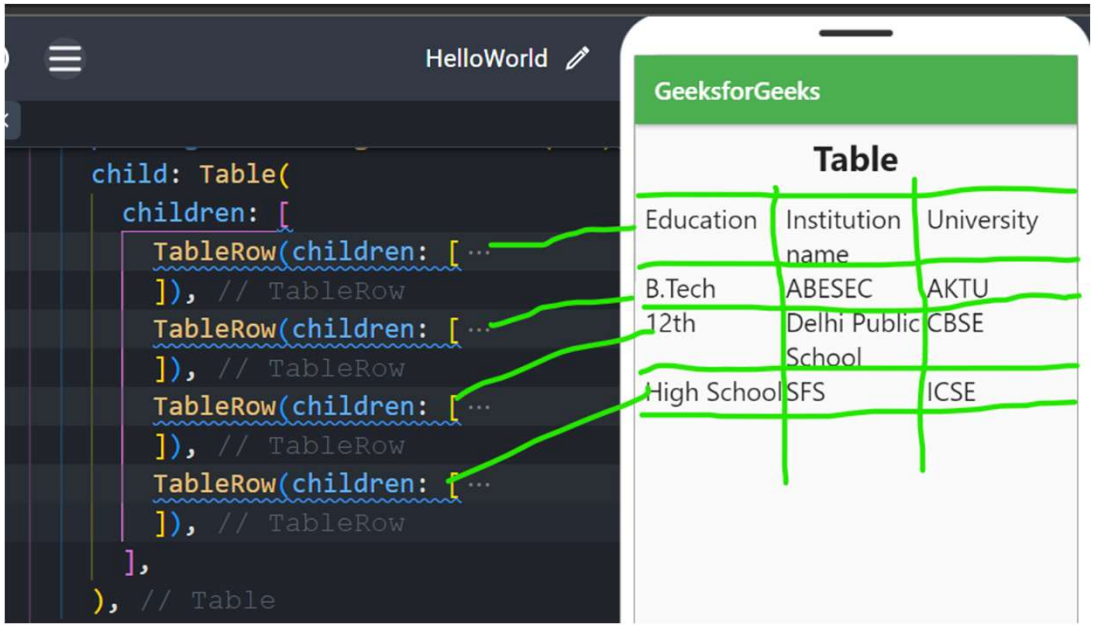


Esta classe é útil se você deseja sobrepor vários filhos de uma forma simples, por exemplo tendo algum texto e uma imagem, sobrepostos com...

https://docs.flutter.dev/ui/widgets/layout

Table

Usando uma **table** você pode posicionar widgets como se fossem uma listview porém sem scroll



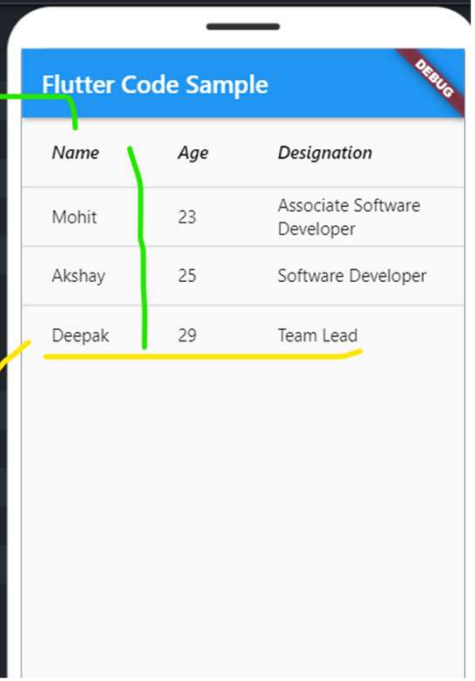
Data Table

Usando uma **datatable** você posiciona os widgets em colunas de dados e linhas de dados.
O dataTable dimensiona as colunas automaticamente.

```
25  @override
26  Widget build(BuildContext context) {
27    return DataTable(
28      columns: const <DataColumn>[ ···
47      ], // <DataColumn>[]
48      rows: const <DataRow>[ ···
70      ], // <DataRow>[]
71    ); // DataTable
72  }
```

