



# ***Compound Pattern***

Giảng viên: Huỳnh Tuấn Anh  
Khoa CNTT – Đại học Nha Trang

03/09/2023



## **Khái niệm Compound Pattern**

- ❖ Các mẫu thường được sử dụng cùng với nhau và được phối hợp trong cùng một giải pháp thiết kế
- ❖ Một Compound pattern phối hợp hai hay nhiều Pattern trong một solution để giải quyết một vấn đề diễn ra lặp đi lặp lại hay vấn đề có tính chất khái quát



## *Kiến trúc MVC*



### **Giới thiệu MVC**

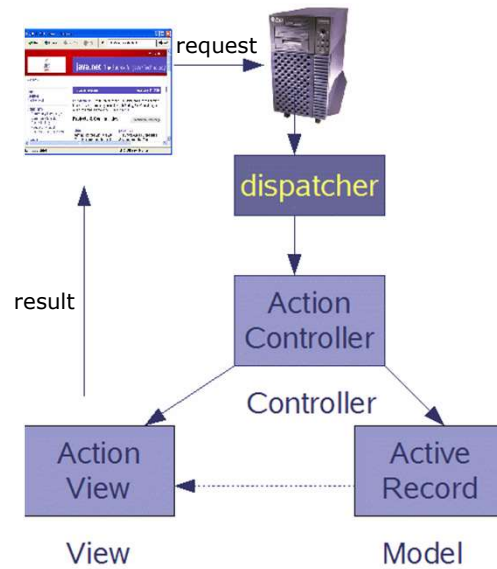
03/09/2023

#### ❖ MVC: Model – View – Controller:

- Được nghiên cứu vào 1978-1979 bởi Trygve Reenskaug.
- Hiện tại ít được sử dụng trong môi trường lập trình Desktop
- Được sử dụng rất rộng rãi và là kiến trúc cơ bản trong môi trường lập trình Web.

03/09/2023

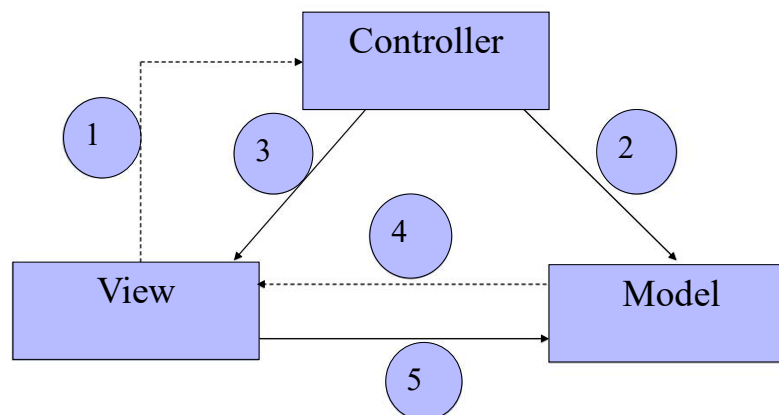
## MVC và Web



5

03/09/2023

## Các thành phần của MVC



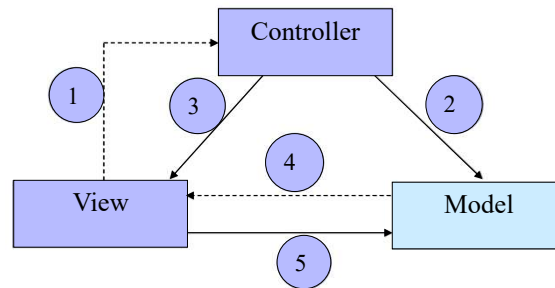
6

03/09/2023

## Các thành phần của MVC

### ❖ Model:

- Chứa dữ liệu, trạng thái và các application logic mà ứng dụng hướng tới.
- Model không biết cấu trúc của View và Controller
- Cung cấp một giao diện để View và Controller sử dụng và truy vấn nó.
- Gửi thông báo về sự thay đổi trạng thái của nó tới các observer (View).



