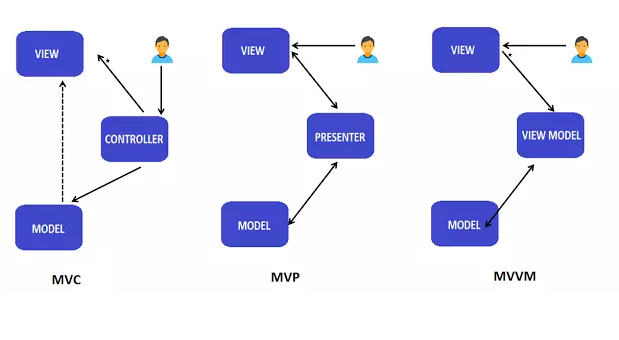
# MVC – MVVM – MVP

## Tổng quan:

// 1 slide

MVC, MVP, và MVVM là một số mô hình chung để hướng dẫn lập trình hướng tới việc tạo ra các giải pháp chuyên môn hoá. Các thành phần phổ biến trong MVC, MVP và MVVM là:

* 1. Data Layer / Business Logic (Model): Đây là thành phần chứa các business logic thao tác trên dữ liệu của ứng dụng.
* 2. Presentation Layer / UI ( View ): Đảm nhận việc hiển thị thông tin, tương tác với người dùng
* 3. Application Logic ( Controller, Presentation or View Model ): Giữ nhiệm vụ nhận điều hướng các yêu cầu từ người dùng và gọi đúng những phương thức xử lý chúng



## MVC (Model – View – Controller)

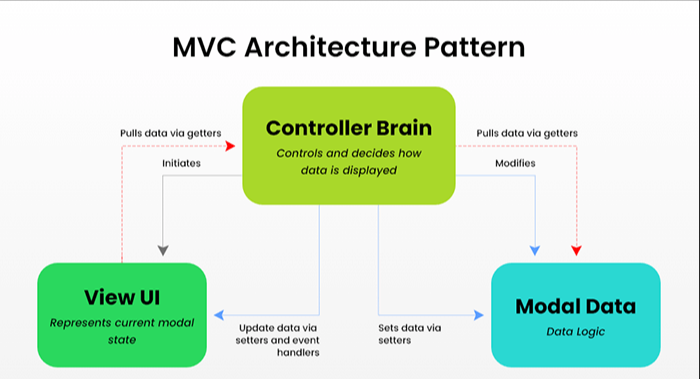
// làm 1 slide cũng được, kh thì chỉ để nói lúc thuyết trình thôi cũng được => đồng nhất các mục trong 3 phần là được

1. Giới thiệu :

MVC là viết tắt của Model View Controller – một mô hình thiết kế được sử dụng rộng rãi trong những phần mềm. Mô hình được thiết kế bởi Trygve Reenskaug vào năm 1979. Trong thời gian đầu mô hình này được gọi là View Controller Editor. Cho đến 1994, MVC trở nên phổ biến.

1. Thành phần: // làm ppt mấy phần highlight xám + ảnh minh họa thôi : 1 slide

* Model (M) – quản lý xử lí các dữ liệu, chịu trách nhiệm chính cho các logic nghiệp vụ (business logic) của ứng dụng và quản lý các trạng thái của ứng dụng. Có thể nói rõ chức năng của Model là đọc, ghi dữ liệu trạng thái của ứng dụng một cách liên tục, và có thể xử lý các tác vụ liên quan đến dữ liệu như mạng và xác thực dữ liệu.
* View (V) – hiển thị dữ liệu cho người dùng và xử lý tương tác
* Controller (C) – điều khiển sự tương tác giữa Model và View: xử lý sự kiện từ View và cập nhật Model tương ứng



1. Cách thức hoạt động: // 1 slide

+ Đầu tiên, thông qua View, người dùng sẽ đưa ra các yêu cầu (user request).

+ Controller sau đó ghi nhận những yêu cầu này và tiếp tục gửi chúng đến Model để tiến hành xử lý.

+ Khi dữ liệu đã được xử lý xong, Controller sẽ thông báo cho View biết.

+ Và View sẽ trực tiếp lấy dữ liệu từ Model để hiển thị trên giao diện UI với các định dạng trực quan như HTML, XML, JSON,… để người dùng có thể hiểu được.

* Như vậy, qua mô hình hoạt động trên, ta có thể thấy phương thức (method) xử lý user request được quyết định bởi Controller. Phương thức xử lý dữ liệu (UI logic) để hiển thị trên View thì được Model quyết định. Còn View thì chỉ có chức năng tiếp nhận yêu cầu người dùng và hiển thị dữ liệu cuối cùng mà thôi.

// ưu – nhược chung 1 slide, có thể chia 2 cột

1. Ưu điểm:

* Nhẹ, tiết kiệm tài nguyên (băng thông)
* Kết cấu đơn giản.
* Dễ dàng kiểm tra và phát hiện lỗi.
* Dễ dàng phân tách các phần Model và View.