main.dart M

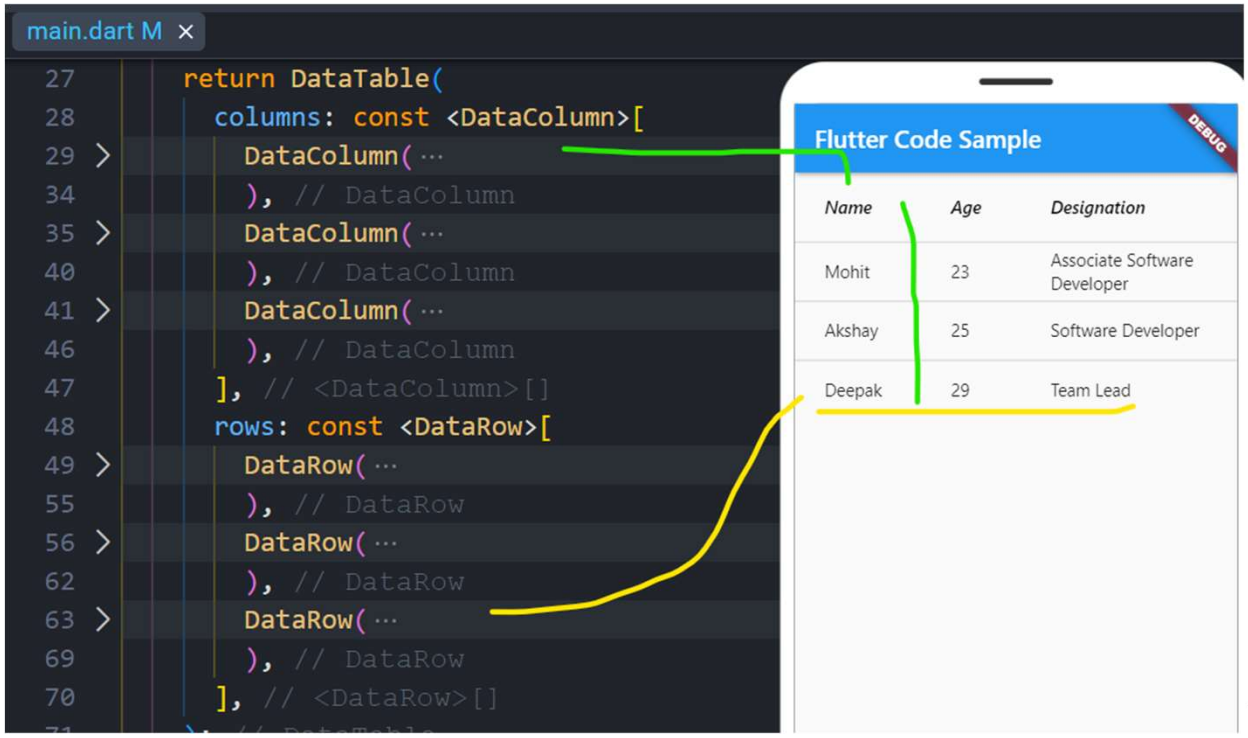
```
27    return DataTable(
28      columns: const <DataColumn>[
29        DataColumn( ···
34        ), // DataColumn
35        DataColumn( ···
40        ), // DataColumn
41        DataColumn( ···
46        ), // DataColumn
47      ], // <DataColumn>[]
48      rows: const <DataRow>[
49        DataRow( ···
55        ), // DataRow
56        DataRow( ···
62        ), // DataRow
63        DataRow( ···
69        ), // DataRow
70      ], // <DataRow>[]
```



