ИСПОЛЬЗОВАНИЕ АРХИТЕКТУРНОГО ШАБЛОНА VIPER ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Полуэктов Н.А.

Институт информационных технологий Белорусского государственного университета информатики и радиоэлектроники
г. Минск, Республика Беларусь

Савенко А.Г. – старший преподаватель, м.т.н.

Аннотация. Работа посвящена архитектуре VIPER, ее преимуществу перед другими аналогами и ее недостаткам. В работе также проведен анализ преимуществ и недостатков альтернативных архитектурных шаблонов, приведена иллюстрация состава модулей VIPER и их взаимосвязь между друг другом.

Архитектурный шаблон VIPER разрабатывался и получил свое распространение исключительно для iOS приложений [1]. Он является альтернативой MVC или MVVM. Был разработан для создания приложений с чистой архитектурой, которая эффективно разделяет различные необходимые функции и обязанности, такие как пользовательский интерфейс, бизнес-логика, хранилище данных и сеть. Затем их легче тестировать, поддерживать и расширять.

MVC является распространенной концепцией, которая делит проект на три части:

1. Model – сущности.
2. View – интерфейс для взаимодействия с пользователем.
3. Controller – обеспечивает взаимодействие между View и Model.

MVC – это архитектура, которую компания Apple предлагает использовать в приложениях. Но проекты бывают с довольно большим и сложным функционалом: поддержка сетевых запросов, парсинг данных, доступ к моделям данных, преобразование данных для выдачи, реакции на интерфейсные события и т.д. Схема архитектурного шаблона MVC представлена на рисунке 1. В итоге получаются огромные контроллеры, которые решают вышеперечисленные задачи и много кода, который невозможно переиспользовать. Другими словами, при использовании шаблона MVC трудно поддерживать приложение длительное время [2].

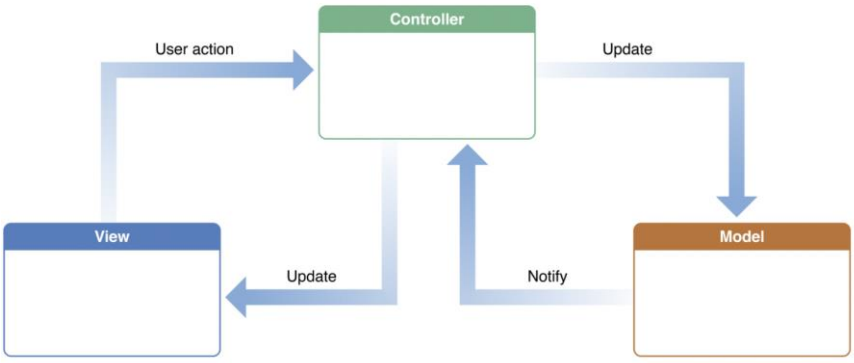

Рисунок 1 – Схема архитектурного шаблона MVC

Тем не менее, архитектурный шаблон MVC имеет следующие преимущества:
– простота и скорость разработки;
– быстрое внедрение нового разработчика;
– идеально подходит для маленьких и средних проектов;
К недостаткам архитектурного шаблона MVC можно отнести:
– Massive View Controller. Большое количество кода в Controller-е. Controller отвечает за все;
– Controller отвечает за все;
– проблема написания unit тестов;
– не рекомендуется для больших проектов с большим количеством разработчиков.

Для решения проблем MVC и был разработан VIPER (View–Interactor–Presenter–Entity–Router), который лишен данных недостатков [3]. Схема архитектурного шаблона VIPER представлена на рисунке 2.

