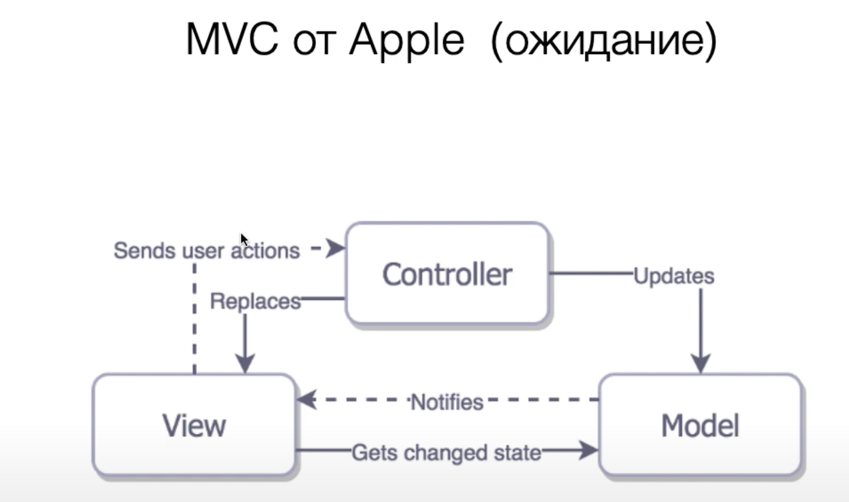
Архитектурные паттерны iOS. Конспект. Ильдар

MVC

Model View Controller

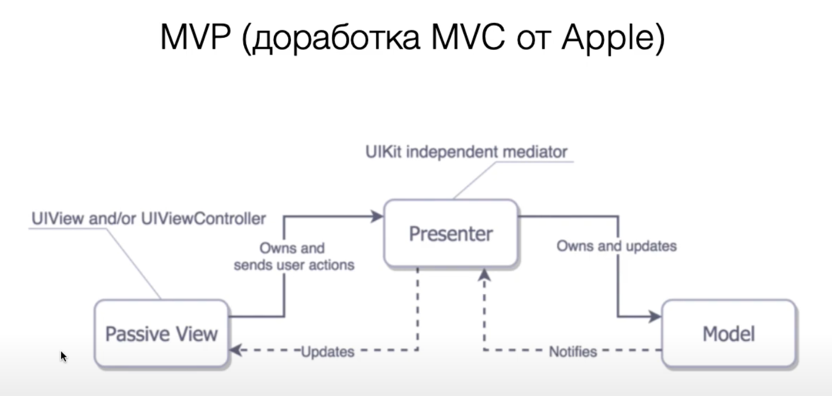


1. Пользовательские действия поступают в “Controller”. Это может быть что-то вроде нажатия кнопки или ввода текста.
2. Controller обрабатывает эти действия и обновляет “Model”. “Model” - это часть программы, которая содержит основные данные и логику.
3. Когда “Model” обновляется, он уведомляет “View”.
4. View — это то, что пользователь видит на экране. Когда “View” получает уведомление от “Model” о том, что данные изменились, он обновляет то, что пользователь видит.

Основной минус - превращение в MassiveVC - когда в контроллере сосредоточено слишком много логики приложения, что делает его сложным для понимания и поддержки.

MVP

Model View Presenter



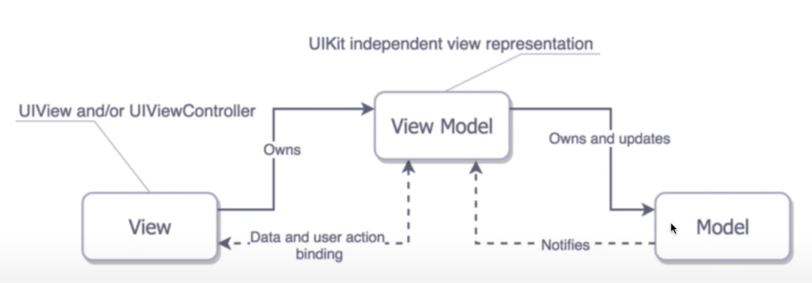
1. View (UIView/UIViewController): это то, что пользователь видит на экране. View взаимодействует с Presenter, чтобы обновить данные на экране или реагировать на действия пользователя.
2. Presenter обрабатывает логику, связанную с обновлением View и взаимодействием с Model. Он действует как посредник между Model и View.
3. Model содержит основные данные и бизнес-логику приложения. Он обновляется Presenter’ом и может уведомлять Presenter об изменениях данных.