Paginated Data Table

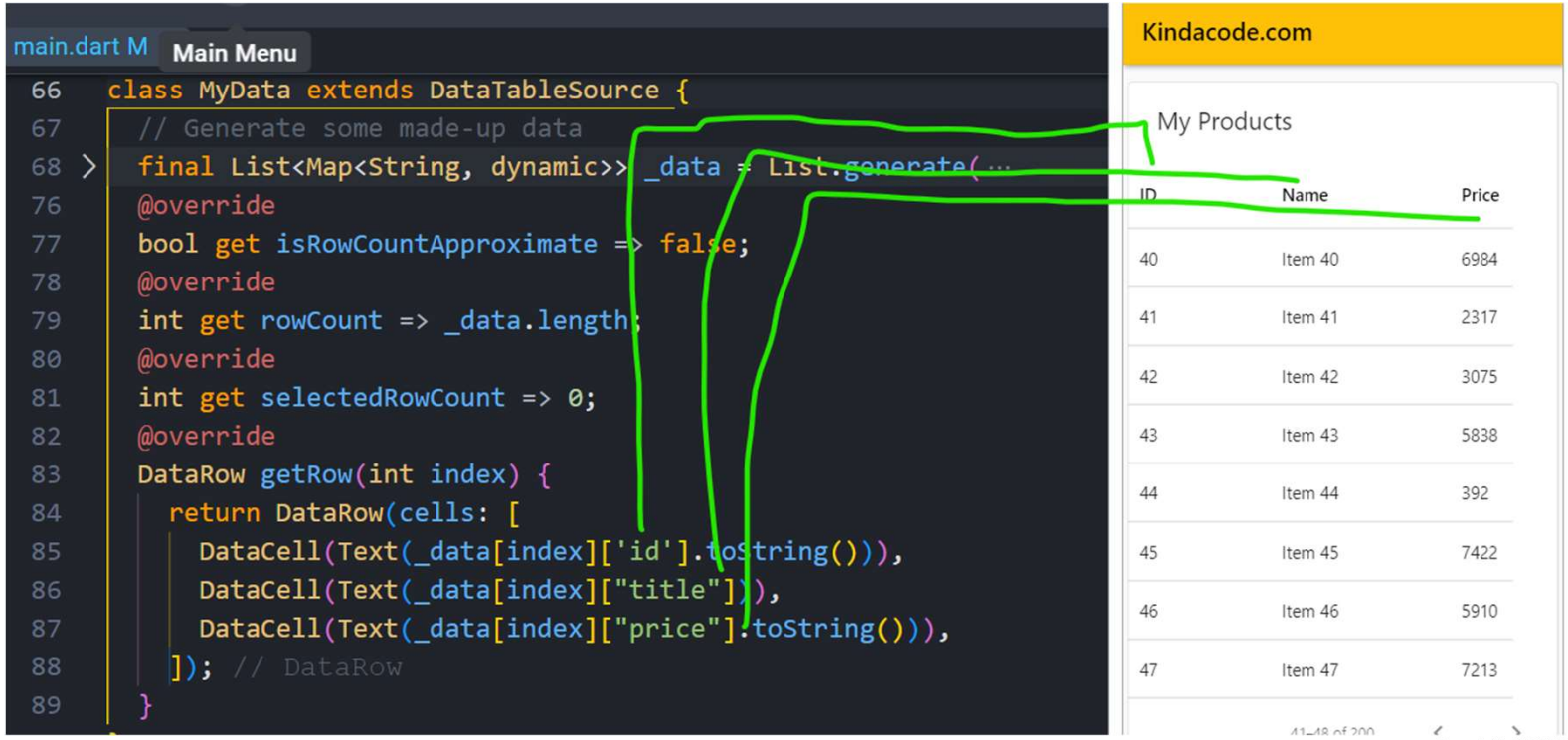
Usando uma **PaginatedDatatable** você pagina os todos os dados. Para isso é preciso definir a fonte de dados.

```
25   @override
26   Widget build(BuildContext context) {
27     return DataTable(
28 >     columns: const <DataColumn>[ ···
47       ], // <DataColumn>[]
48 >     rows: const <DataRow>[ ···
70       ], // <DataRow>[]
71     ); // DataTable
72   }
```



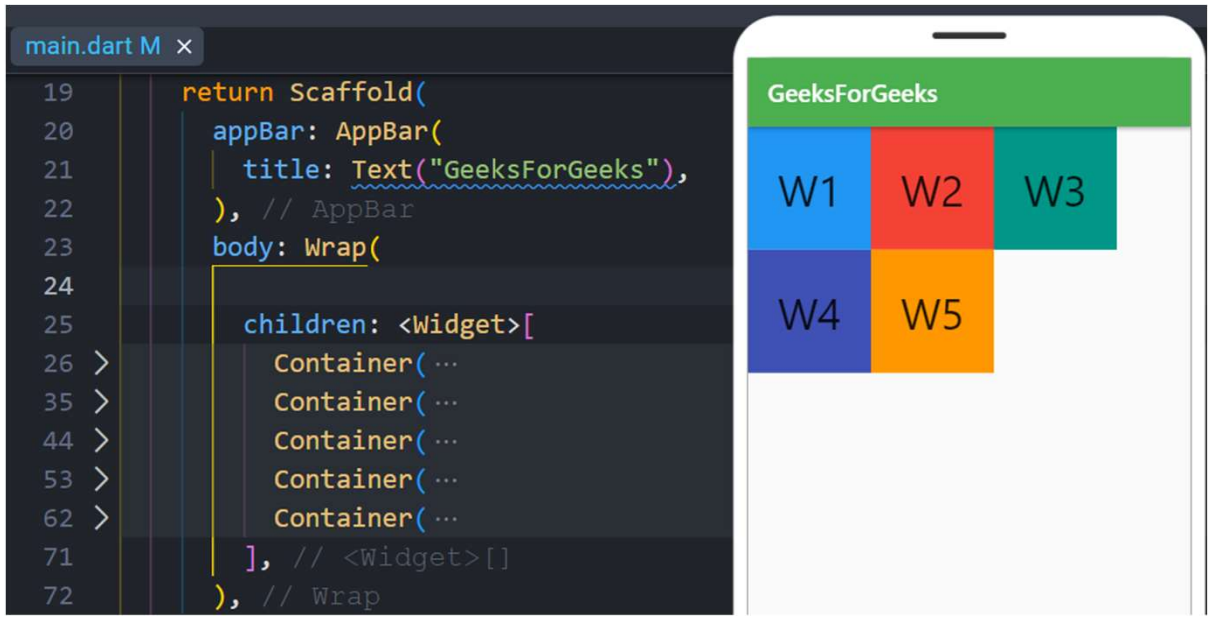
Paginated Data Table

Definindo a fonte de dados.