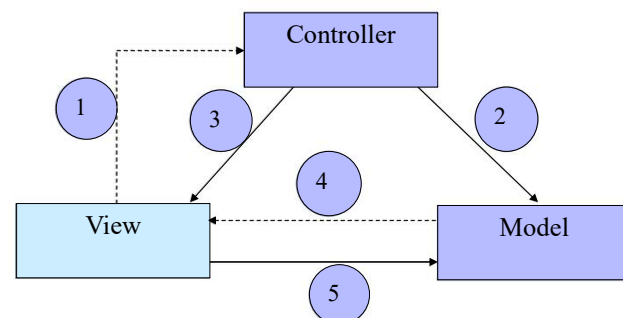
7

03/09/2023

## Các thành phần của MVC

### ❖ View:

- Trình bày, hiển thị Model cho người sử dụng. Thông thường, View lấy các state và data trực tiếp từ các Model để hiển thị.
- Cung cấp các menu, nút bấm, hộp đối thoại, chọn lựa ..., để người dùng có thể thêm, xóa, sửa, tìm kiếm và thực hiện các thao tác khác đối với dữ liệu trong hệ thống.



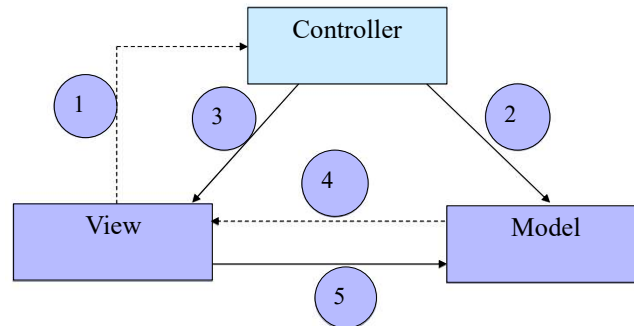
8

03/09/2023

## Các thành phần của MVC

### ❖ Controller:

- Là nơi chứa các xử lý logic nghiệp vụ (Business logic) của ứng dụng.
- Tiếp nhận input từ User và tính toán xử lý chúng trước khi gửi chúng cho Model.
- Cầu nối giữa người dùng và ứng dụng.



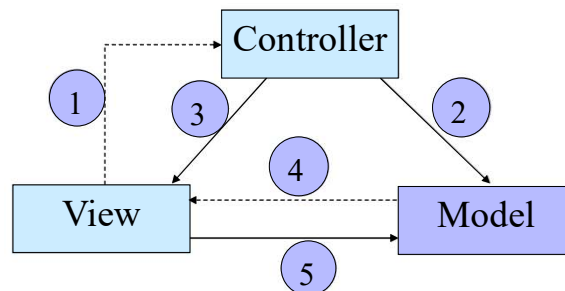
9

03/09/2023

## Ứng dụng các mẫu GoF để thiết kế MVC

### ❖ Strategy pattern:

- View là một object được cấu hình với một Strategy nào đó. View chỉ quan tâm đến giao diện người dùng và ủy quyền cho Controller quyết định về các hành vi của nó
- Controller cung cấp Strategy cho View
- Tách rời View và Model: Thông qua controller, View lấy các kết quả request của User từ Model



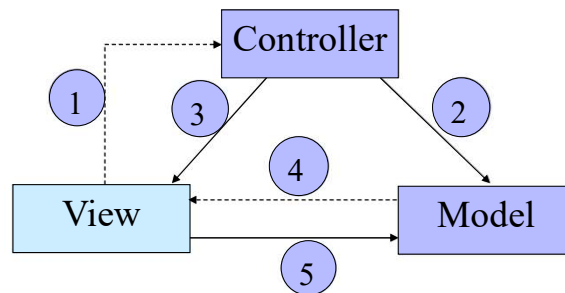
10

03/09/2023

## Ứng dụng các mẫu GoF để thiết kế MVC

### ❖ Composite Pattern:

- View chứa một tập các User Graphic Control, mỗi control được xem như là một Component
- Composite component cao nhất của View nhận yêu cầu cập nhật và chuyển đến component thích hợp



