

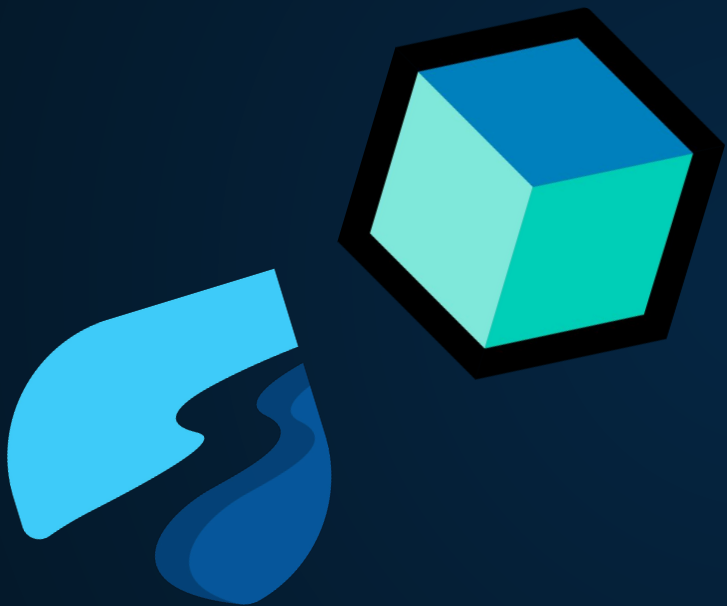


- Alioune KANOUTÉ
- Meïssa B.C MBAYE

# Plan

---

<b>Introduction: C'est quoi le state management ?</b>	<b>01</b>	<b>06</b>	<b>Présentation de la maquette et implémentation avec Riverpod</b>
<b>Riverpod: qu'est-ce que c'est ?</b>	<b>02</b>	<b>07</b>	<b>Qu'est-ce que le pattern BLoC ?</b>
<b>Relation entre Riverpod &amp; Provider</b>	<b>03</b>	<b>08</b>	<b>Les concepts derrières le pattern BLoC</b>
<b>Qu'est-ce qu'un "Provider" dans Riverpod ?</b>	<b>04</b>	<b>09</b>	<b>Flutter BLoC</b>
<b>Riverpod Workflow</b>	<b>05</b>	<b>10</b>	<b>Implémentation avec Flutter BLoC</b>




## C'est quoi le state management ?

---

Un **état(state)** est une information qui peut être lu lorsqu'un widget est créé, il peut changer ou être modifié.

Le **state management** est le processus qui permet de gérer les états tout au long du cycle de vie d'une application.



Riverpod, késako ?

**Qu'est ce que Riverpod ?**

# Riverpod: qu'est ce que c'est ?

---

Framework réactif de gestion d'état et I.D

Plus qu'un "state management system"

Évolution et successeur de Provider

Créé et maintenu par [Remi Rousselet](#)



# Relation entre Riverpod & Provider ?

---

Riverpod est un anagramme de Provider :)

Réécriture complète de Provider

Core concepts de Provider toujours présent

Aide à établir une bonne architecture (s'il est bien utilisé)



# Qu'est ce qu'un "Provider" dans Riverpod ?

- Objet qui encapsule un état
- Possibilité d'écouter les changements de cet état
- Les "Providers" sont au coeur de tout dans Riverpod
  - ✓ Remplace certains **designs patterns** (DI, Singleton, SL)
  - ✓ Facile de stocker/accéder au state, "n'importe où"
  - ✓ Permet d'optimiser les performances de votre app
  - ✓ Rend votre code plus testable

# Qu'est ce qu'un "Provider" dans Riverpod ?

Riverpod propose plusieurs types de **Providers** :

Provider Type	Provider Create Function	Example Use Case
<code>Provider</code>	Returns any type	A service class / computed property (filtered list)
<code>StateProvider</code>	Returns any type	A filter condition / simple state object
<code>FutureProvider</code>	Returns a Future of any type	A result from an API call
<code>StreamProvider</code>	Returns a Stream of any type	A stream of results from an API
<code>StateNotifierProvider</code>	Returns a subclass of StateNotifier	A complex state object that is immutable except through an interface
<code>ChangeNotifierProvider</code>	Returns a subclass of ChangeNotifier	A complex state object that requires mutability