1. Nhược điểm:

* Bên cạnh những ưu điểm MVC mang lại thì nó cũng có một số nhược điểm cần khắc phục.
* MVC đa phần phù hợp với công ty chuyên về website hoặc các dự án lớn thì mô hình này phù hợp hơn so với với các dự án nhỏ, lẻ vì khá là cồng kềnh và mất thời gian.
* Khó triển khai

## MVVM (Model- View – ViewModel)

1. Giới thiệu
   * MVVM viết tắt là Model View ViewModel – được phát triển bởi Microsoft và sử dụng cho Windows Presentation Foundation (WPF) và Silverlight. Đến 2005, được John Grossman công bố bằng một bài blog về Avalon (mật mã của Windows Presentation Foundation).
   * Mô hình MVVM phát triển dựa trên cả MVC và MVP, MVVM được phát triển để tách biệt giao diện UI khỏi logic nghiệp vụ và hành vi trong một ứng dụng. Dữ liệu được ràng buộc giữa ViewModel và View thông qua các cơ chế binding.

A diagram of a presentation

Description automatically generated

1. Thành phần

* Model (M) – tương tự như MVC, Model bao gồm dữ liệu cơ bản để chạy được phần mềm.
* View (V) – là một giao diện đồ hoạ giữa người dùng và mô hình tương tự MVC. Dùng để hiển thị kết quả sau khi xử lý dữ liệu.

View-Model (VM) – Là lớp trung gian của Model và View, chứa logic xử lý và giữ dữ liệu mà View cần.

1. Cách thức hoạt động

* Ở mô hình MVVM, ViewModel là nơi thực hiện hầu hết tất cả các business logic, quản lý methods, properties khác để duy trì viewstate. Điều này đồng nghĩa với việc View trở thành một ánh xạ hiển thị trạng thái của ViewModel tại mọi thời điểm.
* Trong MVVM, các tầng bên dưới sẽ không biết được thông tin, dữ liệu của tầng bên trên. Điều này giúp đem đến sự tách biệt rõ ràng giữa giao diện người dùng và ứng dụng logic.

1. Ưu điểm // làm slide mấy phần gạch chân thôi

* Đơn Giản Hóa Unit Testing:

Người dùng có thể thực hiện unit testing mà không phụ thuộc vào View. Điều này làm tăng khả năng kiểm thử và duy trì code dễ dàng hơn.

* Khả Năng Test Mà Không Cần Mockup:

Trong MVVM, không cần phải tạo mockup như MVP. Chỉ cần xác nhận biến observable thích hợp, giúp giảm độ phức tạp của quá trình kiểm thử.

* Sử Dụng XAML Cho View:

MVVM sử dụng XAML cho View, giúp dễ dàng chỉnh sửa giao diện mà không gây ảnh hưởng đến code logic. Điều này tăng tính tương tác giữa nhóm phát triển và thiết kế giao diện.

* Phân Rõ Ràng Ba Phần:

Kiến trúc MVVM phân rất rõ ràng thành ba phần: Model, View, và ViewModel. Điều này làm cho việc lập trình, kiểm thử, và sửa lỗi code trở nên dễ dàng hơn.

1. Nhược điểm

* Rủi Ro Tăng Dần Của Logic Trong View:

Khi gán nhiều biến vào View, các logic có thể rải rác và tăng dần theo thời gian, gây khó khăn cho việc kiểm soát code và duy trì.

* Hao Tốn Tài Nguyên Bộ Nhớ Với Data Binding Hai Chiều:

Data binding hai chiều có thể gây hao tốn tài nguyên bộ nhớ, đặc biệt là khi có nhiều dữ liệu được liên kết và theo dõi.

## MVP (Model – View – Presenter)

A diagram of a model

Description automatically generated

1. Giới thiệu

* Mô hình MVP cũng gần giống với mô hình MVC. Nó được kế thừa từ mô hình MVC, trong đó Controller được thay thế bới Presenter.

1. Thành phần

* Model: đại diện cho tập hợp các lớp mô tả business logic. Nó cũng định nghĩa các business rules cho dữ liệu (nghĩa là cách mà dữ liệu thay đổi và được dùng)
* View: là thành phần tương tác trực tiếp với người dùng như XML, Activity, fragments. Nó không bao gồm bất kỳ việc xử lý logic nào.
* + Presenter : sẽ nhận input 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(với cái nó cần truyền dữ liệu).

1. Cách thức hoạt động:

* Luồng đi của mô hình này như sau:

+ View nhận tương tác của người dùng.

+ Sau đó, yêu cầu được truyền đến Presenter và xử lý thông qua Model.

+ Khi Model thực hiện thay đổi, xử lý dữ liệu xong, Presenter lấy dữ liệu từ Model để truyền đến cho View.

* Trong mô hình MVP, View là tầng duy nhất tương tác với người dùng (khác với MVC, cả 2 tầng View và Controller đều có thể ghi nhận tương tác).