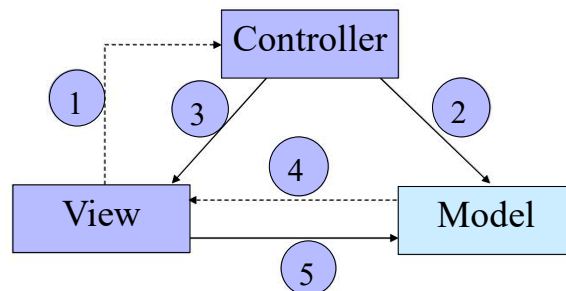
11

03/09/2023

## Ứng dụng các mẫu GoF để thiết kế MVC

### ❖ Observer Pattern:

- Model thực thi Observer pattern để đảm bảo các đối tượng observer đã đăng ký với nó cập nhật trạng thái khi model thay đổi



12

03/09/2023

## Vấn đề của MVC

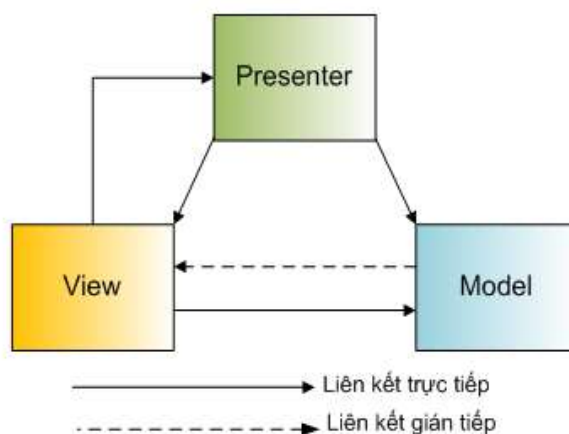
- ❖ Trong môi trường lập trình hiện đại một số control đã được hỗ trợ các xử lý sự kiện → Chia sẻ bớt việc xử lý cho các control → Không tận dụng các hỗ trợ của các Control
- ❖ Khó khăn trong việc lưu trữ các trạng thái của giao diện người dùng trong mô hình MVC?
- ❖ Khó khăn trong việc cài đặt các observer pattern cho Model

13

## Kiến trúc MVP

03/09/2023

## MVP (Model - View - Presenter)

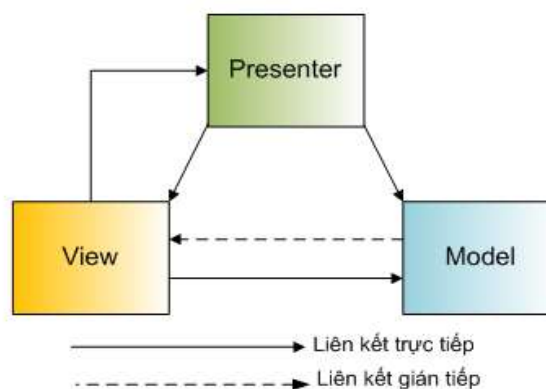


15

03/09/2023

## Các thành phần MVP

- ❖ **Model** chứa dữ liệu và các tính toán xử lý logic (application logic) để giải quyết vấn đề nghiệp vụ (business logic) mà phần mềm hướng tới.
- ❖ **View** là thành phần đảm nhận trình bày từ những dữ liệu của Model. View bao gồm những gì thể hiện trên màn hình như các control, form, widget,...
- ❖ **Presenter** là thành phần đảm nhận các xử lý thể hiện cần đến sự tương tác trên dữ liệu.