Основной плюс - хорошо разделяет логику представления от бизнес-логики.

Основной минус - может стать перегруженным из-за большого количества презентеров.

MVVM

Model-View-ViewModel (реактивное программирование)



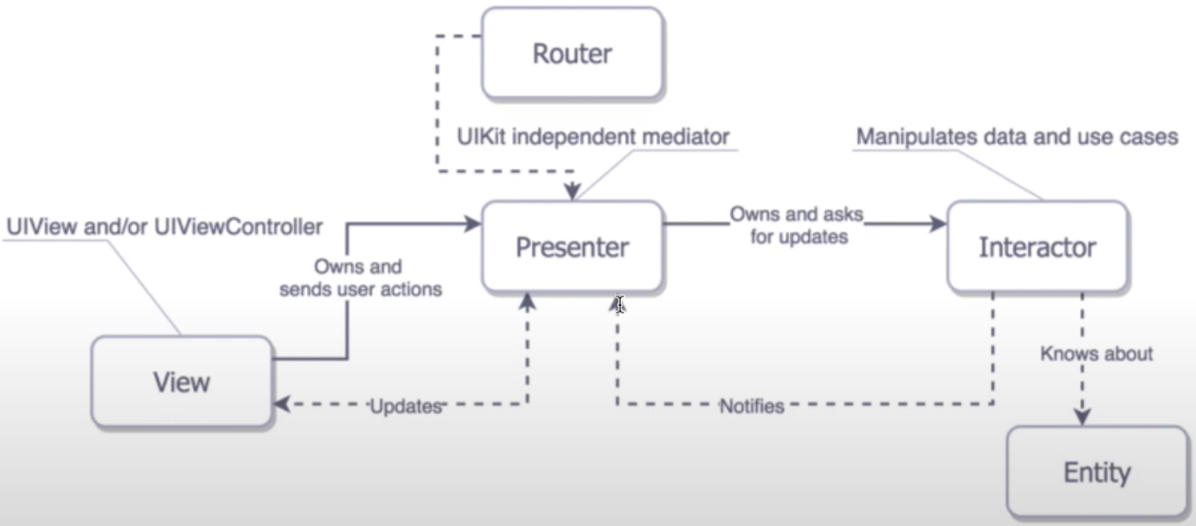
1. View (UIView/UIViewController) - это то, что пользователь видит на экране. Он взаимодействует с ViewModel, чтобы обновить данные на экране или реагировать на действия пользователя.
2. ViewModel обрабатывает логику, связанную с обновлением View и взаимодействием с Model. Он действует как посредник между Model и View, но в отличие от Presenter в MVP, ViewModel также может иметь состояние.
3. Model содержит основные данные и бизнес-логику приложения. Он обновляется ViewModel и может уведомлять ViewModel об изменениях данных.

Основной плюс - позволяет легко связывать данные и представления.

Основной минус - может привести к перегруженности моделей представления.

VIPER

View, Interactor, Presenter, Entity, Router



1. View - это то, что пользователь видит на экране. Он взаимодействует с Presenter, чтобы обновить данные на экране или реагировать на действия пользователя.
2. Interactor содержит бизнес-логику приложения. Он обрабатывает данные, которые затем передаются Presenter.
3. Presenter обрабатывает логику, связанную с обновлением View и взаимодействием с Interactor и Router.
4. Entity представляет собой модель данных в приложении.
5. Router управляет навигацией между различными частями приложения.

Основной плюс - хорошо разделяет ответственности между различными компонентами.

Основной минус - требует больше времени на разработку из-за большего числа компонентов.

Главное отличие VIPER от MV(x) моделей – появление router`a – решение проблемы навигации.