**Chủ đề:** Tìm hiểu về các mô hình kiến trúc trong phát triển trong ứng dụng mobile

**Giới thiệu:**

1. **Model - View - Controller (MVC Pattern)**

**1.1 Mô tả**

Mô hình MVC tách biệt ra thành 3 thành phần chính bao gồm

**- Model (M - Model ) :** Đây là thành phần chịu trách nhiệm cho xử lý dữ liệu và logic kinh doanh của ứng dụng. Model là nơi lưu trữ thông tin, thực hiện các thao tác cập nhật và truy vấn dữ liệu, mà không quan tâm đến cách dữ liệu được hiển thị hoặc tương tác với người dùng.

**- View (V - View) :** View là thành phần hiển thị giao diện người dùng và tương tác với người dùng, đảm nhận trách nhiệm hiển thị thông tin nhận được từ controller. View không có logic kinh doanh và chỉ chịu trách nhiệm về việc hiển thị dữ liệu.

**- Controller (C - Controller) :** Controller là thành phần điều phối và xử lý sự kiện từ người dùng. Nó nhận lệnh từ người dùng thông qua View, sau đó cập nhật Model dựa trên những thay đổi này và điều hướng hiển thị trở lại View. Controller giữ vai trò là điều phối viên giữa View và Model

**1.2 Ưu nhược điểm**

**Ưu điểm**

**- Tách biệt logic:**

MVC giúp tách biệt logic ứng dụng thành các phần riêng biệt làm cho mã nguồn dễ hiểu và dễ bảo trì.

**- Tái sử dụng mã nguồn:**

Các thành phần trong MVC có thể được tái sử dụng một cách linh hoạt. Model và View có thể được sử dụng lại trong nhiều phần của ứng dụng hoặc trong các ứng dụng khác nhau.

- **Dễ mở rộng:**

Do sự tách biệt giữa các thành phần, việc mở rộng và thay đổi ứng dụng trở nên dễ dàng hơn. Một thành phần có thể được thay thế hoặc mở rộng mà không làm ảnh hưởng đến các thành phần khác.

**- Kiểm thử dễ dàng:**

Bạn có thể kiểm thử mỗi thành phần một cách độc lập, giảm rủi ro gây ảnh hưởng lẫn nhau.

**Nhược điểm**

**- Không phù hợp với các ứng dụng nhỏ:**

Đối với các ứng dụng nhỏ hoặc đơn giản, việc triển khai mô hình MVC làm tăng độ phức tạp của mã nguồn khiến nó trở nên khó hiểu và tốn thời gian.

**- Cần thời gian để tìm hiểu:**

Đối với bất kỳ mô hình kiến trúc nào bạn cũng phải bỏ thời gian ra tìm hiểu để có thể áp dụng nó 1 cách đúng đắn

1. **Mô hình Model - View - Presenter (MVP Pattern)**
   1. **Mô tả**

- MVP là một mô hình kiến trúc gần giống như MVC. MVP có nhiều điểm được kế thừa từ MVC nhưng Controller được thay bằng Presenter. Mô hình này gồm có 3 phần chính, bao gồm:

**- Model (M - Model ) :** Cũng giống như MVC Model là thành phần chịu trách nhiệm cho xử lý dữ liệu và logic kinh doanh của ứng dụng

**- View (V - View) :** View là thành phần hiển thị giao diện người dùng và tương tác với người dùng.

**- Presenter (P - Presenter):** Presenter có nhiệm vụ xử lý tất cả các sự kiện UI thay cho View. Nó sẽ xử lý các dữ liệu của người dùng qua Model và trả về kết quả ở View .Không giống như View và Controller, Presenter và View hoàn toàn tách rời nhau và giao tiếp với nhau thông qua một Interface.

* 1. **Ưu nhược điểm**

**Ưu điểm**

- **Dễ hiểu, dễ dùng**

Cấu trúc code rõ ràng hơn so với MVC.

**- Dễ dàng để viết unit test cho Presenter**

Vì nó hoạt động độc lập với View và không gắn với bất cứ API nào của Android

**- Giảm khối lượng công việc phải làm**

Giảm chi phí phát sinh ban đầu

**Nhược điểm**

- Nhược điểm lớn nhất của mô hình MVP là theo thời gian, Presenter sẽ dần lớn lên do bị thêm các business logic rải rác. Người dùng sẽ rất khó để kiểm soát và chia nhỏ code khi Presenter đã quá lớn.