# Riverpod Workflow



```
# By @the_it_dev
dependencies:
  flutter:
    sdk: flutter
  flutter_riverpod: ^2.2.0
```

Installation



```
// By @the_it_dev
void main() {
  // Wrap the entire app with a ProviderScope so that widgets
  // will be able to read/manipulate providers
  runApp(
    ProviderScope(
      child: MyApp(),
    ),
  );
}
```

“Injection” de Riverpod

# Riverpod Workflow



```
// By @the_it_dev

// Provider that returns a string value
final helloWorldProvider = Provider<String>((ref) {
  return 'Hello world';
});
```

**Création d'un Provider**



```
// By @the_it_dev
class HelloWorldWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text(
      /* How to read the provider value here? */,
    );
  }
}
```

**Comment obtenir la valeur  
de ce provider ?**

# Riverpod Workflow

```
// By @the_it_dev

final helloWorldProvider = Provider<String>((_) => 'Hello world');

// 1. Widget class now extends [ConsumerWidget]
class HelloWorldWidget extends ConsumerWidget {
  @override
  // 2. Build method has an extra [WidgetRef] argument
  Widget build(BuildContext context, WidgetRef ref) {
    // 3. Use ref.watch() to get the value of the provider
    final helloWorld = ref.watch(helloWorldProvider);
    return Text(helloWorld);
  }
}
```

Lecture d'un Provider  
Part.1

```
// By @the_it_dev

final helloWorldProvider = Provider<String>((_) => 'Hello world');

class HelloWorldWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      // 1. Add a Consumer
      body: Consumer(
        // 2. Specify the builder and obtain a WidgetRef
        builder: (_, WidgetRef ref, __) {
          // 3. Use ref.watch() to get the value of the provider
          final helloWorld = ref.watch(helloWorldProvider);
          return Text(helloWorld);
        },
      ),
    );
  }
}
```

Lecture d'un Provider  
Part.2

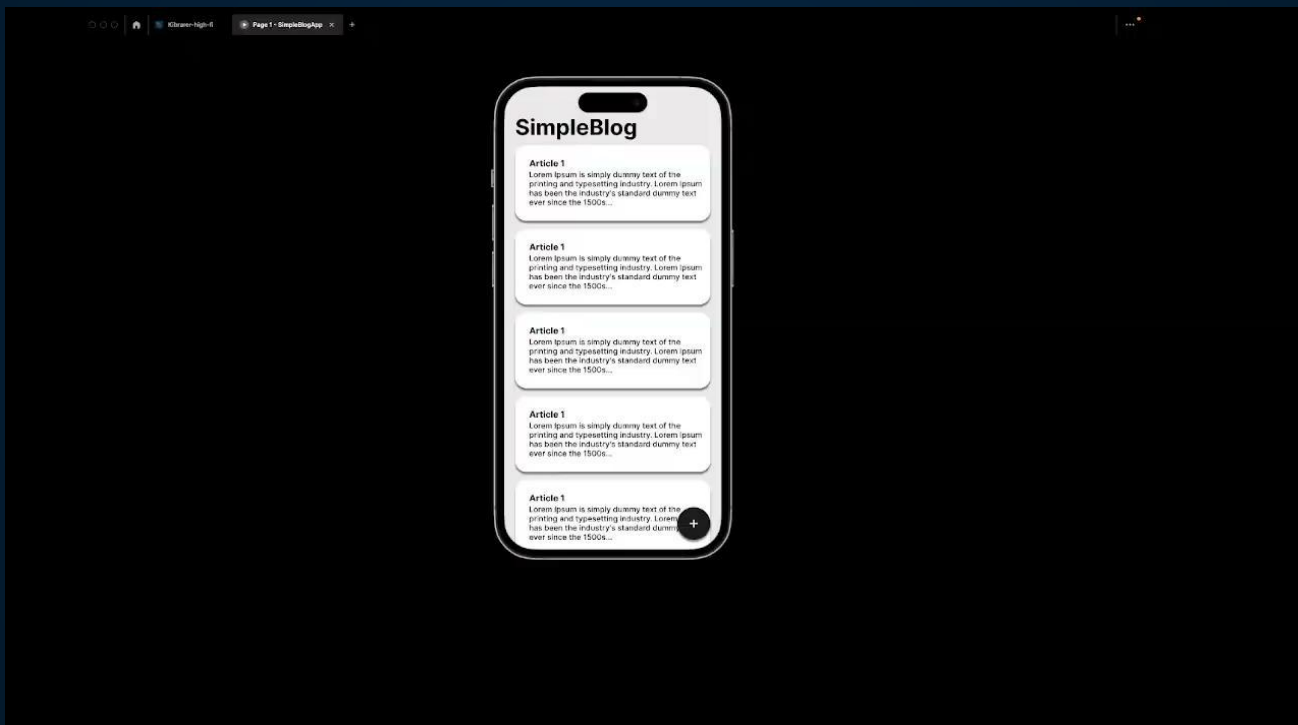


Présentation de la maquette et du  
Use Case

&

**Implémentation avec Riverpod**

# Maquette



# Maquette

## SimpleBlog

### Article 1

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s...

