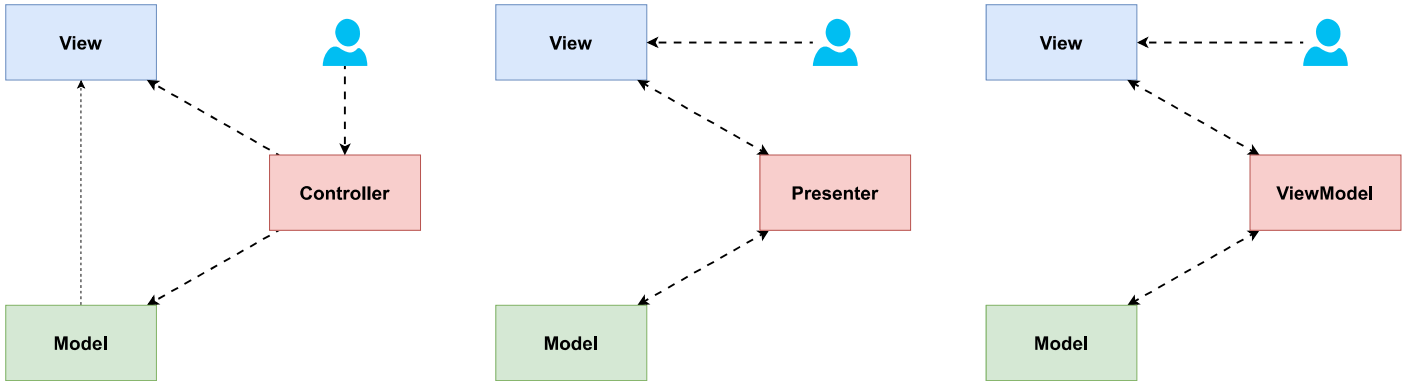
KIẾN TRÚC MVC, MVP, MVVM

MVC, MVP, và MVVM là 3 mô hình thông dụng khi phát triển phần mềm.



Model View Controller (MVC)

Mô hình MVC chia ứng dụng ra thành 3 thành phần chính: `Model`, `View` và `Controller`.

Model

`Model` nghĩa là các dữ liệu cần thiết để hiển thị ở `View`. `Model` đại diện cho một tập hợp các lớp mô tả business logic (business model và data model). Nó cũng định nghĩa các business rules cho dữ liệu (nghĩa là cách mà dữ liệu thay đổi và được dùng)

View

`View` đại diện cho các thành phần giao diện như XML, HTML. `View` sẽ hiển thị dữ liệu đã qua xử lý từ `Controller`. `Model` và `View` tương tác với nhau qua Observer pattern.

Controller

`Controller` có trách nhiệm xử lý các yêu cầu (request) được gửi đến. Nó sẽ xử lý các dữ liệu của người dùng qua `Model` và trả về kết quả ở `View`

Mặt hạn chế của Controller

- *Khả năng kiểm thử*: `Controller` bị ràng buộc với các thành phần khác nên sẽ khó để thực hiện unit test.
- *Tính linh hoạt*: `Controller` liên quan khá chặt chẽ với các view. Nếu chúng ta thay đổi `View` chúng ta sẽ phải thay đổi lại ở `Controller`.
- *Khả năng duy trì*: Qua thời gian, `Controller` sẽ ngày càng phình to ra do việc thêm code dẫn đến việc khó kiểm soát.

Model View Presenter (MVP)

Mô hình MVP cũng gần giống với mô hình MVC. Nó được kế thừa từ mô hình MVC, trong đó Controller được thay thế bởi Presenter. Mô hình này chia ứng dụng thành 3 phần chính: Model, View và Presenter

Model

`Model` đại diện cho một tập hợp các lớp mô tả business logic (business model và data model). Nó cũng định nghĩa các business rules cho dữ liệu (nghĩa là cách mà dữ liệu thay đổi và được dùng)

View

`View` là thành phần tương tác trực tiếp với người dùng. Nó không bao gồm bất kỳ việc xử lý logic nào.

Presenter

`Presenter` sẽ nhận yêu cầu của người dùng thông qua `View`, rồi xử lý dữ liệu của người dùng với sự trợ giúp của `Model` và trả kết quả về `View`. `Presenter` giao tiếp với `View` qua interface. Interface được định nghĩa trong lớp `Presenter`.

Trong cấu trúc MVP, presenter thao túng model và cập nhật ở view. View và Presenter tách biệt với nhau hoàn toàn và giao tiếp với nhau qua thông qua interface. Vì nếu tách riêng từng phần ở view sẽ dễ dàng cho việc kiểm thử ứng dụng ở MVP hơn so với mô hình MVC.

MVP sẽ rõ ràng hơn so với MVC. Chúng ta có thể dễ dàng viết unit test cho presenter vì nó không gắn với bất cứ view và API nào của Android và nó cũng cho phép chúng ta làm việc với các view khác miễn là view đó implement interface liên kết.

Mặt hạn chế của Presenter

- *Khả năng duy trì*: Cũng giống như `Controller`, `Presenter` dễ dàng bị thêm các business logic rải rác qua thời gian. Các nhà phát triển sẽ rất khó để chia nhỏ `Presenter` khi đã quá lớn.

Model View ViewModel (MVVM)

Mô hình MVVM hỗ trợ two-way data binding giữa View và View-Model. Điều này cho phép tự động lan truyền sự thay đổi, trong state của View-Model đến View. Tổng quan, View-Model sử dụng mô hình obsever để thông báo sự thay đổi trong View-Model đến Model.

ViewModel

Nó có trách nhiệm phơi bày các phương thức, lệnh, và các properties khác giúp duy trì state của view, vận dụng model như là kết quả của các hành động ở view, và trigger các sự kiện ở chính bản thân view. View có reference đến View-Model nhưng View-Model không có thông tin gì về View. Có mối quan hệ many-to-one giữa View và View-Model nghĩa là nhiều View có thể được map với một View-Model. View hoàn toàn độc lập.

Bi-directional data binding hay two-way data binding giữa View và View-Model đảm bảo rằng các models và properties ở View-Model được đồng bộ với View. Mô hình MVVM sẽ phù hợp với những ứng dụng cần được hỗ trợ bi-directional data binding.