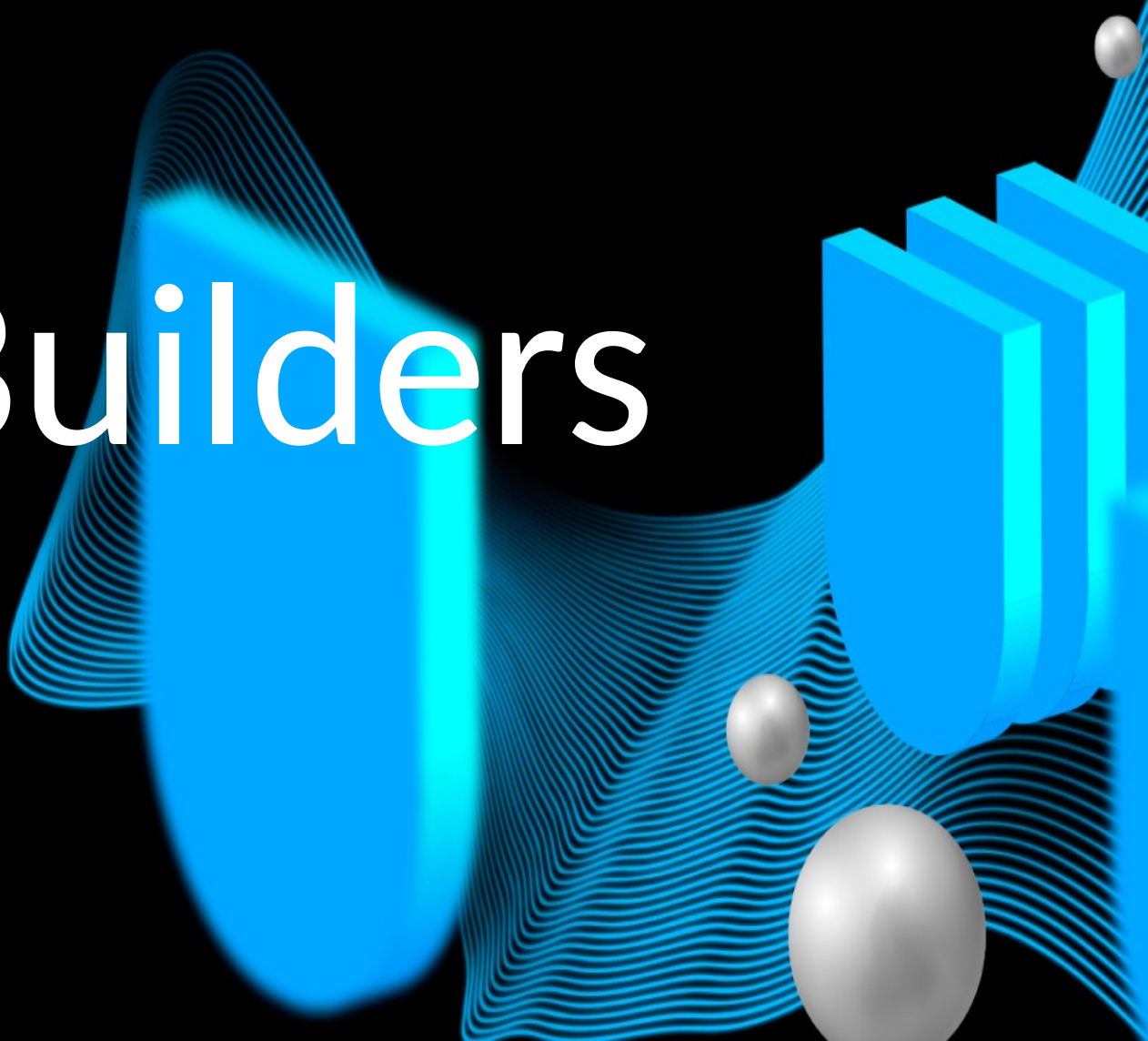

UNITAU
Universidade de Taubaté

# Builders



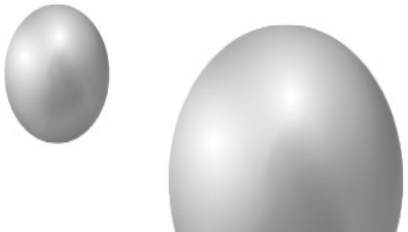
# Builders

Flutter possui alguns objetos voltados para a criação de widgets, os builders.

Existem vários builders que permitem criar algum widget específico ou permitem criar widgets segundo alguma condição ou evento.

Os builders permitem o uso de comandos imperativos para criar os widgets em tempo de execução e devem retornar o widget criado.

Existem também alguns widgets que possuem construtores que permitem criar o widget dinamicamente.



# Builder

**Builder** permite criar qualquer widget em tempo de execução passando para esse widget o contexto da execução do builder.

Crie uma nova aplicação Flutter com o nome **builder**.

# Builder

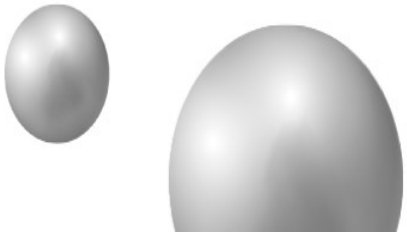
## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

# Builder

## Inclua a classe com a tela da aplicação:

```
class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Builder"),
      ),
      body: Container(),
      floatingActionButton: FloatingActionButton(onPressed:
() {
        Scaffold.of(context).showSnackBar(
          SnackBar(
            content: Text('SnackBar'),
          ),
        );
      }),
    );
  }
}
```

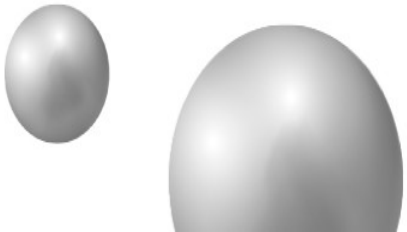


# Builder

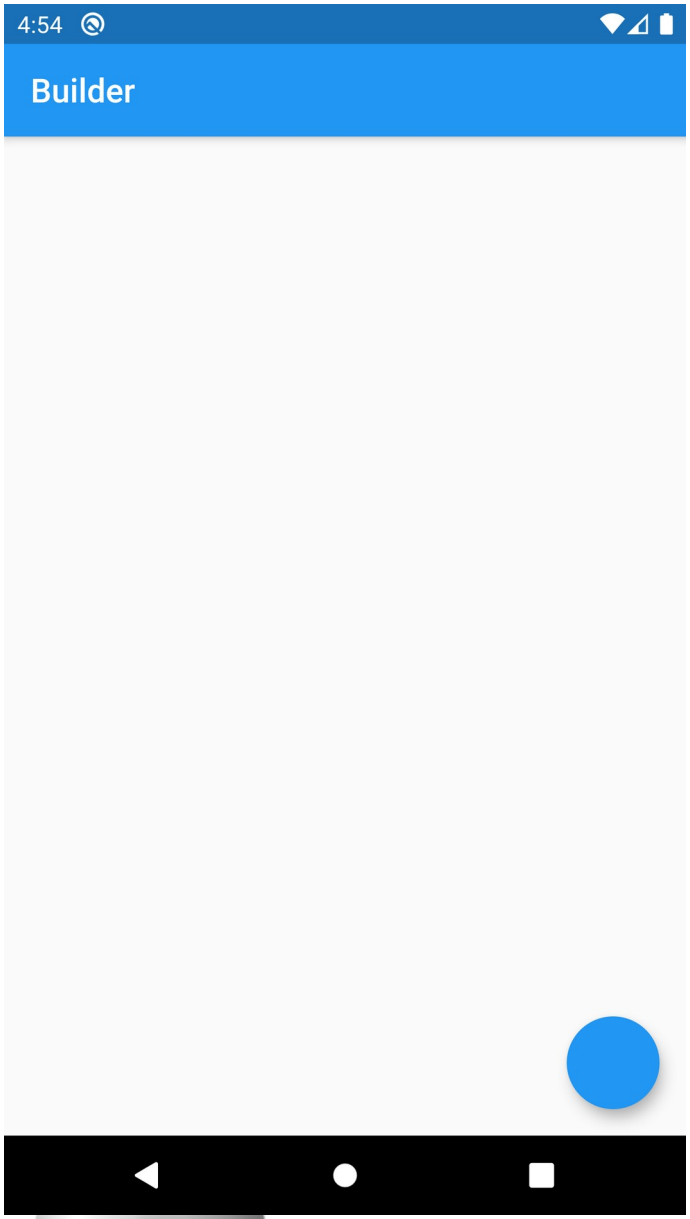
A aplicação deveria exibir uma mensagem no Snackbar, porém não irá funcionar por tentar usar o contexto para encontrar o Scaffold da aplicação:

```
Scaffold.of(context).showSnackBar(
```

No momento da criação do FloatingActionButton, o Scaffold ainda não existe e o contexto passado não é parte de um Scaffold.



# Builder



# Builder

```
An Observatory debugger and profiler on Android SDK built for x86 64 is available at:
http://127.0.0.1:43231/7X4vWQaMOnM=/
D/EGL_emulation( 8356): eglMakeCurrent: 0x7b3a0adda2c0: ver 2 0 (tinfo 0x7b39302b0ac0)
I/flutter ( 8356): ══╡ EXCEPTION CAUGHT BY GESTURE ╞═══════════════════════════════════
═══════════════════════════

I/flutter ( 8356): The following assertion was thrown while handling a gesture:
I/flutter ( 8356): Scaffold.of() called with a context that does not contain a Scaffold.
I/flutter ( 8356): No Scaffold ancestor could be found starting from the context that was
passed to Scaffold.of(). This
I/flutter ( 8356): usually happens when the context provided is from the same StatefulWidg
et as that whose build
I/flutter ( 8356): function actually creates the Scaffold widget being sought.
I/flutter ( 8356): There are several ways to avoid this problem. The simplest is to use a
Builder to get a context that
I/flutter ( 8356): is "under" the Scaffold. For an example of this, please see the documen
tation for Scaffold.of():
I/flutter ( 8356):   https://api.flutter.dev/flutter/material/Scaffold/of.html
I/flutter ( 8356): A more efficient solution is to split your build function into several
widgets. This introduces a
I/flutter ( 8356): new context from which you can obtain the Scaffold. In this solution, y
ou would have an outer widget
I/flutter ( 8356): that creates the Scaffold populated by instances of your new inner widg
ets, and then in these inner
I/flutter ( 8356): widgets you would use Scaffold.of().
I/flutter ( 8356): A less elegant but more expedient solution is assign a GlobalKey to the
 Scaffold, then use the
```

