Dr Milan Milanović • 2nd
Chief Roadblock Remover and Learning Enabler | Software Development Expert | Author | Lead...
View my newsletter
1h • 🌐

Follow

What is the difference between MVC, MVP, MVI, MVVM, MVVM-C, and VIPER architecture patterns?

Here is the list of the most critical architectural patterns:

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

It is one of the earliest and most adopted design patterns. Its primary goal is to separate the application's data, user interface, and control logic into three interconnected components.

Here, the Model manages data and logic, the View displays the information, and the controller connects the Model and View, handling user input.

Usage: For Web applications with a clear separation between data handling and UI.

2. MVP (**Model-View-Presenter**):

The pattern evolved from MVC, aiming to address its shortcomings in event-driven environments by decoupling the View from the Model, with the Presenter acting as a middleman.

Here, the Model manages data, views display data, and sends user commands to the Presenter, while the Presenter retrieves data from the Model and presents it to the View.

Usage: Applications emphasizing testing and UI logic, such as Android apps.

3. MVI (**Model-View-Intent**):

MVI is a reactive architecture embracing unidirectional data flow, ensuring that, given a state, the UI remains consistent.

Here, the Model represents the state, View reflects the state, while intent represents user actions that change the state.

Usage: Reactive applications or frameworks like RxJava with a focus on state consistency.

4. **MVVM** (**Model-View-ViewModel**):

MVVM arose to address complexities in UI development, promoting a decoupled approach with the ViewModel handling view logic without knowing the UI components.

Here, the Model manages and displays data, while ViewModel holds and contains UI-related data.

Usage: UI-rich applications or platforms with data-binding, such as WPF or Android with LiveData.

5. **MVVM-C** (**MVVM with Coordinator**):

MVVM-C builds upon MVVM, introducing the Coordinator to handle navigation, decoupling it from View and ViewModel.

Usage: larger applications, especially iOS, where complex navigation needs separation from view logic.

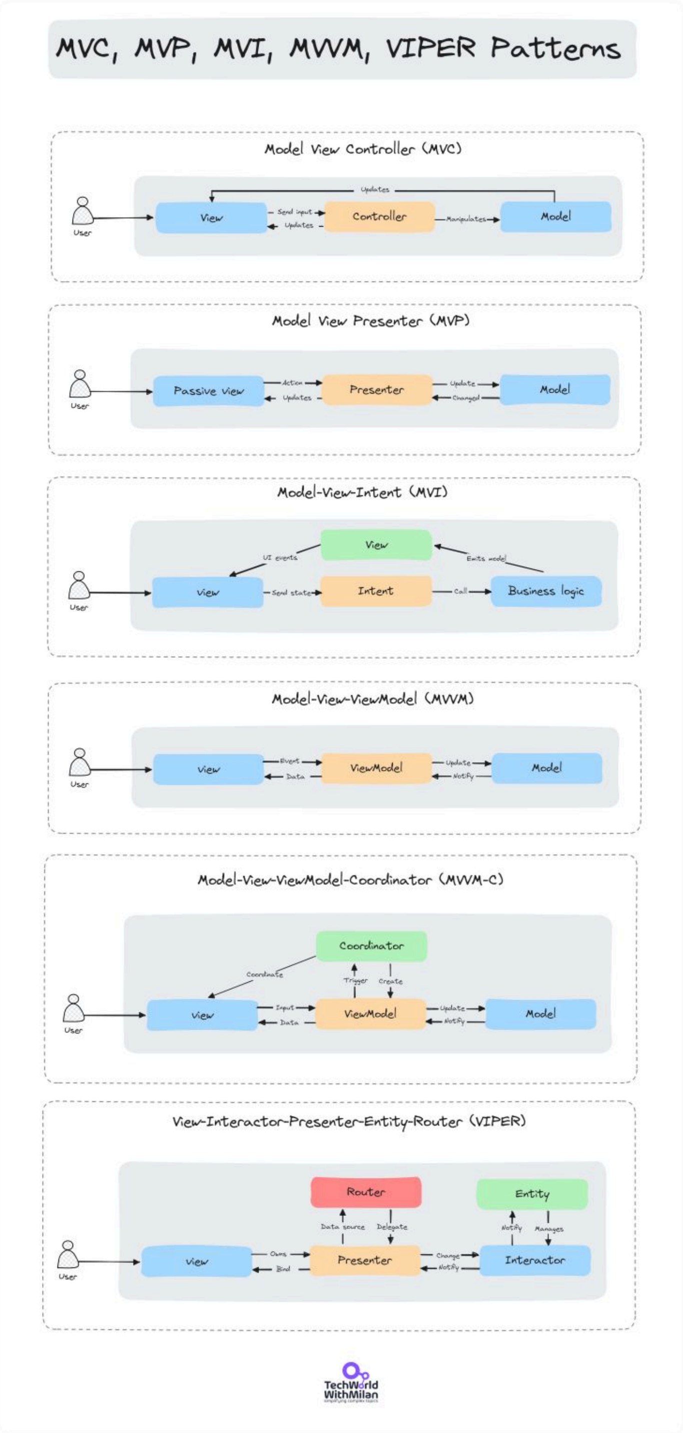
6. **VIPER** (**View-Interactor-Presenter-Entity-Router**):

VIPER is a modular architecture akin to Clean Architecture. It emphasizes testability and the Single Responsibility Principle by breaking down application logic into distinct components.

Here, the View displays what the Presenter sends, the interactor contains business logic per use case, the Presenter contains view logic for preparing content, the entity includes a primary model object, and the router contains navigation logic.

Usage: Complex applications, especially iOS, need modularity, testability, and clarity.

#technology #softwareengineering #programming #techworldwithmilan #softwaredesign



44

11 comments 1 repost

Like

Comment

Repost

Send

Anton Martyniuk • Following

1h

Microsoft MVP | Helping 30K+ Software Engineers Improve .NET Skills and Craft Better Softwar...

Nice breakdown.

MVC - ASP .NET Core MVC and Razor Pages in .NET

MVP, MVVM - WPF in .NET

Like 👍 2 | Reply 1 reply

Dr Milan Milanović Author 1h
Chief Roadblock Remover and Learning Enabler | Software Development Expert | Author |...

Anton Martyniuk yes!