УДК 004.413.2

05.00.00 Технические науки

ПРИНЦИПЫ РАЗРАБОТКИ ПРИЛОЖЕНИЙ ПОД ОПЕРАЦИОННУЮ СИСТЕМУ IOS

Великанова Лариса Олеговна
к. э. н., профессор
SPIN-code = 2123-1838

Мурлин Алексей Георгиевич
к.т.н., доцент
SPIN-code = 4991-8507

Гайвук Анжела Рудиковна
студент
SPIN-code =1996-5100
ФГБОУ ВО «Кубанский государственный аграрный университет имени И. Т. Трубилина, Краснодар, Россия

В статье представлены принципы разработки мобильных приложений под операционную систему iOS. Подробно рассмотрены распространенные шаблоны проектирования и их особенности

Ключевые слова: IOS, APPLE, OBJECTIVE-C, SWIFT, АРХИТЕКТУРА, ОПЕРАЦИОННАЯ СИСТЕМА, ШАБЛОН ПРОЕКТИРОВАНИЯ

Doi: 10.21515/1990-4665-127-037

UDC 004.413.2

Technical sciences

PRINCIPLES OF DEVELOPMENT OF APPLICATIONS FOR IOS OPERATING SYSTEM

Velikanova Larisa Olegovna
Cand.Econ.Sci., professor
SPIN-code = 2123-1838

Murlin Aleksey Georgeeich
Cand.Tech.Sci, associate professor
SPIN-code = 4991-8507

Gayvuk Anjela Rudikovna
master student
SPIN-code =1996-5100
Kuban State Agrarian University, Krasnodar, Russia

The article presents the principles of the development of mobile applications for the operating system called iOS. Details were considered for popular patterns and their features

Keywords: IOS, APPLE, OBJECTIVE-C, SWIFT, ARCHITECTURE, OPERATING SYSTEM, PATTERN

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. Смартфоны, планшеты и другие устройства имеют значимую роль как в повседневной жизни, так и на работе: стало возможным легко и быстро прочитать файл, зайти на почту, напечатать документ при помощи сетевого принтера и многое другое. В связи с увеличением продаж мобильных устройств, постепенно сформировалось отдельное направление программного обеспечения – мобильные приложения. Технологии вливаются в бизнес-процессы современных предприятий все с большей силой, постепенно становясь все более удобными в использовании и

захватывают все больше отраслей в компании. Но встает вопрос об оперативности и удобстве получения данных – и тут на помощь электронному бизнесу приходят мобильные технологии. Трудно назвать бизнес, который бы не был затронут нарастающей волной устройств, приложений и технологий, все более активно используемых покупателями, партнерами и сотрудниками. Мобильные технологии – самая последняя и наиболее актуальная тенденция развития рынка информационных технологий для автоматизации бизнес-процессов. Смартфоны и планшеты все больше проникают в корпоративную среду и превращаются для многих руководителей и сотрудников из личного мобильного устройства в рабочий инструмент. Многие компании и организации задумываются о том, чтобы централизованно внедрять и использовать эти технологии в интересах бизнеса.

Разработка программ для мобильных устройств в сфере бизнеса необходима для продвижения товаров или услуг широкой аудитории. Качественное и удобное приложение пользователи будут активно использовать и рекомендовать друзьям и знакомым. Если создавать приложение с учетом интересов предполагаемых пользователей, то это повысит лояльность и доверие к бренду, а так же поможет оптимизировать выполнение решаемых задач. Так, например, мобильный банк дает возможность контролировать, оплачивать счета, осуществлять переводы, где бы вы ни были. Разработка мобильных приложений актуальна во многих сферах, таких как транспортные услуги, сети ресторанов и кафе, интернет-магазины, журнальный бизнес, организация корпоративного общения и многих других предприятия производства и торговли продукцией.

На данный момент существует наиболее популярные операционной системы (ОС) для мобильных устройств: ОС iOS – компании Apple и ОС Android – компании Google. В четвёртом квартале 2016 года совместная

доля мобильных ОС от Apple и Google на мировом рынке составляет 98,4%, что на 2% больше, чем аналогичный показатель в минувшем году – 96,4%.