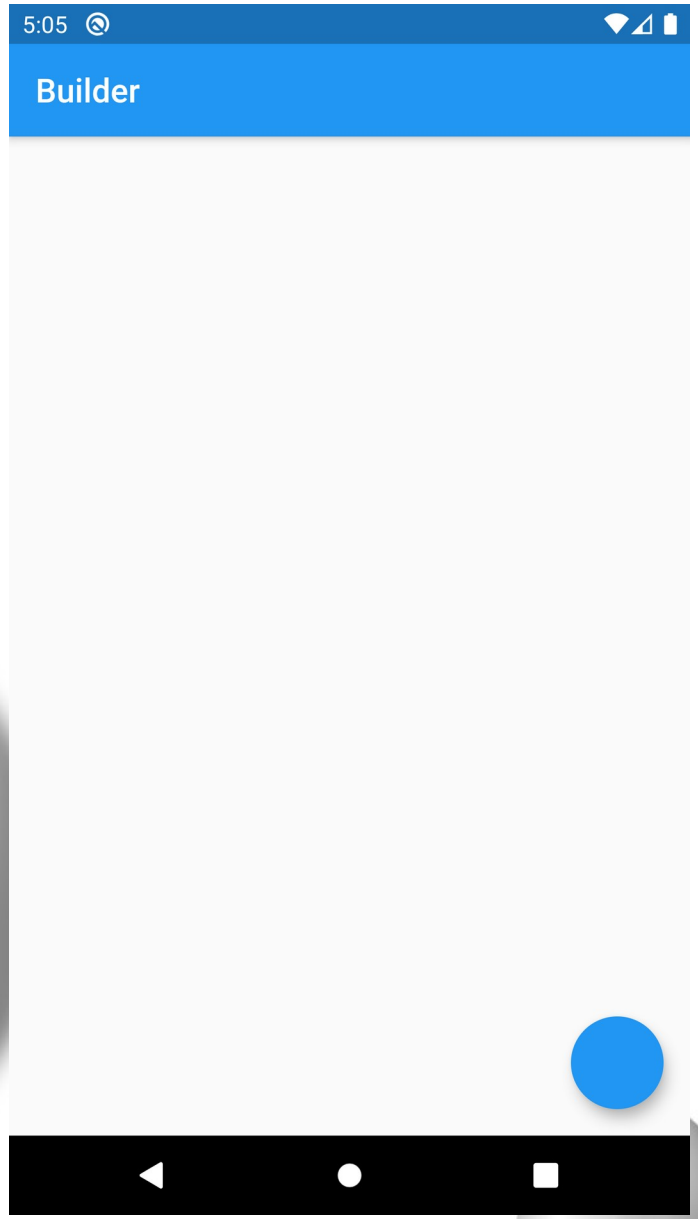
# Builder

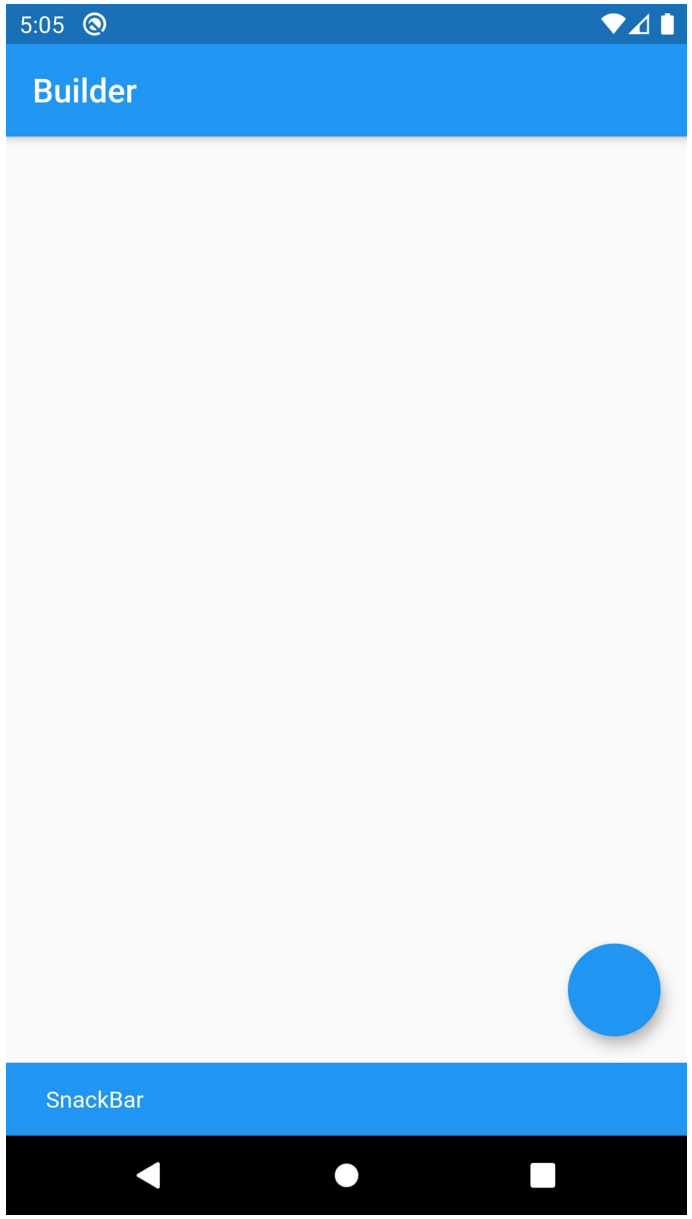
Para corrigir esse problema, pode ser usado o **Builder** que irá criar o botão em tempo de execução e irá fornecer um contexto dentro da árvore do **Scaffold**.

Altere o **floatingActionButton** da tela da aplicação para:

```
      floatingActionButton: Builder(builder: (BuildContext
context) {
        return FloatingActionButton(onPressed: () {
          Scaffold.of(context).showSnackBar(
            SnackBar(
              backgroundColor: Colors.blue,
              content: Text('SnackBar'),
            ),
          );
        });
      }),
```

# Builder



# ListView.builder

**ListView.builder** é um construtor que permite criar um **ListView** em tempo de execução. Uma das possibilidades é criar o **ListView** baseado em um **List**.

Crie uma nova aplicação Flutter com o nome **builder_listview**.

# ListView.builder

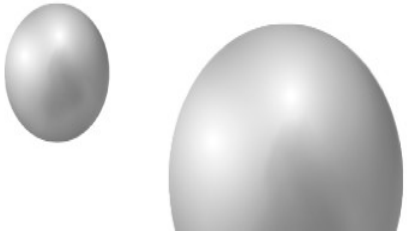
## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

# ListView.builder

## Inclua a classe com a tela da aplicação:

```
class Home extends StatefulWidget {
  Home({Key key}) : super(key: key);

  @override
  _HomeState createState() => _HomeState();
}
```



# ListView.builder

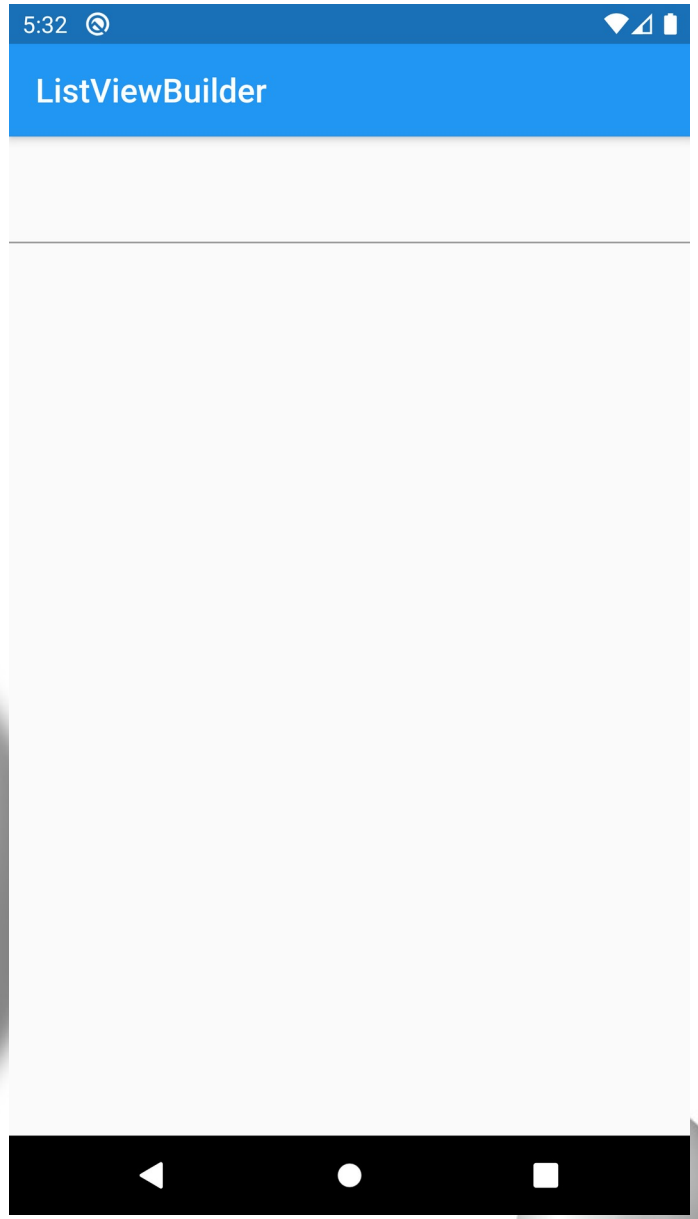
## Inclua a classe para gerenciar o estado da tela da aplicação:

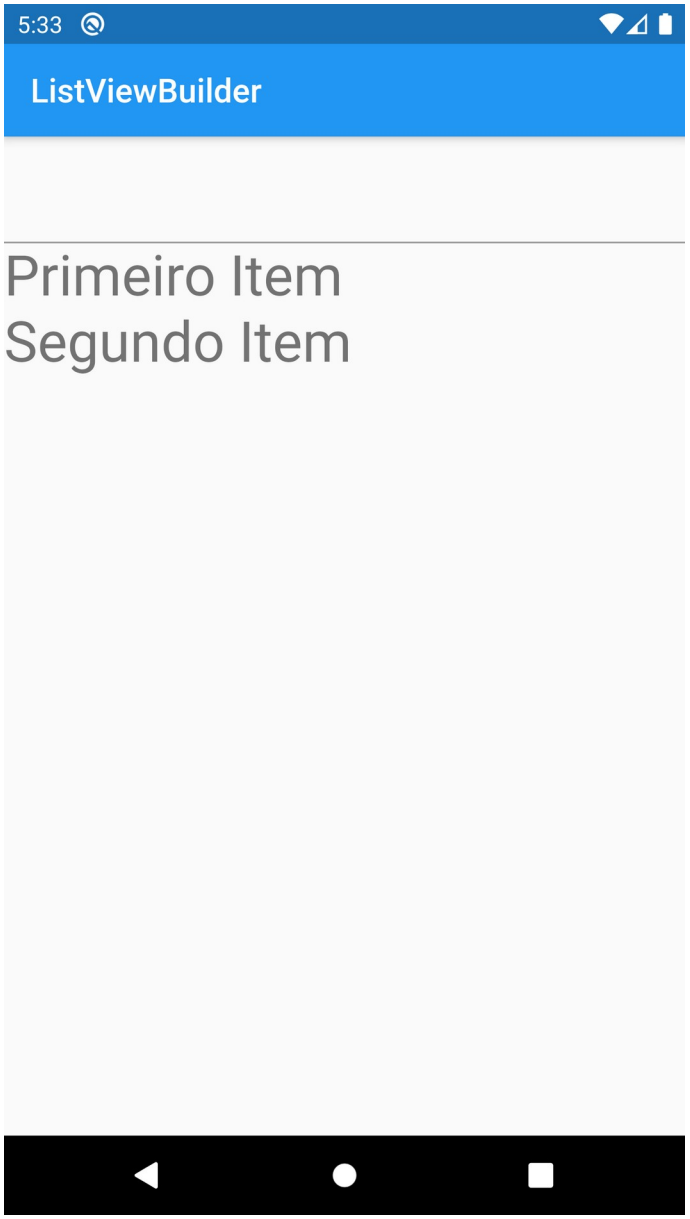
```
class _HomeState extends State<Home> {
  List<String> litems = [];
  final TextEditingController eCtrl =
TextEditingController();
  @override
  Widget build (BuildContext ctxt) {
    return Scaffold(
      appBar: AppBar(title: Text("ListViewBuilder"),),
      body: Column(
        children: <Widget>[
          TextField(
            style: Theme.of(context).textTheme.display1,
            controller: eCtrl,
            onSubmitted: (text) {
              litems.add(text);
              eCtrl.clear();
              setState(() {});
            },
          ),
```

# ListView.builder

```
            Expanded(
              child: ListView.builder(
                itemCount: litems.length,
                itemBuilder: (BuildContext ctxt, int Index) {
                  return Text(litems[Index],
                    style:
Theme.of(context).textTheme.display1,
                  );
                }
              )
            )
          ],
        )
      );
  }
}
```

# ListView.builder



# GridView.builder

**GridView.builder** é um construtor que permite criar um **GridView** em tempo de execução. Uma das possibilidades é criar o **GridView** baseado em um **List**.

Crie uma nova aplicação Flutter com o nome **builder_gridview**.

Crie o diretório **assets/images**.

Altere o arquivo **pubspec.yaml** para referenciar o diretório **assets/images**.

Copie os arquivos do diretório **Flags** para o diretório **assets/images**.

# GridView.builder

## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

# GridView.builder

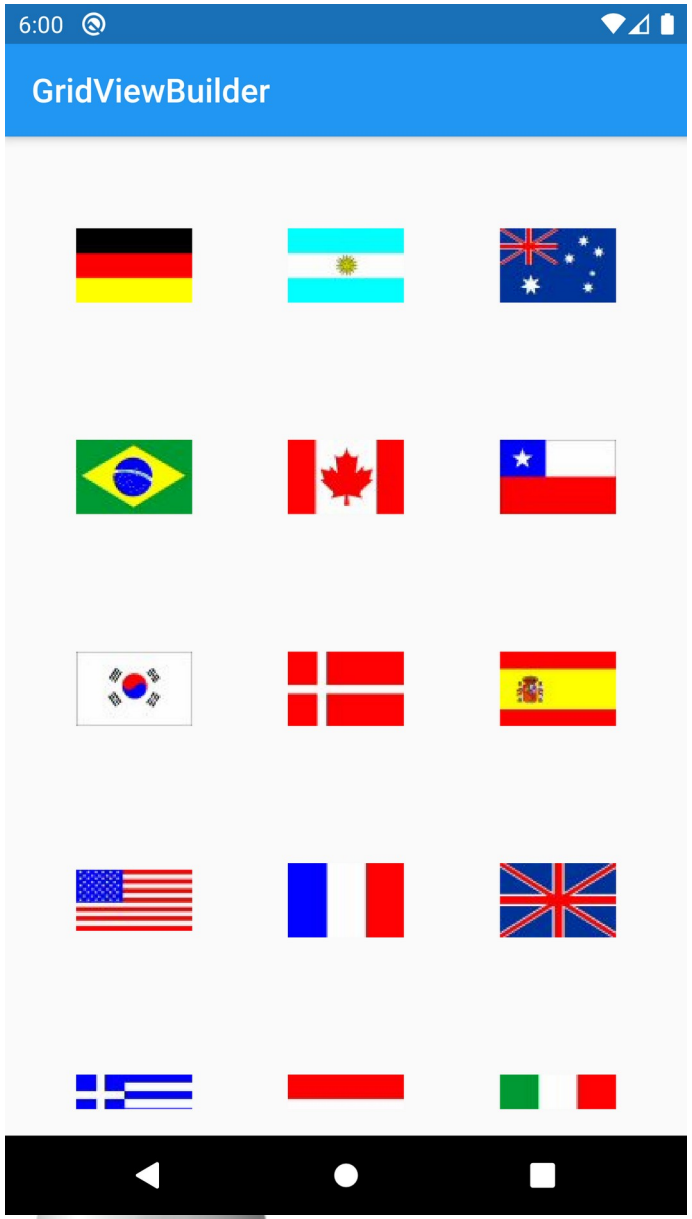
## Inclua a classe com a tela da aplicação:

```
class Home extends StatelessWidget {
  List<String> flags = [
    "assets/images/alemanha.jpg",
    "assets/images/argentina.jpg",
    "assets/images/australia.jpg",
    "assets/images/brasil.jpg",
    "assets/images/canada.jpg",
    "assets/images/chile.jpg",
    "assets/images/coreiadosul.jpg",
    "assets/images/dinamarca.jpg",
    "assets/images/espanha.jpg",
    "assets/images/estadosunidos.jpg",
    "assets/images/franca.jpg",
    "assets/images/gra-bretanha.jpg",
    "assets/images/grecia.jpg",
    "assets/images/holanda.jpg",
    "assets/images/italia.jpg",
    "assets/images/japao.jpg",
    "assets/images/mexico.jpg",
    "assets/images/noruega.jpg",
```

# GridView.builder

```
    "assets/images/portugal.jpg",
    "assets/images/uruguai.jpg",
  ];
  @override
  Widget build (BuildContext ctxt) {
    return Scaffold(
      appBar: AppBar(title: Text("GridViewBuilder"),),
      body: Container(
        padding: EdgeInsets.all(16.0),
        child: GridView.builder(
          itemCount: flags.length,
          gridDelegate:
SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 3,
crossAxisSpacing: 4.0, mainAxisSpacing: 4.0),
          itemBuilder: (BuildContext context, int index){
            return Image.asset(flags[index]);
          },
        )
      ),
    );
  }
}
```

# GridView.builder



# LayoutBuilder

**LayoutBuilder** permite criar widgets de acordo com o tamanho do widget superior na árvore. O **builder** do **LayoutBuilder** irá receber como parâmetro o contexto e um objeto **BoxConstraints** com as dimensões disponíveis para criar os widgets.

Crie uma nova aplicação Flutter com o nome **builder_layout**.

# LayoutBuilder

## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

# LayoutBuilder

## Inclua a classe com a tela da aplicação:

```
class Home extends StatelessWidget {

  @override
  Widget build (BuildContext ctxt) {
    print("Criando Home");
    return Scaffold(
      appBar: AppBar(title: Text("LayoutBuilder"),),
      body: Container(
        padding: EdgeInsets.all(16.0),
        child: LayoutBuilder(
          builder: (context, constraints) {
            print("Recriando LayoutBuilder");
            return constraints.maxWidth > 600
              ? _twoContainers() : _oneContainer();
          },
        ),
      ),
    );
  }
}
```

# LayoutBuilder

**Inclua na classe Home o widget para telas com largura pequena:**

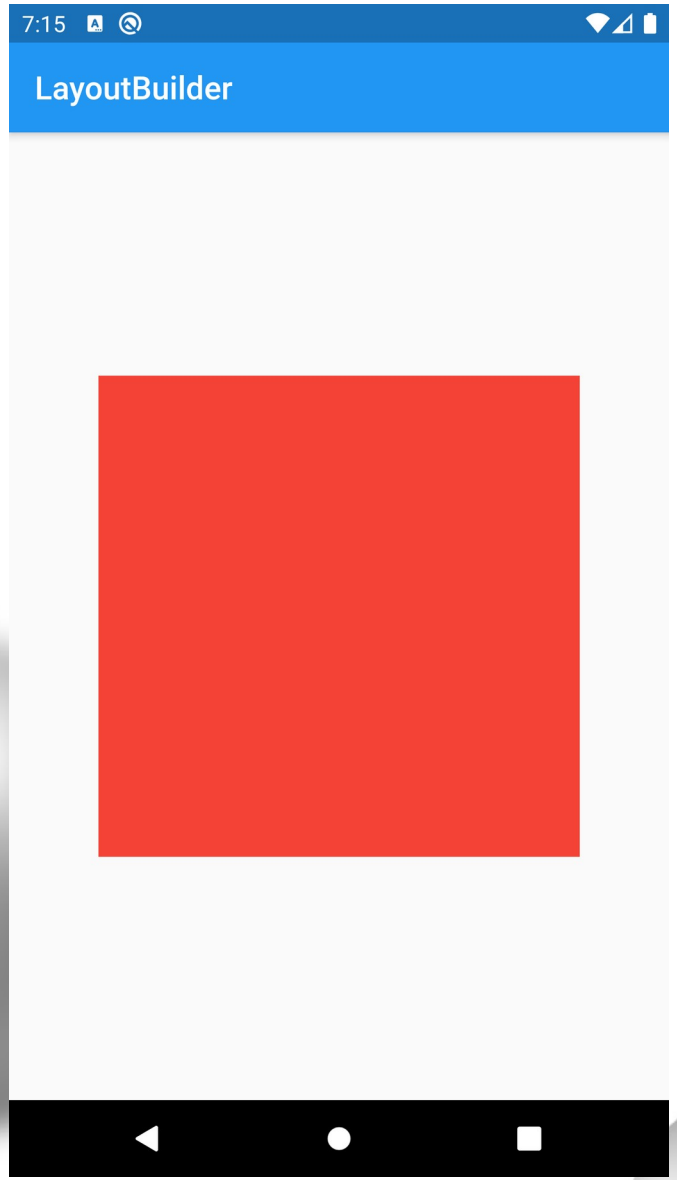
```
Widget _oneContainer() {
  return Center(
    child: Container(
      height: 300.0,
      width: 300.0,
      color: Colors.red,
    ),
  );
}
```

