What distinguishes MVC, MVP, MVVM, MVVM-C, and VIPER architecture patterns from each other?
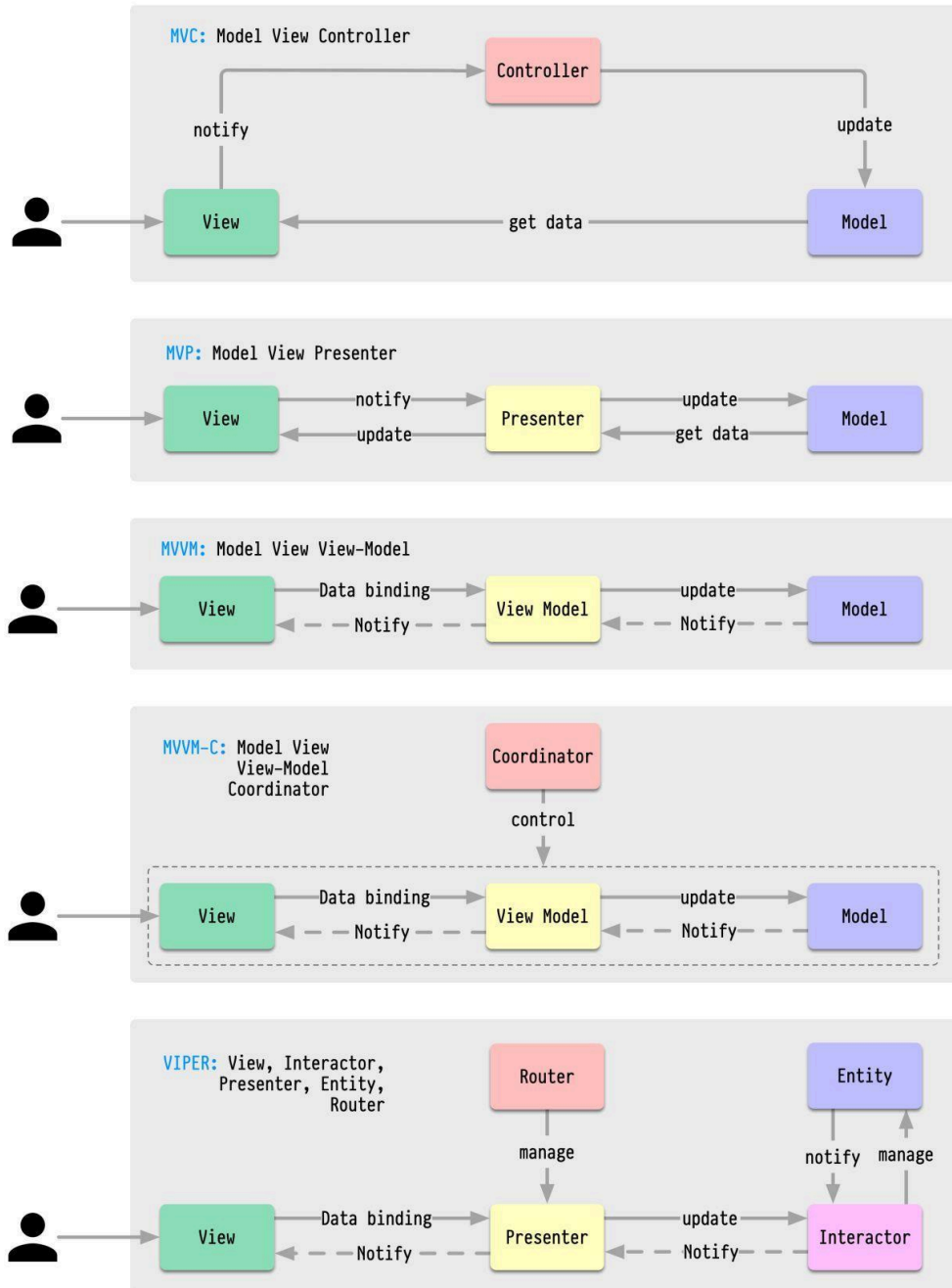
MVC, MVP, MVVM, VIPER patterns

ByteByteGo.com

MVC: Model View Controller

View — notify → Controller — update → Model
Model — get data → View

MVP: Model View Presenter

View — notify → Presenter — update → Model
View ← update — Presenter ← get data — Model

MVVM: Model View View-Model

View — Data binding → View Model — update → Model
View ← Notify -- View Model ← Notify -- Model

MVVM-C: Model View View-Model Coordinator

Coordinator — control →

View — Data binding → View Model — update → Model
View ← Notify -- View Model ← Notify -- Model

VIPER: View, Interactor, Presenter, Entity, Router

Router — manage →
Entity
notify / manage

View — Data binding → Presenter — update → Interactor
View ← Notify -- Presenter ← Notify -- Interactor

These architecture patterns are among the most commonly used in app development, whether on iOS or Android platforms. Developers have introduced them to overcome the limitations of earlier patterns. So, how do they differ?

- MVC, the oldest pattern, dates back almost 50 years
- Every pattern has a "view" (V) responsible for displaying content and receiving user input
- Most patterns include a "model" (M) to manage business data
- "Controller," "presenter," and "view-model" are translators that mediate between the view and the model ("entity" in the VIPER pattern)
- These translators can be quite complex to write, so various patterns have been proposed to make them more maintainable

Over to you: keep in mind that this is not an exhaustive list of architecture patterns. Other notable patterns include Flux and Redux. How do they compare to the ones mentioned here?