### Article 1

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s...

### Article 1

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s...

### Article 1

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s...

### Article 1

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s...



## Article 1

Ecrit par John Doe

21 Novembre 2022

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it?

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Where does it come from?

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

Where can I get some?

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some



## SimpleBlog

### Ajouter un article

Titre

Contenu

AJOUTER



# Qu'est-ce que le pattern BLoC?

---

Patron de conception créé par Google

Permet de séparer la logique de la partie UI d'une application

Pour la mise en œuvre de ce patron de conception sur Flutter, il y a la librairie [flutter\\_bloc](#) qui est maintenue par [Felix Angelo](#).



# Les concepts derrière le pattern BLoC: Stream

BLoC est basé sur les Stream

Basiquement, un Stream est une rivière qui envoie des données de manière asynchrone d'un expéditeur à un receveur.

Sauf que ce dernier doit écouter sur le Stream, pour savoir quand le bateau va arriver.





```
1 Stream<int> streamOfInt() async* {
2   for(int i = 1; i <= 10; i++) {
3     await Future.delayed(Duration(seconds: 3));
4
5     yield i;
6   }
7 }
8
9 void main() {
10  Stream<int> stream = streamOfInt();
11
12  stream.listen((event) {
13    print("Current value $event");
14  });
15 }
16
```

# Exemple: Stream

Ceci est une fonction qui retourne un **Stream** de Integer.

Elle effectue une boucle de 1 à 10, et attend 3 secondes avant de renvoyer la valeur.

**async\* = asynchronous generation** function, cela veut dire que la fonction retourne des données asynchrones.

L'utilisation de **await** indique l'attente de l'exécution du processus avant de passer à l'instruction suivante.

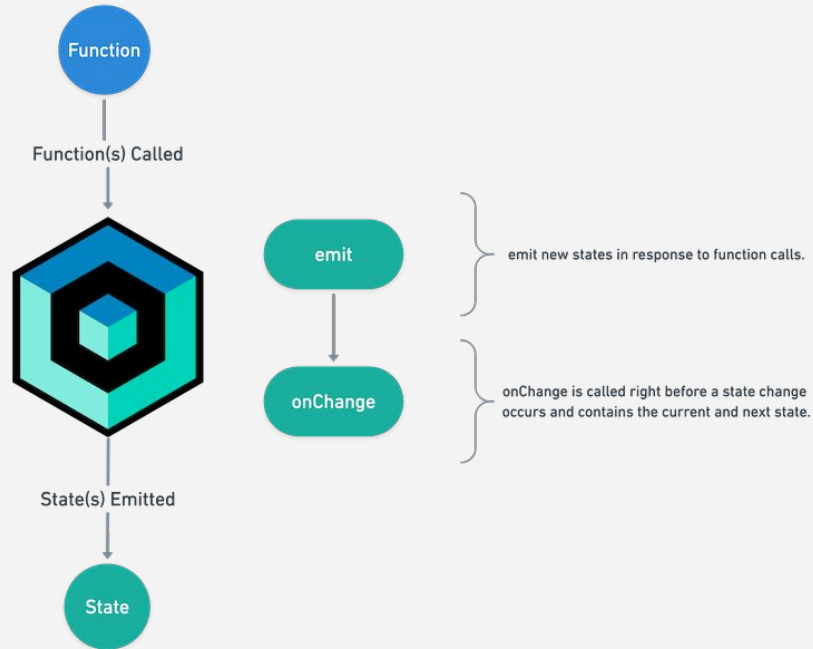
Le mot clé **yield** permet d'envoyer la nouvelle donnée sur le Stream.

Le **Stream** contient une méthode **listen**, qui permet d'écouter sur le **Stream** et récupérer les données.