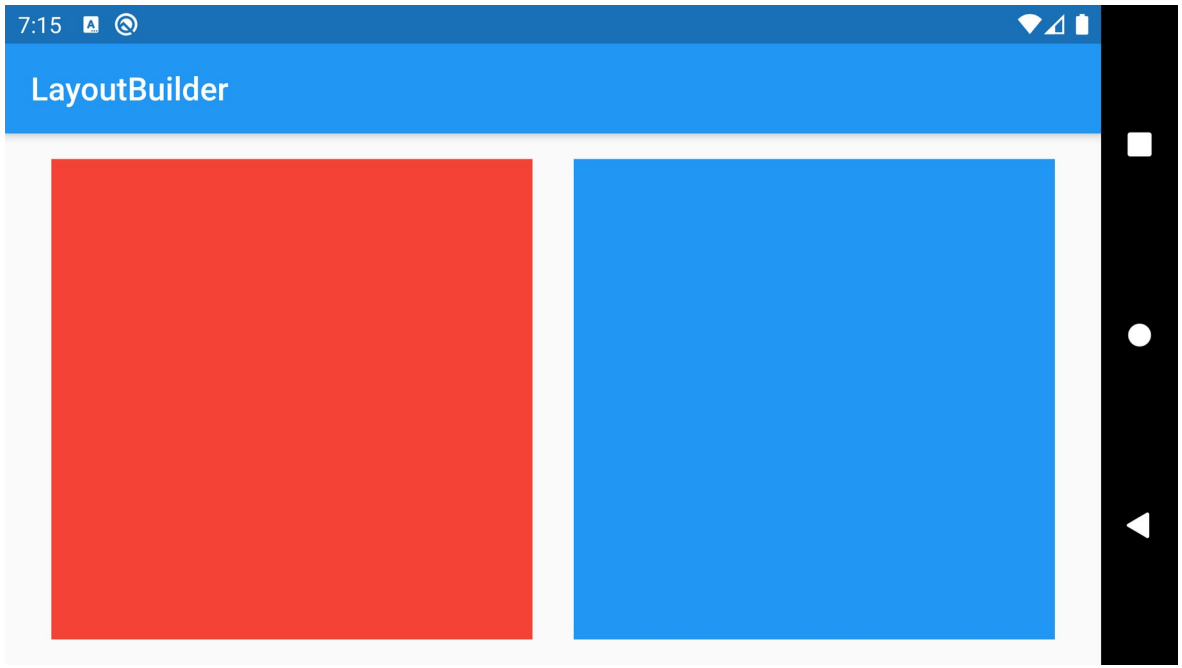
# LayoutBuilder

**Inclua na classe Home o widget para telas com largura grande:**

```
Widget _twoContainers() {
  return Center(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: <Widget>[
        Container(
          height: 300.0,
          width: 300.0,
          color: Colors.red,
        ),
        Container(
          height: 300.0,
          width: 300.0,
          color: Colors.blue,
        ),
      ],
    ),
  );
}
```

# LayoutBuilder



# OrientationBuilder

**OrientationBuilder** permite criar widgets de acordo com a orientação da tela. O **builder** do **OrientationBuilder** irá receber como parâmetro o contexto e a orientação da tela.

Crie uma nova aplicação Flutter com o nome **builder_orientation**.

# OrientationBuilder

## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

# OrientationBuilder

## Inclua a classe com a tela da aplicação:

```
class Home extends StatelessWidget {

  @override
  Widget build (BuildContext ctxt) {
    print("Criando Home");
    return Scaffold(
      appBar: AppBar(title: Text("OrietationBuilder"),),
      body: Container(
        padding: EdgeInsets.all(16.0),
        child: OrientationBuilder(
          builder: (context, orientation) {
            print("Recriando OrientationBuilder");
            return orientation == Orientation.landscape
              ? _landscape() : _portrait();
          },
        ),
      ),
    );
  }
}
```

# OrientationBuilder

**Inclua na classe Home o widget para orientação vertical:**

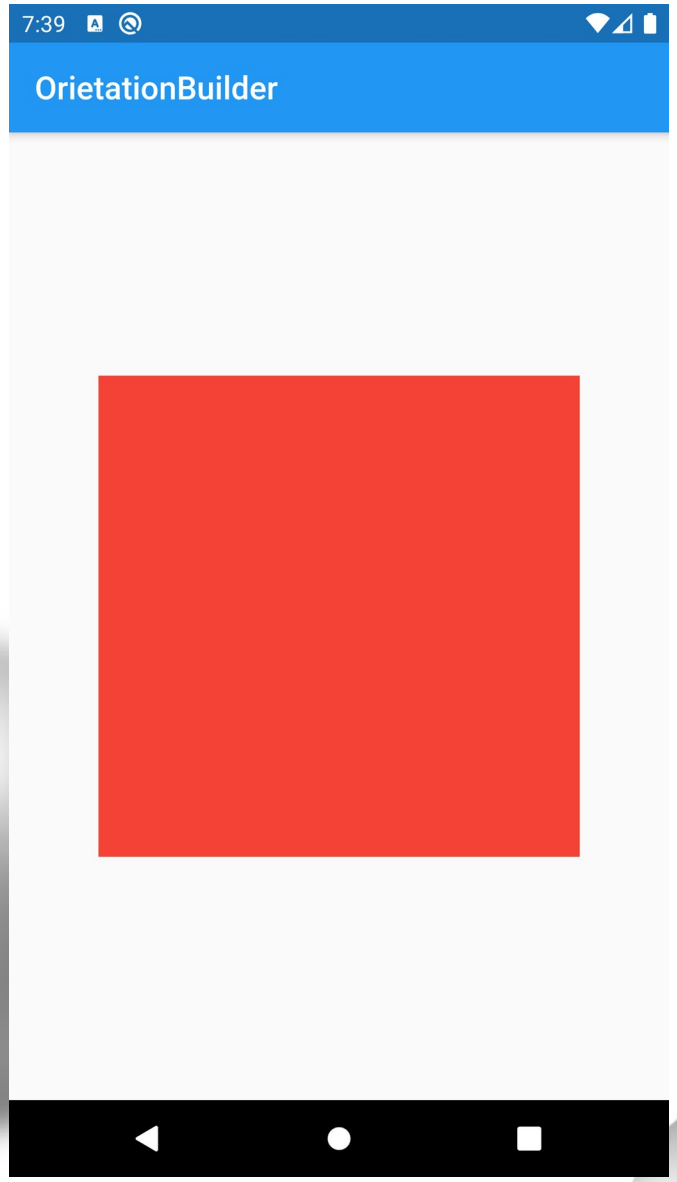
```
Widget _portrait() {
  return Center(
    child: Container(
      height: 300.0,
      width: 300.0,
      color: Colors.red,
    ),
  );
}
```

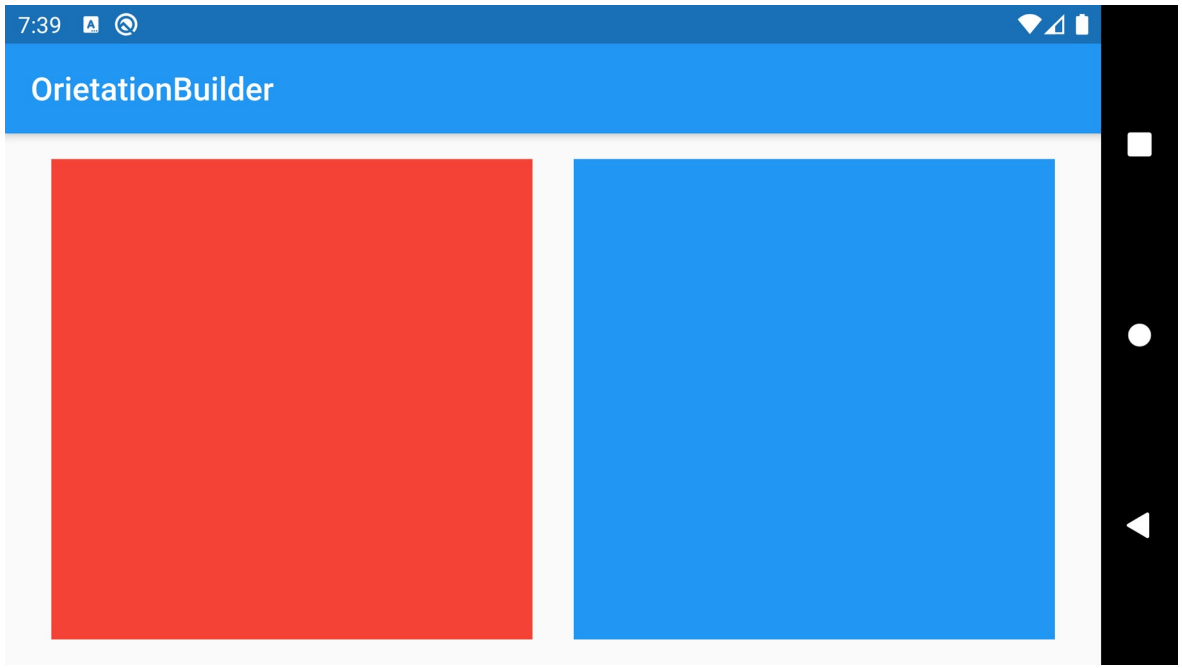
# OrientationBuilder

**Inclua na classe Home o widget para orientação horizontal:**

```
Widget _landscape() {
  return Center(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: <Widget>[
        Container(
          height: 300.0,
          width: 300.0,
          color: Colors.red,
        ),
        Container(
          height: 300.0,
          width: 300.0,
          color: Colors.blue,
        ),
      ],
    ),
  );
}
```

# OrientationBuilder



# StatefulBuilder

**StatefulBuilder** permite criar dinamicamente um **StatefulWidget** e o seu estado.

Crie uma nova aplicação Flutter com o nome **builder_stateful**.