1. **Mô hình Model View ViewModel (MVVM Pattern)**
   1. **Mô tả**

Là mô hình hỗ trợ two-way data binding (ràng buộc dữ liệu) giữa View và ViewModel gồm các thành phần chính sau:

**- Model**

Giống như mô hình MVP và MVC, Model trong MVVM là thành phần chịu trách nhiệm cho xử lý dữ liệu và logic kinh doanh của ứng dụng

**- View**

View là thành phần hiển thị giao diện người dùng và tương tác với người dùng.

**- View Model**

Đây là lớp trung gian giữa View và Model và xem như thành phần thay thế cho Controller trong mô hình MVC.

* 1. **Ưu nhược điểm**

**Ưu điểm**

**- Tách biệt logic và giao diện:** MVVM giúp phân tách rõ ràng các phần của ứng dụng làm cho mã nguồn dễ bảo trì hơn

**- Hỗ trợ khả năng mở rộng:** MVVM giúp dễ dàng mở rộng ứng dụng bằng cách thêm các ViewModel mới hoặc mở rộng các Model hiện có mà không làm ảnh hưởng đến các phần còn lại của ứng dụng.

**- Tái sử dụng mã nguồn:** Các thành phần trong MVVM có thể được tái sử dụng một cách linh hoạt.

**Nhược điểm**

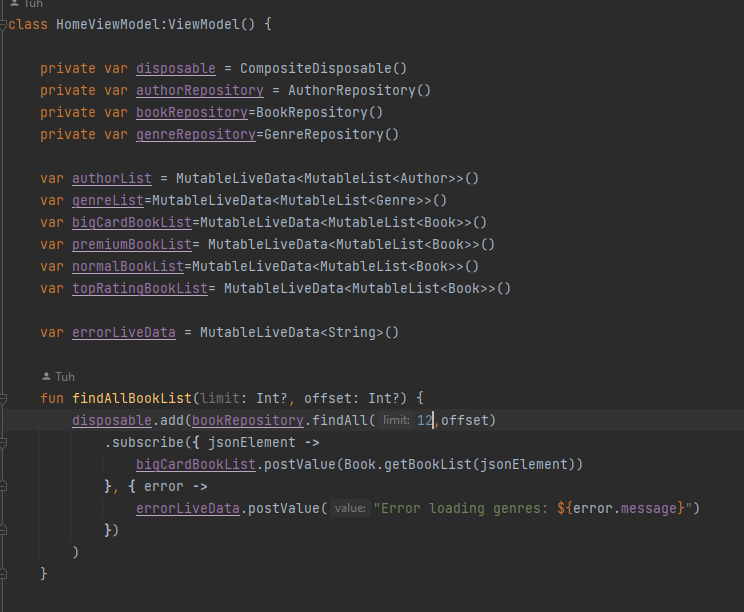
– MVVM không phù hợp với các dự án nhỏ vì khá cồng kềnh, tốn kém và mất thời gian.

– Khả năng liên kết dữ liệu của tất cả các thành phần khiến sẽ khiến cho cơ sở dữ liệu của hệ thống trở nên phức tạp, gây khó khăn trong quá trình debug.

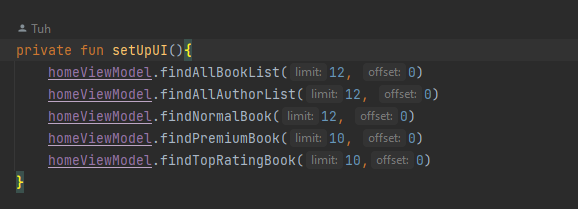
**Sử dụng**

- Hiện tại em đang sử dụng mô hình mvvm để phát triển ứng dụng đọc sách vì em thấy nó thuận lợi cho quá trình phát triển khi muốn truy cập dữ liệu từ các thành phần view khác nhau trong hệ thống

**ViewModel**



**Fragment**





**Mô tả chi tiết**

Ở HomeViewModel em sử dụng các biến observable như mutable live data bigCardBookList để có quan sát dữ liệu khi có sự thay đổi

Ở FragmentHome em tiến hành call api từ viewmodel để lấy dữ liệu danh sách book

Khi việc lấy danh sách được thực hiện thành công em quan sát dữ liệu bookList và tiến hành thực hiện các hành động cần thiết