# Les concepts derrière le pattern BLoC: Cubit

- Cubit peut être vu comme une version minimale de BLoC
- On dit que Cubit est un type spécial du composant Stream, basé sur l'appel des fonctions depuis le UI, ces fonctions reconstruisent le UI en émettant différents états sur le Stream.
- Par contre, les fonctions d'un Cubit ne font pas partie d'un Stream, c'est une liste prédéfinie de ce qui peut être fait dans le Cubit.
- Le seul flux existant dans un Cubit est un flux d'états vides.

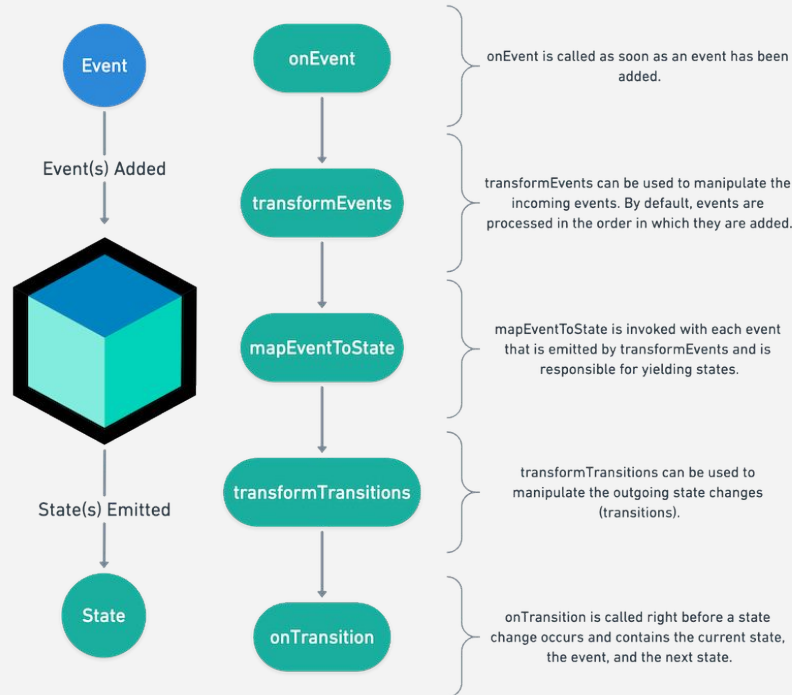
# Cubit: Workflow



# Cubit: Example

```
1 // counter_cubit
2
3 class CounterCubit extends Cubit<int> {
4   CounterCubit(): super(0);
5
6   void increment() => emit(state + 1);
7   void decrement() => emit(state - 1);
8 }
9
10 void main() {
11   final counterCubit = CounterCubit();
12   print(counterCubit.state);
13
14   counterCubit.increment();
15   print(counterCubit.state);
16   counterCubit.increment();
17   print(counterCubit.state);
18   counterCubit.decrement();
19   print(counterCubit.state);
20   counterCubit.decrement();
21   print(counterCubit.state);
22
23 }
24
25 //0
26 //1
27 //2
28 //1
```