# StatefulBuilder

## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

# StatefulBuilder

## Inclua a classe com a tela da aplicação:

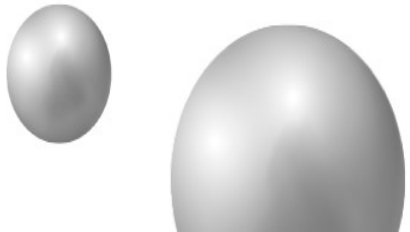
```
class Home extends StatelessWidget {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    print("Criando Home");
    return Scaffold(
      appBar: AppBar(
        title: Text("SatefulBuilder"),
      ),
```

# StatefulBuilder

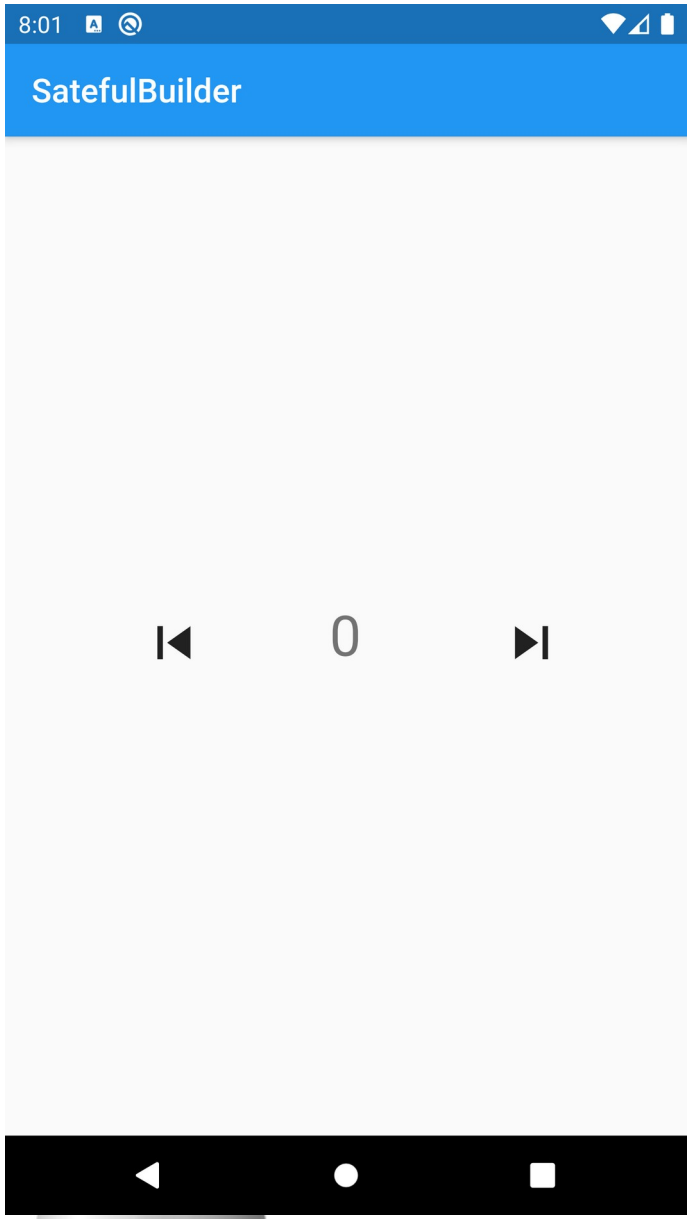
```
      body: Center(
        child: StatefulBuilder(
          builder: (BuildContext context, StateSetter
setState) {
            print("Recriando StatefulWidget");
            return Row(
              mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
              children: <Widget>[
                IconButton(
                  icon: Icon(Icons.skip_previous,size:
40.0,),
                  onPressed: () {
                    setState(() => _counter--);
                  },
                ),
                Text(
                  "$_counter",
                  style:
Theme.of(context).textTheme.display1,
                ),
```

# StatefulBuilder

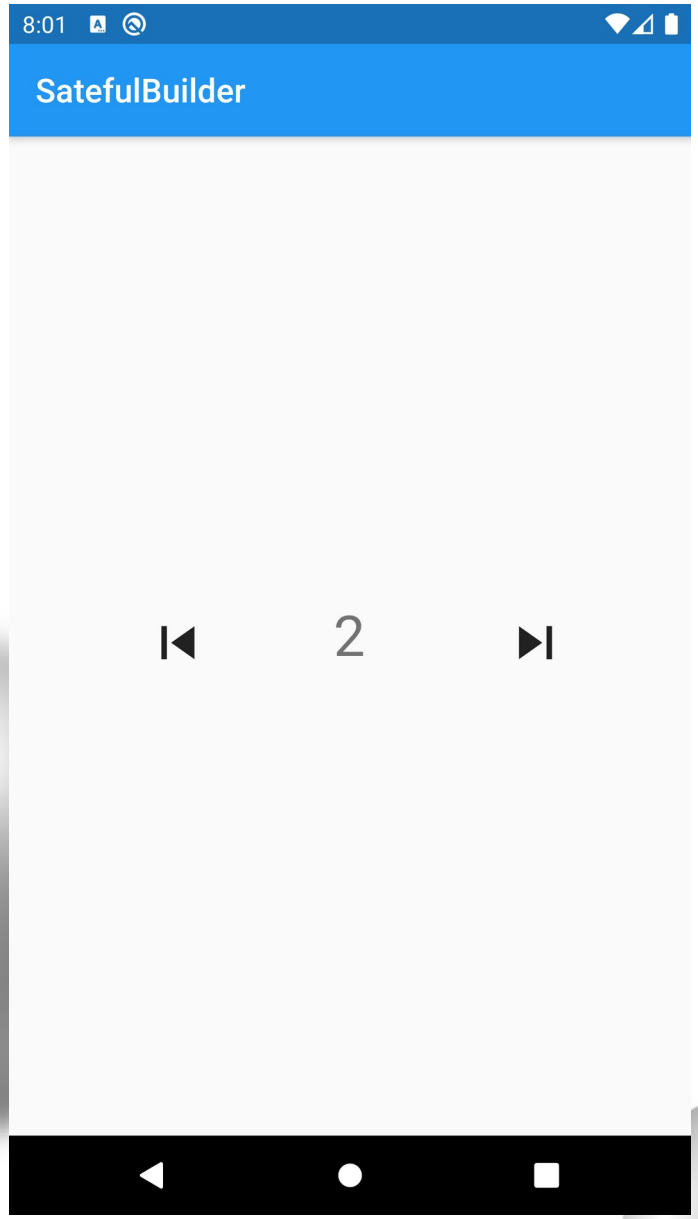
```
                IconButton(
                  icon: Icon(Icons.skip_next,size: 40.0,),
                  onPressed: () {
                    setState(() => _counter++);
                  },
                ),
              ],
            );
          },
        ),
      ),
    );
  }
}
```



# StatefulBuilder



# StatefulBuilder




```
An Observatory debugger and profiler on Androt
http://127.0.0.1:38195/ryRAr6gVAeY=/
D/EGL_emulation(31792): eglMakeCurrent: 0x76d2
I/flutter (31792): Criando Home
I/flutter (31792): Recriando StatefulWidget
I/flutter (31792): Recriando StatefulWidget
I/flutter (31792): Recriando StatefulWidget
```

# FutureBuilder

**FutureBuilder** permite criar widgets segundo o estado de um **Future**. O **builder** do **FutureBuilder** irá receber como parâmetro o contexto e um objeto **AsyncSnapshot** para obter o resultado do **Future**. O método **hasData** retorna **true** quando o **Future** estiver completo com resultado. O método **hasError** retorna **true** quando o **Future** estiver completo com erro.

Crie uma nova aplicação Flutter com o nome **builder_future**.

# FutureBuilder

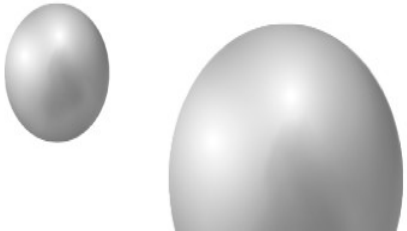
## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

# FutureBuilder

## Inclua a classe com a tela da aplicação:

```
class Home extends StatefulWidget {
  Home({Key key}) : super(key: key);

  @override
  _HomeState createState() => _HomeState();
}
```



# FutureBuilder

## Inclua a classe para gerenciar o estado da tela da aplicação:

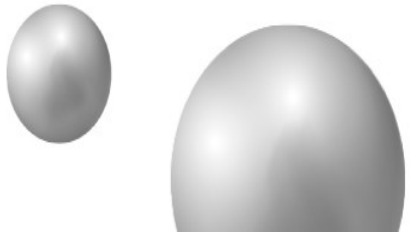
```
class _HomeState extends State<Home> {
  int times = 0;

  Widget build(BuildContext context) {
    print("Recriando Home");
    return Scaffold(
      appBar: AppBar(
        title: Text("FutureBuilder"),
      ),
      body: Center(
        child: FutureBuilder<String>(
          future: Future<String>.delayed( Duration(seconds:
10),
            () {
              times++;
              print("Calculando");
              return '${times} times';
            }
          ),
```

# FutureBuilder

```
          builder: (BuildContext context,
AsyncSnapshot<String> snapshot) {
            print("Recriando FutureBuilder");
            if (snapshot.hasData) {
              return Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment:
CrossAxisAlignment.center,
                children: <Widget> [
                  Icon(
                    Icons.check_circle_outline,
                    color: Colors.green,
                    size: 60,
                  ),
                  Padding(
                    padding: const EdgeInsets.only(top: 16),
                    child: Text('Result: ${snapshot.data}'),
                  )
                ]
            );
```

# FutureBuilder

```
            } else if (snapshot.hasError) {
              return Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment:
CrossAxisAlignment.center,
                children: <Widget> [
                  Icon(
                    Icons.error_outline,
                    color: Colors.red,
                    size: 60,
                  ),
                  Padding(
                    padding: const EdgeInsets.only(top: 16),
                    child: Text('Error: ${snapshot.error}'),
                  )
                ],
              );
```