1. Ưu điểm

* Cấu trúc rõ ràng và trực quan hơn MVC.
* Dễ dàng để viết unit test cho Presenter vì MVP hoạt động độc lập với View và không gắn với bất cứ API nào của Android.

1. Nhược điểm

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

## So sánh :

* **Điểm chung // 1 slide**

+ Điểm chung của cả 3 mô hình trên là chúng đều có mục đích: tách biệt các chức năng trực quan hoá (Visualization), xử lý (Processing) và quản lý dữ liệu (Data) cho các ứng dụng UI.

+ Cả 3 mô hình trên đều nhắm đến mục tiêu:

* Modularity: khả năng phân chia module.
* Flexibility: sự nhanh nhẹn trong quá trình xử lý.
* Testability: khả năng kiểm tra.
* Maintainability: khả năng bảo trì.
* **Điểm khác biệt: // có thể kẻ bảng gồm 2 cột và 5 dòng để trong 1 slide**

1. Đánh giá hiệu suất

* Đối với ứng dụng đơn giản và trung bình, MVC có thể cung cấp hiệu suất tốt.
* MVP là có độ tin cậy cao nhất và ít trở ngại khi hiển thị hình ảnh.
* MVVM khi thực hiện các tác vụ phức tạp sẽ phải tải rất nhiều dữ liệu gây ảnh hưởng đến hiệu suất.

1. Đánh giá mức độ tương thích

* MVVM có độ tương thích cao nhất do tính liên kết dữ liệu tạo ra tác động tích cực.
* MVP hoạt động tốt hơn MVC, vì dù gì MVP là mô hình nâng cấp của MVC.

1. Đánh giá khả năng bảo trì

* Hai mô hình MVP và MVVM có mức độ ổn định cao hơn và ít cần phải bảo trì hơn, việc bảo trì cũng dễ dàng hơn.
* Mô hình MVC cần phải thường xuyên thay đổi và cập nhật trong phần lớn trường hợp sử dụng.

1. References (tham chiếu)
   * "Tham chiếu" trong trường hợp này có nghĩa là quyền truy cập hoặc kết nối giữa các thành phần khác nhau trong kiến trúc. // không cần bỏ vào slide

* Trong MVC, View không có khả tham chiếu đến Controller.
* Trong MVP, View có thể tham chiếu đến Presenter.
* Trong MVVM, View tham chiếu đến View-Model.

1. Entry point (Điểm nhập)
   * // phần khái niệm này kh cần bỏ vào slide
   * Điểm nhập là nơi mà ứng dụng bắt đầu thực thi. Nó là điểm mà quá trình chạy của chương trình bắt đầu thực hiện mã máy. Đối với ứng dụng desktop hoặc web, điểm nhập thường được đặt trong hàm main của chương trình hoặc trong một lớp đặc biệt.

* Điểm nhập của MVC là Controller.
* Điểm nhập của MVP và MVVM là View.

## Kết luận:

// 1 hay 2 slide gì cũng được

* **MVC, MVP và MVVM phù hợp sử dụng khi nào?**
* MVC phù hợp khi việc kết nối giữa View và các phần còn lại của ứng dụng phải luôn sẵn có, ví dụ như web API chẳng hạn.
* MVP phù hợp khi bạn không thể binding dữ liệu qua DataContext, ví dụ như Windows Forms.
* MVVM phù hợp khi bạn có thể binding dữ liệu thông qua DataContext, ví dụ như: Windows Presentation Foundation, javascript sử dụng Knockout.
* **Giữa MVC, MVP và MVVM, mô hình nào là tốt nhất?**
* Rất khó để trả lời câu hỏi này, vì câu trả lời phụ thuộc vào loại phần mềm, ứng dụng bạn muốn xây dựng. MVC sẽ phù hợp cho các phần mềm đơn giản. MVP và MVVM lại phù hợp cho những dự án có mức độ phức tạp cao.

// 1 slide

### Tài liệu tham khảo:

* [Sự khác biệt của các design pattern MVC, MVP và MVVM (viblo.asia)](https://viblo.asia/p/su-khac-biet-cua-cac-design-pattern-mvc-mvp-va-mvvm-DXOkRzQVkdZ)
* [MVC MVP và MVVM là gì? So sánh ưu và nhược điểm 3 mô hình (svnhostingcomparison.com)](https://www.svnhostingcomparison.com/mvc-mvp-va-mvvm-la-gi/)
* [MVC là gì? Ứng dụng của mô hình MVC trong lập trình (vietnix.vn)](https://vietnix.vn/tim-hieu-mo-hinh-mvc-la-gi/)
* [Một chút về MVC, MVP và MVVM (viblo.asia)](https://viblo.asia/p/mot-chut-ve-mvc-mvp-va-mvvm-3P0lPyMv5ox)
* [MVC, MVP, MVVM là gì? Thông tin cần biết về các mô hình lập trình - Trung tâm hỗ trợ kỹ thuật | MATBAO.NET](https://wiki.matbao.net/mvc-mvp-mvvm-la-gi-thong-tin-can-biet-ve-cac-mo-hinh-lap-trinh/)