**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

**---oOo---**

Logo

Description automatically generated

**TÀI LIỆU SEMINAR**

**BLoC trong Flutter**

Lớp: 19\_3

Môn học: Phát triển ứng dụng cho thiết bị di động nâng cao

Thành viên: 19120061 – Ngô Trọng Đức

19120383 – Huỳnh Tấn Thọ

19120426 – Phan Đặng Diễm Uyên

Giảng viên: Phạm Hoàng Hải

**Thành phố Hồ Chí Minh, tháng 11 năm 2022**

**MỤC LỤC**

[1. Khái niệm 1](#_Toc120100278)

[1.1. 1](#_Toc120100279)

[1.2. 1](#_Toc120100280)

[2. Tại sao, khi nào 1](#_Toc120100281)

[2.1. 1](#_Toc120100282)

[3. Ứng dụng của BloC trong Flutter 1](#_Toc120100283)

[3.1. Tính hiệu quả 1](#_Toc120100284)

[3.2. Mức độ phổ biến 1](#_Toc120100285)

[4. Các thư viện về BloC mà Flutter hỗ trợ 1](#_Toc120100286)

[TÀI LIỆU THAM KHẢO 2](#_Toc120100287)

# Khái niệm

## 

## 

# Tại sao, khi nào dùng BLoC

## 

# Ứng dụng của BLoC trong Flutter

## Tính hiệu quả

### Tách giao diện ra khỏi business logic

* Một trở ngại khi lập trình di động sử dụng Flutter là Flutter không dùng một ngôn ngữ trung gian nào dành cho việc dựng giao diện. Không như Android có XML, Flutter dựng giao diện người dùng ngay bên trong code.
* Do đó việc tách giao diện ra khỏi business logic (quy trình nghiệp vụ ứng dụng) là rất cần thiết. BloC hỗ trợ chúng ta đạt được mục tiêu đó theo cách hiệu quả nhất:
  + Sử dụng Stream để quản lý và lan truyền các thay đổi về State
  + Tách giao diện ra khỏi business logic, khiến chúng độc lập với nhau, từ đó giải quyết vấn đề của việc thiếu ngôn ngữ trung gian cho giao diện

### Quản lý State của ứng dụng hiệu quả hơn

* BloC không phải cách duy nhất để quản lý State ứng dụng. Flutter cung cấp sẵn cho ta một số cách để quản lý State. Tuy nhiên mỗi cách trong số đó đều tồn tại các nhược điểm khác nhau
  + Dùng setState: toàn bộ widget tree đều phải được rebuild dù cho chỉ có duy nhất widget thay đổi state, nó cũng không giúp ích trong việc tách business logic ra khỏi giao diện

|  |
| --- |
| setState(() { myState = newValue }); |

* + Dùng InheritedWidget: các State được khai báo là final, do đó chúng không thể thay đổi về sau. Ngoài ra, nó không có các phương thức dispose để giải phóng tài nguyên, có thể gây rò rỉ bộ nhớ.

|  |
| --- |
| class FrogColor extends InheritedWidget {  const FrogColor({  super.key,  required this.color,  required super.child,  });  final Color color;  static FrogColor of(BuildContext context) {  final FrogColor? result = context.dependOnInheritedWidgetOfExactType<FrogColor>();  assert(result != null, 'No FrogColor found in context');  return result!;  }  @override  bool updateShouldNotify(FrogColor old) => color != old.color;  } |

### Giúp ứng dụng dễ scale hơn

* Các thành phần trên giao diện không liên lạc hay trao đổi trực tiếp với nhau, mà chúng chỉ có thể liên lạc trực tiếp với BLoC
* Do đó, ứng dụng sẽ rất dễ scale thêm các tính năng mới mà không cần viết lại code, tái cấu trúc dự án, hay làm giảm performance

### Dễ kiểm thử

* Việc sử dụng BLoC giúp giảm chi phí, thời gian, công sức khi thực hiện kiểm thử. Ta có thể dễ dàng dự đoán và kiểm tra được các state của ứng dụng, nhờ UI và business logic đã tách ra.s
* Khi có sự thay đổi về state, BLoC sẽ thông báo đến các widget lắng nghe nó trên toàn ứng dụng để thực hiện việc rebuild giao diện. Ta không cần kiểm tra từng widget riêng biệt.

## Mức độ phổ biến

* Trừ những ứng dụng cực kỳ đơn giản, chỉ có từ 3 màn hình trở xuống. Còn lại đa phần các ứng dụng đều dùng BLoC
* Do BLoC là một design patten, nên nó không bị ràng buộc về ngôn ngữ, framework. Không chỉ dùng được trên Flutter mà còn cả AngularDart và React

## Ưu, nhược điểm của việc sử dụng BLoC

### Ưu điểm

* Dễ dàng tách giao diện ra khỏi business logic
* Dễ kiểm thử, tái sử dụng code
* Performance tốt, tăng khả năng scale của ứng dụng

### Nhược điểm

* Chỉ có hiệu quả rõ rệt khi dùng trong các ứng dụng lớn
* Phụ thuộc rất nhiều vào Stream, do đó code có thể bị boilerplate

# Các thư viện về BLoC mà Flutter hỗ trợ

## bloc

## flutter\_bloc

# TÀI LIỆU THAM KHẢO

1. BLoC pattern in Flutter | FlutterDevs

<https://medium.flutterdevs.com/bloc-pattern-in-flutter-part-1-flutterdevs-128f90059f5c>

1. State management in Flutter using the BLoC design pattern

<https://blog.logrocket.com/state-management-flutter-bloc-pattern/>

1. How to Implement the BLoC Architecture in Flutter: Benefits and Best Practices

<https://www.mobindustry.net/blog/how-to-implement-the-bloc-architecture-in-flutter-benefits-and-best-practices/>

1. Getx Vs Bloc Which Is Faster With Code Examples

<https://www.folkstalk.com/2022/09/getx-vs-bloc-which-is-faster-with-code-examples.html>

1. Top 5 Benefits of Using BLoC Pattern

<https://www.mitrais.com/news-updates/top-5-benefits-of-using-bloc-pattern/>

1. Flutter State Management: What to Choose- Provider, BLoC, or Redux?

<https://medium.com/dhiwise/flutter-state-management-what-to-choose-provider-bloc-or-redux-214160adbae4>