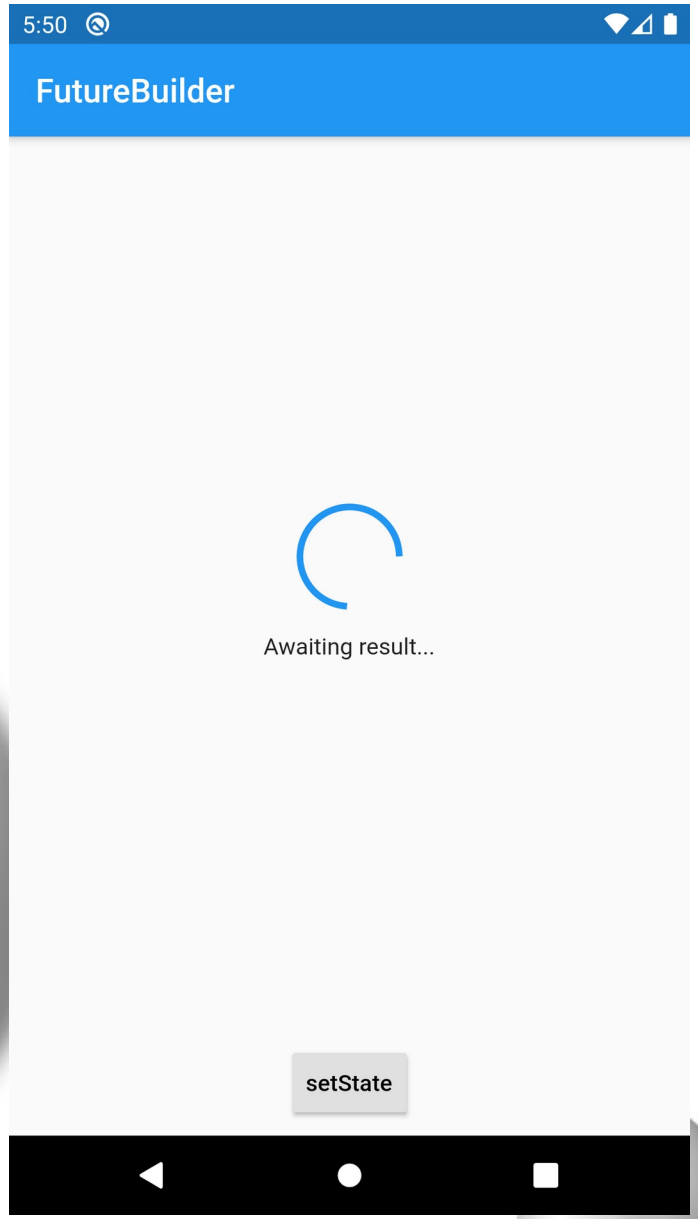
# FutureBuilder

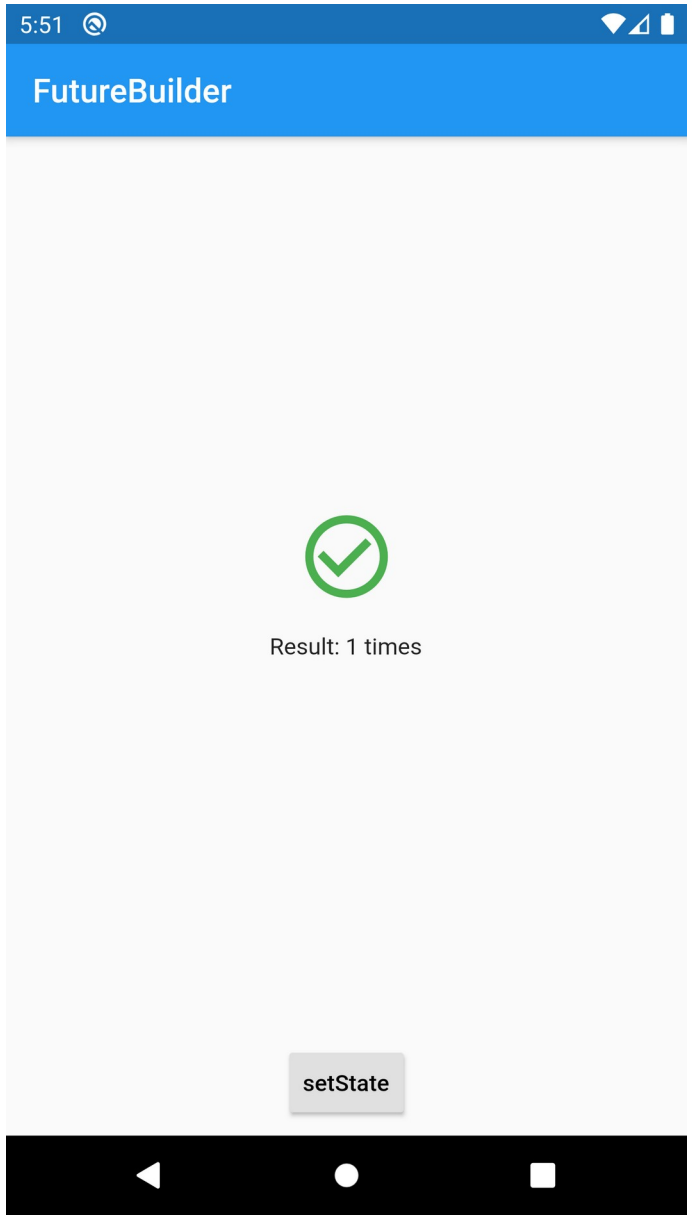
```
            } else {
              return Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment:
CrossAxisAlignment.center,
                children: <Widget> [
                  SizedBox(
                    child: CircularProgressIndicator(),
                    width: 60,
                    height: 60,
                  ),
                  const Padding(
                    padding: EdgeInsets.only(top: 16),
                    child: Text('Awaiting result...'),
                  )
                ],
              );
            }
          },
        ),
      ),
```

# FutureBuilder

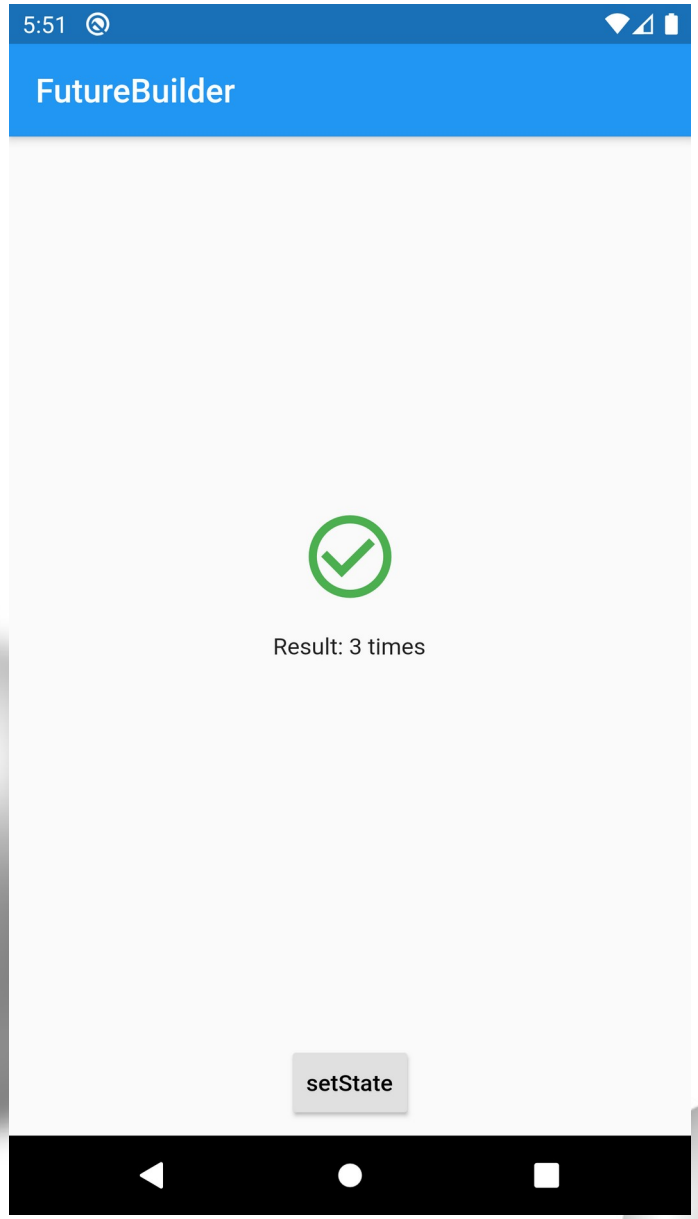
```
      bottomNavigationBar: ButtonBar(
        alignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          RaisedButton(
            child: Text("setState"),
            onPressed: () {
              setState(() {});
            },
          ),
        ],
      ),
    );
  }
}
```

# FutureBuilder



# FutureBuilder




```
An Observatory debugger and profiler on Andro
http://127.0.0.1:37069/eCiIgWdV7KA=/
I/flutter (13887): Recriando FutureBuilder
I/flutter (13887): Calculando
I/flutter (13887): Recriando FutureBuilder
I/flutter (13887): Recriando Home
I/flutter (13887): Recriando FutureBuilder
I/flutter (13887): Calculando
I/flutter (13887): Recriando FutureBuilder
I/flutter (13887): Recriando Home
I/flutter (13887): Recriando FutureBuilder
I/flutter (13887): Calculando
I/flutter (13887): Recriando FutureBuilder
```

# FutureBuilder

Se o **Future** é criado dentro do **FutureBuilder**, sempre que o **FutureBuilder** é recriado, o **Future** é novamente executado.

Se o **Future** deve ser executado uma única vez e não deve ser executado novamente quando o **FutureBuilder** for recriado, é necessário criar o **Future** fora do **FutureBuilder**, é recomentado que seja no **initState**, porém não é obrigatório que seja nesse local.