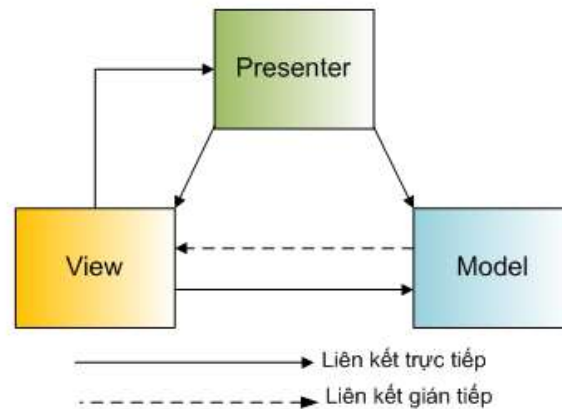
16



03/09/2023

## Các thành phần của MVP

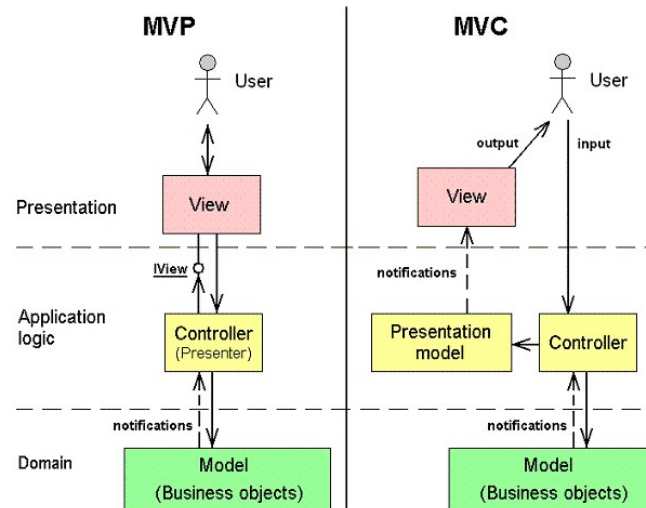
- ❖ Việc xử lý các dữ liệu input được View đảm nhận và được chuyển cho Presenter khi có yêu cầu tương tác đến Model
  - Trong một vài trường hợp dữ liệu được cập nhật trực tiếp vào Model (như sử dụng thay đổi dữ liệu trên một TextBox). Hầu hết các trường hợp khác việc thao tác với Model được giao cho Presenter đảm trách



17

03/09/2023

## Sự khác nhau giữa MVC và MVP



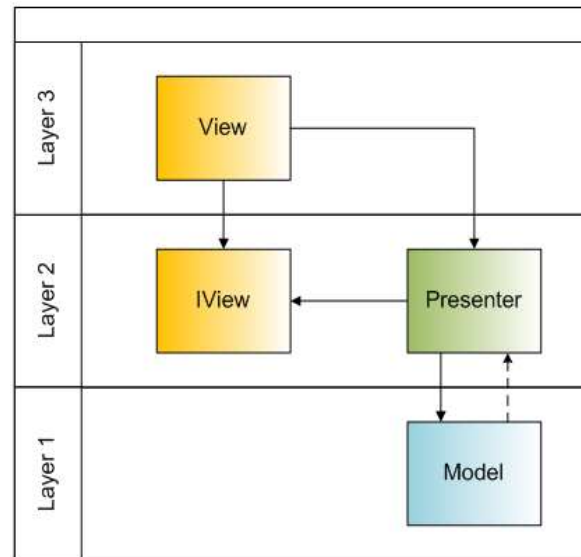
Hình ảnh tham khảo từ website mvccsharp.com

18

03/09/2023

## Phiên bản MVP của Microsoft

- ❖ Presenter: Truy cập đến View thông qua giao diện IView
- ❖ View: Thực thi giao diện IView

