УДК 004.413.2

ПРИНЦИПЫ РАЗРАБОТКИ И ШАБЛОНЫ ПРОЕКТИРОВАНИЯ ПРИЛОЖЕНИЙ ПОД ОПЕРАЦИОННУЮ СИСТЕМУ IOS

Е.В. КОНЯЕВ
(Представлено: И.С. РУСЕЦКИЙ)

В статье рассматриваются основные принципы разработки приложений под операционную систему iOS, подробно рассмотрены распространенные шаблоны проектирования и их особенности.

Ключевые слова: мобильное приложение, разработка, архитектура, Swift, Apple.

С развитием современных технологий, которые позволяют создавать различные мобильные устройства, рынок программных продуктов получает мощнейший стимул к расширению и развитию. Современную мобильную технику люди носят с собой всегда и везде. За основу взята мобильная ОС iOS и рассмотрение принципов разработки программных продуктов для мобильных устройств будет производиться на ее примере. Для грамотного создания приложения необходимо учитывать принципы простоты, удобства и интуитивно-понятного управления. Исходя из входных параметров выбирается наиболее подходящий шаблон архитектуры, влияющий на последующее использование приложения конечным пользователем.

Написание приложения под ОС iOS осуществляется на компьютерах и ноутбуках производства компании Apple в рабочей среде X-Code, имеется возможность написания приложения на двух языках: Objective-C или Swift. Остановимся подробнее на среде разработки Xcode — это приложение для Mac, предназначенное для разработки других приложений для Mac и iOS. Его можно загрузить бесплатно из App Store для Mac. Xcode тесно интегрирован с framework Cocoa, который состоит из библиотек, API, и сред, которые формируют слой разработки для всех Mac OS X. Данный набор инструментов включает в себя: Xcode IDE (для написания кода, создания и отладки приложений), Interface Builder (для разработки пользовательского интерфейса), инстументы для анализа поведения и производительности и другое.

Рассмотрим подробнее языки программирования [1]. Начнем с Objective-C, который является расширением языка программирования C и предоставляет объектно-ориентированные возможности, а также динамическую систему времени выполнения. В нем имеются привычные элементы, например, типы данных (int, float и т.д.), структуры, функции, указатели и управляющие конструкции (while, if...else и оператор for). Также имеется доступ к функциям стандартной библиотеки программирования C, например, к тем, которые объявлены в stdlib.h и stdio.h. По аналогии с интерфейсами в Java, в Objective C есть протоколы. Протокол определяет набор селекторов, которые должен поддерживать объект, реализующий протокол.

Swift является более молодым и, на данный момент, стремительно развивающимся языком программирования. Он разрабатывается компанией Apple, как будущая замена Objective-C [2]. Swift задумывался как быстрый и эффективный язык программирования с откликом в реальном времени, который легко можно вставить в готовый код Objective-C. Swift берет довольно многое из Objective-C, однако он определяется не указателями, а типами переменных, которые обрабатывает компилятор. Swift работает с фреймворками Cocoa и Cocoa Touch и совместим с основной кодовой базой Apple, написанной на Objective-C. Swift разрабатывается как более безопасный язык в сравнении с Objective-C, к тому же Swift, в отличии от Objective-C, легко читаемый язык, как и Python. На данный момент эти языки взаимно дополняют друг друга, но основная часть приложений написана на Objective-C.

Важным аспектом разработки мобильного приложения является выбор правильного шаблона проектирования. В первую очередь рассмотрим особенности хорошей архитектуры приложения: сбалансированное распределение функций между действующими объектами, тестируемость приложения и удобство использования.

В настоящее время существует много вариантов архитектуры шаблонов проектирования, наиболее популярными из которых являются:

- Model-View-Controller (MVC);
- Model-View-Presenter (MVP);
- Model-View-ViewModel (MVVM).

MVC – схема использования нескольких модулей, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные [3]. Традиционное представление MVC модели представлено на рисунке 1.

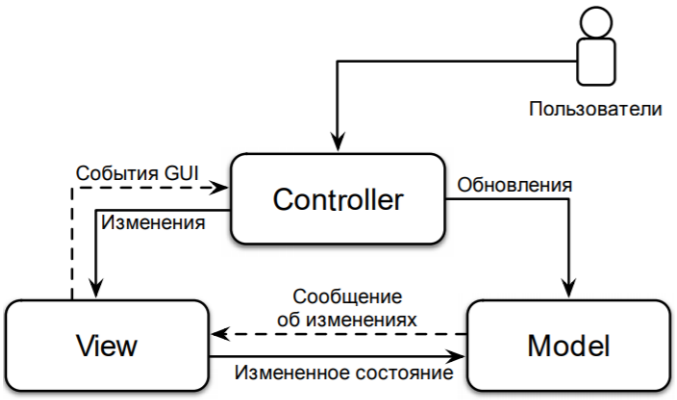

Рисунок 1. – Традиционное представление MVC модели

Модель (Model) содержит основные данные, например, переменные, подключения к внешним RSS-каналам или изображения, подробные функции и числовую информацию. Вид (View) отвечает за стиль отображения информации на дисплее. В качестве представления выступает экран с графическими элементами на котором, например, можно изменить стиль или удалять элементы. Контроллер (Controller) управляет запросами пользователя и использует модель для реализации необходимой реакции.

