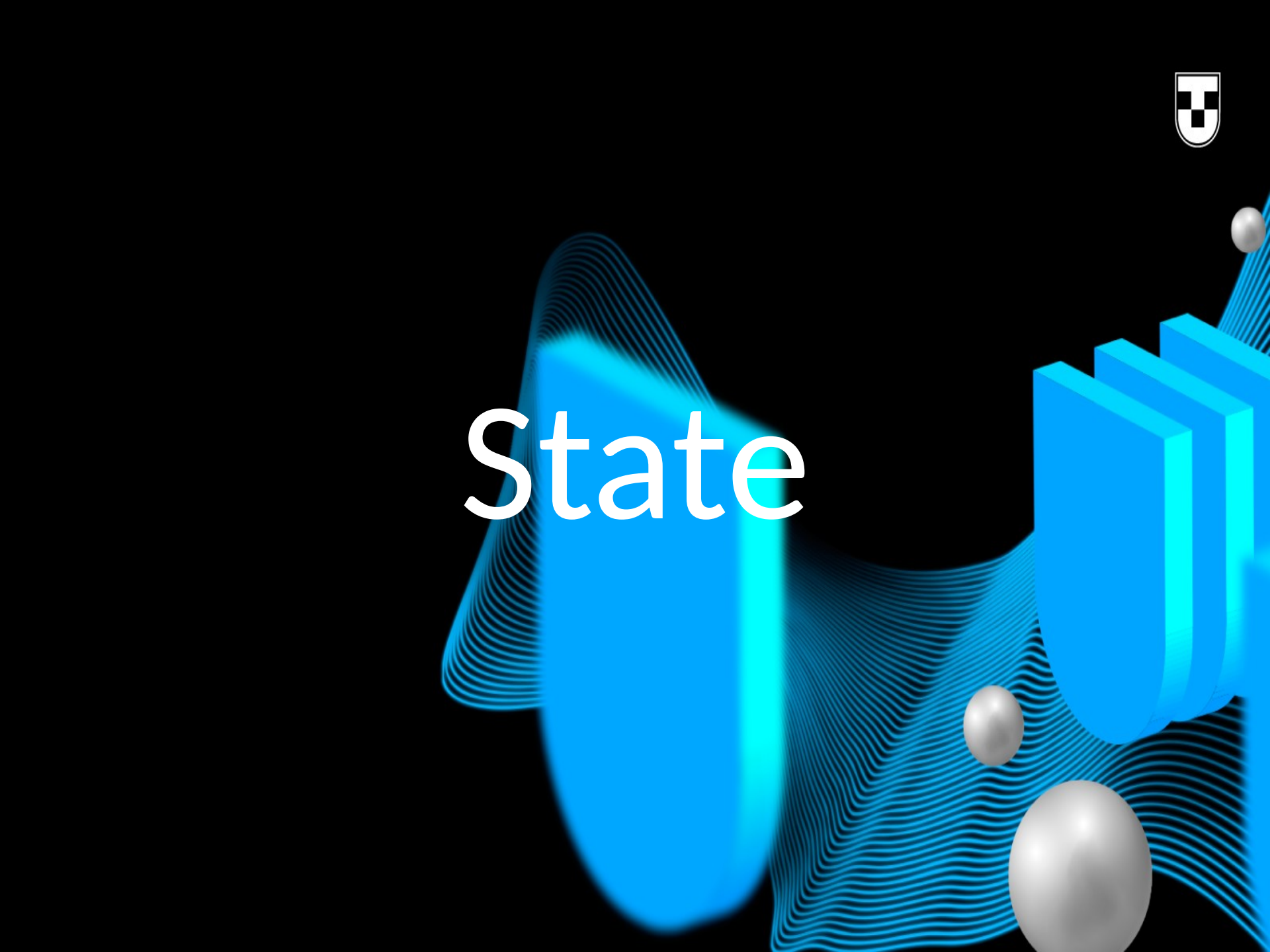

UNITAU
Universidade de Taubaté


State

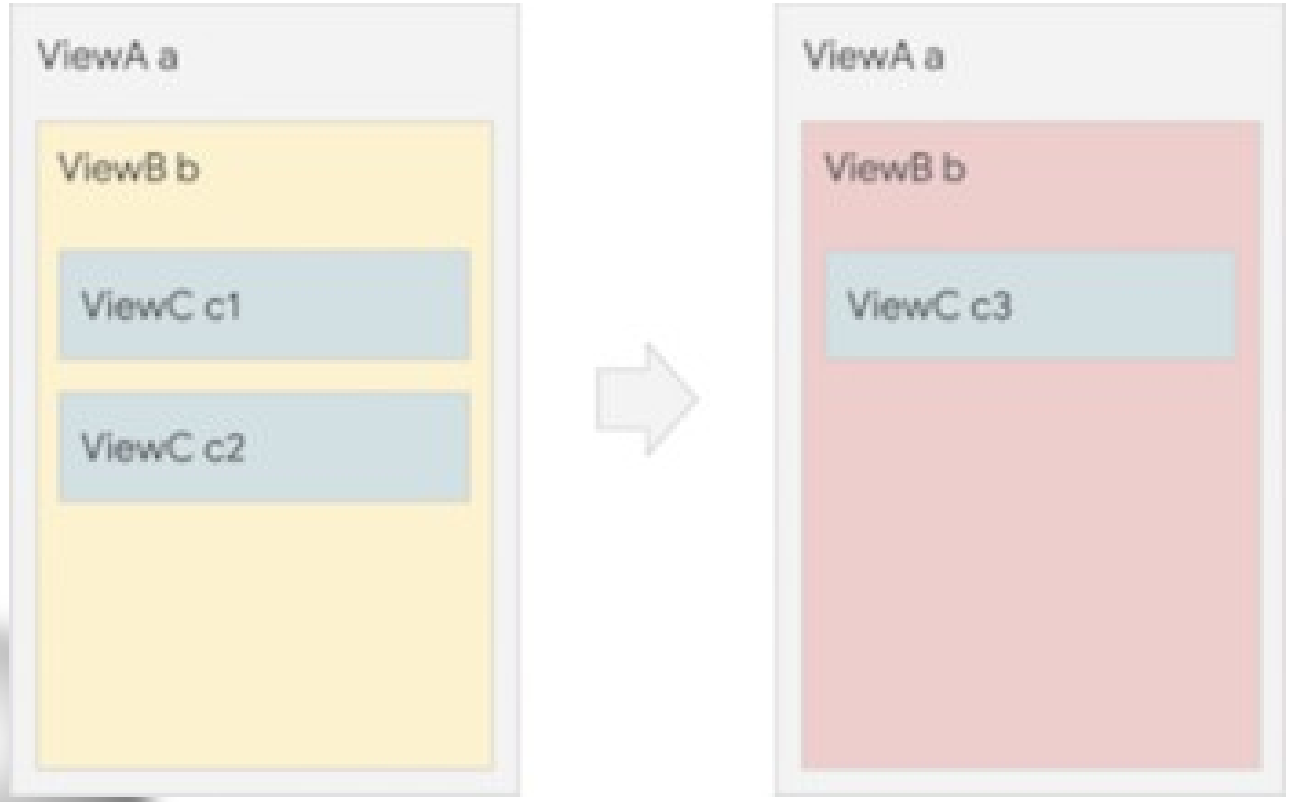
UI Declarativa

Diversas linguagens e frameworks usam um estilo imperativo para a UI da aplicação. A UI é manualmente construída e posteriormente alterada por métodos e atribuições.

Flutter é declarativo, isto é, constrói a UI para refletir o estado da aplicação e quando esse estado é alterado, a UI é refeita.

https://flutter.dev/docs/development/data-and-backend/state-mgmt/declarative

UI Declarativa




```
// Imperative style
b.setColor(red)
b.clearChildren()
ViewC c3 = new
ViewC(...)
b.add(c3)
```

```
// Declarative style
return ViewB(
  color: red,
  child: ViewC(...),
)
```

https://flutter.dev/docs/get-started/flutter-for/declarative

Gerenciamento de Estado

Como a UI reflete o estado da aplicação, é fundamental o gerenciamento desse estado, permitindo a alteração do estado e propagação dessas alterações para os elementos da UI.

Existem alguns métodos para o gerenciamento de estados no Flutter. Alguns exemplos são:

- StatefulWidget
- InheritedWidget / InheritedModel
- Provider
- BloC / RxDart
- MobX

StatefulWidget

StatefulWidget é a forma básica para armazenar estados. É possível transmitir as informações do estado para widgets nos níveis abaixo na árvore de widgets. Porém é necessário repassar esses valores por cada widget na linha até o widget que utilizará esses valores.

StatefulWidget

Crie uma nova aplicação Flutter com o nome state_stateful e altere o arquivo main.dart:

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

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

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

StatefulWidget

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no diretório **lib/pages**:

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

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}
```

StatefulWidget

Acrescente a classe para gerenciar o estado da tela no arquivo home.dart:

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

  @override
  Widget build(BuildContext context) {
    print("Recriando Home");
    return Scaffold(
      appBar: AppBar(
        title: Text("Stateful Widget"),
      ),
      body: WidgetA(_counter),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: Icon(Icons.add),
      ),
    );
  }
}
```

StatefulWidget

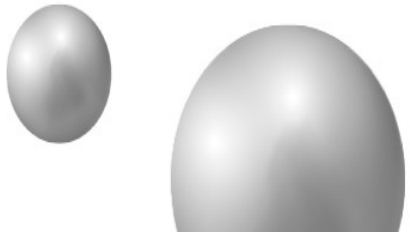
Acrescente o método para incrementar o contado na classe _HomeState:

```
void _incrementCounter() {
  setState(() {
    _counter++;
  });
}
```

StatefulWidget

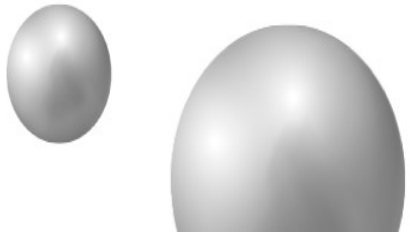
Crie o subdiretório **lib/pages**. Crie o arquivo **widgetA.dart** no diretório **lib/utils**:

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

class WidgetA extends StatelessWidget {
  final int _counter;

  WidgetA(this._counter);
```



StatefulWidget

```
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetA");
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            'You have pushed the button this many times:',
          ),
          WidgetB(_counter),
        ],
      ),
    );
  }
}
```

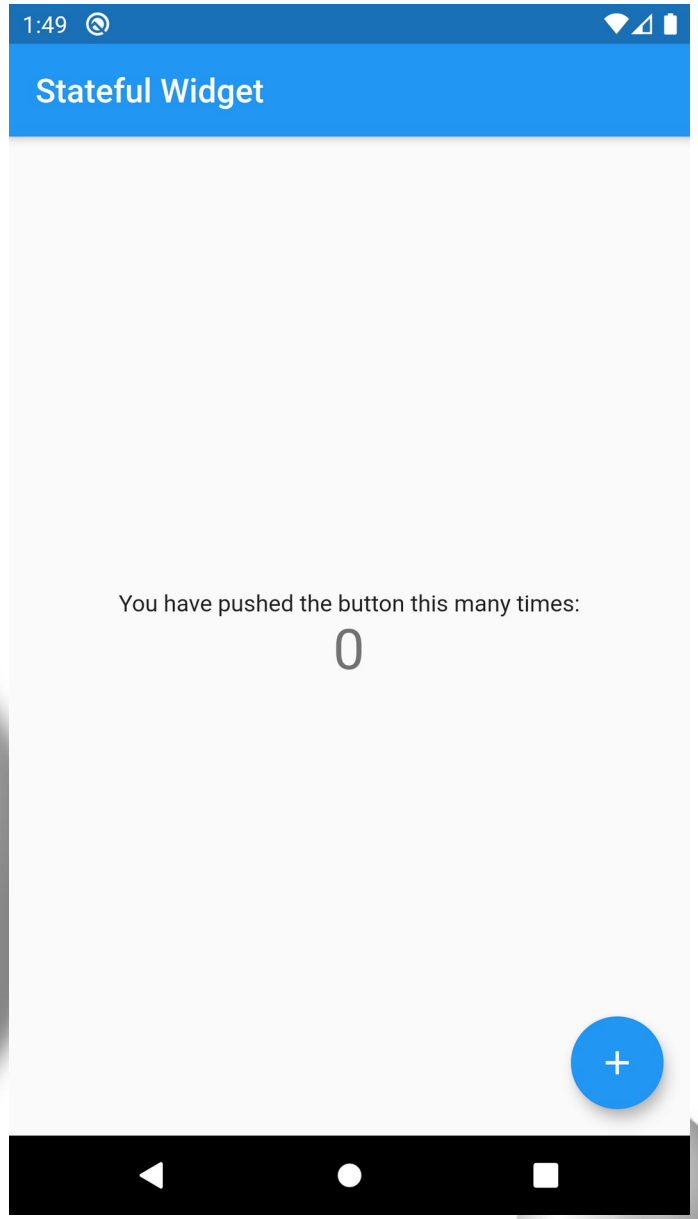


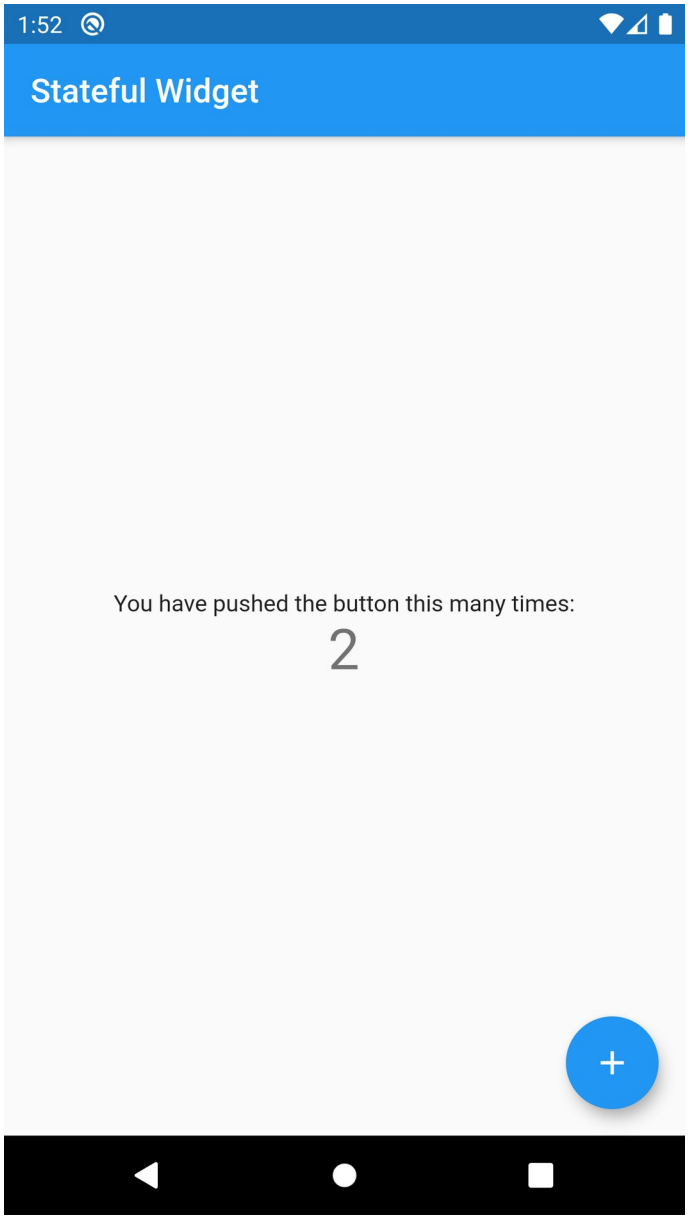
StatefulWidget

Crie o arquivo widgetB.dart no diretório lib/utils:

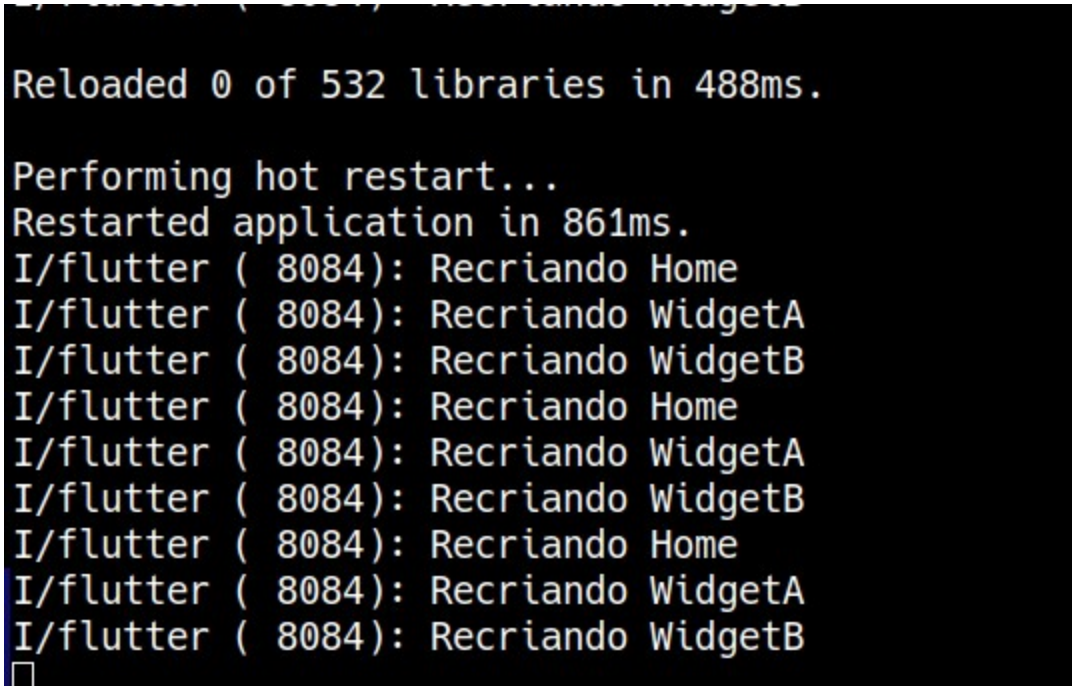
```
import 'package:flutter/material.dart';

class WidgetB extends StatelessWidget {
  final int _counter;

  WidgetB(this._counter);

  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetB");
    return Center(
      child: Text(
        '$_counter',
        style: Theme.of(context).textTheme.display1,
      ),
    );
  }
}
```

StatefulWidget



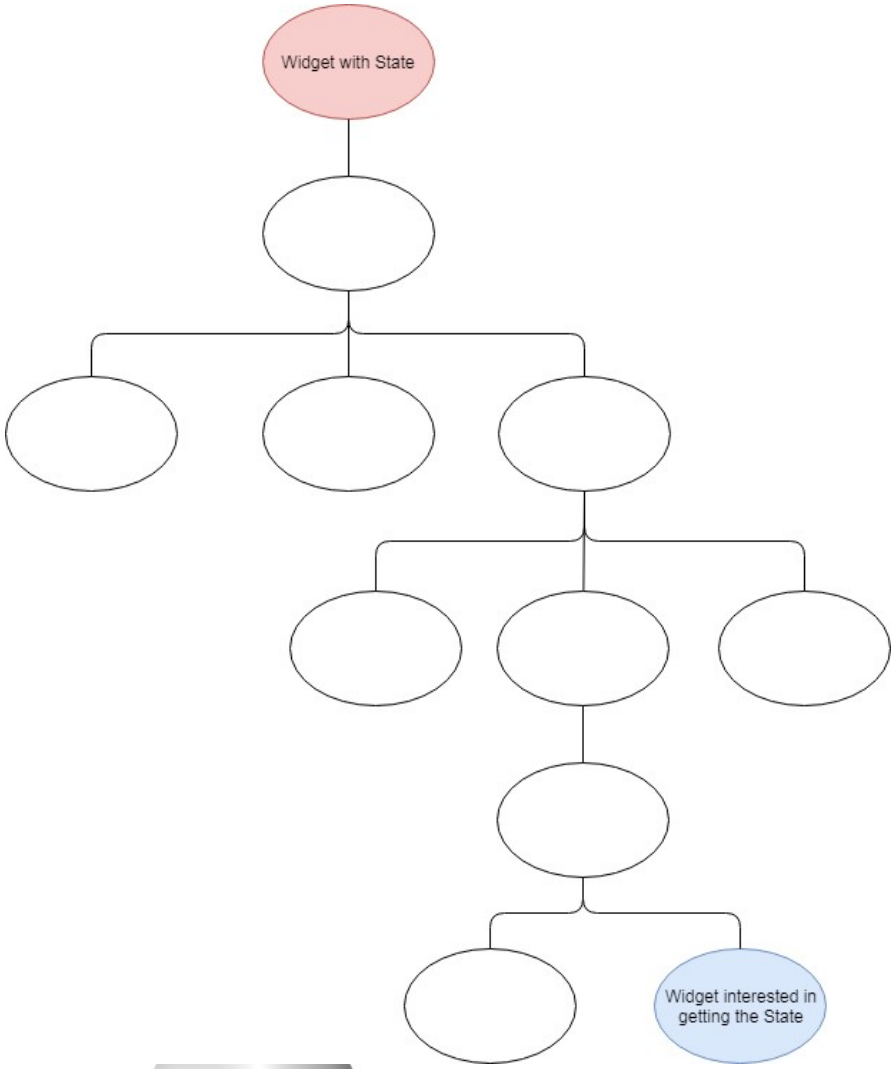


StatefulWidget



StatefulWidget

Isso se tornaria mais complexo em uma árvore com muitos níveis.



https://www.didierboelens.com/2018/06/widget-state-context-inheritedwidget/

StatefulWidget

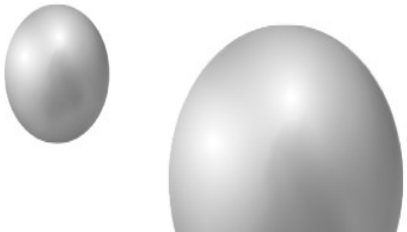
Os métodos alternativos ao StatefulWidget facilitam o acesso e a recriação dos widgets que dependem desse estado, sem a necessidade de transmitir o estado por toda a árvore.

InheritedWidget

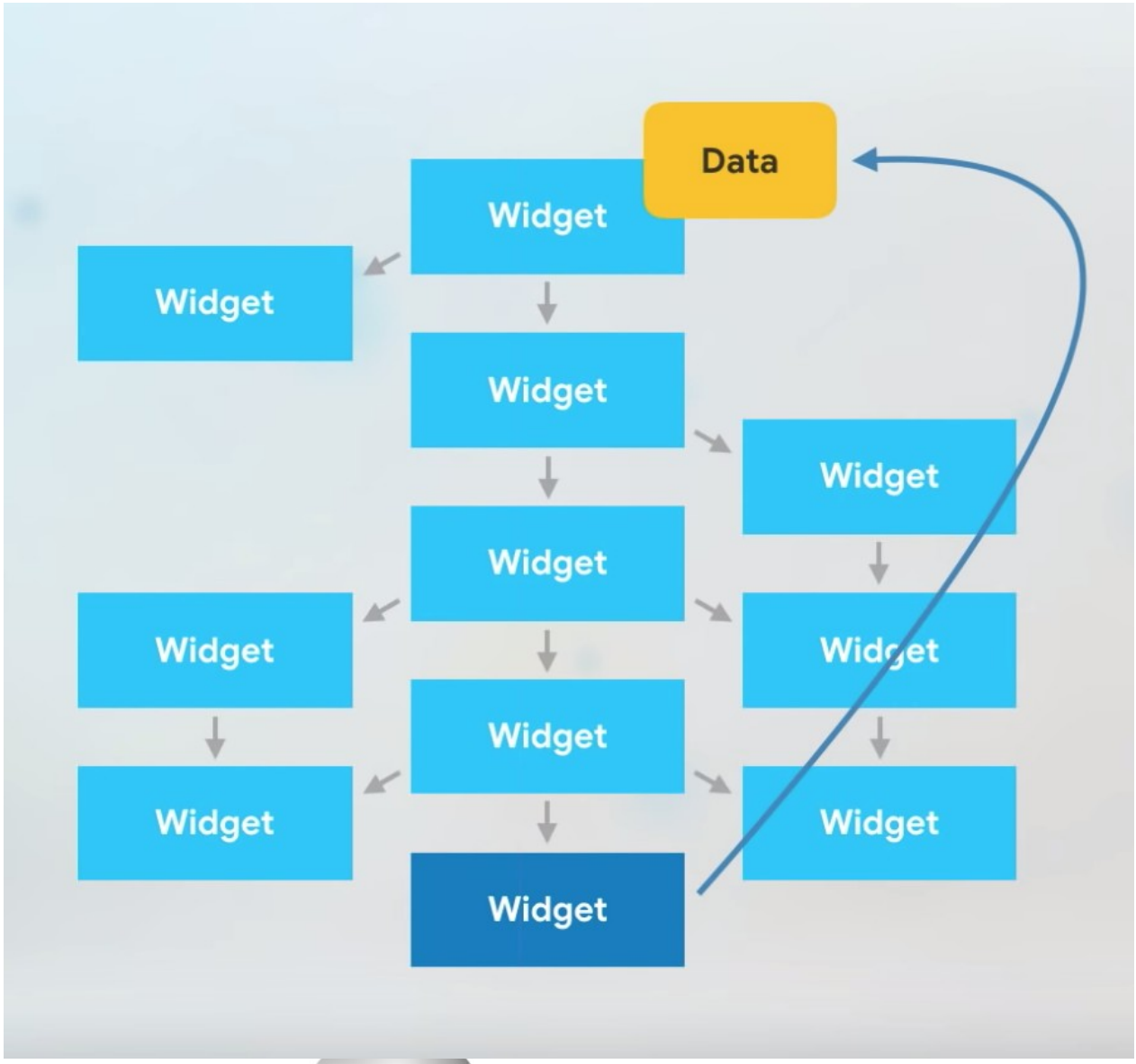
InheritedWidget é um widget que tem por objetivo, armazenar um estado e propaga esse estado para widgets da arvore de widgets.

O **InheritedWidget** deve ser colocado no topo da subárvore para ser acessado por widgets embaixo dessa subárvore.

Quando o estado do **InheritedWidget** é alterado, os widgets da subárvore registrados para ter acesso ao **InheritedWidget** são recriados.

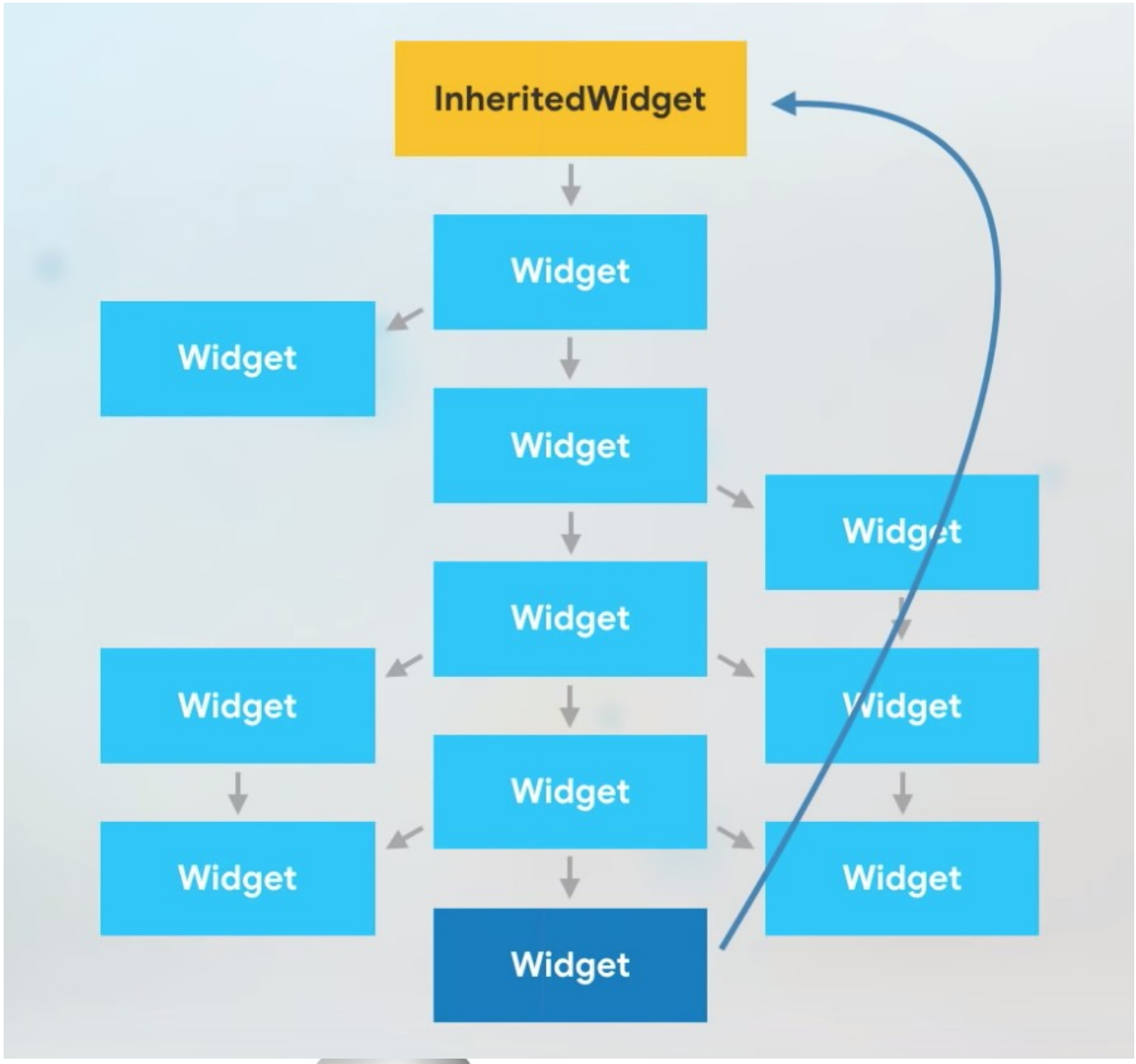


InheritedWidget



https://www.coderzheaven.com/2019/01/18/flutter-tutorials-inherited-widgets/

InheritedWidget



https://www.coderzheaven.com/2019/01/18/flutter-tutorials-inherited-widgets/

InheritedWidget

Crie uma nova aplicação Flutter com o nome **state_inheritedwidget** e altere o arquivo **main.dart**:

```
import 'package:flutter/material.dart';
import 'pages/home.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(),
    );
  }
}
```

InheritedWidget

Crie os subdiretórios **lib/models**. Crie o arquivo **item.dart** no diretório **lib/models**:

```
//
// Modelo para item
//
class Item {
  String reference;

  Item(this.reference);
}
```

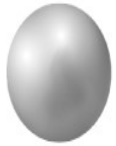
InheritedWidget

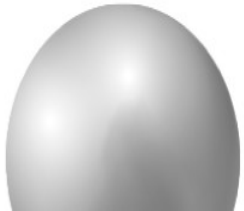
Crie os subdiretórios lib/pages. Crie o arquivo home.dart no diretório lib/pages:

```
import 'package:flutter/material.dart';
import '../utils/controlwidget.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando Home");
    // Cria a subarvoce a partir do InheritedWidget
    return ControlWidget(
      child: Scaffold(
        appBar: AppBar(
          title: Text('Inherited Widget'),
        ),
```

InheritedWidget

```
        body: Column(
          children: <Widget>[
            WidgetA(),
            Container(
              child: Row(
                children: <Widget>[
                  Icon(Icons.shopping_cart),
                  WidgetB(),
                ],
              ),
            ),
            WidgetC(),
          ],
        ),
      ),
    );
  }
}
```

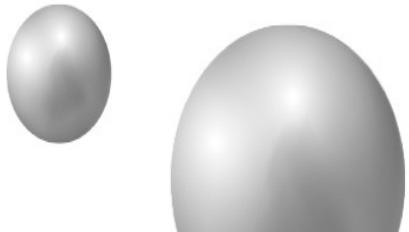




InheritedWidget

Crie os subdiretórios **lib/utils** e crie o arquivo **controlwidget.dart** no diretório **lib/utils**:

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

class ControlWidget extends StatefulWidget {

  ControlWidget({Key key,this.child,}):
    super(key: key);

  // Subarvore a ser criada sob o InheritedWidget
  final Widget child;

  @override
  ControlWidgetState createState() => ControlWidgetState();

}
```



InheritedWidget

Acrescente o método que dá acesso ao **InheritedWidget** para os demais widgets da subárvore na classe **ControlWidget**:

```
  // Permite acesso a _MyInheritedWidget
  static ControlWidgetState of(BuildContext context){
    return
(context.inheritFromWidgetOfExactType(_MyInheritedWidget)
      as _MyInheritedWidget
    ).data;
  }
```

InheritedWidget

Acrescente a classe para gerenciar o estado do widget no arquivo **controlwidget.dart**:

```
class ControlWidgetState extends State<ControlWidget>{
  List<Item> _items = <Item>[];
  int get itemsCount => _items.length;

  void addItem(String reference){
    setState((){
      _items.add(Item(reference));
    });
  }

  @override
  Widget build(BuildContext context){
    print("Recriando ControlWidget");
    return _MyInheritedWidget(
      data: this,
      child: widget.child,
    );
  }
}
```

InheritedWidget

Acrescente a classe do InheritedWidget no arquivo controlwidget.dart:

```
//
// InheritedWidget
//
class _MyInheritedWidget extends InheritedWidget {
  _MyInheritedWidget({
    Key key,
    @required Widget child,
    @required this.data,
  }): super(key: key, child: child);

  // Acesso ao State do MyInheritedWidget
  final ControlWidgetState data;

  // Determina se os widgets registrados devem ser recriados
  @override
  bool updateShouldNotify(_MyInheritedWidget oldWidget) {
    return true;
  }
}
```

InheritedWidget

Crie o arquivo widgetA.dart no diretório lib/utils:

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

class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetA");
    // Acesso ao estado do InheritedWidget
    // Registra o widget na lista a recriar
    // quando o InheritedWidget e alterado
    final ControlWidgetState state =
ControlWidget.of(context);

    return Container(
      child: RaisedButton(child: Text('Add Item'),
        onPressed: () => state.addItem('new item'),
      ),
    );
  }
}
```

InheritedWidget

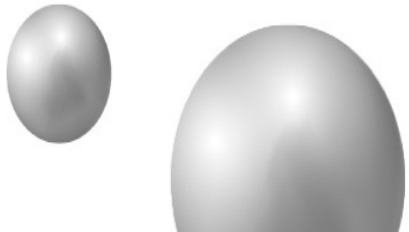
Crie o arquivo widgetB.dart no diretório lib/utils:

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

//
// Widget para exibir a quantidade de itens
//
class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetB");
    // Acesso ao estado do InheritedWidget
    // Registra o widget na lista a recriar
    // quando o InheritedWidget e alterado
    final ControlWidgetState state =
ControlWidget.of(context);

    return Text('${state.itemsCount}');
  }
}
```

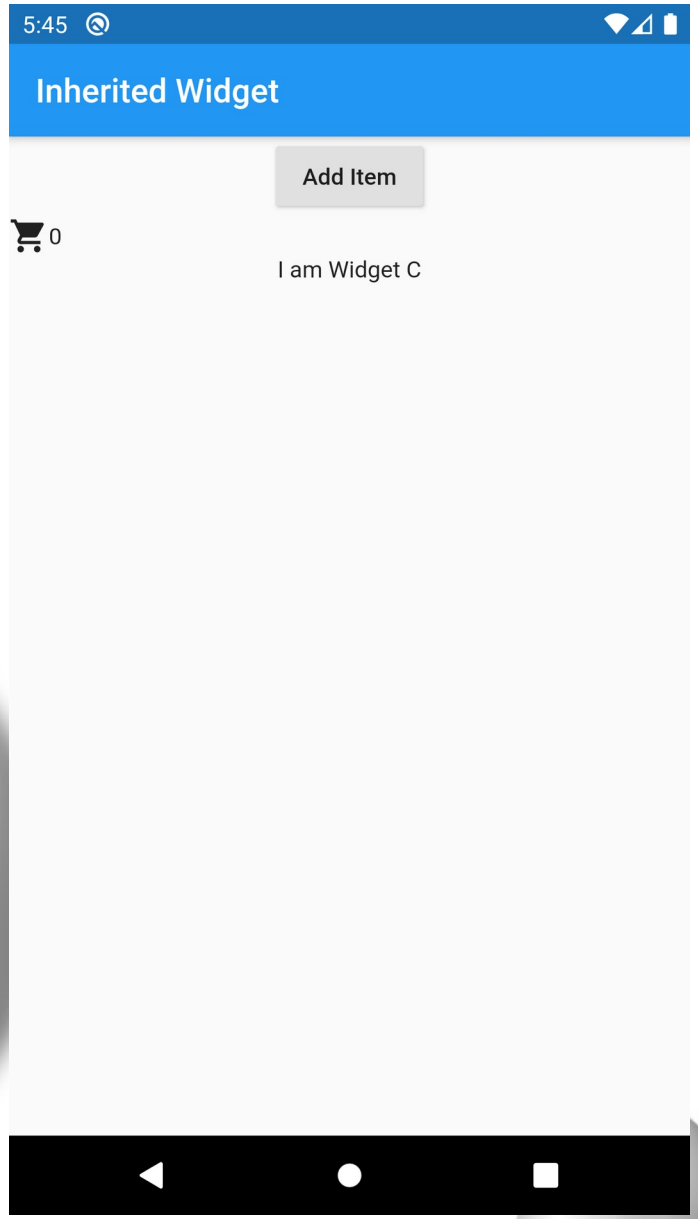
InheritedWidget

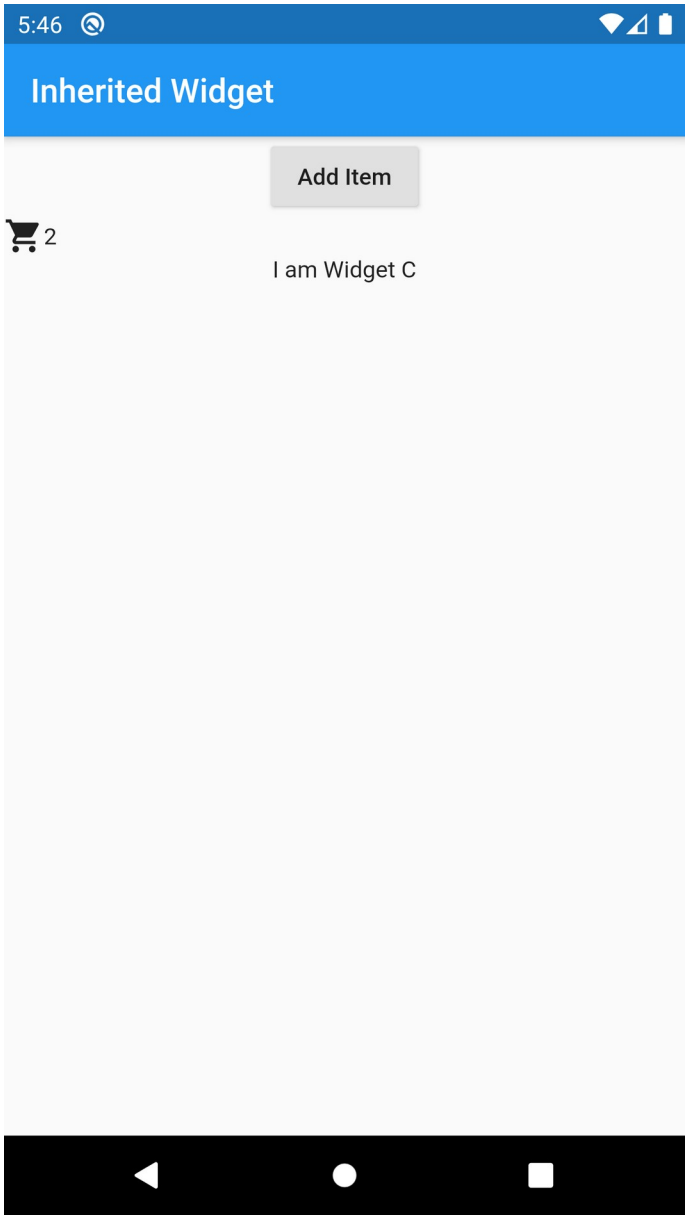
Crie o arquivo widgetC.dart no diretório lib/utils:

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

//
// Widget sem acesso ao InheritedWidget
//
class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetC");
    return Text('I am Widget C');
  }
}
```

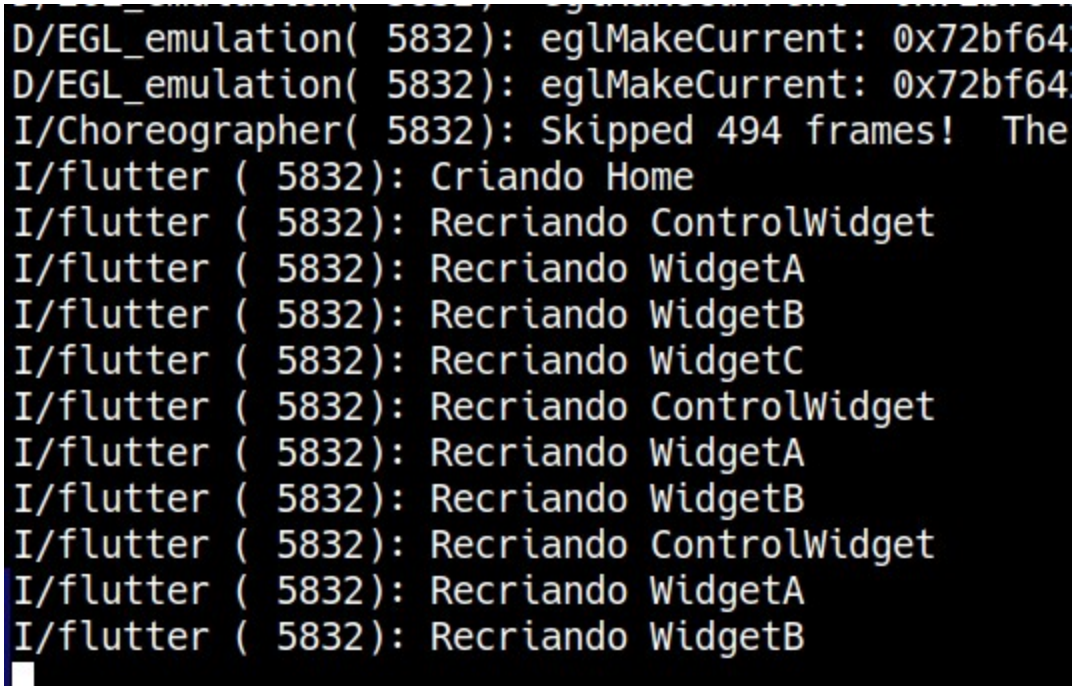


InheritedWidget





InheritedWidget



InheritedWidget

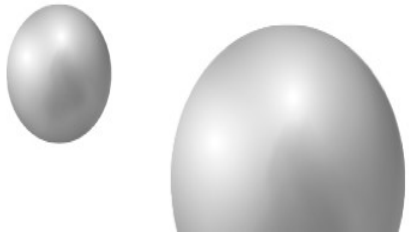
Os widgets WidgetA e WidgetB são recriados cada vez que se clica no botão para adicionar novo item. Porém, WidgetA precisa ter acesso ao estado de ControlWidget mas não precisa ser recriado quando esse estado é alterado. Altere o método que permite o acesso ao estado de _MyInheritedWidget para permitir acesso ao estado de ControleWidget, sem registrar o widget na lista de widgets a recriar quando esse estado for alterado se rebuild for falso:

```
  static ControlWidgetState of([BuildContext context, bool
rebuild = true]){
    return (rebuild ?

context.inheritFromWidgetOfExactType(_MyInheritedWidget)
        as _MyInheritedWidget
    :
      context.ancestorWidgetOfExactType(_MyInheritedWidget)
        as _MyInheritedWidget).data;
  }
```

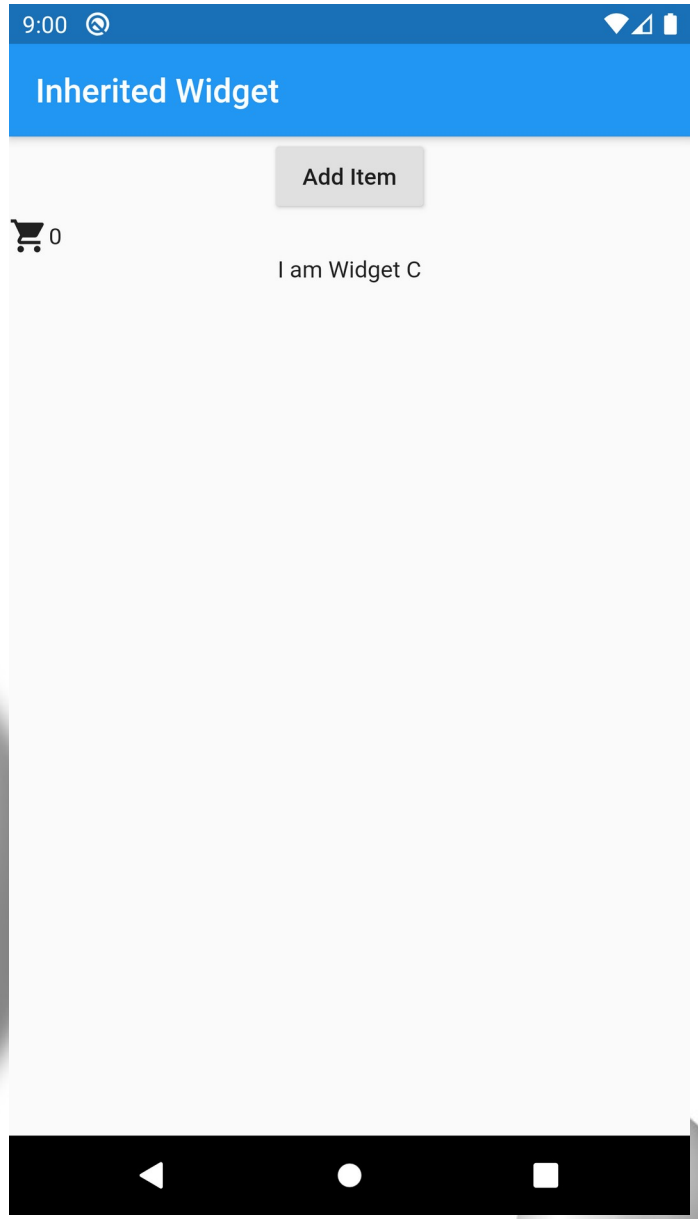
InheritedWidget

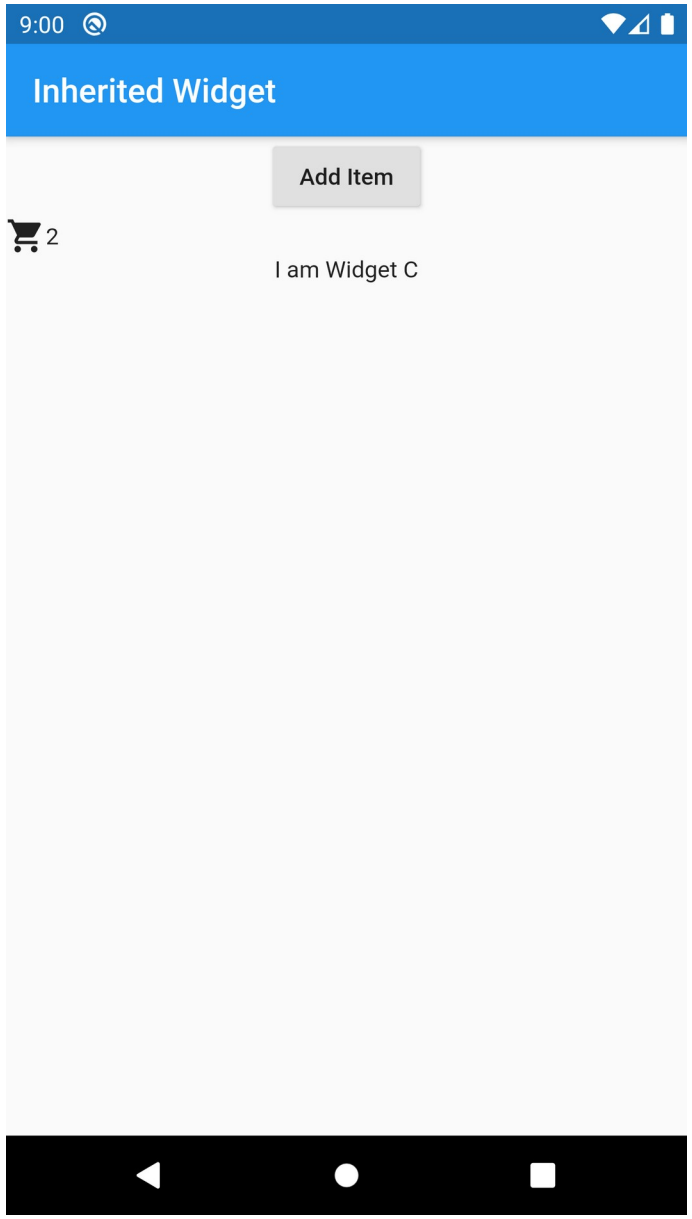
Altere o WidgetA para acessar o estado de ControlWidget sem registrar o widget na lista de widgets a recriar:

```
    // Acesso ao estado do ControlWidget
    final ControlWidgetState state =
ControlWidget.of(context, false);
```

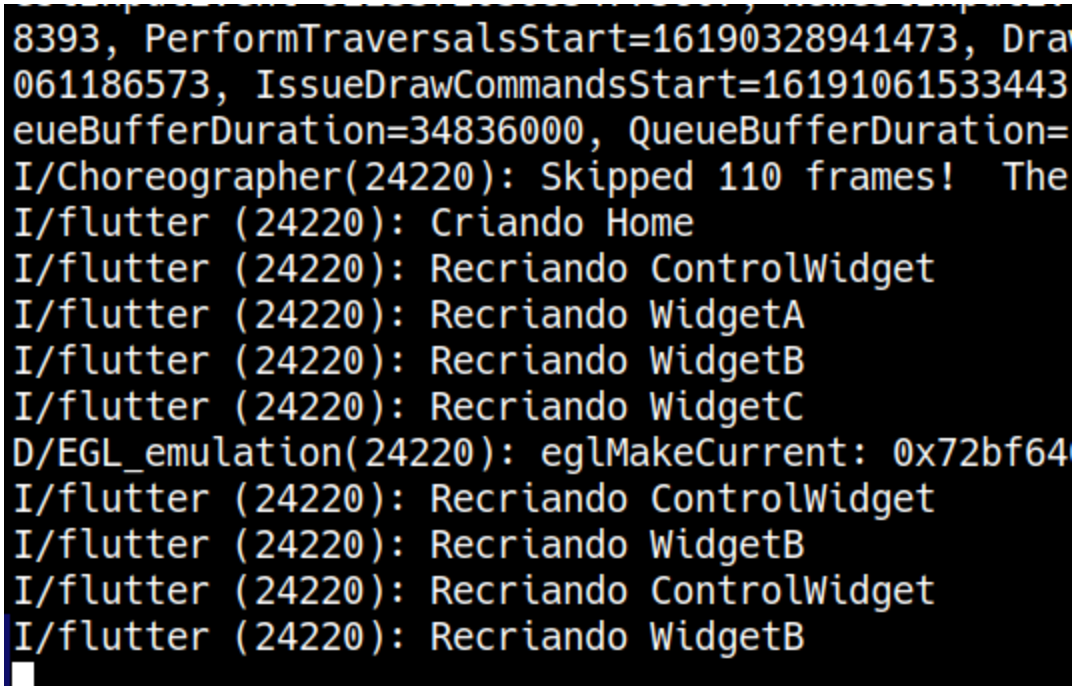


InheritedWidget



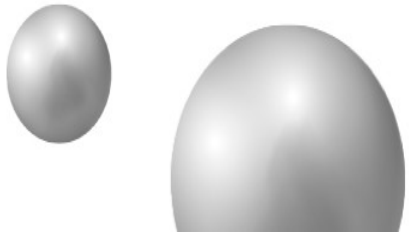


InheritedWidget



InheritedModel

Com **InheritedWidget** todos widgets dependentes são recriados toda vez que **updateShouldNotify** indica que é necessário recriar os widgets. Para permitir recriar apenas os widgets realmente necessários quando apenas uma parte do estado é alterado, **InheritedModel** permite utilizar o método **updateShouldNotifyDependent** para definir uma lista de aspectos para o estado e selecionar os widgets a recriar de acordo com o aspecto ao qual o widget foi associado.



InheritedModel

Crie uma nova aplicação Flutter com o nome **state_inheritedmodel** e Altere o arquivo **main.dart**:

```
import 'package:flutter/material.dart';
import 'pages/home.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(),
    );
  }
}
```

InheritedModel

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no diretório **lib/pages**:

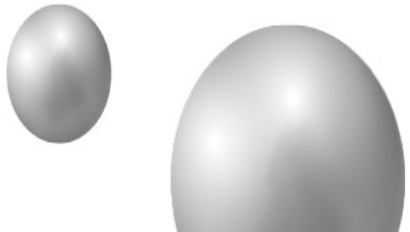
```
import 'package:flutter/material.dart';
import '../utils/controlwidget.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';
import '../utils/widgetD.dart';

final GlobalKey<ControlWidgetState> _key =
GlobalKey<ControlWidgetState>();

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando Home");
    return ControlWidget(
      key: _key,
      child: Scaffold(
        appBar: AppBar(title: Text('Inherited Model')),
```

InheritedModel

```
body: Column(
  children: <Widget>[
    WidgetA(),
    WidgetB(),
    WidgetC(),
    WidgetD(),
  ],
),
bottomNavigationBar: ButtonBar(
  alignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    RaisedButton(
      onPressed: () => _key.currentState.inc(),
      child: Text('Numero')
    ),
```

InheritedModel

```
            RaisedButton(
              onPressed: () => _key.currentState.newColor(),
              child: Text('Cor')
            ),
          ],
        ),
      ),
    );
  }
}
```



InheritedModel

Crie os subdiretórios **lib/utils** e Crie o arquivo **controlwidget.dart** no diretório **lib/utils**:

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

class ControlWidget extends StatefulWidget {

  ControlWidget({Key key, @required this.child}):
    super(key: key);

  // Subarvore a ser criada sob o InheritedWidget
  final Widget child;

  @override
  State<StatefulWidget> createState() =>
ControlWidgetState();
}
```

InheritedModel

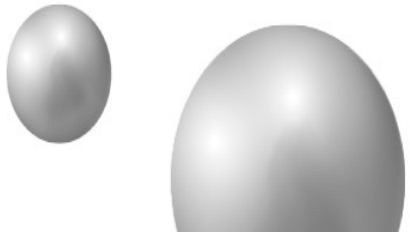
Acrescente a classe para gerenciar o estado do widget no arquivo **controlwidget.dart**:

```
class ControlWidgetState extends State<ControlWidget> {
  // Contador do estado
  int _count;
  // Cor do estado
  Color _color;

  @override
  Widget build(BuildContext context) {
    print("Recriando ControlWidget");
    return MyInheritedModel(
      count: _count,
      color: _color,
      child: widget.child,
    );
  }
}
```

InheritedModel

Acrescente o método para inicializar o estado na classe ControlWidgetState:

```
@override
void initState() {
  super.initState();
  _count = 0;
  _color = UniqueColorGenerator.getColor();
}
```



InheritedModel

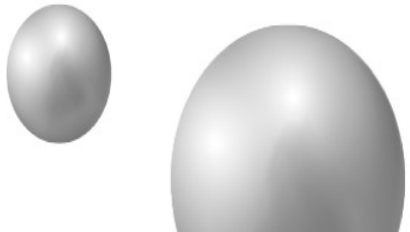
Acrescente o método para incrementar o contador na classe ControlWidgetState:

```
// Incrementar o contador
void inc(){
  setState((){
    _count = _count+1;
  });
}
```

InheritedModel

Acrescente o método para alterar a cor na classe **ControlWidgetState**:

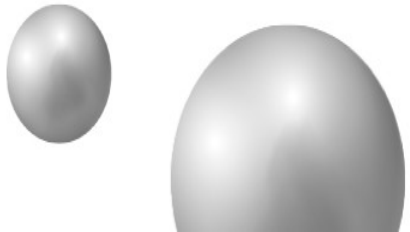
```
// Alterar a cor
void newColor(){
  setState((){
    _color = UniqueColorGenerator.getColor();
  });
}
```



InheritedModel

Acrescente a lista de aspectos no arquivo **controlwidget.dart**:

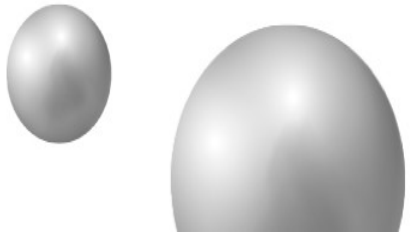
```
// Aspectos do InheritedModel
enum ASPECTS {
  NUMBER,
  COLOR,
}
```



InheritedModel

Acrescente a classe do **InheritedModel** no arquivo **controlwidget.dart**:

```
class MyInheritedModel extends InheritedModel<ASPECTS> {
  final int count;
  final Color color;

  MyInheritedModel({
    @required this.count,
    @required this.color,
    @required Widget child,
  }) : super(child: child);
}
```



InheritedModel

Acrescente o método para verificar se é necessários atualizar os widgets na classe MyInheritedModel:

```
@override
bool updateShouldNotify(MyInheritedModel old) {
  return count != old.count ||
    color != old.color;
}
```

InheritedModel

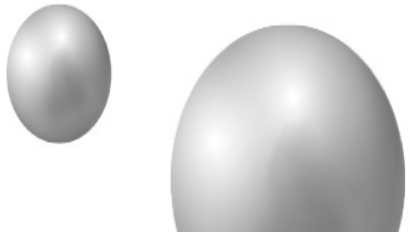
Acrescente o método para verificar quais widgets dependentes devem ser atualizados na classe MyInheritedModel:

```
  @override
  bool updateShouldNotifyDependent(MyInheritedModel old,
Set<ASPECTS> aspects) {
    return (aspects.contains(ASPECTS.NUMBER) && old.count !=
count) ||
      (aspects.contains(ASPECTS.COLOR) && old.color !=
color);
  }
```

InheritedModel

Acrescente o método para dar acesso ao InheritedModel na classe MyInheritedModel:

```
  static MyInheritedModel of(BuildContext context, {ASPECTS
aspect}) {
    return InheritedModel.inheritFrom<MyInheritedModel>(
      context,
      aspect: aspect
    );
  }
```



InheritedModel

Acrescente a classe para gerar uma cor aleatória no arquivo controlwidget.dart:

```
//
// Classe para gerar uma cor aleatoria
//
class UniqueColorGenerator {
  static Random random = Random();
  static Color getColor() {
    return Color.fromARGB(255, random.nextInt(255),
random.nextInt(255), random.nextInt(255));
  }
}
```

InheritedModel

Crie o arquivo widgetA.dart no diretório lib/utils:

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

//
// Widget para exibir contador
//
class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetA");
    // Acesso ao InheritedModel
    final MyInheritedModel model =
      MyInheritedModel.of(context, aspect: ASPECTS.NUMBER);
    return ListTile(
      title: Text('Contador ${model.count}'),
    );
  }
}
```

InheritedModel

Crie o arquivo widgetB.dart no diretório lib/utils:

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

//
// Widget para exibir a cor
//
class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetB");
    // Acesso ao InheritedModel
    final MyInheritedModel model =
      MyInheritedModel.of(context, aspect: ASPECTS.COLOR);
    return Container(
      color: model.color,
      child: ListTile(title: Text('Cor '),),
    );
  }
}
```

InheritedModel

Crie o arquivo widgetC.dart no diretório lib/utils:

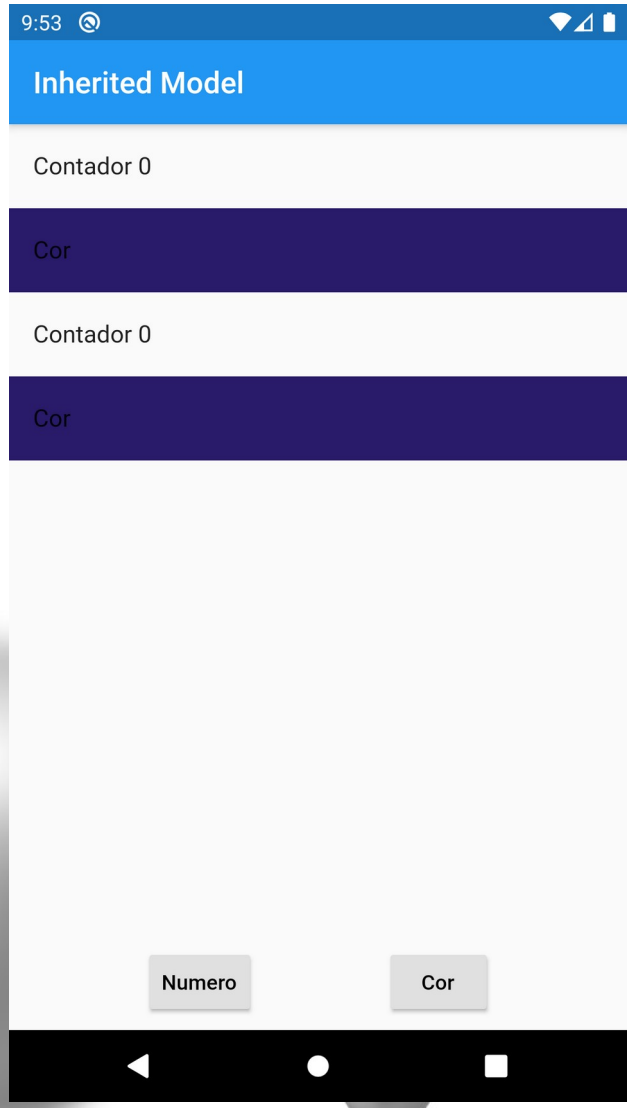
```
import 'package:flutter/material.dart';
import 'controlwidget.dart';

//
// Widget para exibir contador
//
class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetC");
    // Acesso ao InheritedModel
    final MyInheritedModel model =
      MyInheritedModel.of(context, aspect: ASPECTS.NUMBER);
    return ListTile(
      title: Text('Contador ${model.count}'),
    );
  }
}
```

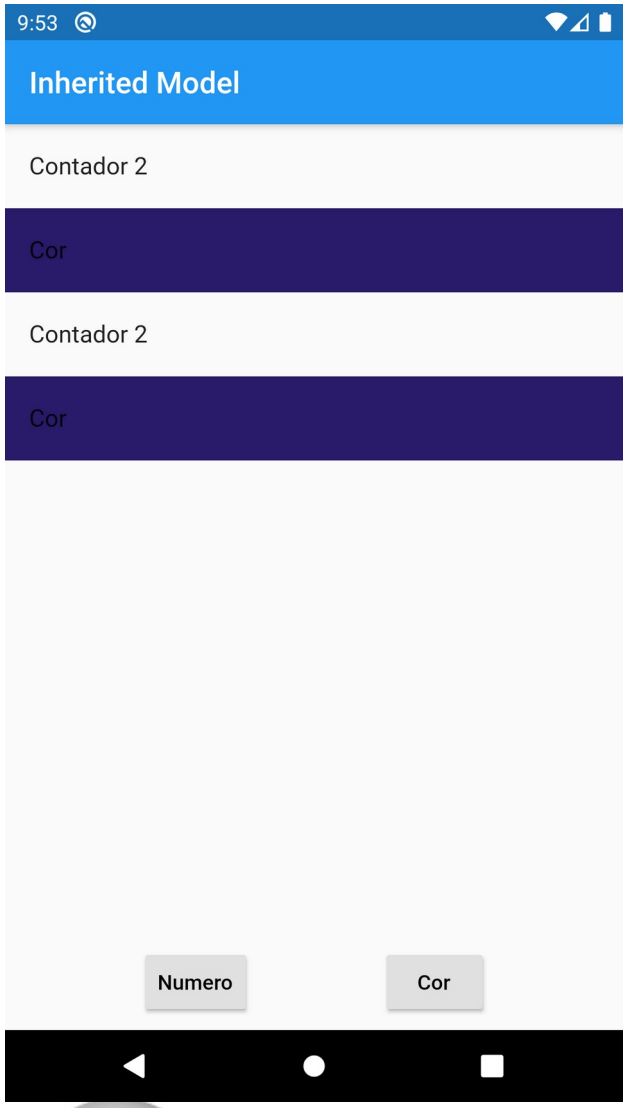
InheritedModel

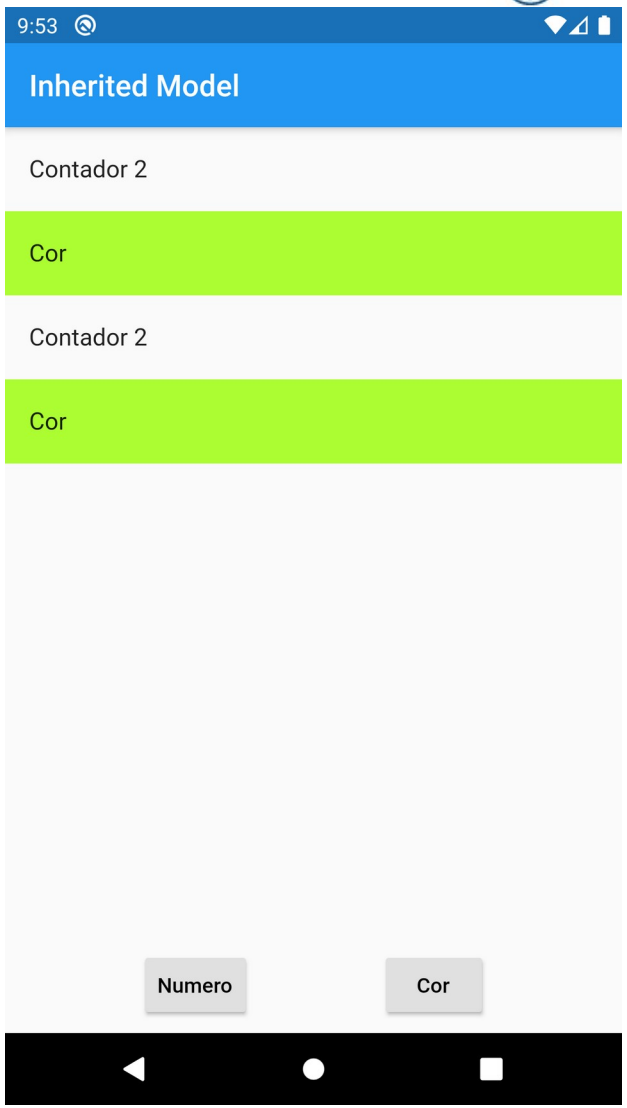
Crie o arquivo widgetD.dart no diretório lib/utils:

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

//
// Widget para exibir a cor
//
class WidgetD extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Recriando WidgetD" );
    // Acesso ao InheritedModel
    final MyInheritedModel model =
      MyInheritedModel.of(context, aspect: ASPECTS.COLOR);
    return Container(
      color: model.color,
      child: ListTile(title: Text('Cor '),),
    );
  }
}
```

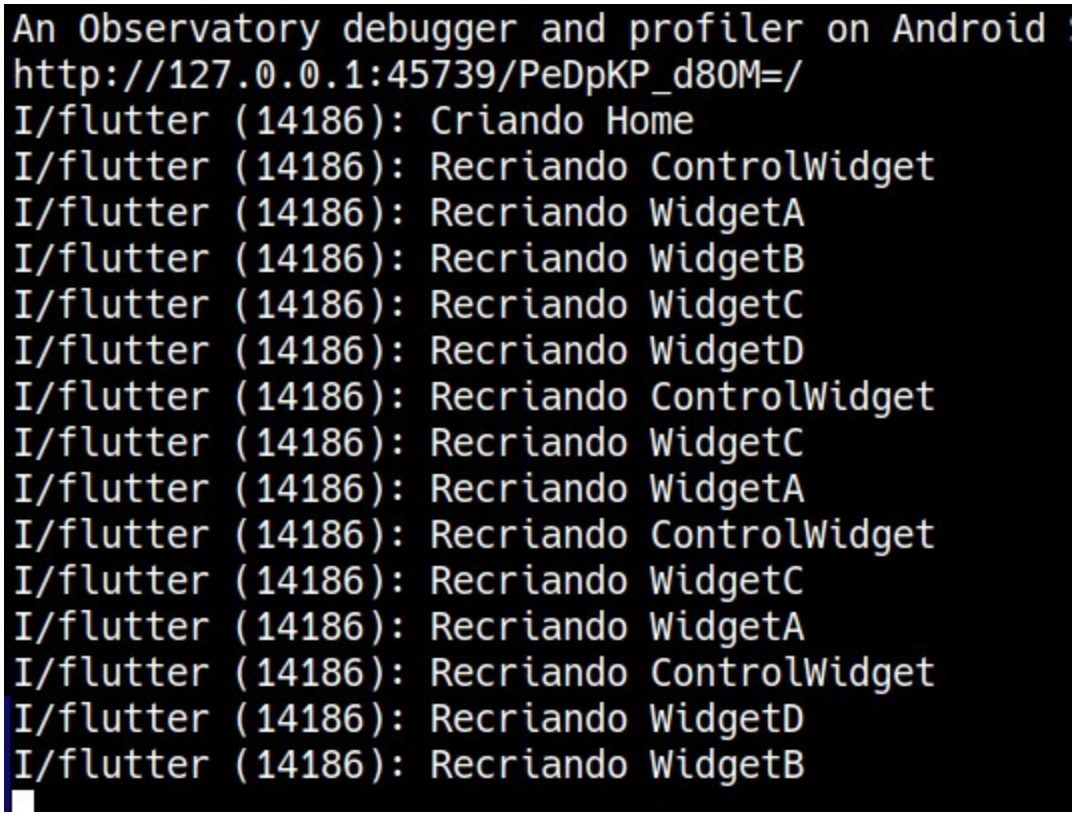
InheritedModel







InheritedModel



Provider

Provider fornece um mecanismo para expor objetos aos widgets da árvore do Provider.

Utilizando **Provider.of<T>(BuildContext context)**, widgets que fazem parte da árvore podem acessar o objeto exposto.

Dart possui vários providers:

- Provider
- ListenableProvider
- ChangeNotifierProvider
- ValueListenableProvider
- StreamProvider
- FutureProvider
- MultiProvider
- ProxyProvider

Provider

ChangeNotifierProvider pode ser utilizado para gerenciamento de estado pois utilizado junto com **ChangeNotifier** permite notificar e recriar os widgets que acessam o objeto utilizando **notifyListeners()**.

ListenableProvider e **ValueListenableProvider** também permitem a recriação dos widgets porém são tão usados.

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

Altere o arquivo **pubspec.yaml** para incluir a dependência de **provider**:

```
dependencies:
  provider: ^4.0.1
  flutter:
    sdk: flutter
```

Instale o pacote adicionado nas dependências.

Provider

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'pages/home.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(),
    );
  }
}
```

Provider

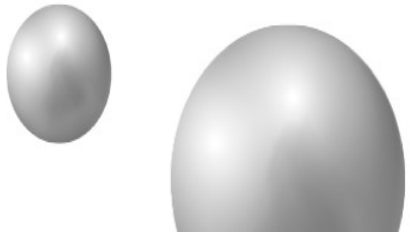
Crie o subdiretório **lib/models** e crie o arquivo **countmodel.dart** no diretório **lib/models**:

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

//
// Modelo para contador do estado
//
class CountModel with ChangeNotifier {
  int count = 0;

  void inc() {
    count = count + 1;
    notifyListeners();
  }
}
```

Provider

Crie o arquivo colormodel.dart no diretório lib/models:

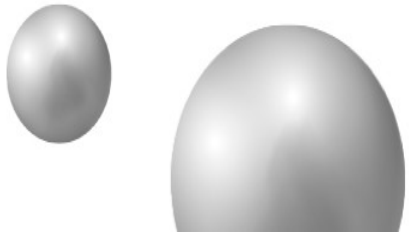
```
import 'package:flutter/material.dart';
import 'package:flutter/foundation.dart';
import 'dart:math';

//
// Modelo para cor do estado
//
class ColorModel with ChangeNotifier {
  Color color = UniqueColorGenerator.getColor();

  void newColor() {
    color = UniqueColorGenerator.getColor();
    notifyListeners();
  }
}
```



Provider

Acrescente a classe para gerar uma cor aleatória no arquivo colormodel.dart:

```
//
// Classe para gerar uma cor aleatoria
//
class UniqueColorGenerator {
  static Random random = Random();
  static Color getColor() {
    return Color.fromARGB(255, random.nextInt(255),
random.nextInt(255), random.nextInt(255));
  }
}
```



Provider

Crie o subdiretório **lib/pages** e crie o arquivo **home.dart** no diretório **lib/pages**:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/countmodel.dart';
import '../models/colormodel.dart';
import '../utils/countbutton.dart';
import '../utils/colorbutton.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';
import '../utils/widgetD.dart';

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando Home");
```