В дальнейшем, как можно предположить, доля Android и iOS на рынке будет лишь увеличиваться. Платформа Windows Phone, которую никак нельзя назвать конкурентом мобильным ОС от Apple и Google, продолжает сдавать позиции, а другие всё так же топчутся на месте.

Возьмем за основу мобильную ОС iOS и рассмотрим принципы разработки программных продуктов для мобильных устройств на ее примере. Для грамотного создания приложения необходимо учитывать принципы простоты, удобства и интуитивно-понятного управления. При проектировании интерфейса необходимо разработать сценарии того, как пользователь будет использовать программу, например какие жесты наиболее удобны. ОС iOS поддерживает множество способов манипуляции с сенсорным экраном благодаря функции Multi-touch, что повышает удобство работы с программным продуктом.

Написания приложения под ОС iOS осуществляется на компьютерах Mac 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 кодирования, создания и отладки приложений);

– Interface Builder (для разработки пользовательского интерфейса);

– инструменты для анализа поведения и производительности;

– десятки дополнительных инструментов.

Рассмотрим подробнее языки программирования. Начнем с Objective-C, который является расширением языка программирования C и предоставляет объектно-ориентированные возможности, а так же 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.

Также в Objective-C добавлена поддержка объектно-ориентированного программирования в стиле Smalltalk, т.е. объектам посылаются сообщения вместо вызова метода.

По аналогии с интерфейсами в Java, в Objective C есть протоколы. Протокол определяет набор селекторов, которые должен поддерживать объект, реализующий протокол. Протоколы должны описывать методы, которые реализуются определенным классом.

Основные задачи для которых они используются:

– ожидание, что класс поддерживающий протокол выполнит описанные в протоколе функции;

– поддержка протокола на уровне объекта, не раскрывая методы и реализацию самого класса;

– в виду отсутствия множественного наследования - объединить общие черты нескольких классов.

Swift является более молодым и, на данный момент, стремительно развивающимся языком программирования. Он разрабатывается компанией Apple, как будущая замена Objective-C. Swift задумывался как быстрый и эффективный язык программирования с откликом в реальном времени, который легко можно вставить в готовый код Objective-C. Теперь

разработчики могут не только писать более надёжные и безопасные коды, но также экономить время и создавать приложения с расширенными возможностями.

Swift берет довольно многое из Objective-C, однако он определяется не указателями, а типами переменных, которые обрабатывает компилятор. По аналогичному принципу работают многие скриптовые языки. В то же время, он предоставляет разработчикам многие функции, которые прежде были доступны в C++ и Java, такие как определяемые наименования, так называемые обобщения и перегрузка (overloading) операторов.

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;
– 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).

Рассмотрим каждый из них.

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. Традиционное представление MVC модели представлено на рисунке 1.