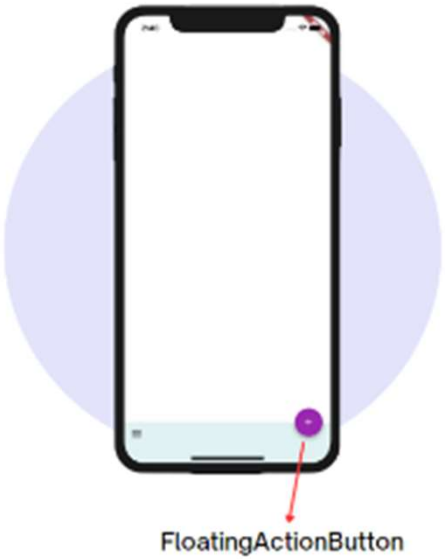
WRAP

Um **Wrap** dispõe cada child e tenta colocar o child adjacente ao child anterior no eixo principal, deixando espaço entre eles. Se não houver espaço suficiente para acomodar o child, Wrap criará um novo trecho adjacente aos children existentes no eixo.



FloatingActionButton

Um floating action button (FAB) executa a ação principal ou mais comum em uma tela.
Ele aparece na frente de todo o conteúdo da tela, normalmente como uma forma circular com um ícone no centro.



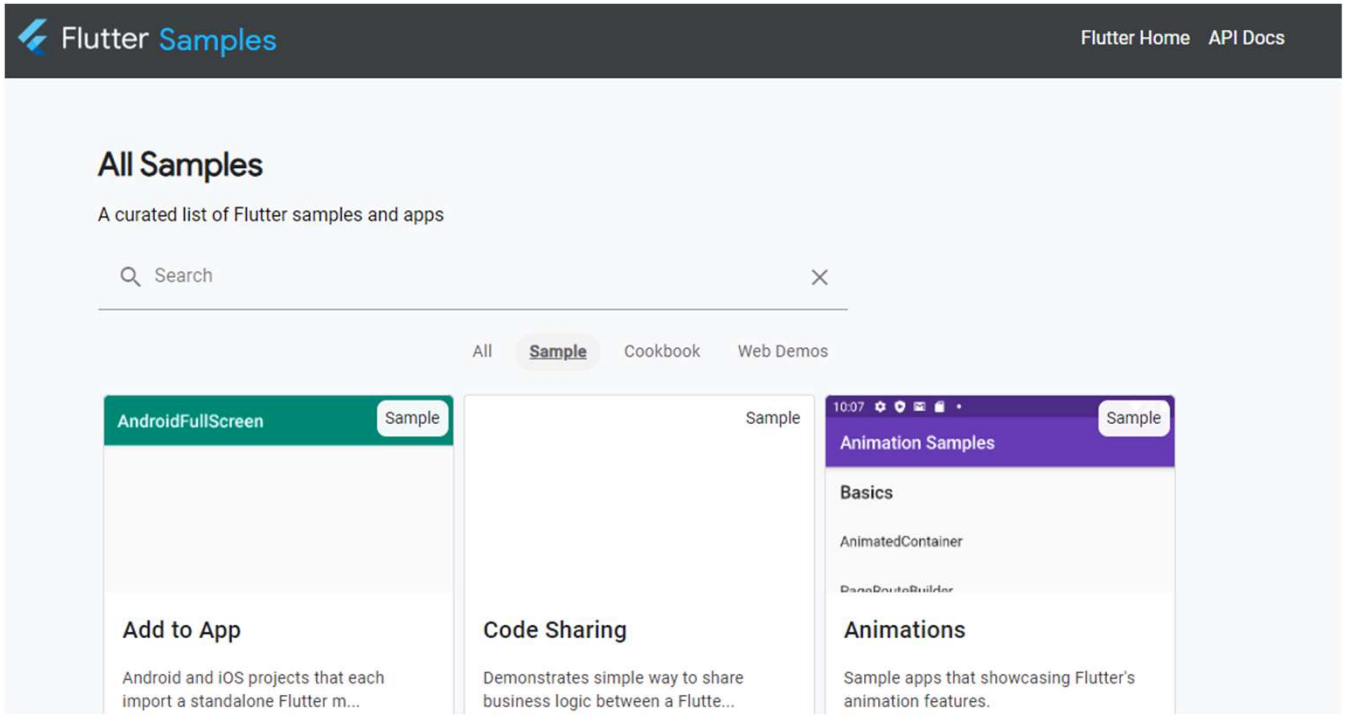
(FAB) são mais comumente usados no Scaffold.

(FAB) é único.

FloatingActionButton.01 e 02

FLUTTER

Exemplos



https://flutter.github.io/samples/#?type=sample