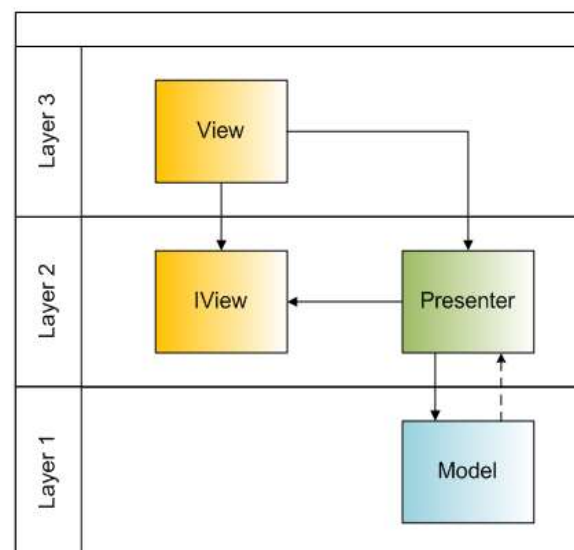


19

03/09/2023


## Phiên bản MVP của Microsoft

- ❖ Mỗi quan hệ giữa Presenter và Model là mối quan hệ một chiều
  - Presenter thực hiện các Business logic trực tiếp trên Model hay thông qua các Application logic của Model
- ❖ Presenter thực thi mẫu Strategy Pattern hay Observer Pattern tùy vào ứng dụng cụ thể



20

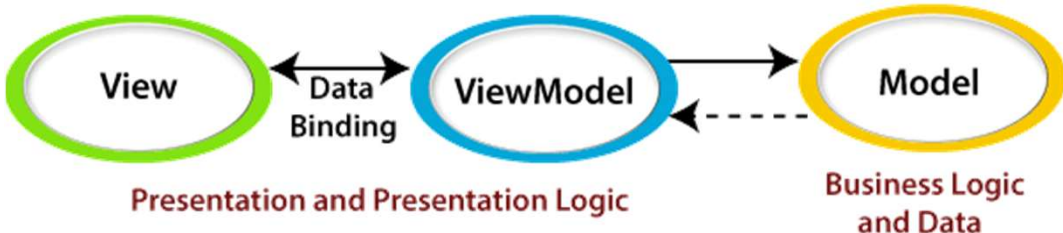
## Kiến trúc MVVM (Model-View-ViewModel)



MVVM

03/09/2023

- ❖ Là một biến thể của mẫu MVP, được phát minh bởi các kiến trúc sư của Microsoft: Ken Cooper và Ted Peters để chú trọng vào việc đơn giản hóa việc lập trình theo hướng sự kiện của giao diện người dùng.
- ❖ Sơ đồ kiến trúc



```

graph LR
    View([View]) <-->|Data Binding| ViewModel([ViewModel])
    ViewModel <-->|Data Binding| Model([Model])
    Model -.->|Data Binding| ViewModel
  
```

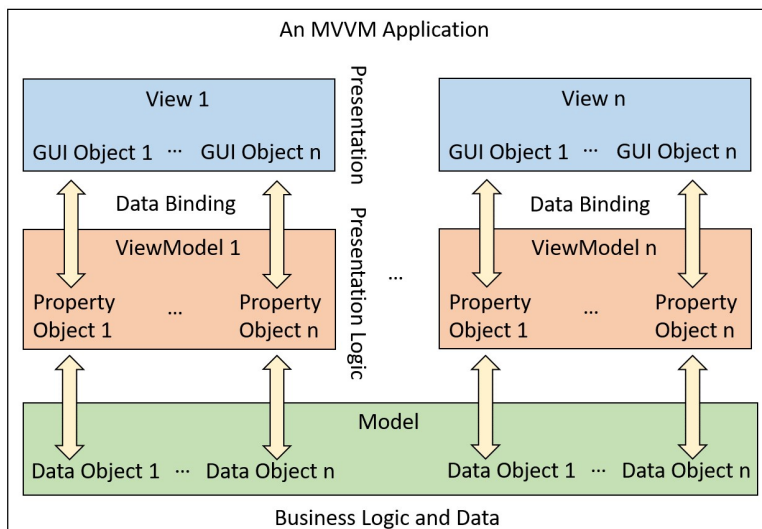
The diagram illustrates the MVVM (Model-View-ViewModel) architecture. It consists of three main components: View, ViewModel, and Model. The View and ViewModel are connected by a solid double-headed arrow labeled "Data Binding". The ViewModel and Model are also connected by a solid double-headed arrow labeled "Data Binding". A dashed arrow points from the Model back to the ViewModel, indicating a one-way data flow from the Model to the ViewModel. Below the View, the text "Presentation and Presentation Logic" is written. Below the Model, the text "Business Logic and Data" is written.