Provider

```
    return  MultiProvider(
      providers: [
        ChangeNotifierProvider<CountModel>(create: (context)
=> CountModel()),
        ChangeNotifierProvider<ColorModel>(create: (context)
=> ColorModel()),
      ],
      child: Scaffold(
        appBar: AppBar(title: Text('Provider')),
        body: Column(
          children: <Widget>[
            WidgetA(),
            WidgetB(),
            WidgetC(),
            WidgetD(),
          ],
        ),
```

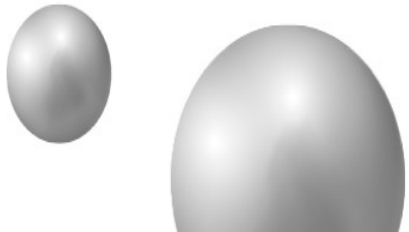
Provider

```
        bottomNavigationBar: ButtonBar(
          alignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            countButton(),
            colorButton(),
          ],
        ),
      ),
    );
  }
}
```

Provider

Crie o subdiretório **lib/utils** e crie o arquivo **countbutton.dart** no diretório **lib/utils**:

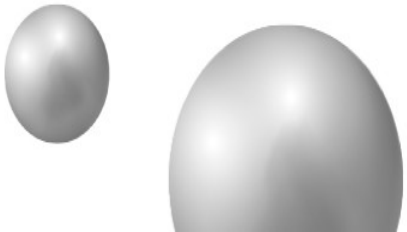
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/countmodel.dart';

class countButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando countButton");
    final model = Provider.of<CountModel>(context, listen:
false);
    return RaisedButton(
      child: Text('Numero'),
      onPressed: () => model.inc(),
    );
  }
}
```



Provider

Crie o arquivo colorbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/colormodel.dart';

class colorButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando colorButton");
    final model = Provider.of<ColorModel>(context);
    return RaisedButton(
      child: Text('Cor'),
      onPressed: () => model.newColor(),
    );
  }
}
```



Provider

Crie o arquivo widgetA.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/countmodel.dart';

// Widget para exibir contador
class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetA");
    return Consumer<CountModel>(
       builder: (context, model, child) {
         print("Recriando texto do WidgetA");
         return ListTile(
           title: Text('Contador ${model.count}'),);
       },
     );
  }
}
```

Provider

Crie o arquivo widgetB.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/colormodel.dart';

// Widget para exibir a cor
class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetB");
    return Consumer<ColorModel>(
      builder: (context, model, child) {
        print("Recriando ListTile do WidgetB");
        return Container(
          color: model.color,
          child: ListTile(title: Text('Cor '),),
        );
      },
    );
  }
}
```

Provider

Crie o arquivo widgetC.dart no diretório lib/utils:

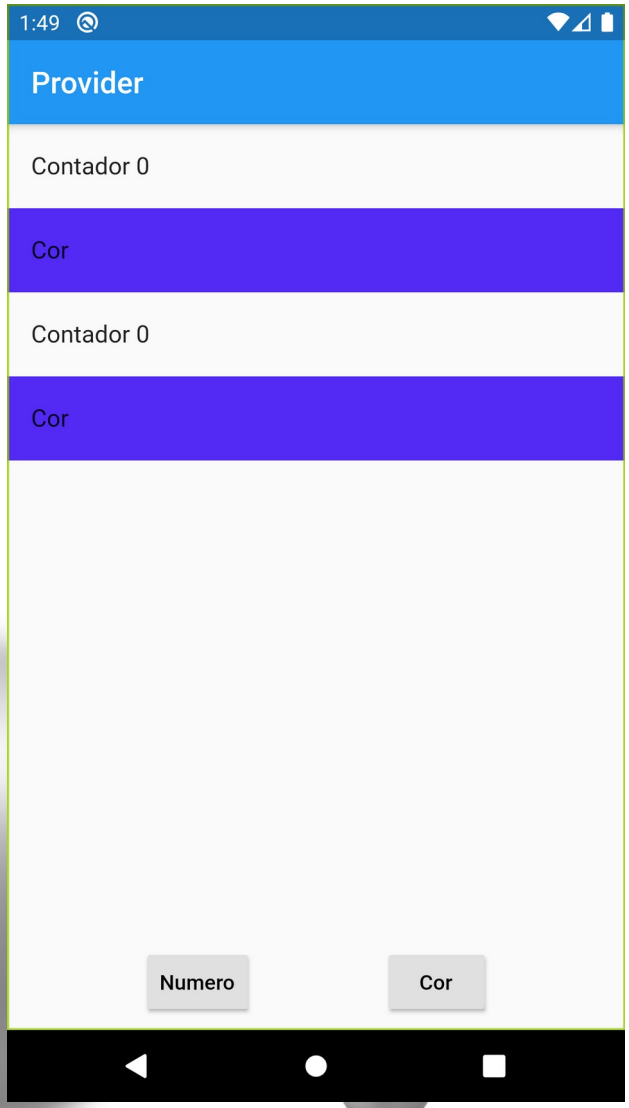
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/countmodel.dart';

// Widget para exibir contador
class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetC");
    return Consumer<CountModel>(
       builder: (context, model, child) {
         print("Recriando texto do WidgetC");
         return ListTile(
           title: Text('Contador ${model.count}'),);
       },
     );
  }
}
```

Provider

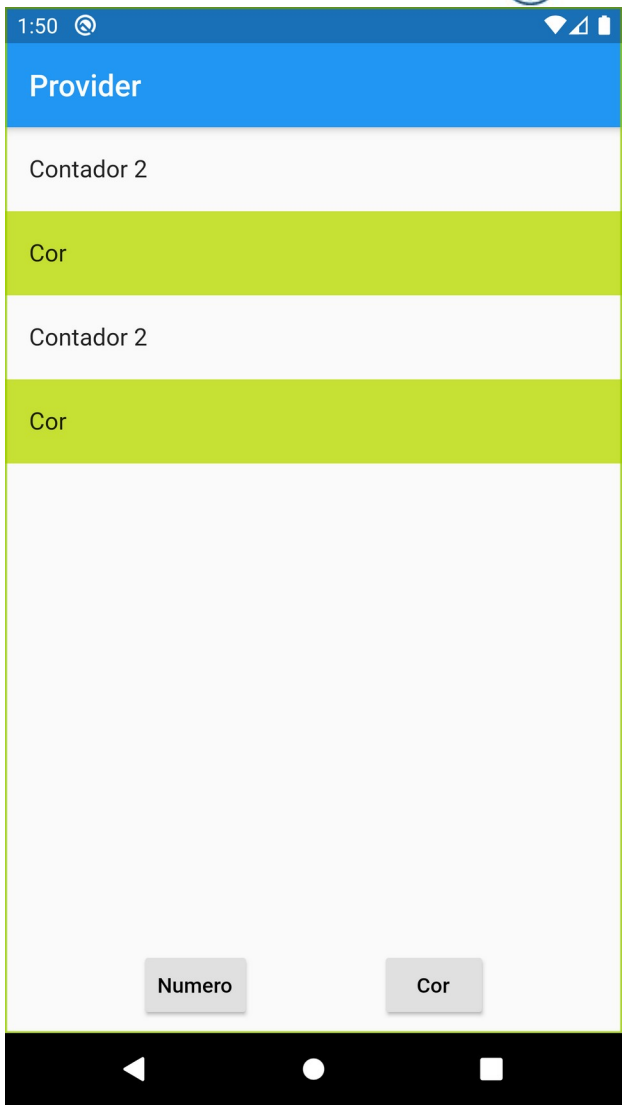
Crie o arquivo widgetD.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/colormodel.dart';

// Widget para exibir a cor
class WidgetD extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetD");
    return Consumer<ColorModel>(
      builder: (context, model, child) {
        print("Recriando ListTile do WidgetD");
        return Container(
          color: model.color,
          child: ListTile(title: Text('Cor '),),
        );
      },
    );
  }
}
```

Provider


1:49
Provider
Contador 0
Cor
Contador 0
Cor
Numero
Cor


1:50
Provider
Contador 2
Cor
Contador 2
Cor
Numero
Cor


1:50
Provider
Contador 2
Cor
Contador 2
Cor
Numero
Cor

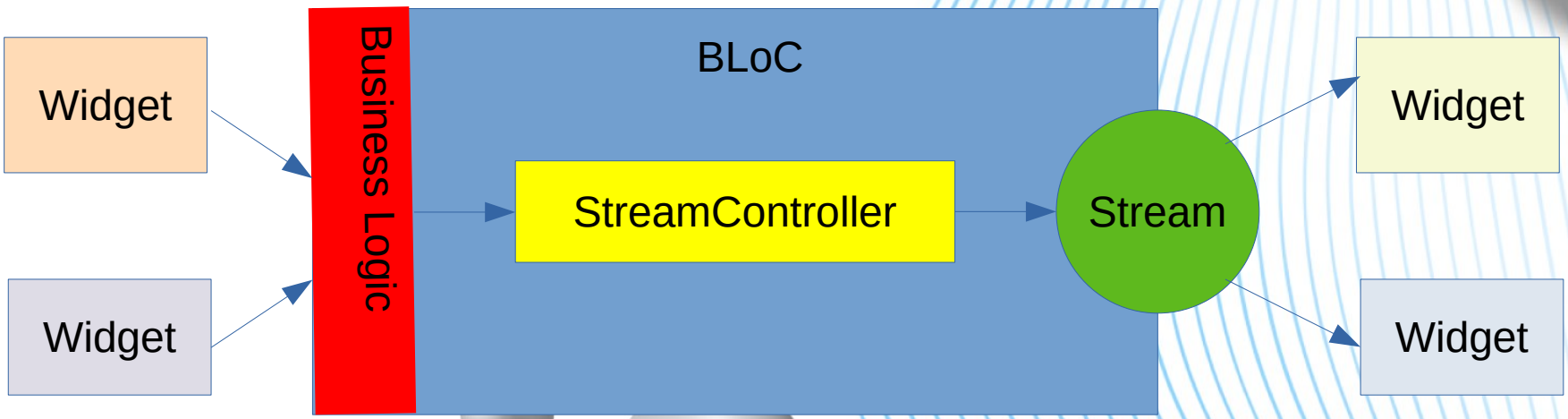
Provider

```
Performing hot restart...
Restarted application in 1.435ms.
I/flutter ( 4805): Criando Home
I/flutter ( 4805): Criando WidgetA
I/flutter ( 4805): Recriando texto do WidgetA
I/flutter ( 4805): Criando WidgetB
I/flutter ( 4805): Recriando ListTile do WidgetB
I/flutter ( 4805): Criando WidgetC
I/flutter ( 4805): Recriando texto do WidgetC
I/flutter ( 4805): Criando WidgetD
I/flutter ( 4805): Recriando ListTile do WidgetD
I/flutter ( 4805): Criando countButton
I/flutter ( 4805): Criando colorButton
I/flutter ( 4805): Recriando texto do WidgetC
I/flutter ( 4805): Recriando texto do WidgetA
I/flutter ( 4805): Recriando texto do WidgetC
I/flutter ( 4805): Recriando texto do WidgetA
I/flutter ( 4805): Recriando ListTile do WidgetB
I/flutter ( 4805): Recriando ListTile do WidgetD
I/flutter ( 4805): Criando colorButton
```

BLoC

BloC (Business Logic Object Components) é um padrão de desenvolvimento que propõe a separação entre as regras de negócios e a UI.

Segundo esse padrão, o estado e as regras de negócios devem ser colocados em classes separadas da UI. A comunicação entre o BloC e a UI deve ser feita através de Stream.



BLoC

Encontramos implementações de BloC sem o uso direto de **Stream**, porém, o conceito inicial é baseado na utilização de **Stream**.

BLoC

Crie uma nova aplicação Flutter com o nome **state_bloc_stream** e altere o arquivo **main.dart**:

```
import 'package:flutter/material.dart';
import 'pages/home.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(),
    );
  }
}
```

BLoC

Crie o subdiretório **lib/blocs** e crie o arquivo **bloc.dart** no diretório **lib/blocs**:

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

class MyBLoC {
  static final MyBLoC _singleton = MyBLoC._internal();
  final _countController =
StreamController<int>.broadcast();
  final _colorController =
StreamController<Color>.broadcast();
  int _count;
  Color _color;

  factory MyBLoC() => _singleton;

  MyBLoC._internal() {
    _count = 0;
    _color = UniqueColorGenerator.getColor(); }
}
```

BLoC

O objeto para o BloC será implementado com um singleton.

Acrescente os getters no arquivo bloc.dart:

```
int get count => _count;
Color get color => _color;
Stream<int> get countStream => _countController.stream;
Stream<Color> get colorStream => _colorController.stream;
```

Acrescente o método para incrementar o contador no arquivo bloc.dart:

```
void inc() {
  _count++;
  _countController.add(_count);
}
```

Acrescente o método para alterar a cor no arquivo bloc.dart:

```
void newColor() {
  _color = UniqueColorGenerator.getColor();
  _colorController.add(_color);
}
```

BLoC

Acrescente o método para fechar os Streams no arquivo **bloc.dart**:

```
  void dispose(){
    _countController.close();
    _colorController.close();
  }
```

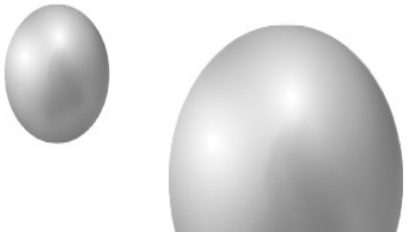
Acrescente a classe para gerar uma cor aleatória no arquivo **bloc.dart**:

```
//
// Classe para gerar uma cor aleatoria
//
class UniqueColorGenerator {
  static Random random = Random();
  static Color getColor() {
          return  Color.fromARGB(255,  random.nextInt(255),
random.nextInt(255), random.nextInt(255));
  }
}
```

BLoC

Crie o subdiretório lib/pages. Crie o arquivo home.dart no diretório lib/pages:

```
import 'package:flutter/material.dart';
import '../blocs/bloc.dart';
import '../utils/countbutton.dart';
import '../utils/colorbutton.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';
import '../utils/widgetD.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}
```



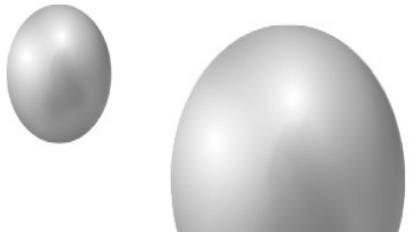
BLoC

Acrescente a classe para gerenciar o estado da tela no arquivo home.dart:

```
class _HomeState extends State {
  MyBLoC bloc = MyBLoC();

  @override
  Widget build(BuildContext context) {
    print("Criando Home");
    return Scaffold(
      appBar: AppBar(title: Text('BLoC')),
      body: Column(
        children: <Widget>[
          WidgetA(),
          WidgetB(),
          WidgetC(),
          WidgetD(),
        ],
      ),
```

BLoC

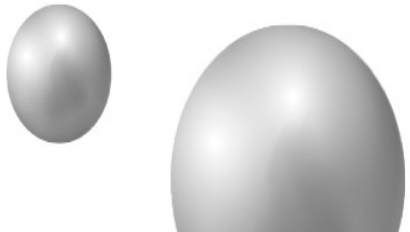
```
      bottomNavigationBar: ButtonBar(
        alignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          countButton(),
          colorButton(),
        ],
      ),
    );
  }

  @override
  dispose() {
    bloc.dispose();
  }
}
```



BLoC

Crie o subdiretório lib/utils e crie o arquivo countbutton.dart no diretório lib/utils:

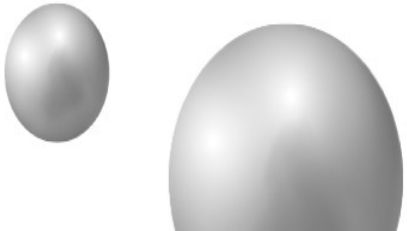
```
import 'package:flutter/material.dart';
import '../blocs/bloc.dart';

class countButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando countButton");
    return RaisedButton(
      child: Text('Numero'),
      onPressed: () => bloc.inc(),
    );
  }
}
```



BLoC

Crie o arquivo colorbutton.dart no diretório lib/utils:

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

class colorButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando colorButton");
    return RaisedButton(
      child: Text('Cor'),
      onPressed: () => bloc.newColor(),
    );
  }
}
```



BLoC

Crie o arquivo widgetA.dart no diretório lib/utils:

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

//
// Widget para exibir contador
//
class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetA");
    return StreamBuilder<int>(
      stream: bloc.countStream,
      initialData: bloc.count,
      builder: (context, snapshot) {
        print("Recriando texto do WindgetA");
```

BLoC

```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return ListTile(
            title: Text('Contador ${snapshot.data}'),
          );
        }
      }
    );
  }
}
```

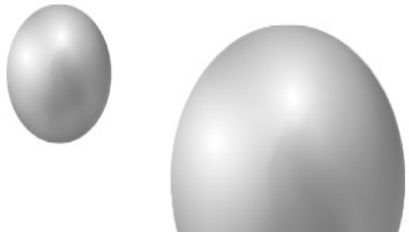
BLoC

Crie o arquivo widgetB.dart no diretório lib/utils:

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

//
// Widget para exibir a cor
//
class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetB");
    return StreamBuilder<Color>(
      stream: bloc.colorStream,
      initialData: bloc.color,
      builder: (context, snapshot) {
        print("Recriando ListTile do WindgetB");
```

BLoC

```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return Container(
            color: snapshot.data,
            child: ListTile(
              title: Text('Cor '),
            ),
          );
        }
      }
    );
  }
}
```



BLoC

Crie o arquivo widgetC.dart no diretório lib/utils:

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

//
// Widget para exibir contador
//
class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetC");
    return StreamBuilder<int>(
      stream: bloc.countStream,
      initialData: bloc.count,
      builder: (context, snapshot) {
        print("Recriando texto do WindgetC");
```

BLoC

```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return ListTile(
            title: Text('Contador ${snapshot.data}'),
          );
        }
      }
    );
  }
}
```

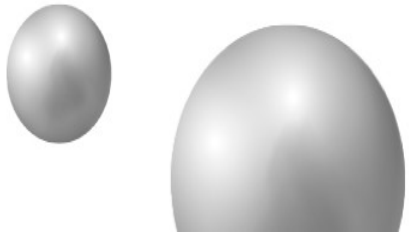
BLoC

Crie o arquivo widgetD.dart no diretório lib/utils:

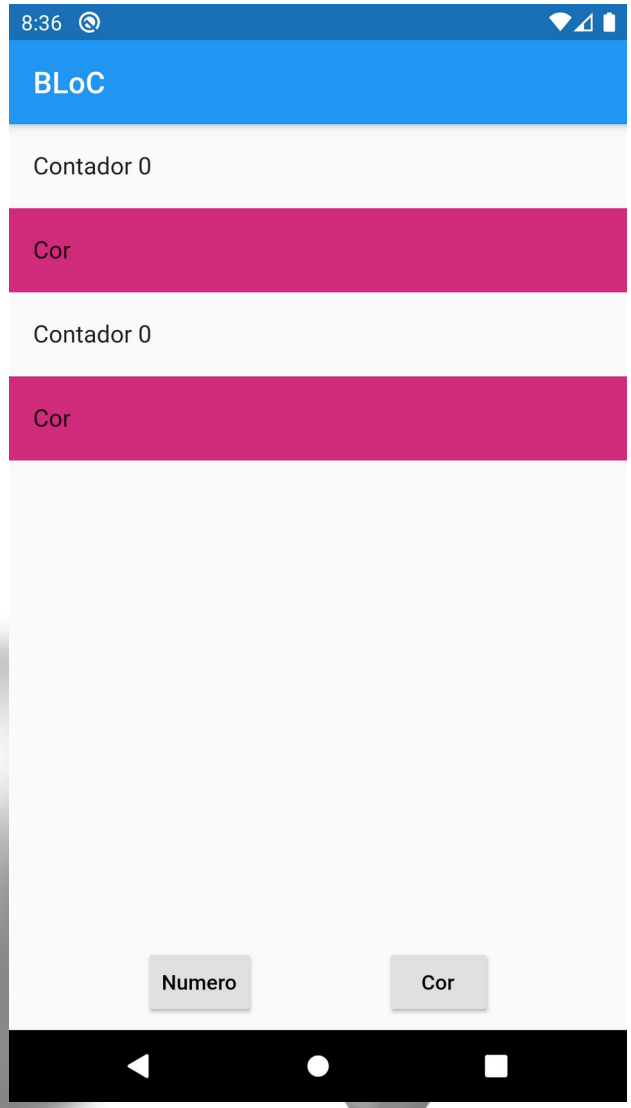
```
import 'package:flutter/material.dart';
import '../blocs/bloc.dart';

