**BLoC Hay MVVM – Đâu là “Chân Lý” cho phát triển dự án bằng Flutter?**

Trước khi đi vào tìm kiếm và so sánh giữa các Kiến trúc khi được triển khai trên Flutter – Dart để xem Kiến trúc nào sẽ phù hợp, tối ưu, thuận tiện, dễ triển khai … hơn thì mình xin phép kể về hành trình và lý do mình viết bài viết này để chia sẻ và mong muốn nhận được góp ý của mọi người.

Với nền tảng đã làm Android/IOS Native nên Team mình tự tin nhận 1 dự án A phát triển bằng Flutter – Dart để vừa làm và vừa học. Do Team mình ít người và cũng không có ai có kinh nghiệm với Flutter trước đó nên team chọn hướng tiếp cận theo phong cách Free Style. Sau khi hoàn thành dự án A, mình cũng nhận thấy rất nhiều nhược điểm của Free Style ảnh hưởng trực tiếp đến chất lượng sản phẩm, trải nghiệm người dùng, khả năng fix bug và maintain …. Nên mình mới tìm kiếm 1 Kiến trúc nào đó tối ưu và phù hợp với Flutter. Mình tìm thấy 2 gợi ý được nhiều người nhắc đến: Bloc, MVVM (Nếu còn cái nào hay hơn nhờ mọi người góp ý giúp mình nhé :D ).

Ở bài viết này mình sẽ cùng nghiên cứu về BLoC và GetX + MVVM để xem cái nào tốt hơn.

Design Pattern:

* BLoC Pattern (Business Logic Component) mục đích là tách code business logic ra khỏi UI thay vì code gộp chung cả logic và UI vô cùng 1 file, để sau này spec mới có yêu cầu sửa code business logic hay sửa UI sẽ dễ dàng sửa hơn.
* MVVM (Model – View – View Model) xây dựng một mô hình mà View Model tương tác với Model để cung cấp dữ liệu cho View.

Sau khi đọc xong lý thuyết thì mình thấy MVVM và BLoC khá giống nhau về bản chất (đều không sử lý logic trong view) vậy chúng khác nhau ở điểm nào ?

Mình có làm sample và tìm đọc 1 số bài so sánh giữa BLoC & MVVM + GetX thì thấy điểm mấu chốt khác nhau chủ yếu là Tương tác giữa BLoC/Controller và View:

* BLoC và View giao tiếp với nhau thông qua các State và Event. BLoC không để lộ bất kỳ phương thức nào ra ngoài.

BLoC thông báo cập nhật cho View qua emit()

emit(SuccessState(data));

View thông báo cập nhật event cho BLoC qua add()

BlocProvider.*of*<LoginBloc>(context).add(LoginUsernameChanged(text)

* Controller và View giao tiếp trực qua instance của Controller

View đăng kí lắng nghe thay đổi của Controller để update View

Obx(() => titleHome(context))

View thông báo cập nhật event cho BLoC qua api của Controller

controller.setUserName(value);

Một điểm khác biệt thú vị như BLoC giao tiếp thông qua các Event không chỉ với UI mà còn với các phần khác nhau của ứng dụng. ví dụ: nó sẽ nhận được một Event sau khi nhận được thông báo về firebase hoặc khi data trong DB thay đổi.

Nhân tiên, Mình cũng có đọc và nghe mọi người bảo GetX chỉ nên dùng với những dự án nhỏ, làm nhanh còn những dự án lớn hơn cần maintain liên tục thì nên sử dụng BLoC. Mình cũng đã tìm hiểu và thấy 1 số lý do sau khiến 1 số người ngại sử dụng GetX:

* GetX chứa rất nhiều thứ như State Management, DI, Navigate, Internationalization, Change Theme, Network, Database … chính vì thế nó lại là nhược điểm vì lắm ứng dụng bạn không cần nhiều module đến thế và chúng cũng dễ bị out of date
* GetX sử dụng Static Context
* Khó khăn trong việc UnitTest
* Module get\_connect hỗ trợ cả API REST và GraphQL trong khi một dự án thự tế không cần nhiều như thế
* Thỉnh thoảng sẽ gặp vấn đề với Hot Reload do cơ chế Dependency Injection bên trong của GetX dẫn đến việc GetXController có thể bị remove khi chúng t Hot Reload
* Nhiều Code không có hoặc thiếu tài liệu
* Tỉ lệ unit test và widget testing thấp

=> Sau khi nghiên cứu và demo BLoC (sử dụng Flutter Bloc Plugin) và MVVM + GetX thì thực sự có cảm tình với BLoC hơn, nhưng nếu để nên bàn cân mình cũng không thấy giữa MVVM và BLoC có gì khác biệt nhiều.

Mình có để link chia sẻ về Demo Base BLoC tại đây

State Management: flutter\_bloc

Dependence Injection: GetIt

Network: Dio, Retrofit

Mock Network: http\_mock\_adapter

Database: floor

Router: auto\_route

Nếu mình có nhận xét cái gì sai hoặc thiếu sót thì nhờ mn góp ý nhé.