Altere a função **_HomeState**:

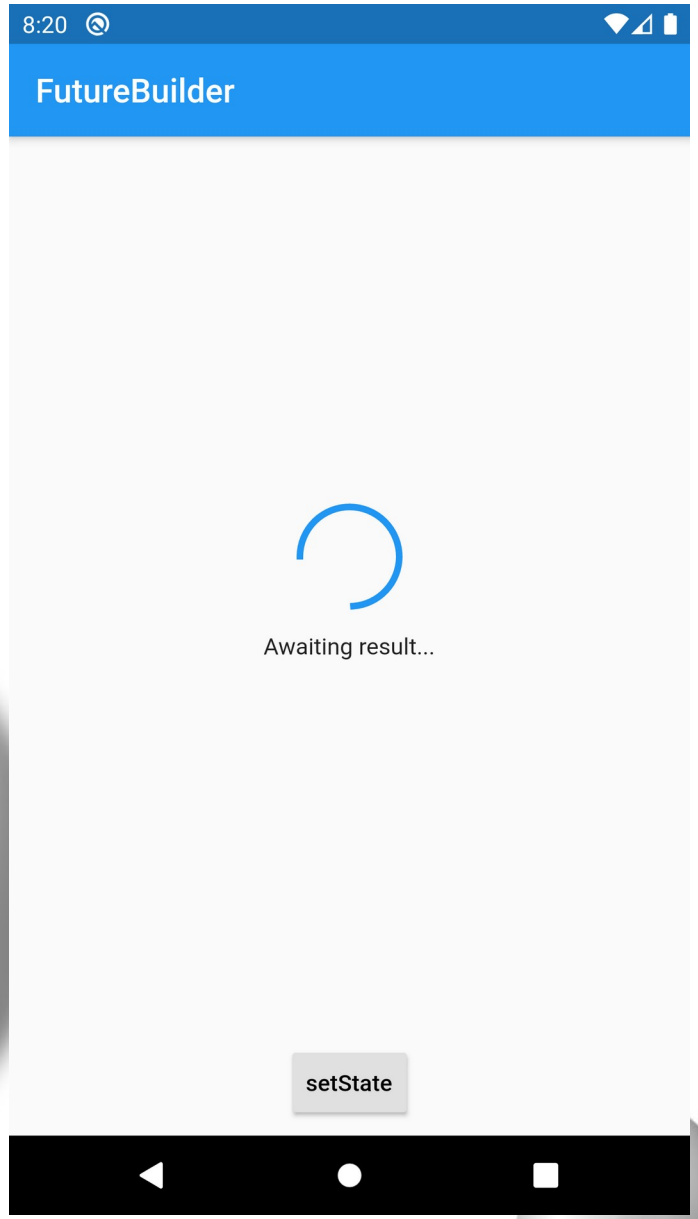
```
int times = 0;

class _HomeState extends State<Home> {
  Future<String> _calculation =
Future<String>.delayed( Duration(seconds: 10),
    () { times++;
      print("Calculando");
      return '${times} times';
    }
  );
```

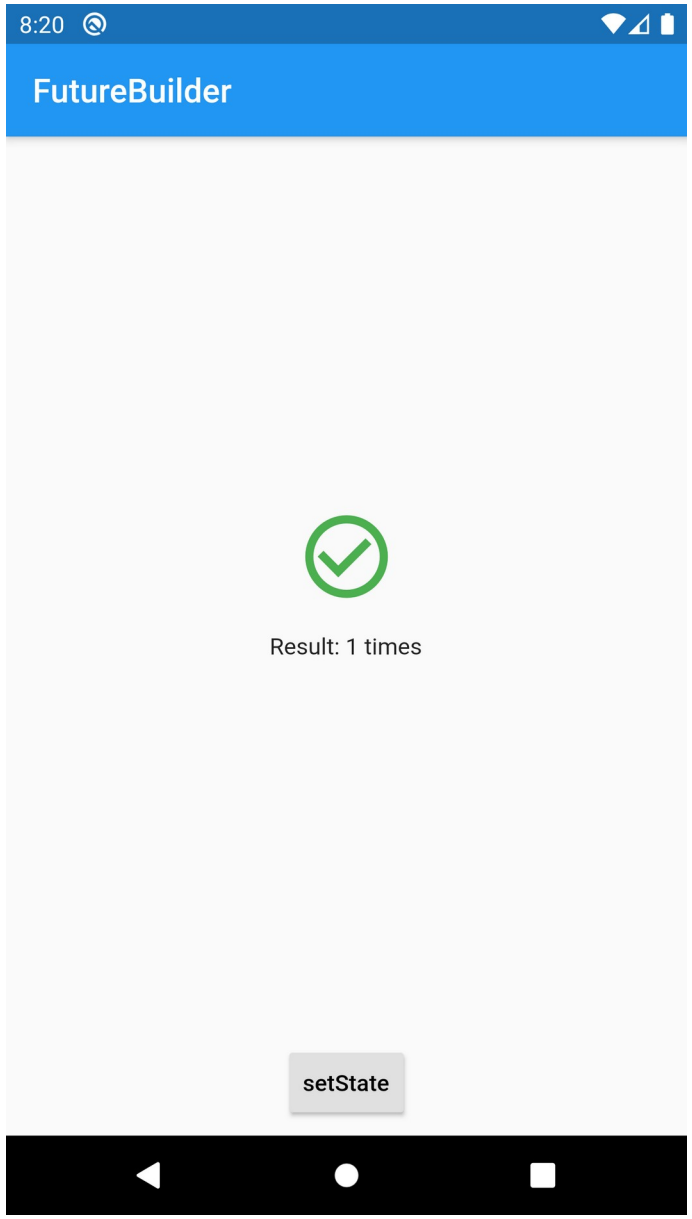
# FutureBuilder

## Altere o FutureBuilder:

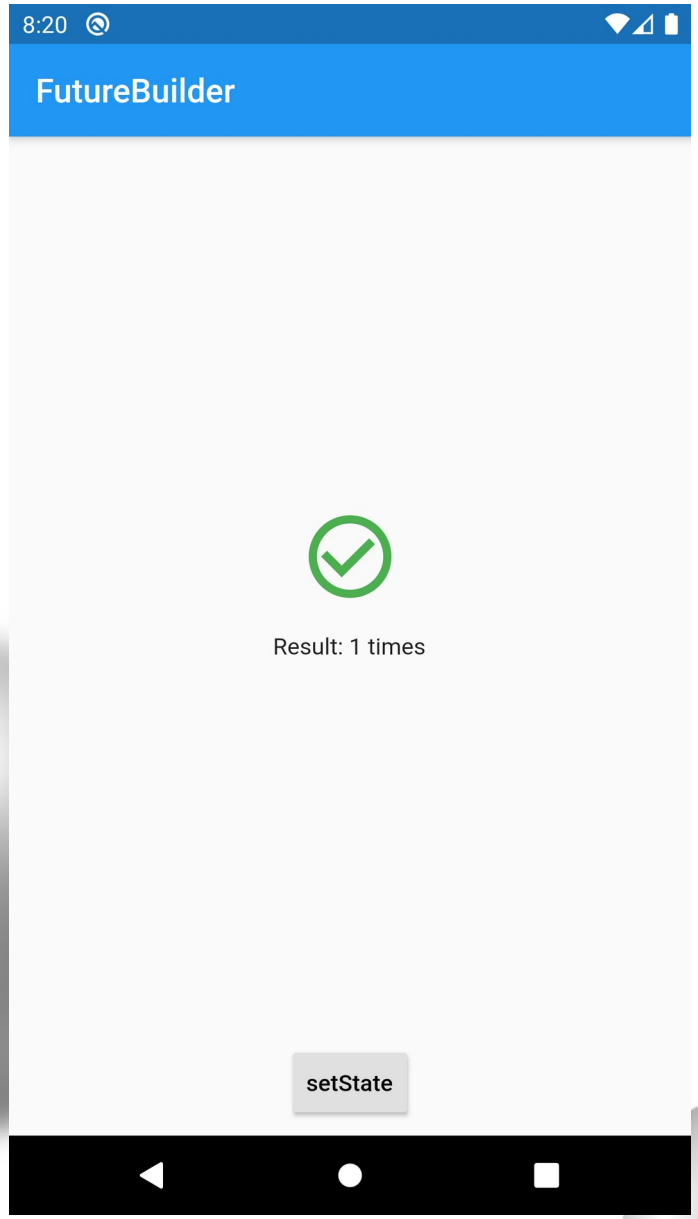
```
          child: FutureBuilder<String>(
            future: _calculation,
            builder: (BuildContext context,
AsyncSnapshot<String> snapshot) {
```

# FutureBuilder



# FutureBuilder




```
http://127.0.0.1:37719/iMRlKaU5R4g=/
D/EGL_emulation( 5764): eglMakeCurrent: 0x7a7
D/EGL_emulation( 5764): eglMakeCurrent: 0x7a7
I/flutter ( 5764): Recriando Home
I/Choreographer( 5764): Skipped 137 frames!
 its main thread.
I/flutter ( 5764): Recriando FutureBuilder
I/flutter ( 5764): Calculando
I/flutter ( 5764): Recriando FutureBuilder
I/flutter ( 5764): Recriando Home
I/flutter ( 5764): Recriando FutureBuilder
I/flutter ( 5764): Recriando Home
I/flutter ( 5764): Recriando FutureBuilder
```

# StreamBuilder

**StreamBuilder** permite criar widgets segundo os eventos de um **Stream**. O **builder** do **StreamBuilder** irá receber como parâmetro o contexto e um objeto **AsyncSnapshot** para obter o evento do **Stream**. O método **hasData** retorna **true** quando o evento é um valor. O método **hasError** retorna **true** quando o evento é um erro.

Crie uma nova aplicação Flutter com o nome **builder_stream**.

# StreamBuilder

## Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: Home(),
    );
  }
}
```

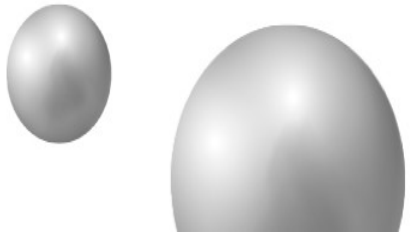
# StreamBuilder

## Inclua a classe com a tela da aplicação:

```
class Home extends StatelessWidget {
  int _counter = 0;
  StreamController<int> _events = StreamController<int>();

  @override
  Widget build(BuildContext context) {
    print("Criando Home");
    return Scaffold(
      appBar: AppBar(
        title: Text("StreamBuilder"),
      ),
```

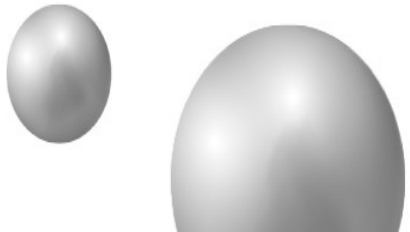
# StreamBuilder