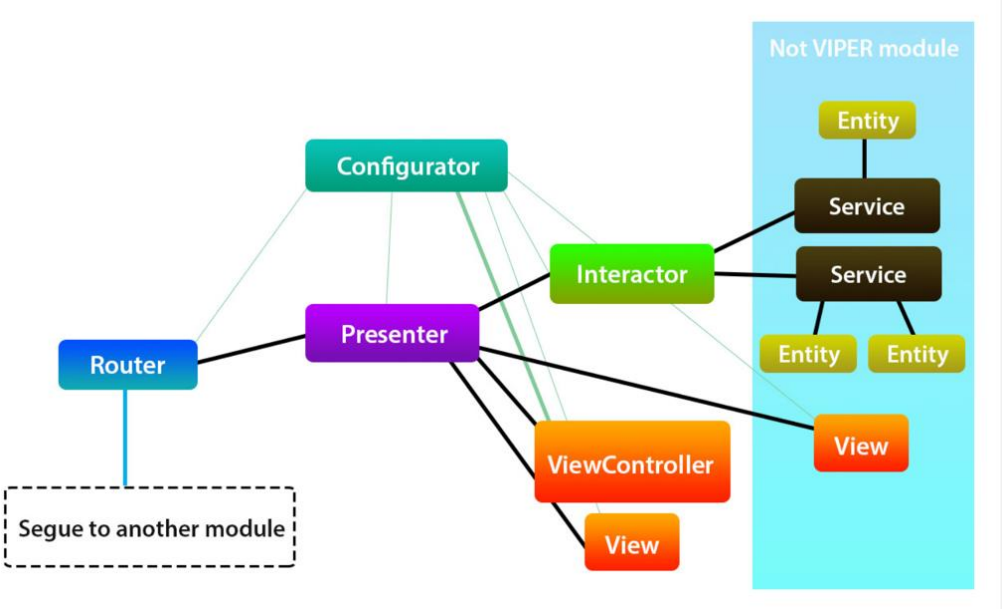

Рисунок 2 – Схема архитектурного шаблона VIPER

Рассмотрим основные особенности данного архитектурного шаблона.

Entity в понятие модуля может не входить, т.к. является самодостаточным классом, который может использоваться в любом модуле или сервисе. Стартовой точкой является Configurator. Он знает о всех зависимостях внутри модуля. В нем устанавливается, что у ViewController будет Presenter, у Presenter будет Interactor и т.д. Presenter отвечает за логику нажатий на кнопки, ввода текста или какое-либо другое взаимодействие с UI. Presenter решает, куда перенаправить действие – на Router или Interactor. Но более важной функцией Presenter является подготовка и передача визуальных данных для View/ViewController, которые будут видны для пользователя.

Router отвечает за навигацию внутри приложения (закрытие текущего экрана или показ нового).

Interactor решает, что делать дальше с поступившими событиями и какой сервис вызвать. В нем содержится логика модуля.

View(View+Controller) получает от презентерора запросы на обновление представлений.

Сервисом в нашей интерпретации называются различные хелперы и другие классы, которые могут быть доступны из разных модулей и частей приложения (логика авторизации, работа с базой, работа с сервером, шифрование и т.п.).

Ответственность сильно поделилась между Презентером и Интерактором:

1. Презентер отвечает за UI: просит данные у интерактора, подготавливает их для View и говорит ему когда, как и что показывать;

2. Интерактор отвечает за данные: когда презентер просит что-то показать, именно Интерактор идёт в базу, делает всякий CRUD, а потом отдает презентеру готовый ответ (данные или ошибка).

Таким образом, можно выделить следующие преимущества архитектурного шаблона VIPER:

– чистая архитектура;

– возможность быстрого и удобного модернизировать экраны;

– удобное покрытие Unit тестами;

– удобно работать с большой командой разработчиков.

К недостаткам можно отнести следующее:

– много сопутствующего кода;

– медленная скорость разработки;

– сложность изучения проекта;

– рекомендуется для больших проектов с большой командой разработчиков.

В заключение можно отметить, что хорошо спроектированная архитектура очень важна для того, чтобы обеспечить длительную поддержку проекта. При использовании архитектуры с высокой декомпозицией модулей приложение легче покрыть Unit-тестами и тем самым уменьшить количество багов в приложении.

Список использованных источников:

1. Get Started with Clean Swift [Электронный ресурс]. – Режим доступа: https://clean-swift.com, – Дата доступа: 04.02.2022.

2. VIPER Swift [Электронный ресурс]. – Режим доступа: http://bytepace.com/ru/blog/viper-swift, – Дата доступа: 04.02.2022.

3. Architecting iOS Apps with VIPER [Электронный ресурс]. – Режим доступа: https://www.objc.io/issues/13-architecture/viper/, – Дата доступа: 14.03.2022.