MVP является шаблоном проектирования производным от MVC, структура шаблона показана на рисунке 2. Основной задачей MVP является сделать вид (view) повторно используемым [4].

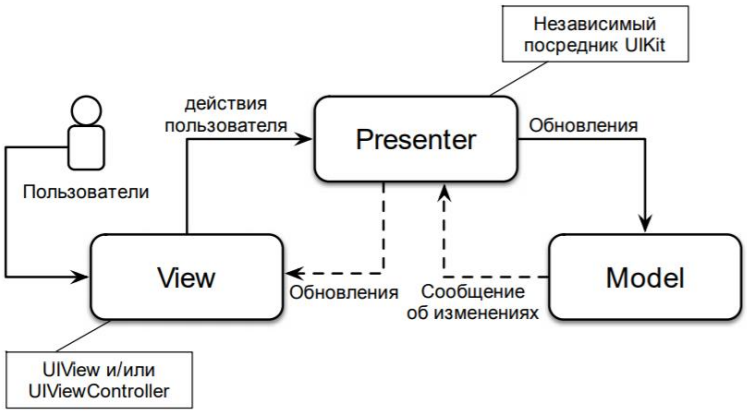

Рисунок 2. – Структура шаблона MVP

MVC и MVP парадигмы очень похожи друг на друга, но их применение зависит от условий использования. Для MVC — это там, где вид (view) обновляется каждый раз по какому-либо событию, а для MVP, когда вид (view) не нужно каждый раз пересоздавать.

Последний из выбранных шаблонов проектирования, MVVM, появился для обхода ограничений паттернов MVC и MVP, и объединяющий некоторые из их сильных сторон. Эта модель впервые появилась в составе фреймворка Small Talk в 80-х, и была позднее улучшена с учетом обновленной модели MVP. Схема работы MVVM представлена на рисунке 3.

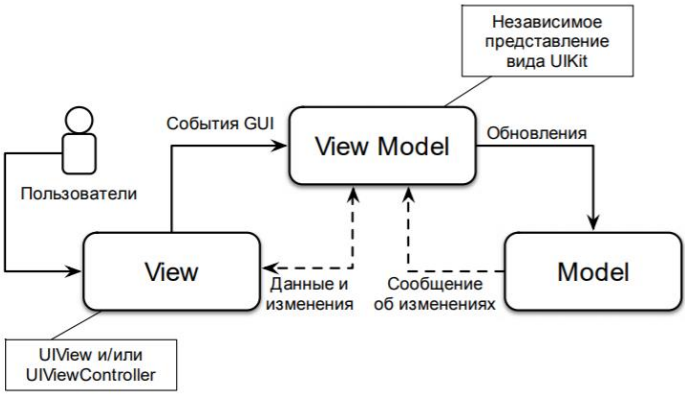

Рисунок 3 – Модель MVVM

MVVM является очень привлекательным вариантом, так как он сочетает в себе преимущества вышеупомянутых подходов, и, кроме того, он не требует дополнительного кода для вида (view) и тестируемость находится на хорошем уровне [5].

Все шаблоны проектирования являются актуальными на данный момент. Выделим некоторые моменты для выбора шаблона проектирования: MVVM используется в ситуации, когда возможно связывание данных без специальных интерфейсов, MVP используется, когда невозможно связывание данных, MVC используется, когда связь между видом (View) и другими частями приложения невозможна.

Заключение. В данной статье были рассмотрены основной инструментарий создания приложения на платформе iOS, используемые для этого языки программирования и их особенности, а также основные шаблоны проектирования приложений и их структурные отличия.

ЛИТЕРАТУРА

1. Live Typing [Электронный ресурс]. Режим доступа: https://livetyping.com/ru/blog/na-chem-pishut-prilozhenija-pod-ios. – Дата доступа: 28.09.2020.
2. GeekBrains. [Электронный ресурс]. Режим доступа: https://geekbrains.ru/posts/swift_vs_obj_c. Дата доступа: 28.09.2020.
3. Stfalcon [Электронный ресурс]. Режим доступа: https://stfalcon.com/ru/blog/post/ios-patterns. Дата доступа: 28.09.2020.
4. Habr [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/215605/ Дата доступа: 29.09.2020.
5. ProfessorWEB [Электронный ресурс]. Режим доступа: https://professorweb.ru/my/WPF/documents_WPF/level36/36_5.php Дата доступа: 20.09.2020.