22

03/09/2023

## Kiến trúc MVVM

- ❖ View phụ thuộc vào ViewModel nhưng "không biết gì" về Model.
- ❖ ViewModel phụ thuộc vào Model, "không biết gì" về View.
- ❖ Model "biết đến" ViewModel nhưng "không biết gì" về View



23

03/09/2023

## Model-View-ViewModel

- ❖ **Model:** Nằm giữ logic nghiệp vụ của ứng dụng và phụ thuộc vào lớp liên quan Resource Model - để truy cập cơ sở dữ liệu. Tương tự như trong MVC, Model chứa logic quản lý dữ liệu và mô tả dữ liệu cơ bản cần thiết cho hoạt động của ứng dụng. Model đáp ứng các request đến từ ViewModel.
- ❖ **View:** Xác định giao diện người dùng, bao gồm tất cả các thành phần trực quan (nút, nhãn, trình chỉnh sửa, v.v.) được liên kết với các thuộc tính và lệnh trong ViewModel.
- ❖ **ViewModel:** Chứa các logic kết nối giữa View và Model. Có nhiệm vụ chuyển đổi dữ liệu từ Model và cung cấp các luồng dữ liệu tới View. ViewModel chứa các hook hay các callback để cập nhật View. Nó được sử dụng để yêu cầu dữ liệu từ Model
- ❖ **Ưu điểm:** Tách biệt sự phát triển của View khỏi phần còn lại của ứng dụng, cho phép phát triển song song GUI và Logic của ứng dụng cũng như các lớp khác

24

03/09/2023

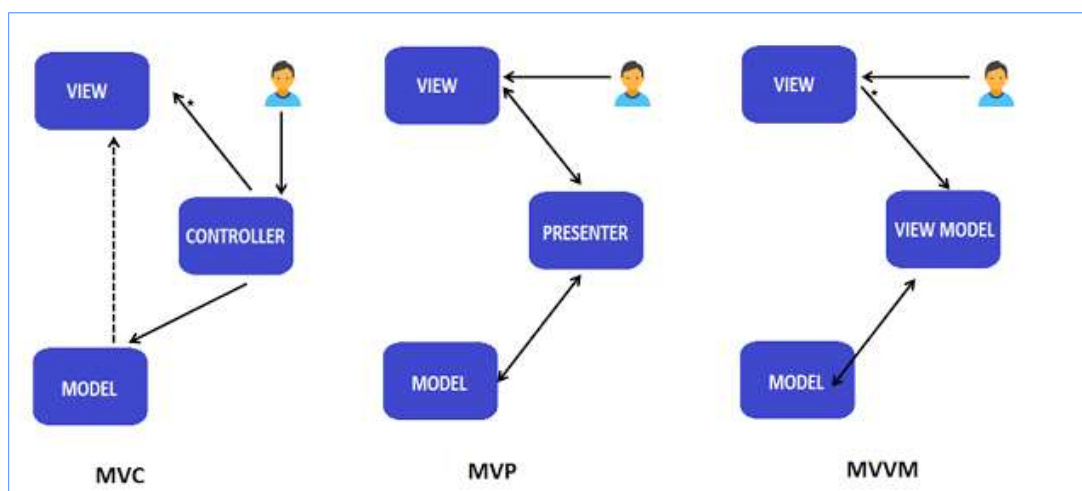
## Điểm giống nhau giữa MVC, MVP, MVVM

- ❖ Các mô hình đều có mục đích tách biệt giữa mô hình dữ liệu, UI và lớp xử lý nghiệp vụ
- ❖ Model: Chứa các lớp (class) mô tả dữ liệu, lớp (layer) truy cập dữ liệu
- ❖ View: Thường là các UI của ứng dụng