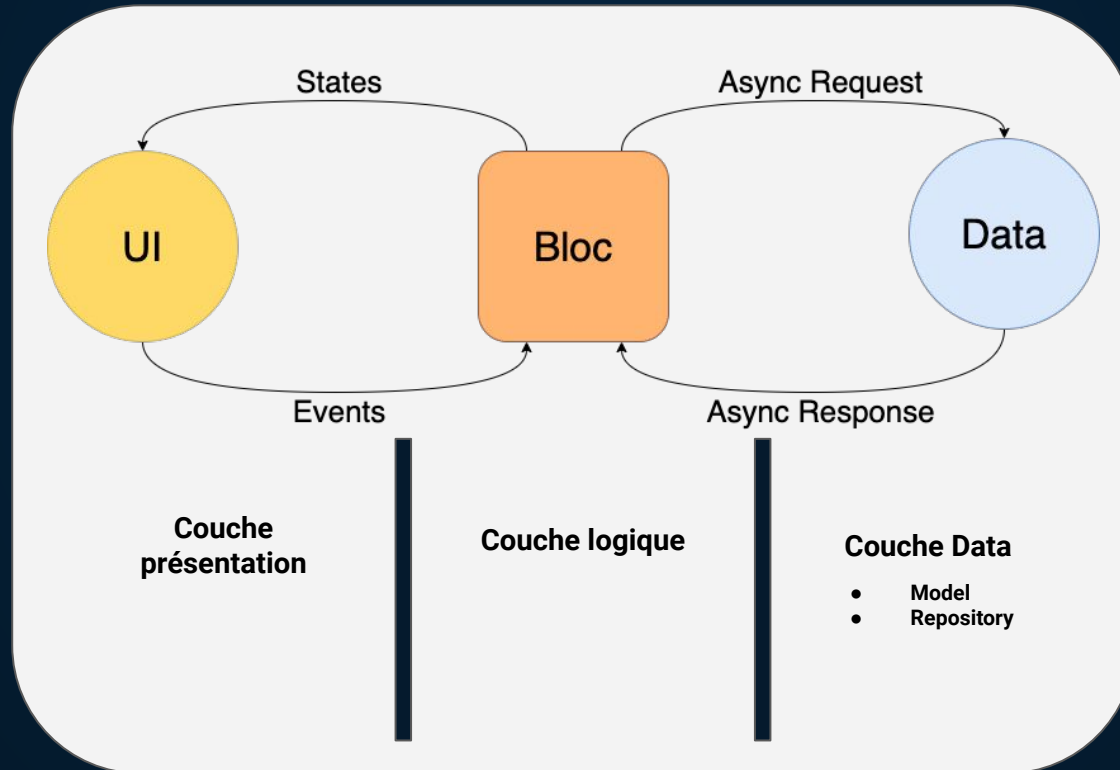
# BLoC: Workflow



# Flutter BLoC

- Cette librairie contient un ensemble de classes, qui permet la mise en œuvre du pattern BLoC dans un projet Flutter.
- **BlocProvider**: permet d'injecter une **seule** instance d'un BLoC dans un widget et ses descendants.
- **BlocBuilder**: permet de mettre à jour le UI en fonction du changement d'état du BLoC.
- **BlocListener**: permet d'écouter le changement d'état d'un BLoC, il est appelé une fois par état, l'état initial étant exclu.
- **BlocConsumer**: combinaison de **BlocBuilder** et **BlocListener**

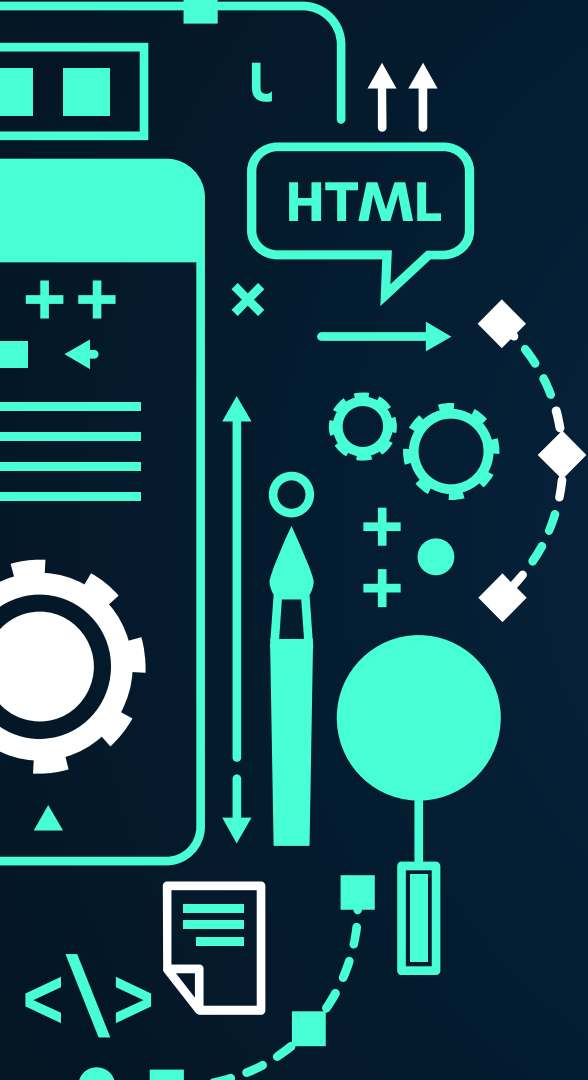
# BLoC: Architecture



A decorative graphic on a dark blue background. It features a large, light blue gear on the left side. In the top left corner, there are two horizontal lines, one light blue and one white. In the top center, there is a small white gear with a light blue circular outline. In the top right corner, there are several horizontal lines in alternating light blue and white colors. In the bottom right corner, there is a small white gear with a light blue circular outline. A large, solid light blue rectangle is positioned in the center of the slide.

**Implémentation avec BLoC**





# THANKS!

Does anyone have any question?