```
body: Center(
  child: StreamBuilder<int>(
    stream: _events.stream,
    builder: (context, snapshot) {
      print("Recriando StreamBuilder");
      if (snapshot.hasError) {
        return Text(snapshot.error);
      } else if (!snapshot.hasData) {
        return Text("No data");
      } else {
        return Text("${snapshot.data.toString()}",
          style: Theme.of(context).textTheme.display1,
        );
      }
    },
  ),
),
```



# StreamBuilder

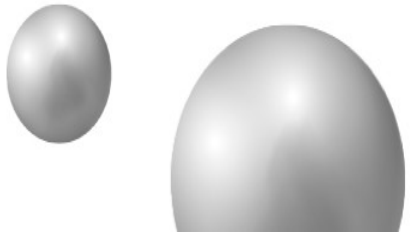
```
      bottomNavigationBar: ButtonBar(
        alignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          RaisedButton(
            child: Icon(Icons.add),
            onPressed: _incrementCounter,
          ),
          RaisedButton(
            child: Icon(Icons.remove),
            onPressed: _decrementCounter,
          ),
        ],
      ),
    );
  }
}
```



# StreamBuilder

Inclua o método para incrementar o contador na classe **Home**:

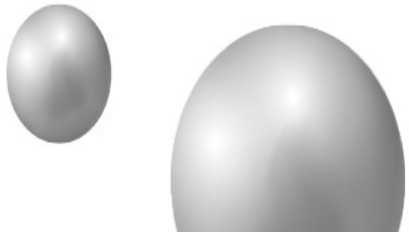
```
void _incrementCounter() {
  print("Incrementando");
  if ( _counter < 5 ) {
    _counter++;
    _events.add(_counter);
  } else {
    _events.addError("Valor maximo");
  }
}
```



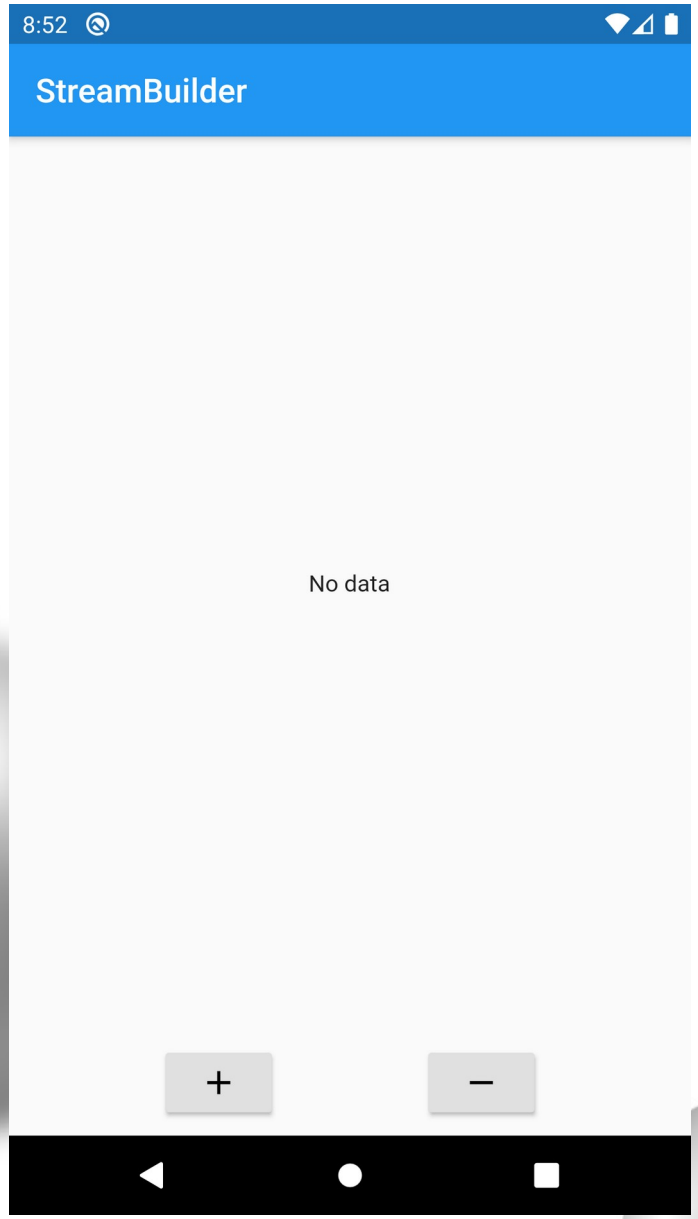
# StreamBuilder

Inclua o método para decrementar o contador na classe **Home**:

```
void _decrementCounter() {
  print("Decrementando");
  if ( _counter > 0 ) {
    _counter--;
    _events.add(_counter);
  } else {
    _events.addError("Valor minimo");
  }
}
```

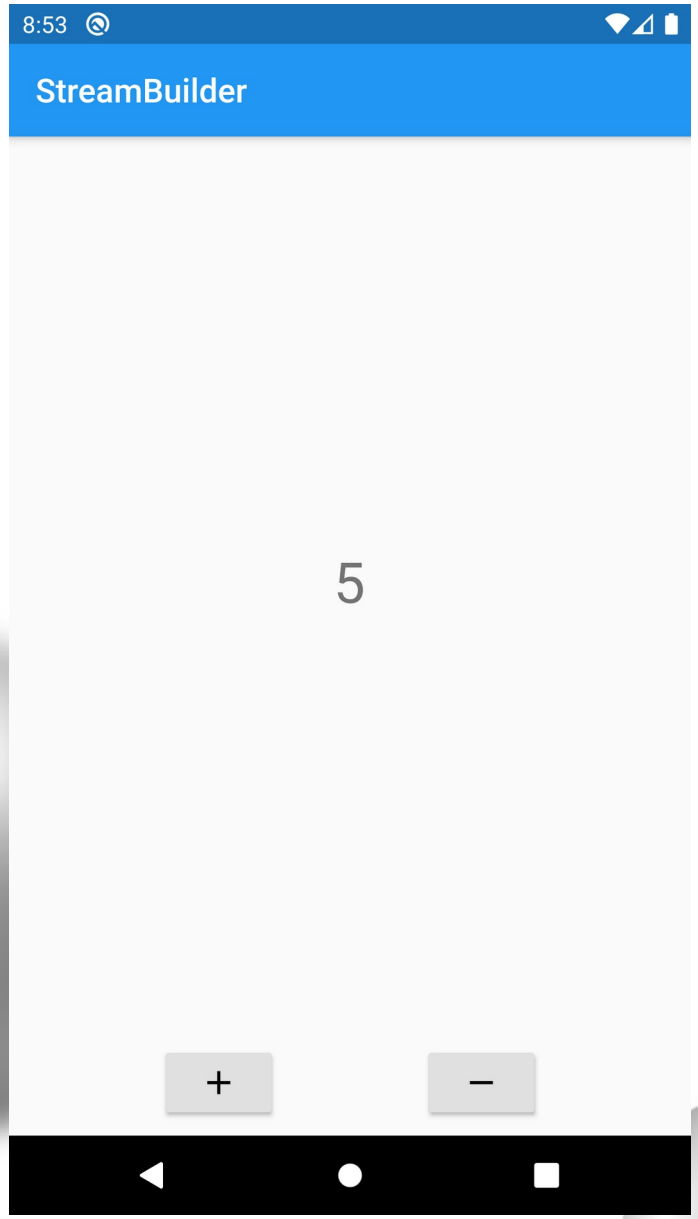


# StreamBuilder



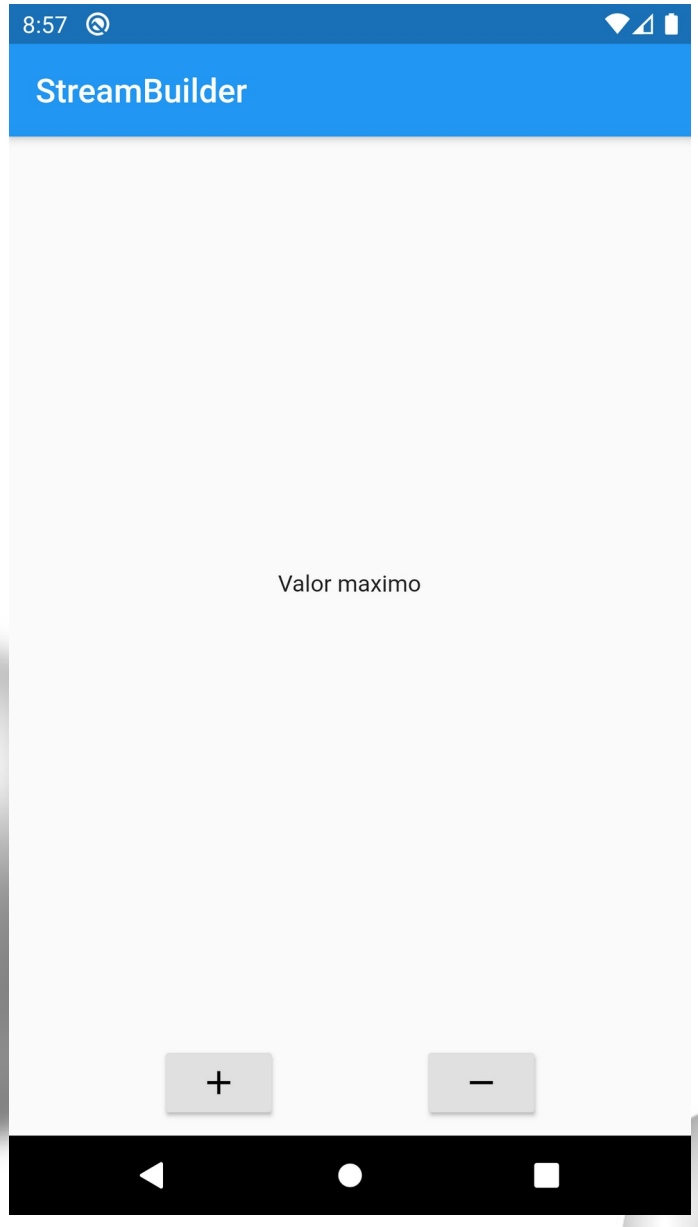

```
3232385534, SwapBuffers=165888470319, FrameCo
8741000, QueueBufferDuration=667000,
I/Choreographer( 6136): Skipped 1268 frames!
n its main thread.
I/flutter ( 6136): Criando Home
I/flutter ( 6136): Recriando StreamBuilder
```

# StreamBuilder




```
3232385534, SwapBuffers=165888470319, FrameCo
8741000, QueueBufferDuration=667000,
I/Choreographer( 6136): Skipped 1268 frames!
n its main thread.
I/flutter ( 6136): Criando Home
I/flutter ( 6136): Recriando StreamBuilder

I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
```

# StreamBuilder




```
I/Choreographer( 6136): Skipped 1268 frames!
n its main thread.
I/flutter ( 6136): Criando Home
I/flutter ( 6136): Recriando StreamBuilder

I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
I/flutter ( 6136): Incrementando
I/flutter ( 6136): Recriando StreamBuilder
```