25

03/09/2023

## So sánh MVC, MVP, MVVM



26

03/09/2023

## So sánh MVC, MVP, MVVM

	Model	Controller/Presenter/ViewModel	View
MVC	- Các lớp mô tả dữ liệu của ứng dụng. - Lớp (layer) truy cập dữ liệu	- Controller, biết View - Tiếp nhận yêu cầu xử lý, lấy dữ liệu từ Model, thực hiện các xử lý nghiệp vụ và hiển thị dữ liệu lên View	- Biết Controller - Có thể truy cập dữ liệu trực tiếp từ Model
MVP		- Presenter - Lắng nghe các sự kiện từ cả View lẫn Model - Trung gian cho các hành vi giữa View và Model. - Tham chiếu với View thông qua các interface (IView)	- Là Các interface, IView, được Presenter sử dụng để tham chiếu đến UI. - Các UI phải implement một hoặc nhiều interface IView mới tương tác được với Presenter
MVVM		- ViewModel - Tạo ra một tập con của Model cho View. Tránh việc phơi bày toàn bộ Model lên View - Không cần tham chiếu đến View	- Thường là các UI của ứng dụng - Có thể bind với các thuộc tính trên ViewModel để hiển thị Model trên View

27

03/09/2023

## Ưu / Nhược điểm của các mô hình

### ❖ MVC

- Ưu điểm: Controller tách biệt View và Model
- Nhược điểm: Controller và View phụ thuộc nhau do đó khó có thể phát triển độc lập

### ❖ MVP:

- Ưu điểm: ba lớp (layer) tách biệt tốt hơn MVC, Presenter và View có thể phát triển độc lập nhau
- Nhược điểm: Phải định nghĩa thêm giao diện để giao tiếp Presenter và View

### ❖ MVVM

- Ưu điểm: Mô hình chia làm 3 lớp (layer) rất rõ ràng, dễ dàng lập trình, kiểm tra sửa lỗi. Thực hiện các data binding giúp viết mã cho View dễ dàng
- Nhược điểm: Data binding hai chiều làm cho tốn kém bộ nhớ. Độ phức tạp sẽ tăng khi View được gán nhiều biến, biểu thức. Do đó nên tránh tính toán ở View, thay vào đó, nên thực hiện tính toán ở ViewModel

28

03/09/2023

## Bài tập thiết kế ứng dụng Mobile sử dụng MVVM, MVC

- ❖ BT1: Xây dựng ứng dụng quản lý môn học và điểm của môn học
  - Định nghĩa Model: Các lớp dữ liệu, dữ liệu lưu trữ trên Firebase (MVVM vì Firebase có khả năng binding dữ liệu tốt với các StreamBuilder Widget)
  - ViewModel: Lớp truy cập dữ liệu để cung cấp cho các widget, sử dụng plugin Provider để cung cấp lớp truy cập dữ liệu này cho các trang của ứng dụng
  - View: Các trang của ứng dụng
  - Chú ý: Sinh viên cần vẽ sơ đồ các thành phần của mô hình MVVM của ứng dụng
- ❖ BT2: Tương tự như bài tập 1 nhưng yêu cầu dữ liệu lưu bằng SQLite (Có thể thực hiện trên các mô hình MVC, MVP, MVVM)
  - Định nghĩa Model: Các lớp dữ liệu, Lớp truy cập dữ liệu
  - ViewModel: Lớp cung cấp các cách hiển thị model (Danh sách môn học, Danh sách điểm của một môn học), Sử dụng plugin provider để thực hiện tầng ViewModel
  - View: Các trang của ứng dụng

29