//
// Widget para exibir a cor
//
class WidgetD extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetD");
    return StreamBuilder<Color>(
      stream: bloc.colorStream,
      initialData: bloc.color,
      builder: (context, snapshot) {
        print("Recriando ListTile do WindgetD");
```

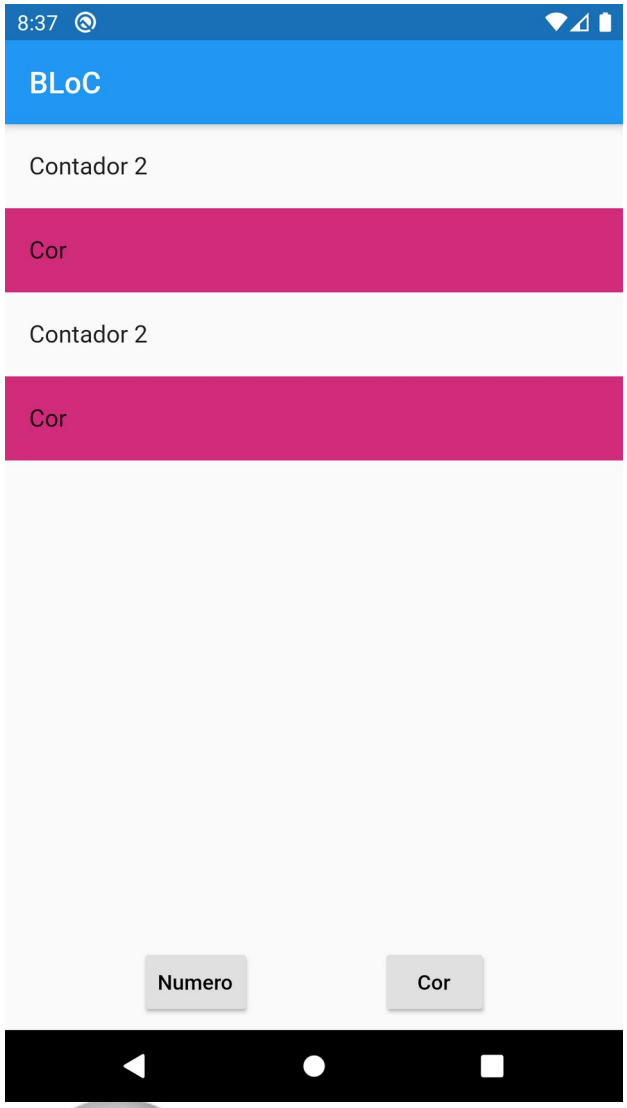
BLoC

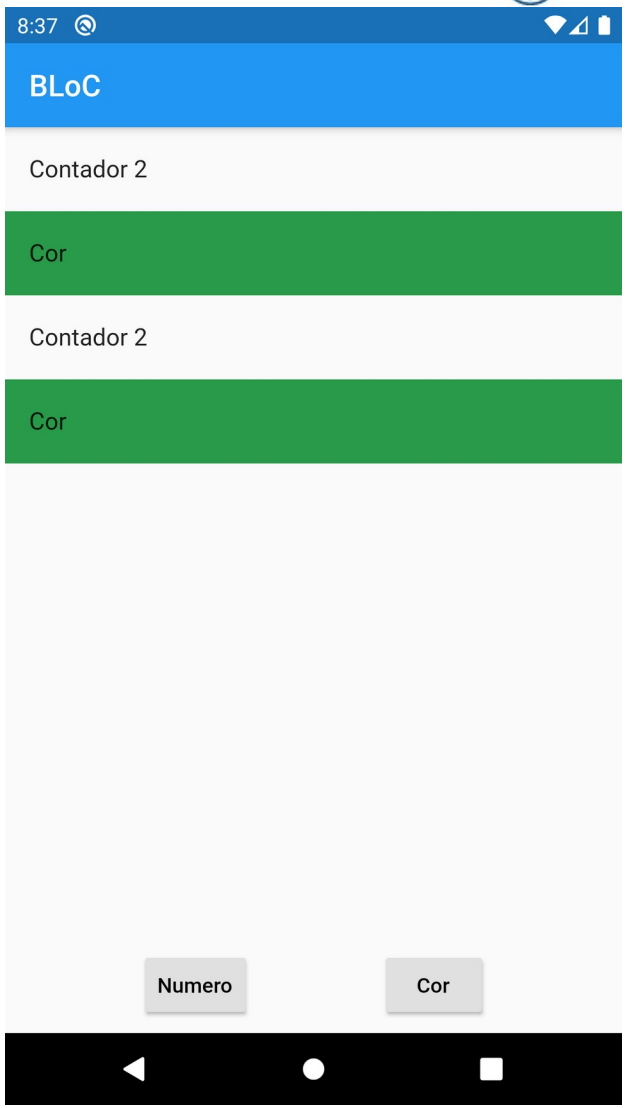
```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return Container(
            color: snapshot.data,
            child: ListTile(
              title: Text('Cor '),
            ),
          );
        }
      }
    );
  }
}
```



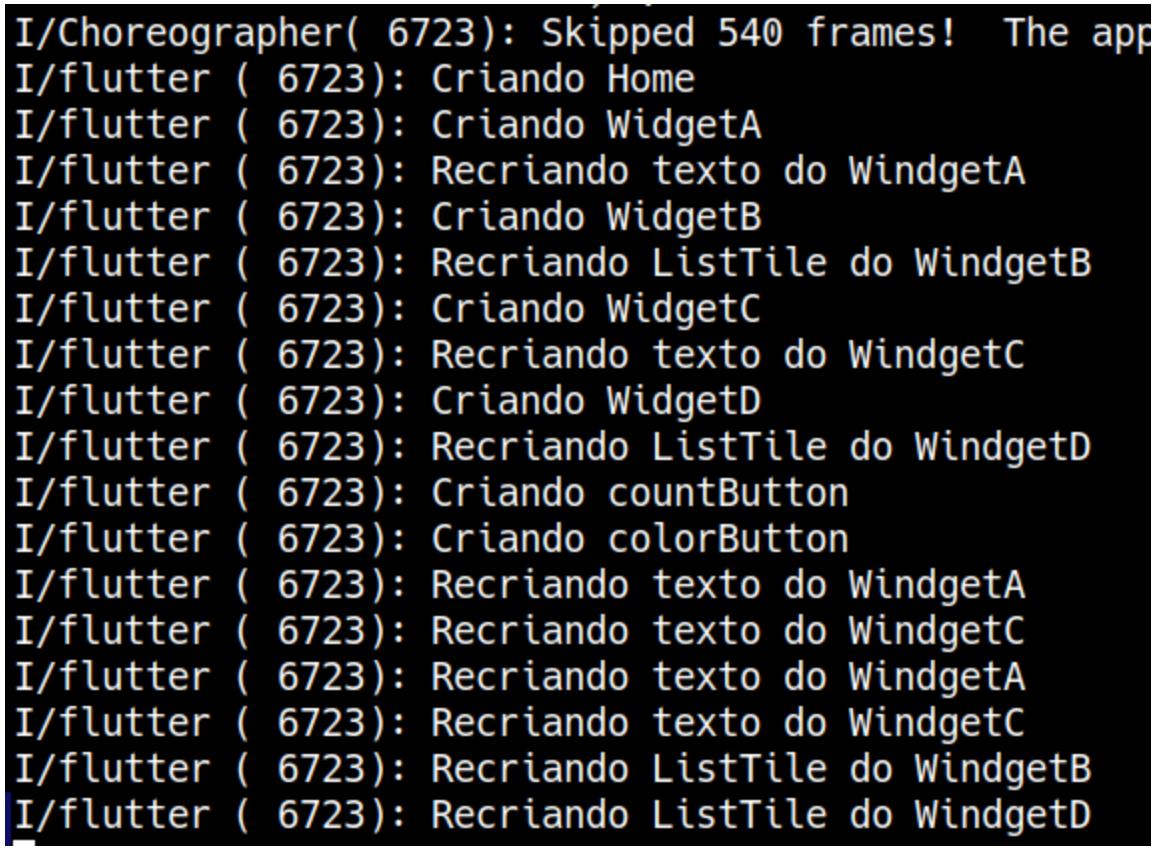
BLoC


8:36
BLoC
Contador 0
Cor
Contador 0
Cor
Numero
Cor


8:37
BLoC
Contador 2
Cor
Contador 2
Cor
Numero
Cor


8:37
BLoC
Contador 2
Cor
Contador 2
Cor
Numero
Cor

BLoC


I/Choreographer(6723): Skipped 540 frames! The app
I/flutter (6723): Criando Home
I/flutter (6723): Criando WidgetA
I/flutter (6723): Recriando texto do WindgetA
I/flutter (6723): Criando WidgetB
I/flutter (6723): Recriando ListTile do WindgetB
I/flutter (6723): Criando WidgetC
I/flutter (6723): Recriando texto do WindgetC
I/flutter (6723): Criando WidgetD
I/flutter (6723): Recriando ListTile do WindgetD
I/flutter (6723): Criando countButton
I/flutter (6723): Criando colorButton
I/flutter (6723): Recriando texto do WindgetA
I/flutter (6723): Recriando texto do WindgetC
I/flutter (6723): Recriando texto do WindgetA
I/flutter (6723): Recriando texto do WindgetC
I/flutter (6723): Recriando ListTile do WindgetB
I/flutter (6723): Recriando ListTile do WindgetD

BLoC

Existem pacotes com classes próprias para implementação de BloC.

Crie uma nova aplicação Flutter com o nome state_bloc.

Altere o arquivo pubspec.yaml para incluir a dependência de flutter_bloc:

```
dependencies:
  flutter_bloc: ^3.2.0
  flutter:
    sdk: flutter
```

Instale o pacote adicionado nas dependências.

BLoC

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'blocs/countbloc.dart';
import 'blocs/colorbloc.dart';
import 'pages/home.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,
      ),
```

BLoC

```
      home: MultiBlocProvider(
        providers: [
          BlocProvider<CountBloc>(
            create: (context) => CountBloc(),
          ),
          BlocProvider<ColorBloc>(
            create: (context) => ColorBloc(),
          ),
        ],
        child: Home(),
      ),
    );
  }
}
```

BLoC

Crie o subdiretório lib/blocs e crie o arquivo countbloc.dart no diretório lib/blocs:

```
import 'package:flutter_bloc/flutter_bloc.dart';

enum CountEvent { increment, decrement }

class CountBloc extends Bloc<CountEvent, int> {
  @override
  int get initialState => 0;

  @override
  Stream<int> mapEventToState(CountEvent event) async* {
    switch (event) {
      case CountEvent.decrement:
        yield state - 1;
        break;
      case CountEvent.increment:
        yield state + 1;
        break; }
  }
}
```

BLoC

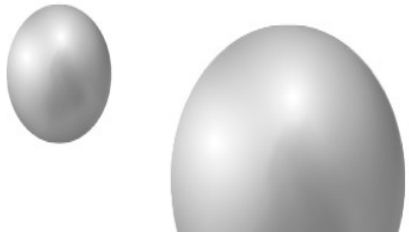
Crie o arquivo colorbloc.dart no diretório lib/blocs:

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

enum ColorEvent { change }

class ColorBloc extends Bloc<ColorEvent, Color> {
  @override
  Color get initialState => UniqueColorGenerator.getColor();

  @override
  Stream<Color> mapEventToState(ColorEvent event) async* {
    switch (event) {
      case ColorEvent.change:
        yield UniqueColorGenerator.getColor();
        break;
    }
  }
}
```

BLoC

Acrescente a classe para gerar uma cor aleatória no arquivo colorbloc.dart:

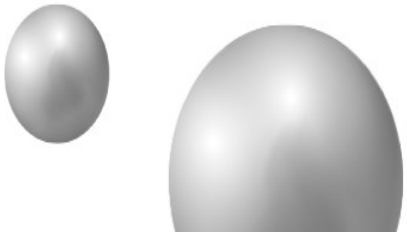
```
//
// Classe para gerar uma cor aleatoria
//
class UniqueColorGenerator {
  static Random random = Random();
  static Color getColor() {
    return Color.fromARGB(255, random.nextInt(255),
random.nextInt(255), random.nextInt(255));
  }
}
```



BLoC

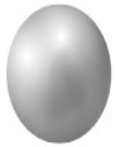
Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no diretório **lib/pages**:

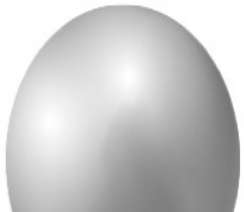
```
import 'package:flutter/material.dart';
import '../utils/incbutton.dart';
import '../utils/decbutton.dart';
import '../utils/colorbutton.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';
import '../utils/widgetD.dart';

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('BLoC')),
```



BLoC

```
      body: Column(
        children: <Widget>[
          WidgetA(),
          WidgetB(),
          WidgetC(),
          WidgetD(),
        ],
      ),
      bottomNavigationBar: ButtonBar(
        alignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          incButton(),
          decButton(),
          colorButton(),
        ],
      ),
    );
  }
}
```





BLoC

Crie o subdiretório lib/utils e crie o arquivo incbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/countbloc.dart';

class incButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final CountBloc countBloc =
BlocProvider.of<CountBloc>(context);
    print("Criando incButton");
    return RaisedButton(
      child: Icon(Icons.add),
      onPressed: () => countBloc.add(CountEvent.increment),
    );
  }
}
```

BLoC

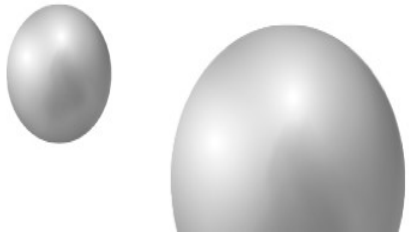
Crie o arquivo decbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/countbloc.dart';

class decButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final CountBloc countBloc =
BlocProvider.of<CountBloc>(context);
    print("Criando decButton");
    return RaisedButton(
      child: Icon(Icons.remove),
      onPressed: () => countBloc.add(CountEvent.decrement),
    );
  }
}
```

BLoC

Crie o arquivo colorbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/colorbloc.dart';

class colorButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final ColorBloc colorBloc =
BlocProvider.of<ColorBloc>(context);
    print("Criando colorButton");
    return RaisedButton(
      child: Text('Cor'),
      onPressed: () => colorBloc.add(ColorEvent.change),
    );
  }
}
```



BLoC

Crie o arquivo widgetA.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/countbloc.dart';

// Widget para exibir contador
class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetA");
    return BlocBuilder<CountBloc, int>(
      builder: (context, count) {
        print("Recriando texto do WindgetA");
        return ListTile(
          title: Text('Contador $count'),
        );
      },
    );
  }
}
```

BLoC

Crie o arquivo widgetB.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/colorbloc.dart';

// Widget para exibir a cor
class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetB");
    return BlocBuilder<ColorBloc, Color>(
      builder: (context, color) {
        print("Recriando ListTile do WindgetB");
        return Container(
          color: color,
          child: ListTile(title: Text('Cor '),),
        );
      },
    );
  }
}
```

BLoC

Crie o arquivo widgetC.dart no diretório lib/utils:

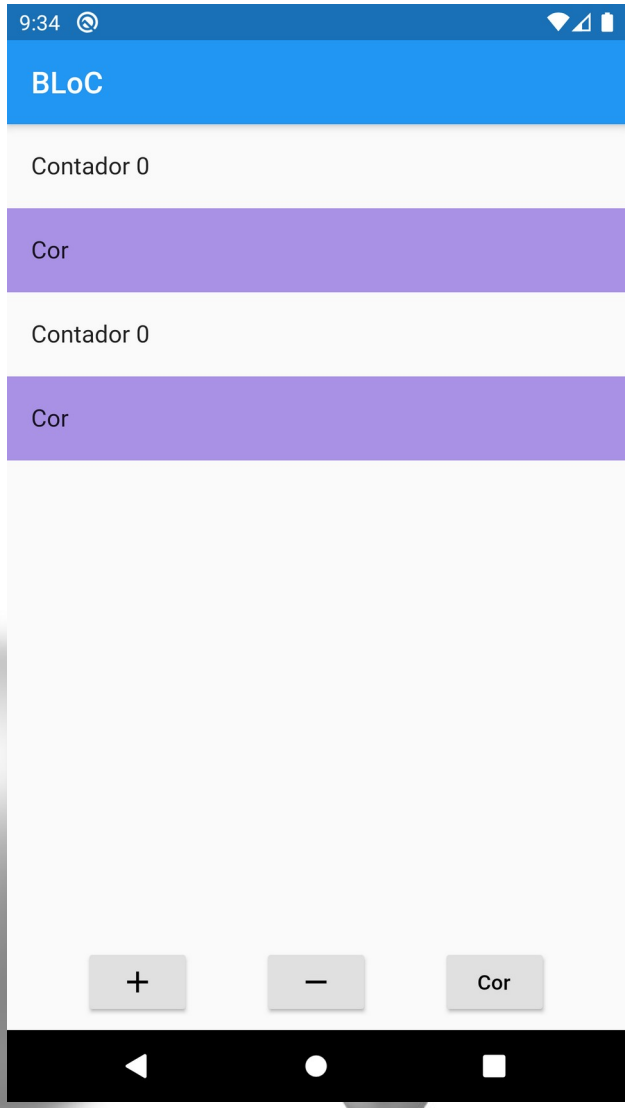
```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/countbloc.dart';

// Widget para exibir contador
class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetC");
    return BlocBuilder<CountBloc, int>(
      builder: (context, count) {
        print("Recriando texto do WindgetC");
        return ListTile(
          title: Text('Contador $count'),
        );
      },
    );
  }
}
```

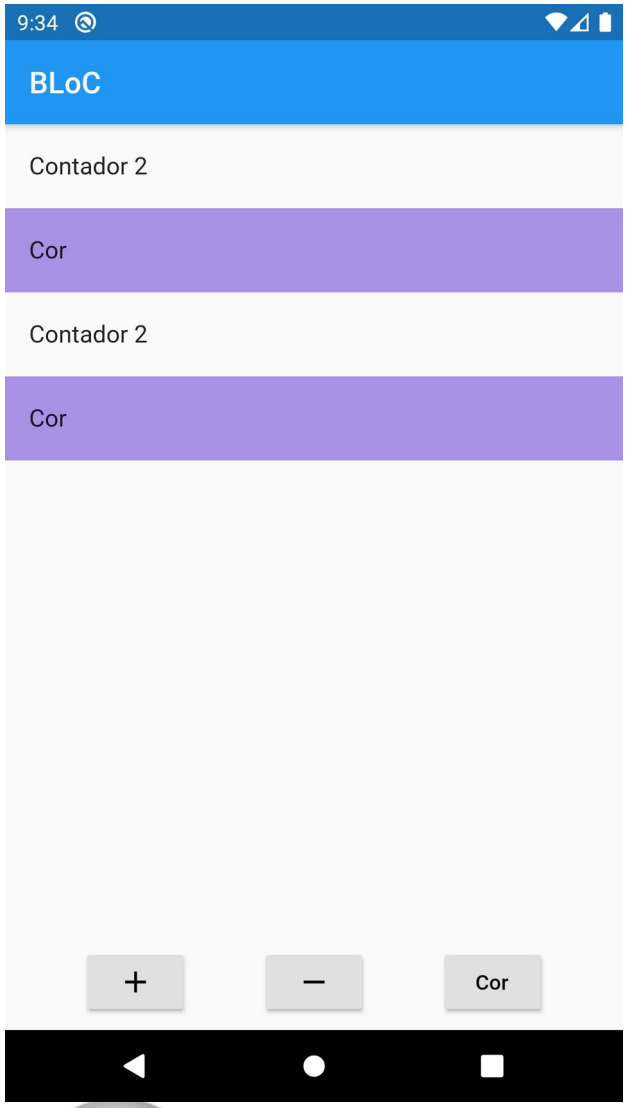
BLoC

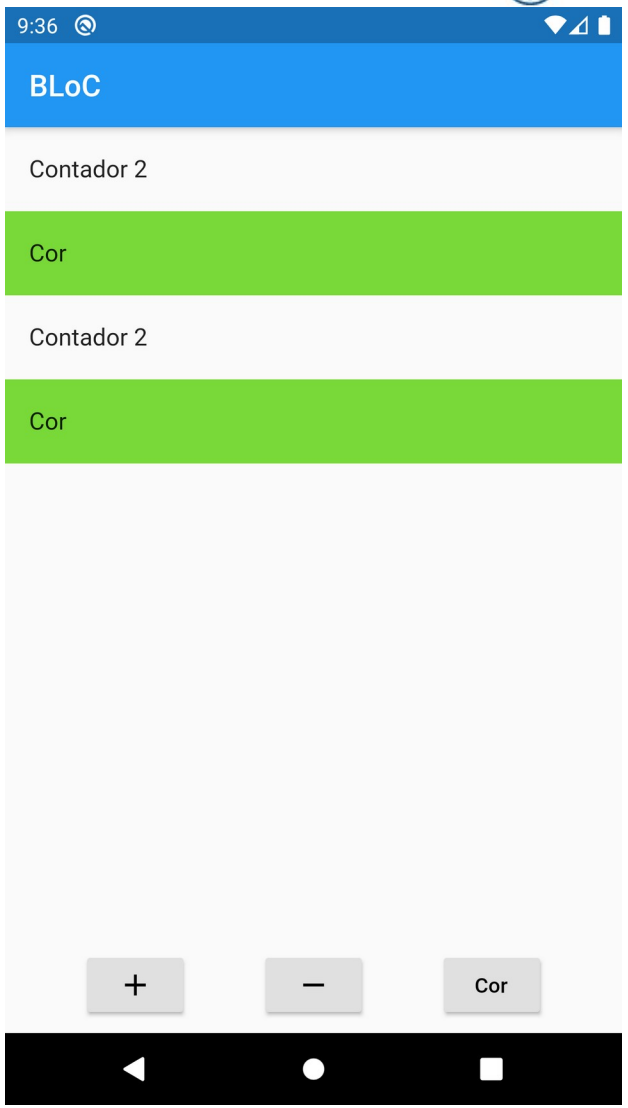
Crie o arquivo widgetD.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/colorbloc.dart';

// Widget para exibir a cor
class WidgetD extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetD");
    return BlocBuilder<ColorBloc, Color>(
      builder: (context, color) {
        print("Recriando ListTile do WindgetD");
        return Container(
          color: color,
          child: ListTile(title: Text('Cor '),),
        );
      },
    );
  }
}
```

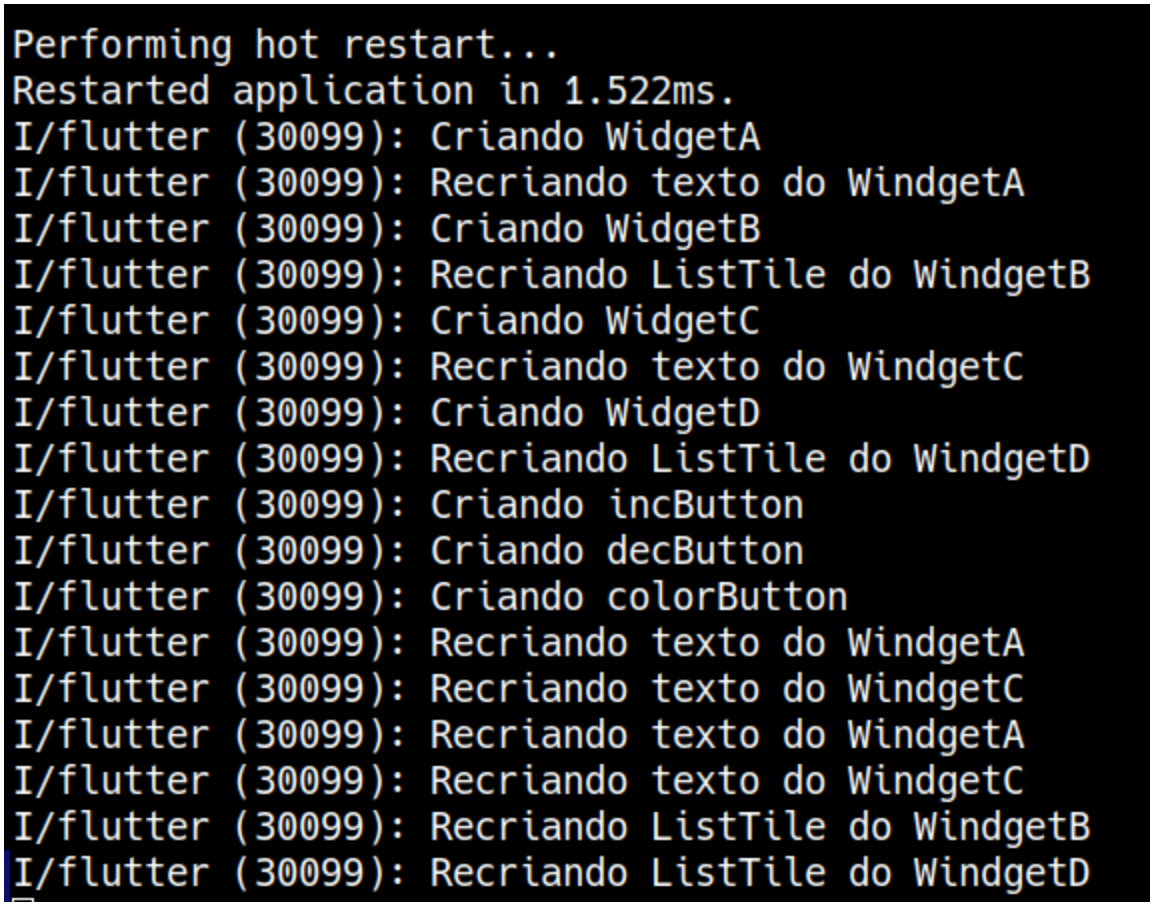
BLoC


9:34
BLoC
Contador 0
Cor
Contador 0
Cor
+
−
Cor


9:34
BLoC
Contador 2
Cor
Contador 2
Cor
+
−
Cor


9:36
BLoC
Contador 2
Cor
Contador 2
Cor
+
−
Cor

BLoC


Performing hot restart...
Restarted application in 1.522ms.
I/flutter (30099): Criando WidgetA
I/flutter (30099): Recriando texto do WindgetA
I/flutter (30099): Criando WidgetB
I/flutter (30099): Recriando ListTile do WindgetB
I/flutter (30099): Criando WidgetC
I/flutter (30099): Recriando texto do WindgetC
I/flutter (30099): Criando WidgetD
I/flutter (30099): Recriando ListTile do WindgetD
I/flutter (30099): Criando incButton
I/flutter (30099): Criando decButton
I/flutter (30099): Criando colorButton
I/flutter (30099): Recriando texto do WindgetA
I/flutter (30099): Recriando texto do WindgetC
I/flutter (30099): Recriando texto do WindgetA
I/flutter (30099): Recriando texto do WindgetC
I/flutter (30099): Recriando ListTile do WindgetB
I/flutter (30099): Recriando ListTile do WindgetD

BLoC

BloC pode ser implementado com vários blocos ou unindo os estados e lógica em um único bloco.

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

Altere o arquivo **pubspec.yaml** para incluir a dependência de **flutter_bloc**:

```
dependencies:
  flutter_bloc: ^3.2.0
  flutter:
    sdk: flutter
```

Instale o pacote adicionado nas dependências.

BLoC

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'blocs/bloc.dart';
import 'pages/home.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,
      ),
```

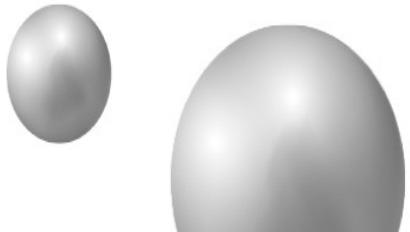
BLoC

```
      home: BlocProvider(
        create: (BuildContext context) {
          return MyBLoC();
        },
        child: Home(),
      ),
    );
  }
}
```

BLoC

Crie o subdiretório **lib/models** e crie o arquivo **state.dart** no diretório **lib/models**:

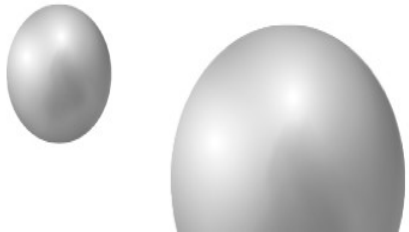
```
import 'package:flutter/material.dart';

class BLoCState {
  int count;
  Color color;

  BLoCState(this.count,this.color);
}
```



BLoC

Crie o subdiretório **lib/blocs** e crie o arquivo **bloc.dart** no diretório **lib/blocs**:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'dart:math';
import '../models/state.dart';

enum Event { increment, decrement, color }

class MyBLoC extends Bloc<Event, BLoCState> {
  @override
  BLoCState get initialState =>
BLoCState(0,UniqueColorGenerator.getColor());
```



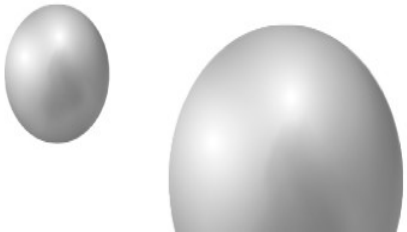
BLoC

```
  @override
  Stream<BLoCState> mapEventToState(Event event) async* {
    switch (event) {
      case Event.decrement:
        yield BLoCState(state.count-1,state.color);
        break;
      case Event.increment:
        yield BLoCState(state.count+1,state.color);
        break;
      case Event.color:
        yield
BLoCState(state.count,UniqueColorGenerator.getColor());
        break;
    }
  }
}
```

BLoC

Acrescente a classe para gerar uma cor aleatória no arquivo bloc.dart:

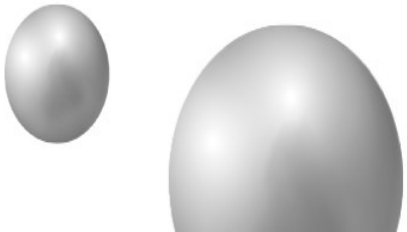
```
//
// Classe para gerar uma cor aleatoria
//
class UniqueColorGenerator {
  static Random random = Random();
  static Color getColor() {
    return Color.fromARGB(255, random.nextInt(255),
random.nextInt(255), random.nextInt(255));
  }
}
```



BLoC

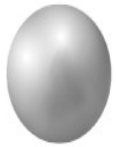
Crie o subdiretório **lib/pages** e crie o arquivo **home.dart** no diretório **lib/pages**:

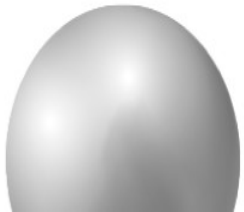
```
import 'package:flutter/material.dart';
import '../utils/incbutton.dart';
import '../utils/decbutton.dart';
import '../utils/colorbutton.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';
import '../utils/widgetD.dart';

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('BLoC')),
```



BLoC

```
      body: Column(
        children: <Widget>[
          WidgetA(),
          WidgetB(),
          WidgetC(),
          WidgetD(),
        ],
      ),
      bottomNavigationBar: ButtonBar(
        alignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          incButton(),
          decButton(),
          colorButton(),
        ],
      ),
    );
  }
}
```





BLoC

Crie o subdiretório lib/utils e crie o arquivo incbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/bloc.dart';

class incButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final MyBLoC bloc = BlocProvider.of<MyBLoC>(context);
    print("Criando incButton");
    return RaisedButton(
      child: Icon(Icons.add),
      onPressed: () => bloc.add(Event.increment),
    );
  }
}
```

BLoC

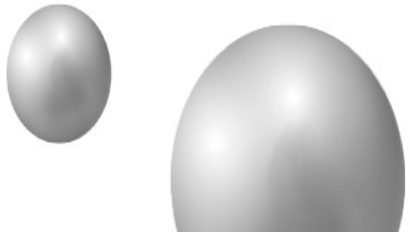
Crie o arquivo decbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/bloc.dart';

class decButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final MyBLoC bloc = BlocProvider.of<MyBLoC>(context);
    print("Criando decButton");
    return RaisedButton(
      child: Icon(Icons.remove),
      onPressed: () => bloc.add(Event.decrement),
    );
  }
}
```

BLoC

Crie o arquivo colorbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/bloc.dart';

class colorButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final MyBLoC bloc = BlocProvider.of<MyBLoC>(context);
    print("Criando colorButton");
    return RaisedButton(
      child: Text('Cor'),
      onPressed: () => bloc.add(Event.color),
    );
  }
}
```



BLoC

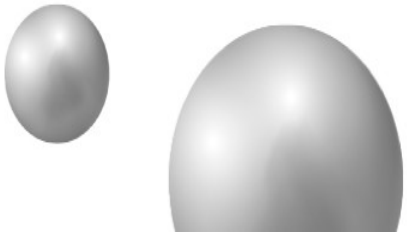
Crie o arquivo widgetA.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/bloc.dart';

class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetA");
    return BlocConsumer(
      bloc: BlocProvider.of<MyBLoC>(context),
      listener: (context, state) {},
      buildWhen: (previous, current) {
        return previous.count != current.count; },
      builder: (context, state) {
        print("Recriando texto do WindgetA");
        return ListTile(
          title: Text('Contador ${state.count}'), );
      },
    );
}}
```

BLoC

Crie o arquivo widgetB.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/bloc.dart';

class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetB");
    return BlocConsumer(
      bloc: BlocProvider.of<MyBLoC>(context),
      listener: (context, state) {
      },
      buildWhen: (previous, current) {
        return previous.color != current.color;
      },
```



BLoC

```
      builder: (context, state) {
        print("Recriando ListTile do WindgetB");
        return Container(
          color: state.color,
          child: ListTile(
            title: Text('Cor '),
          ),
        );
      },
    );
  }
}
```

BLoC

Crie o arquivo widgetC.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/bloc.dart';

class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetC");
    return BlocConsumer(
      bloc: BlocProvider.of<MyBLoC>(context),
      listener: (context, state) {},
      buildWhen: (previous, current) {
        return previous.count != current.count; },
      builder: (context, state) {
        print("Recriando texto do WindgetC");
        return ListTile(
          title: Text('Contador ${state.count}'), );
      },
    );
}}
```

BLoC

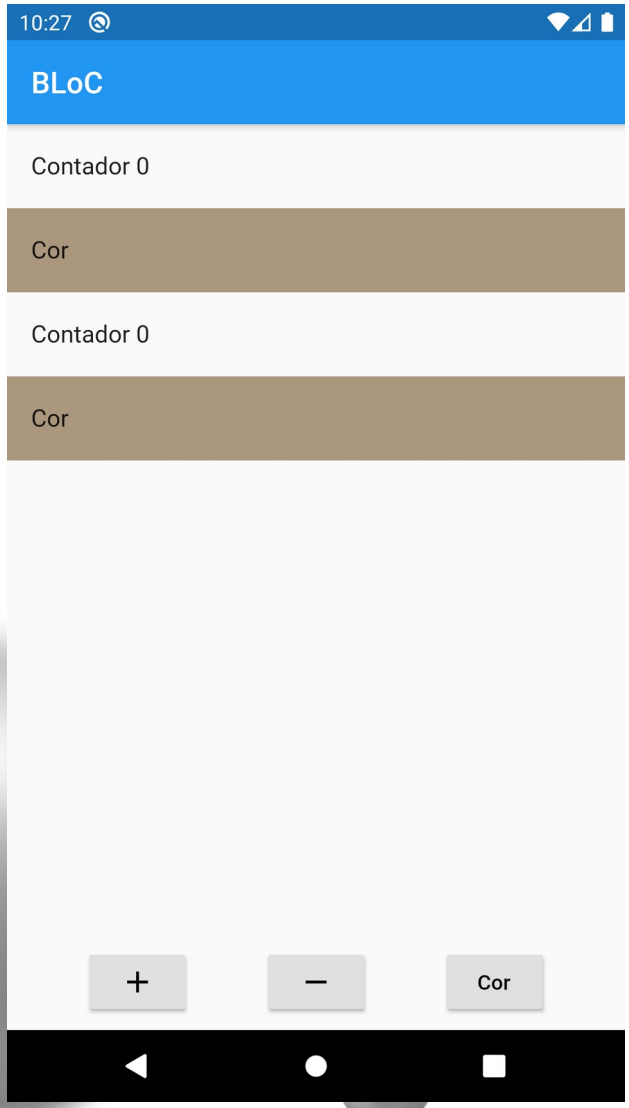
Crie o arquivo widgetD.dart no diretório lib/utils:

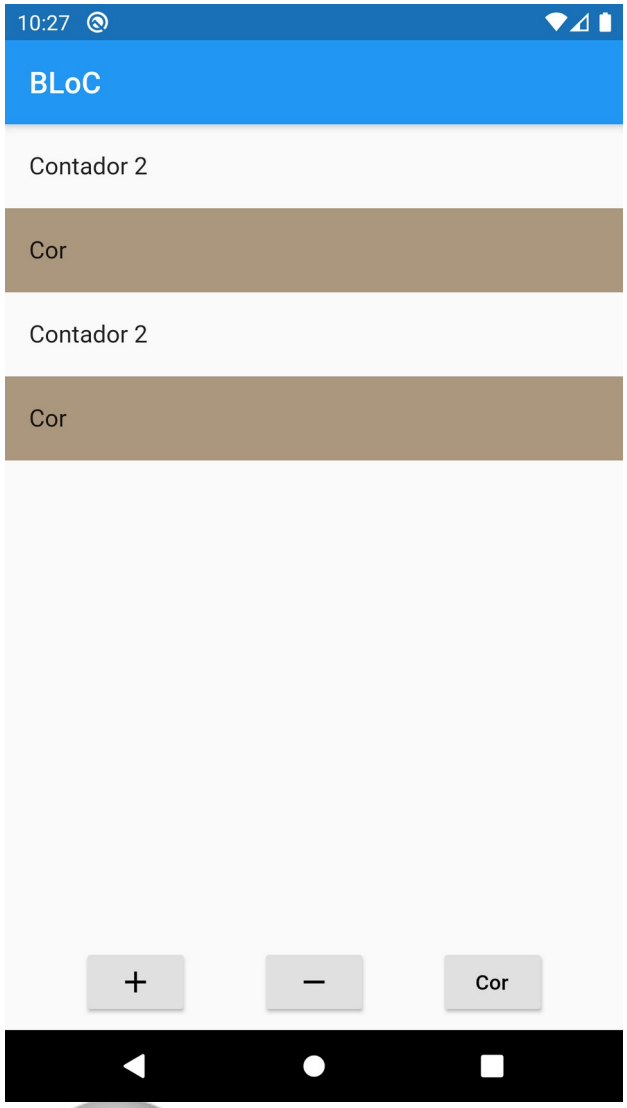
```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '../blocs/bloc.dart';

class WidgetD extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetD");
    return BlocConsumer(
      bloc: BlocProvider.of<MyBLoC>(context),
      listener: (context, state) {
      },
      buildWhen: (previous, current) {
        return previous.color != current.color;
      },
```

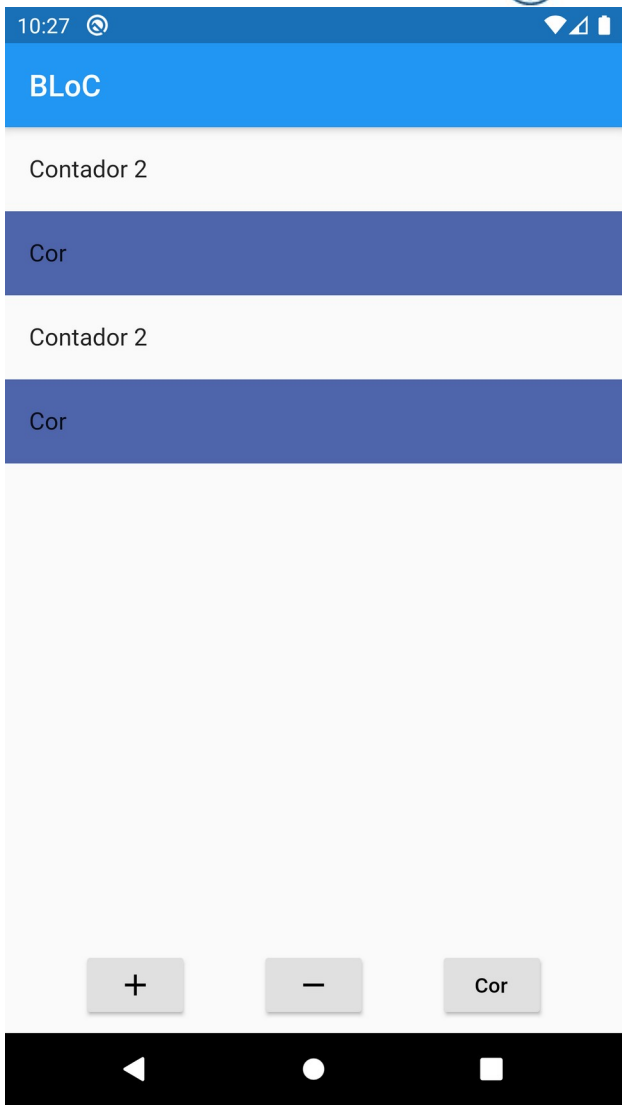
BLoC

```
      builder: (context, state) {
        print("Recriando ListTile do WindgetD");
        return Container(
          color: state.color,
          child: ListTile(
            title: Text('Cor '),
          ),
        );
      },
    );
  }
}
```

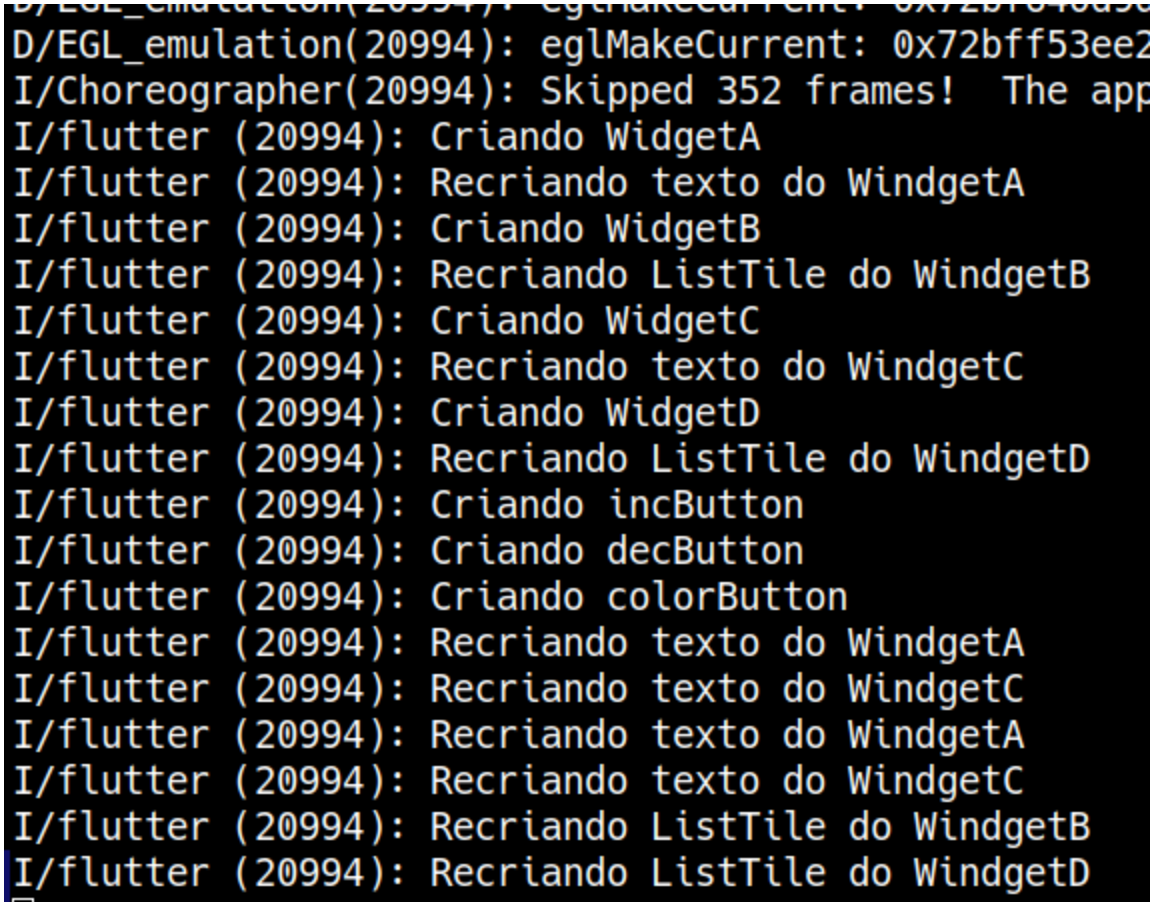
BLoC


10:27
BLoC
Contador 0
Cor
Contador 0
Cor
+
−
Cor


10:27
BLoC
Contador 2
Cor
Contador 2
Cor
+
−
Cor


10:27
BLoC
Contador 2
Cor
Contador 2
Cor
+
−
Cor

BLoC


D/EGL_emulation(20994): eglMakeCurrent: 0x72bff53ee2
I/Choreographer(20994): Skipped 352 frames! The app
I/flutter (20994): Criando WidgetA
I/flutter (20994): Recriando texto do WindgetA
I/flutter (20994): Criando WidgetB
I/flutter (20994): Recriando ListTile do WindgetB
I/flutter (20994): Criando WidgetC
I/flutter (20994): Recriando texto do WindgetC
I/flutter (20994): Criando WidgetD
I/flutter (20994): Recriando ListTile do WindgetD
I/flutter (20994): Criando incButton
I/flutter (20994): Criando decButton
I/flutter (20994): Criando colorButton
I/flutter (20994): Recriando texto do WindgetA
I/flutter (20994): Recriando texto do WindgetC
I/flutter (20994): Recriando texto do WindgetA
I/flutter (20994): Recriando texto do WindgetC
I/flutter (20994): Recriando ListTile do WindgetB
I/flutter (20994): Recriando ListTile do WindgetD

RxDart

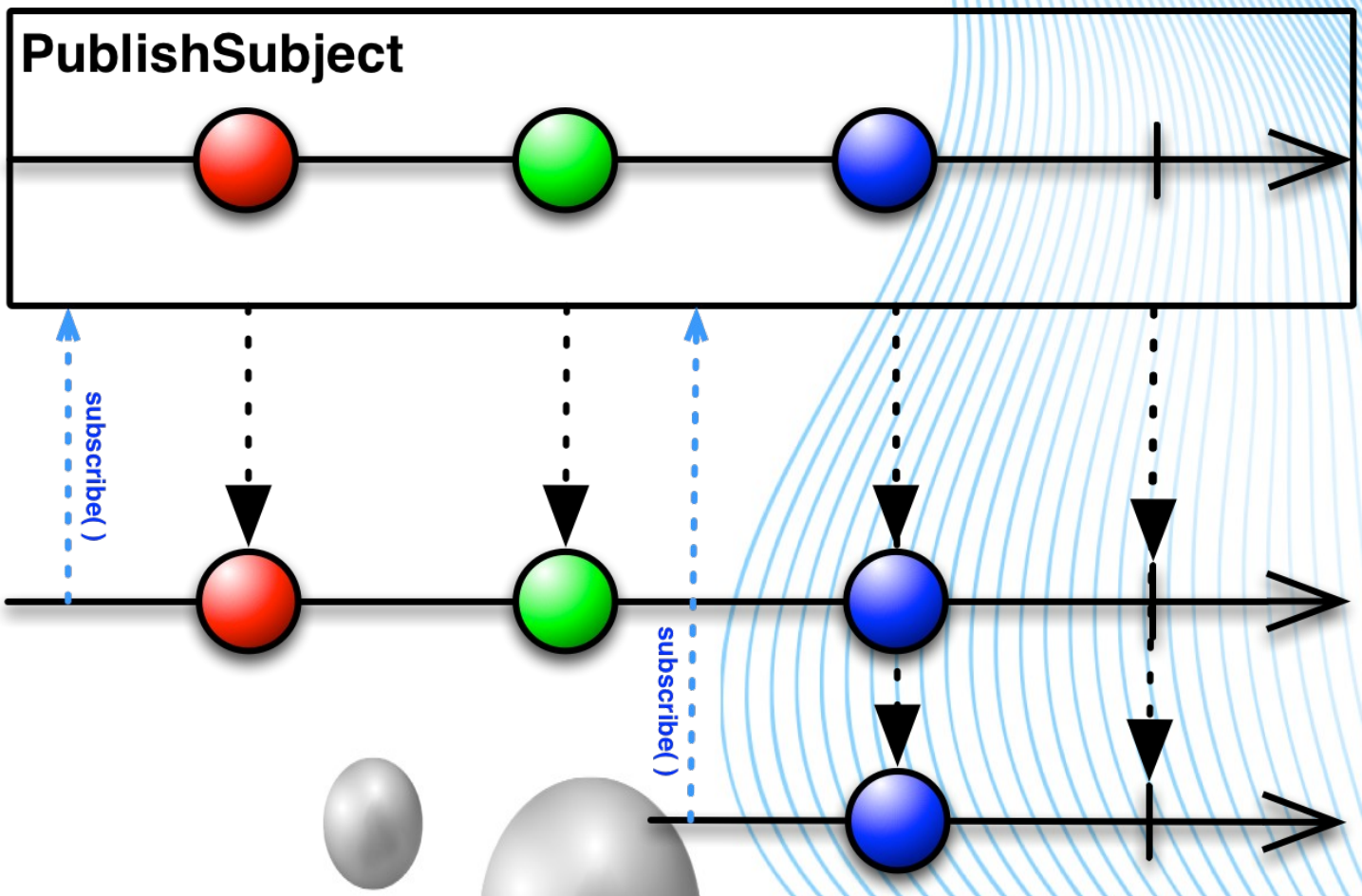
RxDart é a implementação da linguagem Dart para ReactiveX, extendendo Stream.

Dart	RxDart
Stream	Observable
StreamController	Subject

RxDart oferece 3 variações de StreamController:

RxDart

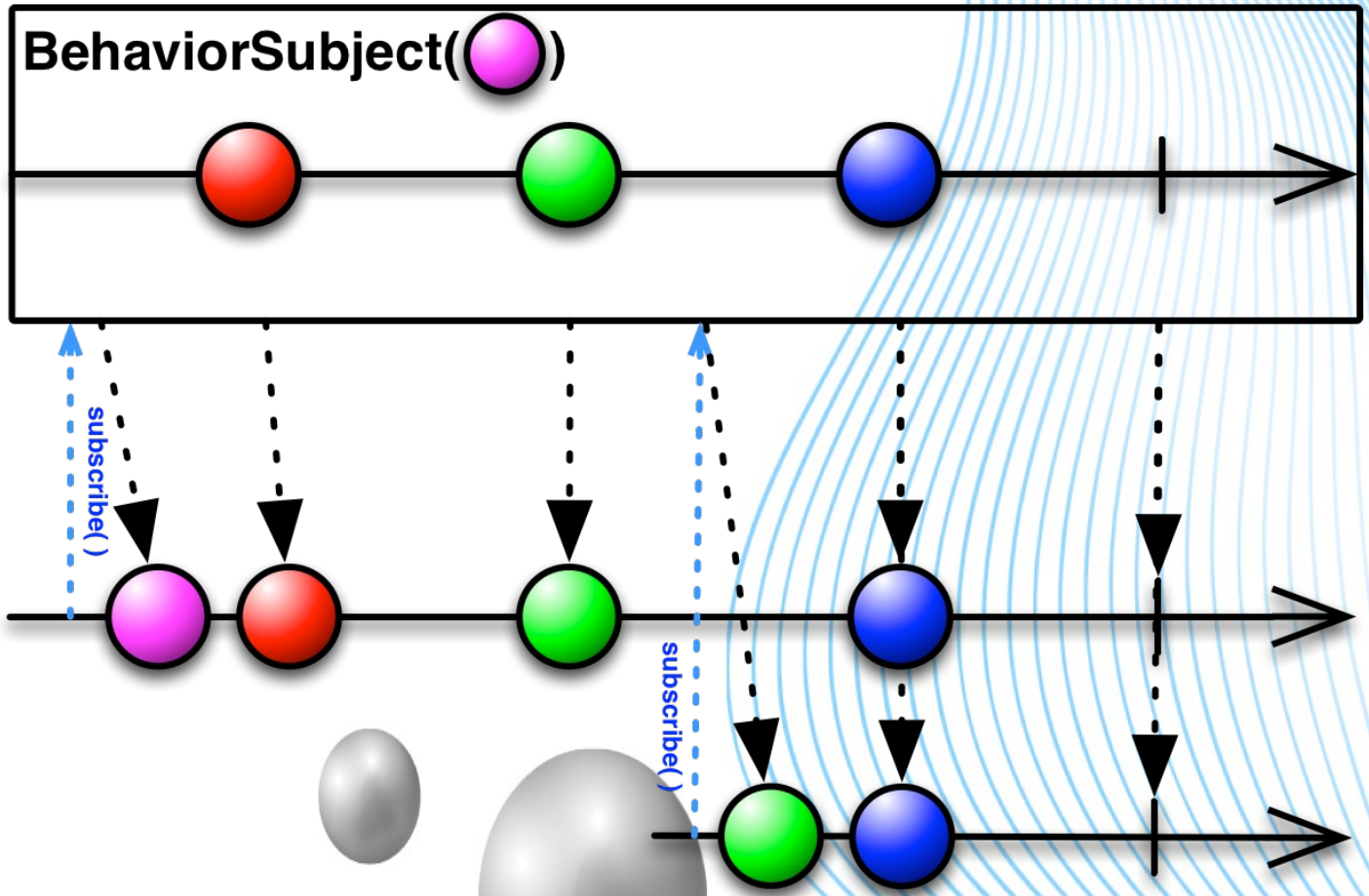
PublishSubject: é um **StreamController** com broadcast que retorna um **Observable** ao invés de um **Stream**.



https://www.didierboelens.com/2018/08/reactive-programming-streams-bloc/

RxDart

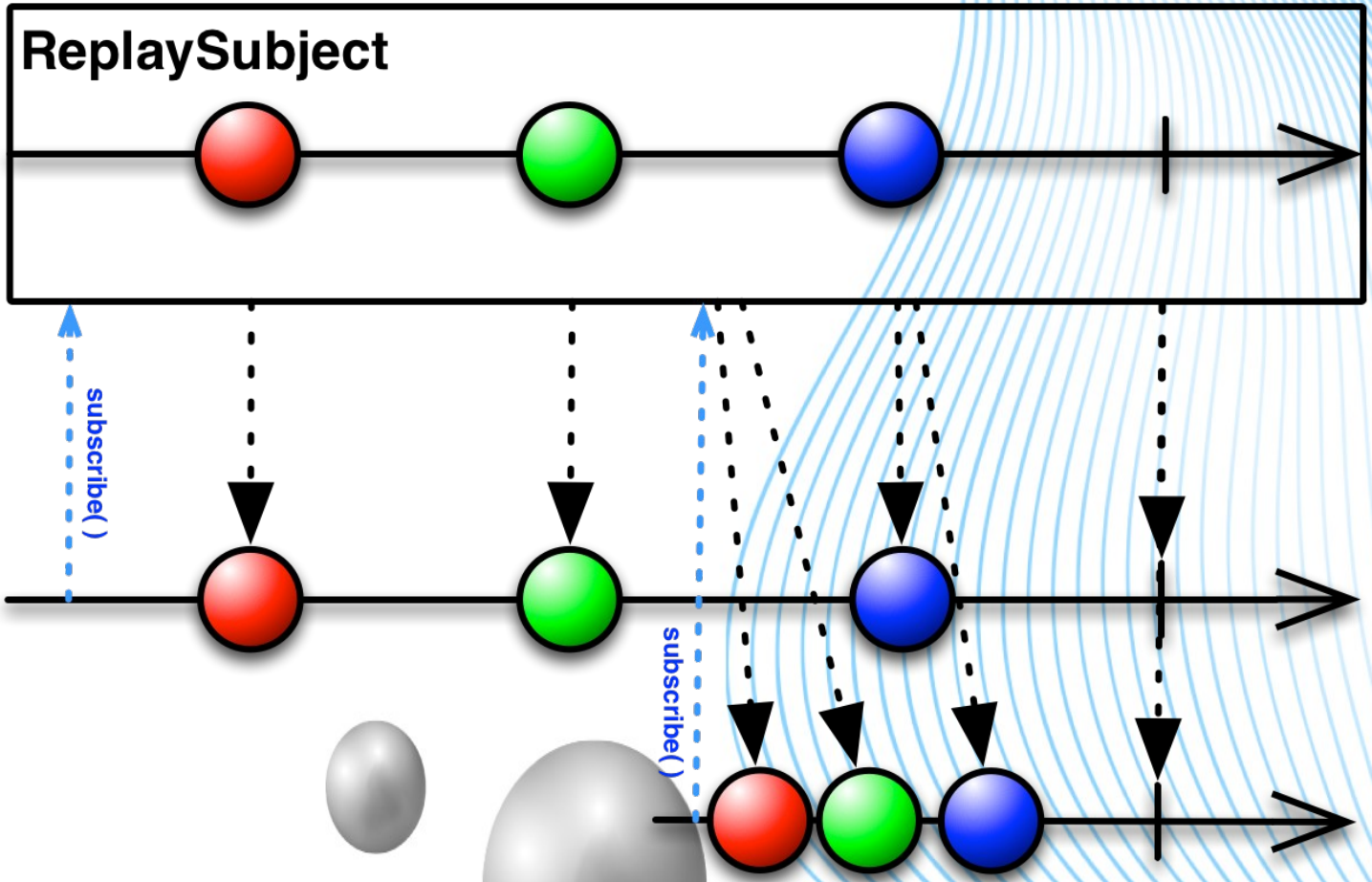
BehaviorSubject: é semelhante a **PublishSubject** , porém qualquer novo objeto que passar a ouvir o **BehaviorSubject** receberá o último evento emitido.



https://www.didierboelens.com/2018/08/reactive-programming-streams-bloc/

RxDart

ReplaySubject: é semelhante a **PublishSubject** , porém qualquer novo objeto que passar a ouvir o **ReplaySubject** receberá todos os eventos emitidos.



https://www.didierboelens.com/2018/08/reactive-programming-streams-bloc/

RxDart

Crie uma nova aplicação Flutter com o nome state_rxdart.

Altere o arquivo pubspec.yaml para incluir a dependência de rxdart:

```
dependencies:
  rxdart: ^0.22.2
  flutter:
    sdk: flutter
```

Instale o pacote adicionado nas dependências.

RxDart

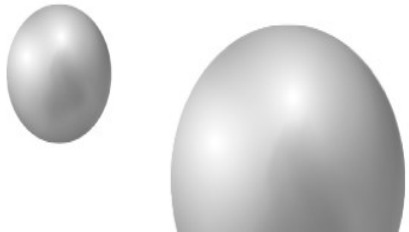
Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'pages/home.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(),
    );
  }
}
```

RxDart

Crie o subdiretório **lib/blocs** e crie o arquivo **bloc.dart** no diretório **lib/blocs**:

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

class MyBLoC {
  static final MyBLoC _singleton = MyBLoC._internal();
  BehaviorSubject<int> _subjectCount;
  ReplaySubject<Color> _subjectColor;
  int _count;
  Color _color;

  factory MyBLoC() => _singleton;
```



RxDart

```
  MyBLoC._internal() {
    _count = 0;
    _color = UniqueColorGenerator.getColor();
    _subjectCount =
BehaviorSubject<int>.seeded(this._count);
    _subjectColor = ReplaySubject<Color>();
    _subjectColor.add(_color);
  }
}
```

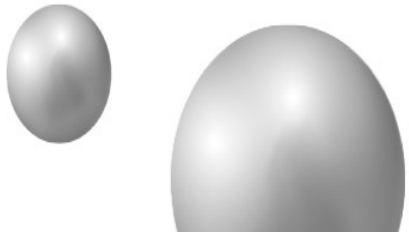
RxDart

Acrescente os getters para os observáveis e o ReplaySubject no arquivo bloc.dart:

```
  Observable<int> get countObservable =>
_subjectCount.stream;
  Observable<Color> get colorObservable =>
_subjectColor.stream;
  ReplaySubject<Color> get colorSubject => _subjectColor;
```

Acrescente os getters para as propriedades do estado no arquivo bloc.dart:

```
  int get count => _count;
  Color get color => _color;
```



RxDart

Acrescente o método para incrementar o contador no arquivo:

```
void inc() {
  _count++;
  _subjectCount.sink.add(_count);
}
```

Acrescente o método para alterar a cor no arquivo **bloc.dart**:

```
void newColor() {
  _color = UniqueColorGenerator.getColor();
  _subjectColor.sink.add(_color);
}
```

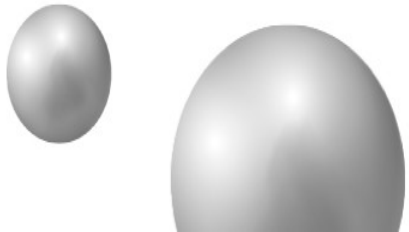
Acrescente o método para fechar os subjects no arquivo **bloc.dart**:

```
void dispose(){
  _subjectCount.close();
  _subjectColor.close();
}
```

RxDart

Acrescente a classe para gerar uma cor aleatória no arquivo bloc.dart:

```
//
// Classe para gerar uma cor aleatoria
//
class UniqueColorGenerator {
  static Random random = Random();
  static Color getColor() {
    return Color.fromARGB(255, random.nextInt(255),
random.nextInt(255), random.nextInt(255));
  }
}
```



RxDart

Crie o subdiretório **lib/pages** e crie o arquivo **home.dart** no diretório **lib/pages**:

```
import 'package:flutter/material.dart';
import '../blocs/bloc.dart';
import '../utils/countbutton.dart';
import '../utils/colorbutton.dart';
import '../utils/historicbutton.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';
import '../utils/widgetD.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}
```

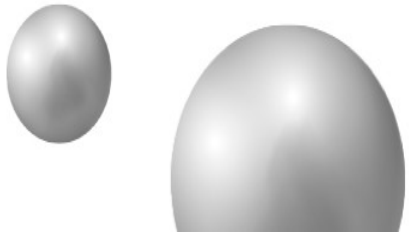
RxDart

Acrescente a classe para gerenciar o estado da tela no arquivo home.dart:

```
class _HomeState extends State {
  MyBLoC bloc = MyBLoC();

  @override
  Widget build(BuildContext context) {
    print("Criando Home");
    return Scaffold(
      appBar: AppBar(title: Text('RxDart')),
      body: Column(
        children: <Widget>[
          WidgetA(),
          WidgetB(),
          WidgetC(),
          WidgetD(),
        ],
      ),
```

RxDart

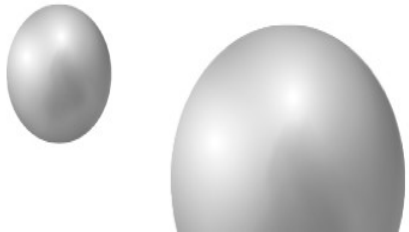
```
      bottomNavigationBar: ButtonBar(
        alignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          countButton(),
          colorButton(),
          historicButton(),
        ],
      ),
    );
  }

  @override
  dispose() {
    bloc.dispose();
  }
}
```



RxDart

Crie o subdiretório lib/utils e crie o arquivo countbutton.dart no diretório lib/utils:

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

class countButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando countButton");
    return RaisedButton(
      child: Text('Numero'),
      onPressed: () => bloc.inc(),
    );
  }
}
```



RxDart

Crie o arquivo colorbutton.dart no diretório lib/utils:

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

class colorButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando colorButton");
    return RaisedButton(
      child: Text('Cor'),
      onPressed: () => bloc.newColor(),
    );
  }
}
```

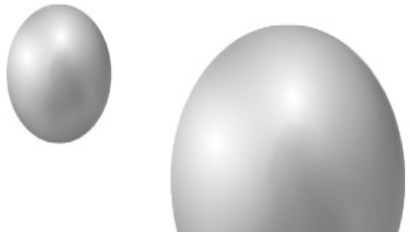
RxDart

Crie o arquivo widgetA.dart no diretório lib/utils:

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

// Widget para exibir contador
class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetA");
    return StreamBuilder<int>(
      stream: bloc.countObservable,
      initialData: bloc.count,
      builder: (context, snapshot) {
        print("Recriando texto do WindgetA");
```

RxDart

```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return ListTile(
            title: Text('Contador ${snapshot.data}'),
          );
        }
      }
    );
  }
}
```



RxDart

Crie o arquivo widgetB.dart no diretório lib/utils:

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

// Widget para exibir a cor
class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetB");
    return StreamBuilder<Color>(
      stream: bloc.colorObservable,
      initialData: bloc.color,
      builder: (context, snapshot) {
        print("Recriando ListTile do WindgetB");
```

RxDart

```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return Container(
            color: snapshot.data,
            child: ListTile(
              title: Text('Cor '),
            ),
          );
        }
      }
    );
  }
}
```

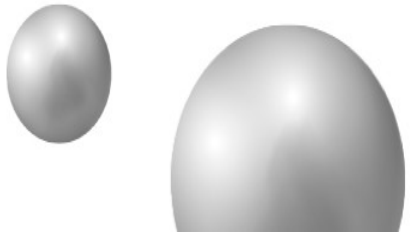
RxDart

Crie o arquivo widgetC.dart no diretório lib/utils:

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

// Widget para exibir contador
class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetC");
    return StreamBuilder<int>(
      stream: bloc.countObservable,
      initialData: bloc.count,
      builder: (context, snapshot) {
        print("Recriando texto do WindgetC");
```

RxDart

```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return ListTile(
            title: Text('Contador ${snapshot.data}'),
          );
        }
      }
    );
  }
}
```



RxDart

Crie o arquivo widgetD.dart no diretório lib/utils:

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

// Widget para exibir a cor
class WidgetD extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    MyBLoC bloc = MyBLoC();
    print("Criando WidgetD");
    return StreamBuilder<Color>(
      stream: bloc.colorObservable,
      initialData: bloc.color,
      builder: (context, snapshot) {
        print("Recriando ListTile do WindgetD");
```

RxDart

```
        if (snapshot.hasError) {
          return Text('Há um erro na Stream');
        } else {
          return Container(
            color: snapshot.data,
            child: ListTile(
              title: Text('Cor '),
            ),
          );
        }
      }
    );
  }
}
```

RxDart

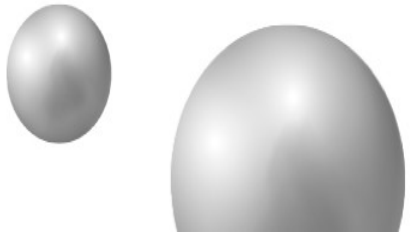
Crie o arquivo historicbutton.dart no diretório lib/utils:

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

class historicButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando historicButton");
    return RaisedButton(
      child: Text('Historico'),
      onPressed: () {
        Navigator.push(context,
          MaterialPageRoute(builder: (context) =>
Historic()),
        );
      },
    );
  }
}
```

RxDart

Crie o arquivo historic.dart no diretório lib/pages:

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


//
// Classe para exibir o historico das cores
//
class Historic extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Historico"),
      ),
      body: _body(),
    );
  }
}
```

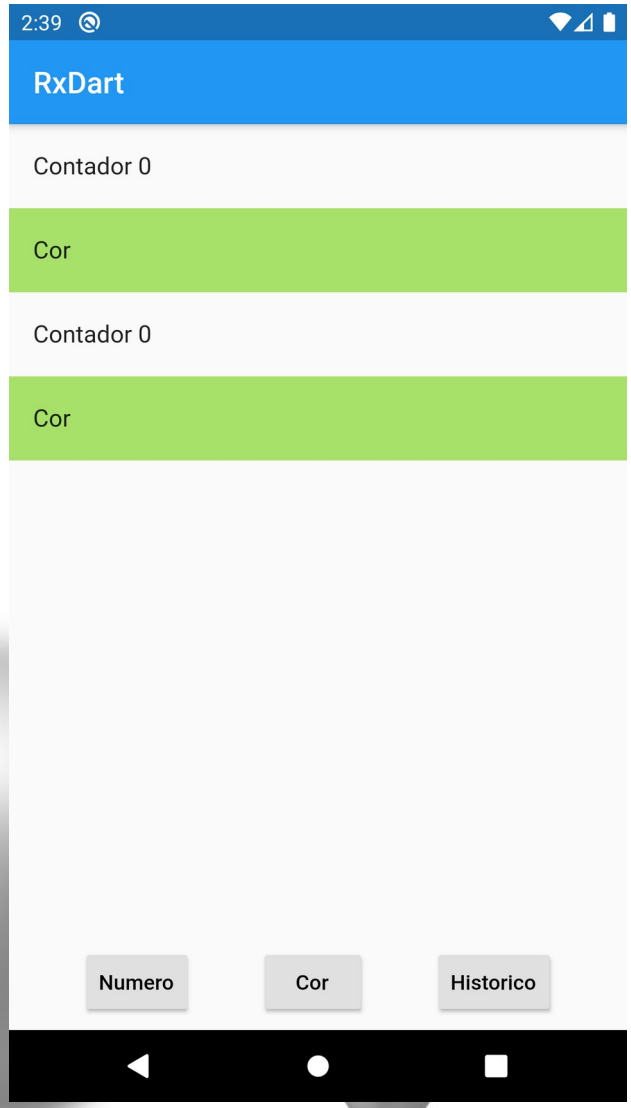


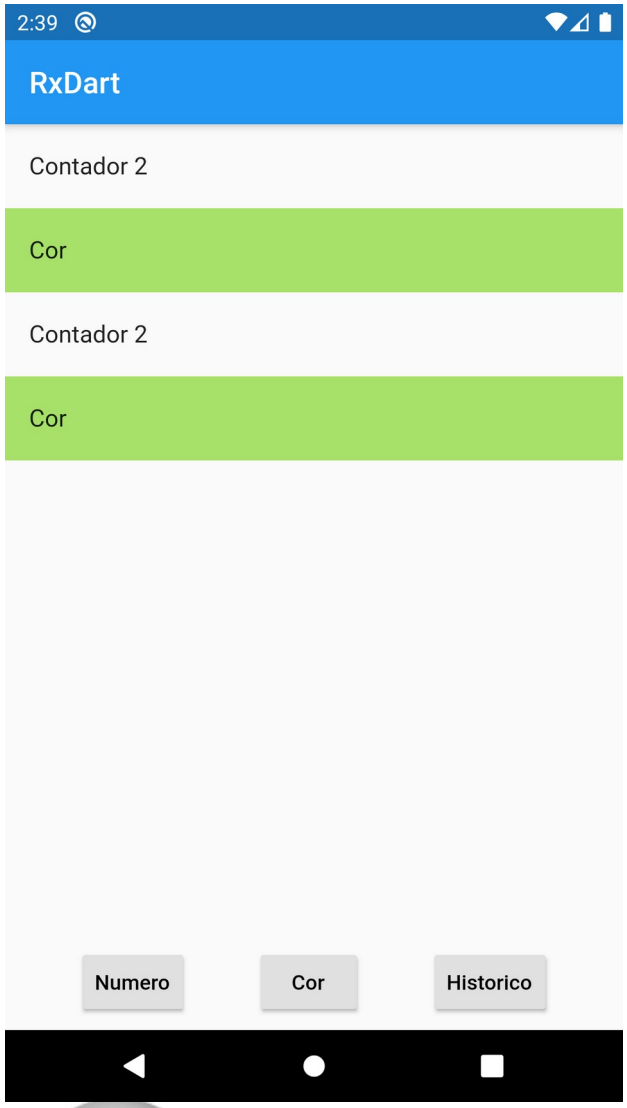
RxDart

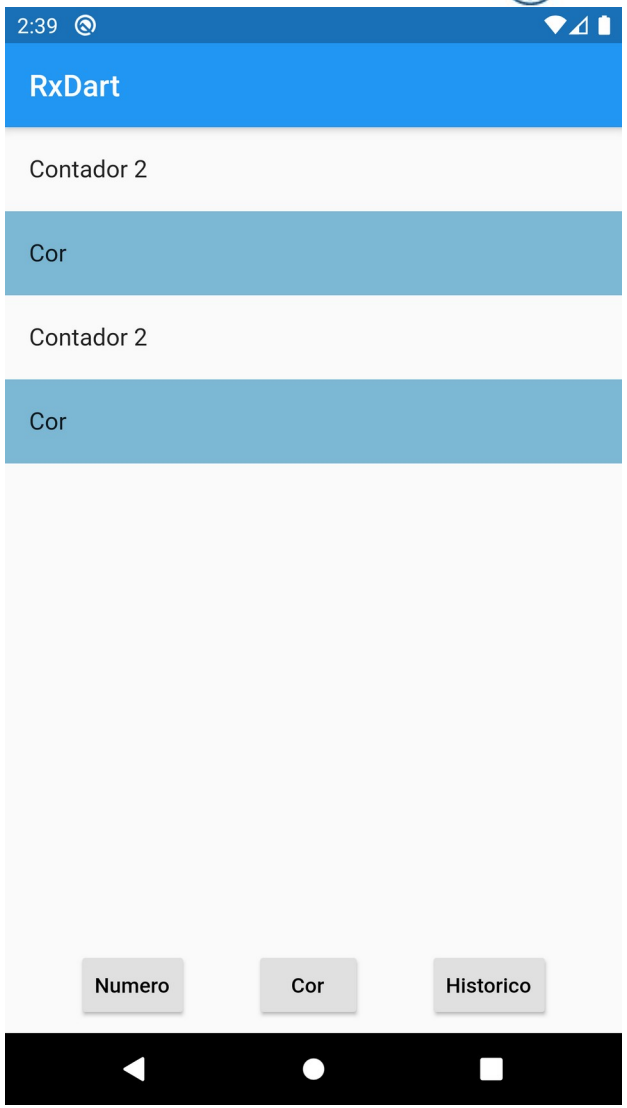
Acrescente a função para montar o corpo da tela de histórico no arquivo historic.dart:

```
// Corpo da tela
ListView _body() {
  print("Criando historico");
  MyBLoC bloc = MyBLoC();
  List<Color> _colors = bloc.colorSubject.values;
  print("colors ${_colors.length}");
  ListView ret = ListView.builder(
    itemCount: _colors.length,
    itemBuilder: (BuildContext ctxt, int index) {
      return Container(
        color: _colors[index],
        child: ListTile(
          title: Text('Cor ${index+1}'),
        ),
      );
    },
  );
  return ret;
}
```

RxDart


2:39
RxDart
Contador 0
Cor
Contador 0
Cor
Numero
Cor
Historico

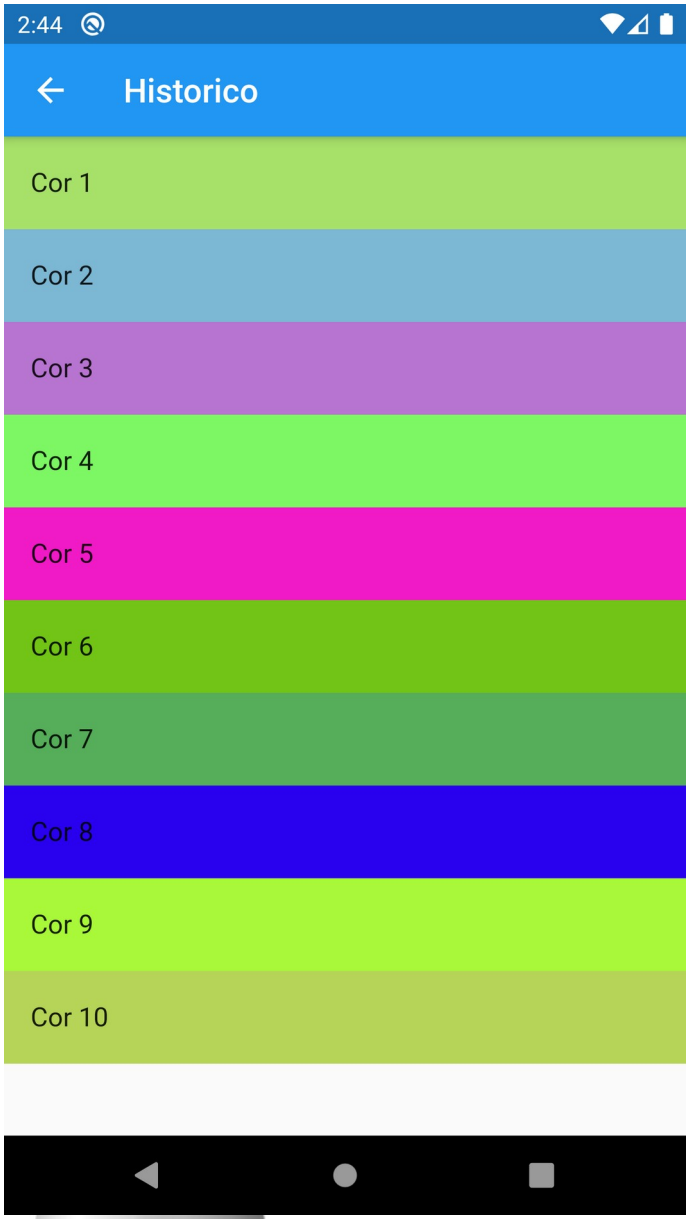

2:39
RxDart
Contador 2
Cor
Contador 2
Cor
Numero
Cor
Historico


2:39
RxDart
Contador 2
Cor
Contador 2
Cor
Numero
Cor
Historico

RxDart


I/Choreographer(893): Skipped 297 frames! The app
I/flutter (893): Criando Home
I/flutter (893): Criando WidgetA
I/flutter (893): Recriando texto do WindgetA
I/flutter (893): Criando WidgetB
I/flutter (893): Recriando ListTile do WindgetB
I/flutter (893): Criando WidgetC
I/flutter (893): Recriando texto do WindgetC
I/flutter (893): Criando WidgetD
I/flutter (893): Recriando ListTile do WindgetD
I/flu Slide 159(893): Criando countButton
I/flutter (893): Criando colorButton
I/flutter (893): Criando historicButton
I/flutter (893): Recriando texto do WindgetA
I/flutter (893): Recriando ListTile do WindgetB
I/flutter (893): Recriando texto do WindgetC
I/flutter (893): Recriando ListTile do WindgetD
I/flutter (893): Recriando texto do WindgetA
I/flutter (893): Recriando texto do WindgetC
I/flutter (893): Recriando texto do WindgetA
I/flutter (893): Recriando texto do WindgetC
I/flutter (893): Recriando ListTile do WindgetB
I/flutter (893): Recriando ListTile do WindgetD

RxDart

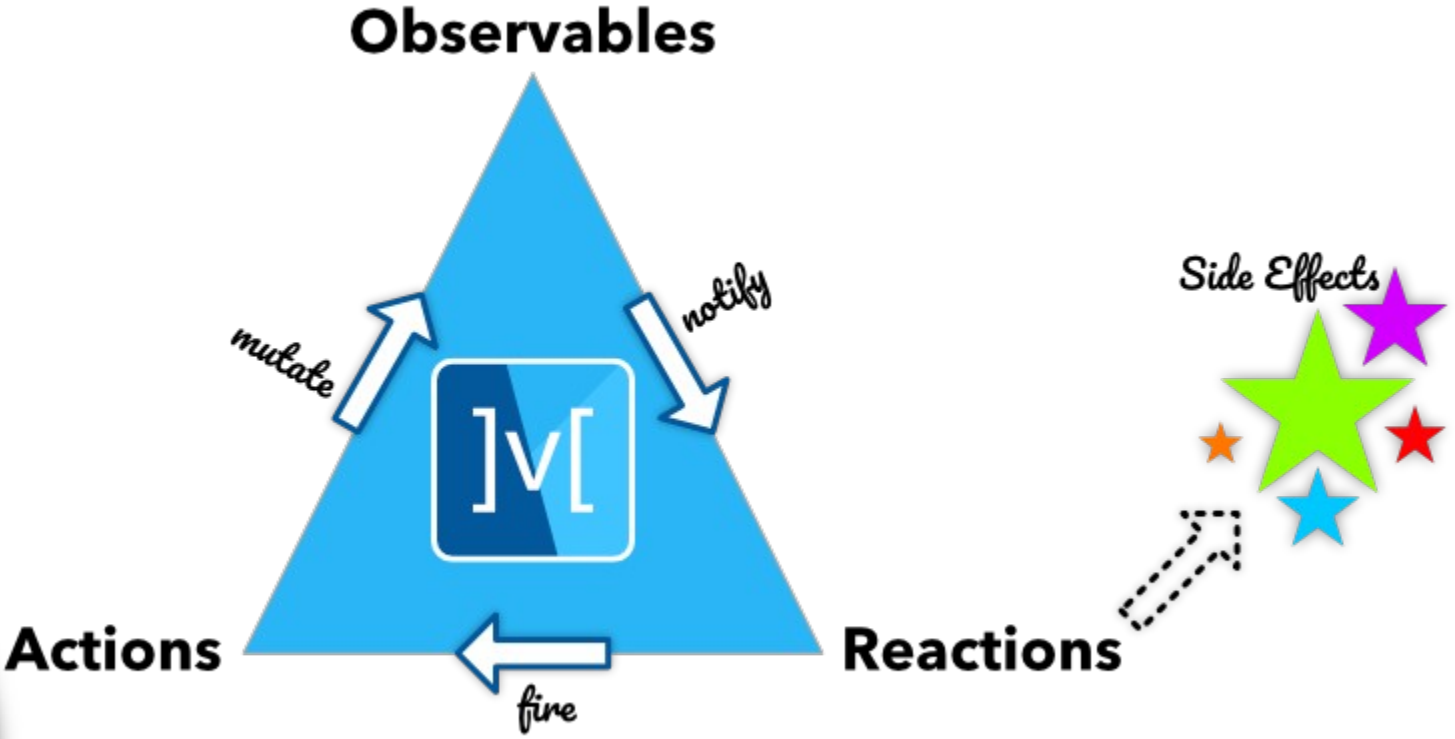

2:44
Historico
Cor 1
Cor 2
Cor 3
Cor 4
Cor 5
Cor 6
Cor 7
Cor 8
Cor 9
Cor 10

MobX

MobX é uma biblioteca para gerenciamento de estado. As propriedades do estado são chamadas observáveis (@observable) e ficam armazenadas em objetos chamados Store. Os métodos para alterar as propriedades são chamados ações (@action). É possível ter propriedades calculadas automaticamente (@computed). Quando um observável é alterado, o MobX pode dispar uma reação (reaction). Os widgets que dependem dos observáveis e devem ser recriados quando o observável é alterado são chamados observadores (Observer).

O código da Store pode ser criado manualmente ou podem ser utilizadas as marcações para que o pacote gerador de código do MobX crie o código da Store.

MobX



https://github.com/mobxjs/mobx.dart

MobX

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

Altere o arquivo **pubspec.yaml** para incluir as dependências de **flutter_mobx**, **mobx** e **provider**:

```
dependencies:
  mobx:
  flutter_mobx:
  provider: ^4.0.2
  flutter:
    sdk: flutter
```

Inclua as dependências dos pacotes para geração do código do Mobx:

```
dev_dependencies:
  build_runner: ^1.3.1
  mobx_codegen:
  flutter_test:
    sdk: flutter
```

Instale os pacotes adicionados nas dependências.

MobX

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'pages/home.dart';
import 'stores/store.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,
      ),
```

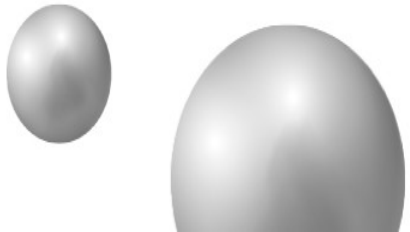
MobX

```
      home: Provider(
        create: (BuildContext context) => MyStore(),
        child: Home(),
      ),
    );
  }
}
```

MobX

Crie o subdiretório **lib/stores** e crie o arquivo **store.dart** no diretório **lib/stores**:

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

// This is our generated file (we'll see this soon!)
part 'store.g.dart';

// We expose this to be used throughout our project
class MyStore = _MyStore with _$MyStore;

// Our store class
abstract class _MyStore with Store {
}
```



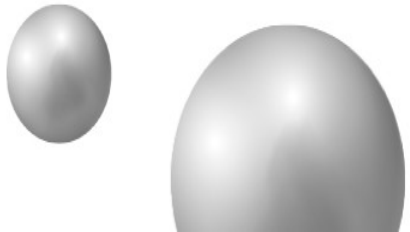
MobX

Acrescente os observáveis no arquivo store.dart:

```
@observable
int count = 0;

@observable
Color color = UniqueColorGenerator.getColor();
```

Acrescente a propriedade calculada do fatorial no arquivo store.dart:

```
@computed
int get fatorial {
  int ret = 1;
  for (int i = 1; i<= count; i++ )
    ret *= i;
  return ret;
}
```



MobX

Acrescente a ação de incrementar o contador no arquivo

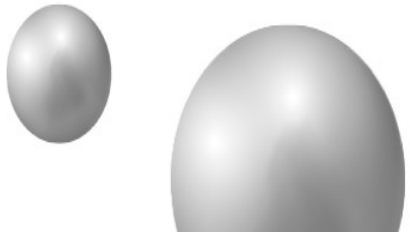
```
@action
void inc() {
  count++;
}
```

Acrescente a ação de decrementar o contador no arquivo

```
@action
void dec() {
  count--;
}
```

Acrescente a ação de mudar a cor no arquivo **store.dart**:

```
@action
void newColor() {
  color = UniqueColorGenerator.getColor();
}
```

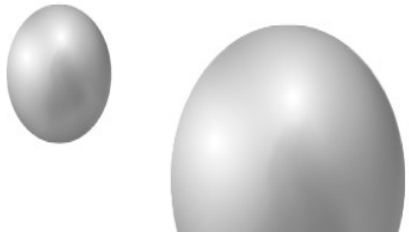


MobX

Acrescente a classe para gerar uma cor aleatória no arquivo **store.dart**:

```
//
// Classe para gerar uma cor aleatoria
//
class UniqueColorGenerator {
  static Random random = Random();
  static Color getColor() {
    return Color.fromARGB(255, random.nextInt(255),
random.nextInt(255), random.nextInt(255));
  }
}
```

Gere o código para o **MobX**.



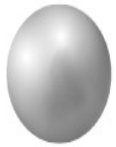
MobX

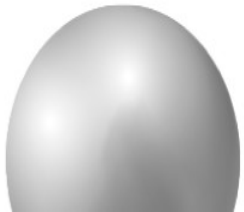
Crie o subdiretório **lib/pages** e crie o arquivo **home.dart** no diretório **lib/pages**:

```
import 'package:flutter/material.dart';
import '../utils/incbutton.dart';
import '../utils/decbutton.dart';
import '../utils/colorbutton.dart';
import '../utils/widgetA.dart';
import '../utils/widgetB.dart';
import '../utils/widgetC.dart';
import '../utils/widgetD.dart';
import '../utils/widgetE.dart';

class Home extends StatelessWidget {

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

MobX

```
      body: Column(
        children: <Widget>[
          WidgetA(),
          WidgetB(),
          WidgetC(),
          WidgetD(),
          WidgetE(),
        ],
      ),
      bottomNavigationBar: ButtonBar(
        alignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          incButton(),
          decButton(),
          colorButton(),
        ],
      ),
    );
  }
}
```

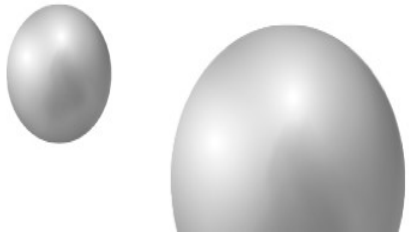




MobX

Crie o subdiretório **lib/utils** e crie o arquivo **incbutton.dart** no diretório **lib/utils**:

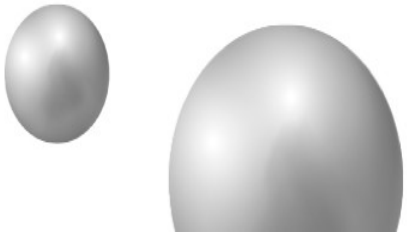
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../stores/store.dart';

class incButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final store = Provider.of<MyStore>(context);
    print("Criando incButton");
    return RaisedButton(
      child: Icon(Icons.add),
      onPressed: () => store.inc(),
    );
  }
}
```



MobX

Crie o arquivo decbutton.dart no diretório lib/utils:

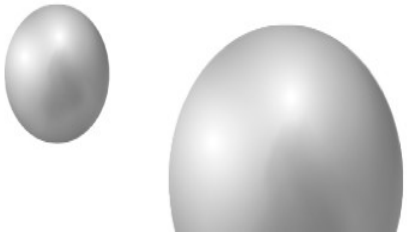
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../stores/store.dart';

class decButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final store = Provider.of<MyStore>(context);
    print("Criando decButton");
    return RaisedButton(
      child: Icon(Icons.remove),
      onPressed: () => store.dec(),
    );
  }
}
```



MobX

Crie o arquivo colorbutton.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../stores/store.dart';

class colorButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final store = Provider.of<MyStore>(context);
    print("Criando colorButton");
    return RaisedButton(
      child: Text('Cor'),
      onPressed: () => store.newColor(),
    );
  }
}
```



MobX

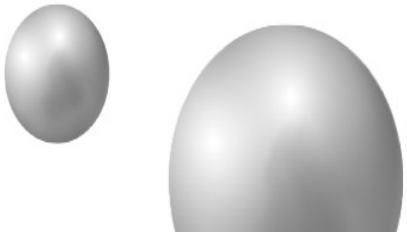
Crie o arquivo widgetA.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:mobx/mobx.dart';
import '../stores/store.dart';

class WidgetA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetA");
    final store = Provider.of<MyStore>(context);
    return Observer(
      builder: (_) {
        print("Recriando texto do WidgetA");
        return ListTile(
          title: Text('Contador ${store.count}'), );
      }
    );
  }
}
```

MobX

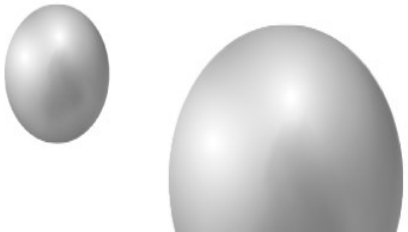
Crie o arquivo widgetB.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:mobx/mobx.dart';
import '../stores/store.dart';

class WidgetB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetB");
    final store = Provider.of<MyStore>(context);
    return Observer(
      builder: (_) {
        print("Recriando ListTile do WindgetB");
```



MobX

```
        return Container(
          color: store.color,
          child: ListTile(
            title: Text('Cor '),
          ),
        );
      }
    );
  }
}
```



MobX

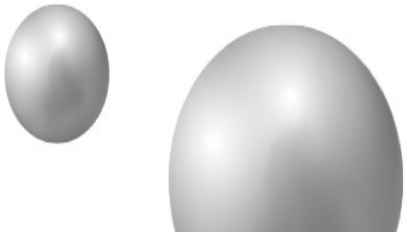
Crie o arquivo widgetC.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:mobx/mobx.dart';
import '../stores/store.dart';

class WidgetC extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetC");
    final store = Provider.of<MyStore>(context);
    return Observer(
      builder: (_) {
        print("Recriando texto do WidgetC");
        return ListTile(
          title: Text('Contador ${store.count}'), );
      }
    );
  }
}
```

MobX

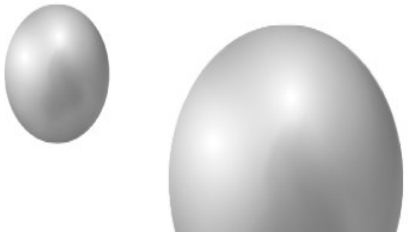
Crie o arquivo widgetD.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:mobx/mobx.dart';
import '../stores/store.dart';

class WidgetD extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetD");
    final store = Provider.of<MyStore>(context);
    return Observer(
      builder: (_) {
        print("Recriando ListTile do WindgetD");
```



MobX

```
        return Container(
          color: store.color,
          child: ListTile(
            title: Text('Cor '),
          ),
        );
      }
    );
  }
}
```

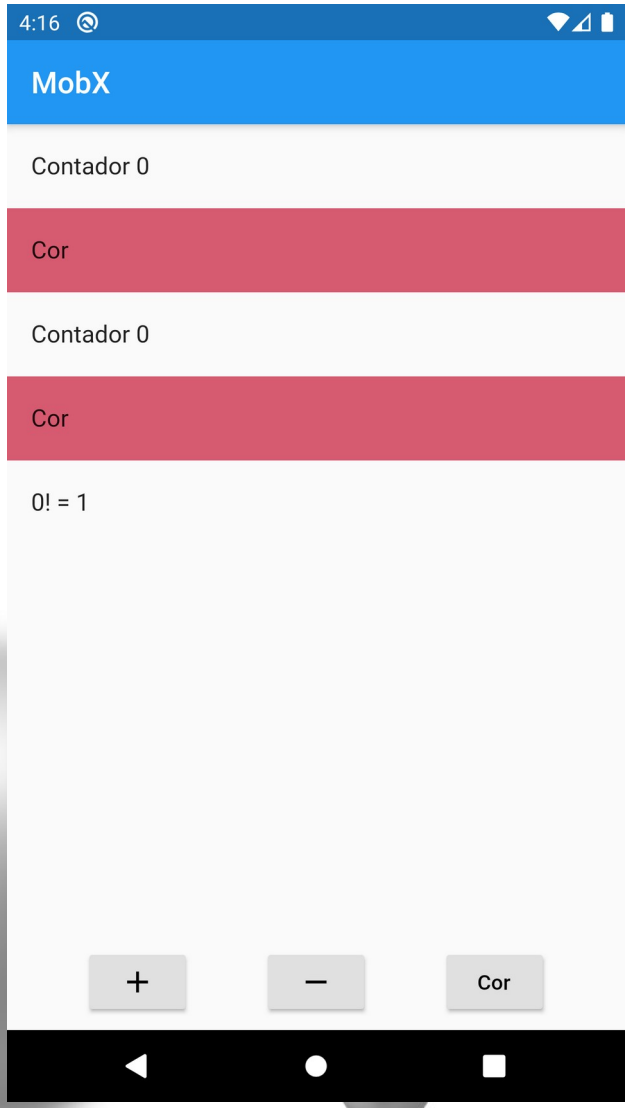


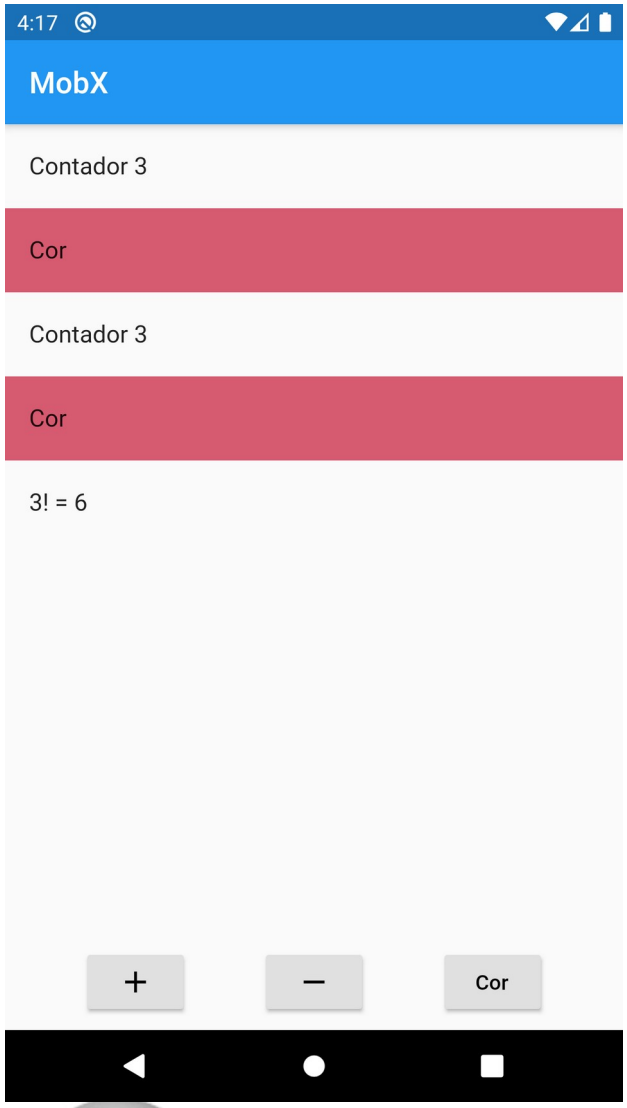
MobX

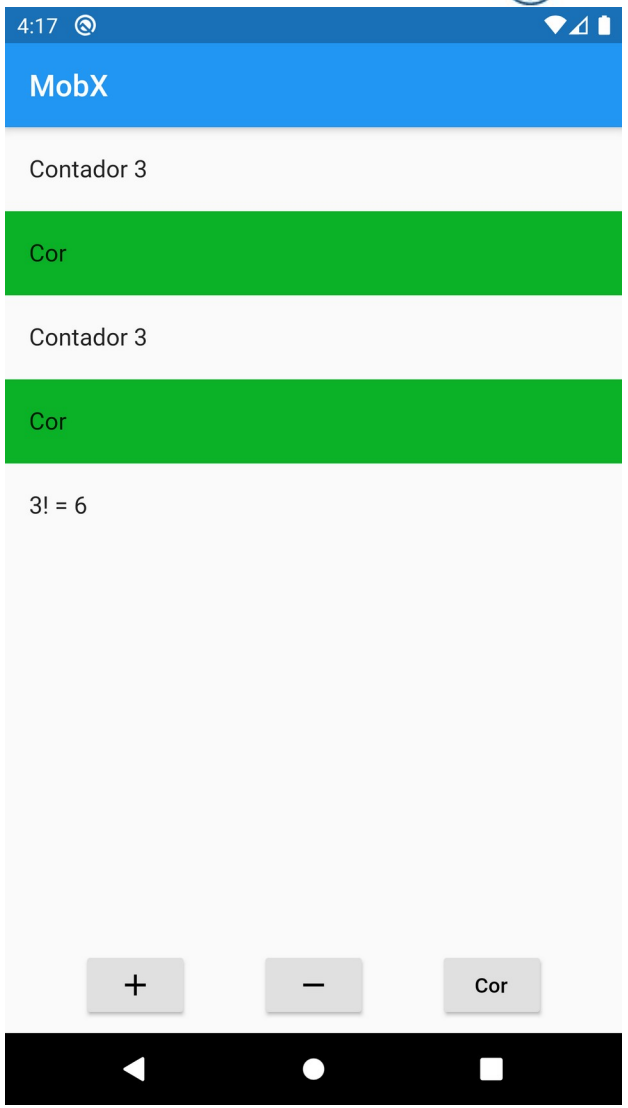
Crie o arquivo widgetE.dart no diretório lib/utils:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:mobx/mobx.dart';
import '../stores/store.dart';

class WidgetE extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("Criando WidgetE");
    final store = Provider.of<MyStore>(context);
    return Observer(
      builder: (_) {
        print("Recriando texto do WidgetE");
        return ListTile(
          title: Text('${store.count}! = $
{store.fatorial}'), );
      }
    );
}}
```

MobX


4:16
MobX
Contador 0
Cor
Contador 0
Cor
0! = 1
+
−
Cor


4:17
MobX
Contador 3
Cor
Contador 3
Cor
3! = 6
+
−
Cor


4:17
MobX
Contador 3
Cor
Contador 3
Cor
3! = 6
+
−
Cor

MobX

```
I/flutter ( 6226): Criando Home
I/flutter ( 6226): Criando WidgetA
I/flutter ( 6226): Recriando texto do WidgetA
I/flutter ( 6226): Criando WidgetB
I/flutter ( 6226): Recriando ListTile do WindgetB
I/flutter ( 6226): Criando WidgetC
I/flutter ( 6226): Recriando texto do WidgetC
I/flutter ( 6226): Criando WidgetD
I/flutter ( 6226): Recriando ListTile do WindgetD
I/flutter ( 6226): Criando WidgetE
I/flutter ( 6226): Recriando texto do WidgetE
I/flutter ( 6226): Criando incButton
I/flutter ( 6226): Criando decButton
I/flutter ( 6226): Criando colorButton
I/flutter ( 6226): Recriando texto do WidgetA
I/flutter ( 6226): Recriando texto do WidgetC
I/flutter ( 6226): Recriando texto do WidgetE
I/flutter ( 6226): Recriando texto do WidgetA
I/flutter ( 6226): Recriando texto do WidgetC
I/flutter ( 6226): Recriando texto do WidgetE
I/flutter ( 6226): Recriando texto do WidgetA
I/flutter ( 6226): Recriando texto do WidgetC
I/flutter ( 6226): Recriando texto do WidgetE
I/flutter ( 6226): Recriando ListTile do WindgetB
I/flutter ( 6226): Recriando ListTile do WindgetD
```