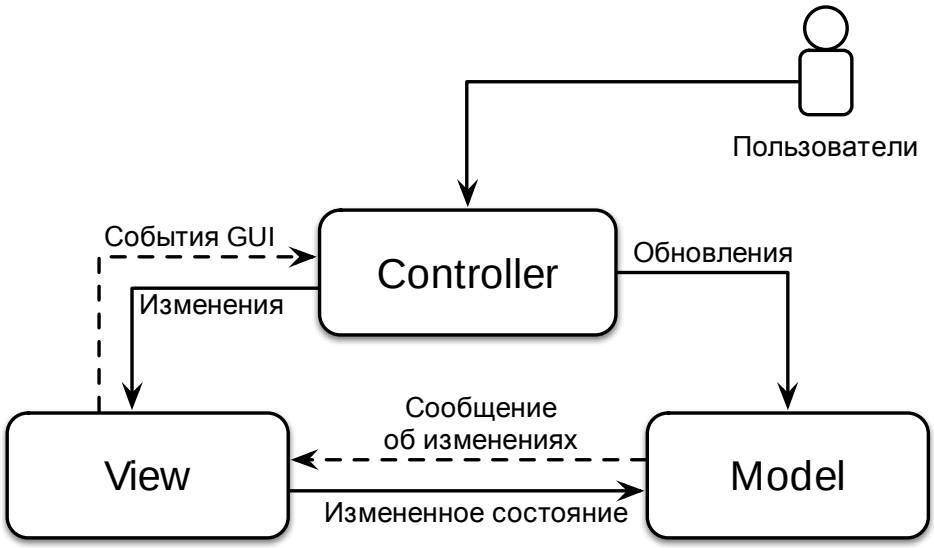

Рисунок 1 – Традиционная MVC модель

Модель (Model) содержит основные данные, например, переменные, подключения к внешним RSS-каналам или изображения, подробные функции и числовую информацию. Этот слой полностью отделяется от визуального оформления, поэтому можно легко изменить вид дисплея, и это не повлияет на данные.

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.

Но традиционная MVC архитектура не применима к современной разработке приложений под мобильную ОС IOS, поэтому компания Apple

модернизировала данный шаблон проектирования под свою архитектуру, которая называется Cocoa MVC (рисунок 2).

Здесь контроллер (controller) является посредником между видом и моделью, так что они не взаимодействуют друг с другом.

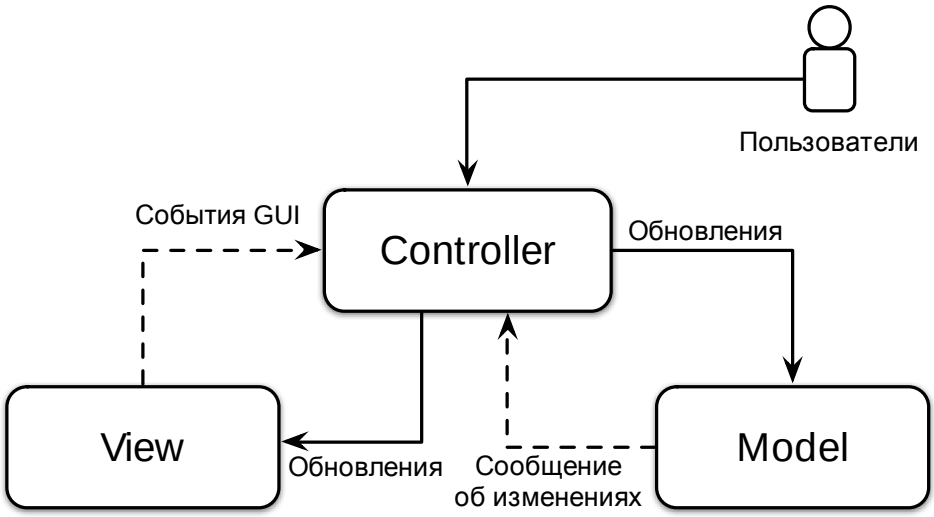

Рисунок 2 – Cocoa MVC модель

Особенностью данной шаблона проектирования MVC является то, что контроллер (controller) настолько вовлечен в жизненный цикл вида (view) посредством подкласса UIViewController, поэтому трудно сказать, что они действительно разделены.

Рассмотрим данную архитектуру с точки зрения качеств, выделенных выше:

– распределение – вид (view) и модель (model) в данном случае полностью разделены, но вид (view) и контроллер (controller) тесно связаны между собой;

– тестируемость – из-за неполного распределения можно тестировать только модель (model);

– простота использования – используется меньшее количество кода по сравнению с остальными моделями, кроме того является самой распространенной моделью.

Сосоа MVC является лучшим шаблоном проектирования с точки зрения простоты и быстрой реализации.

Рассмотрим следующую архитектуру – MVP является шаблоном проектирования производным от MVC (рисунок 3). Основной задачей MVP является сделать вид (view) повторно используемым.

Presenter, как и контроллер (controller) управляет видом (view) и моделью (model), но при этом вид (view) каждый раз не пересоздается.

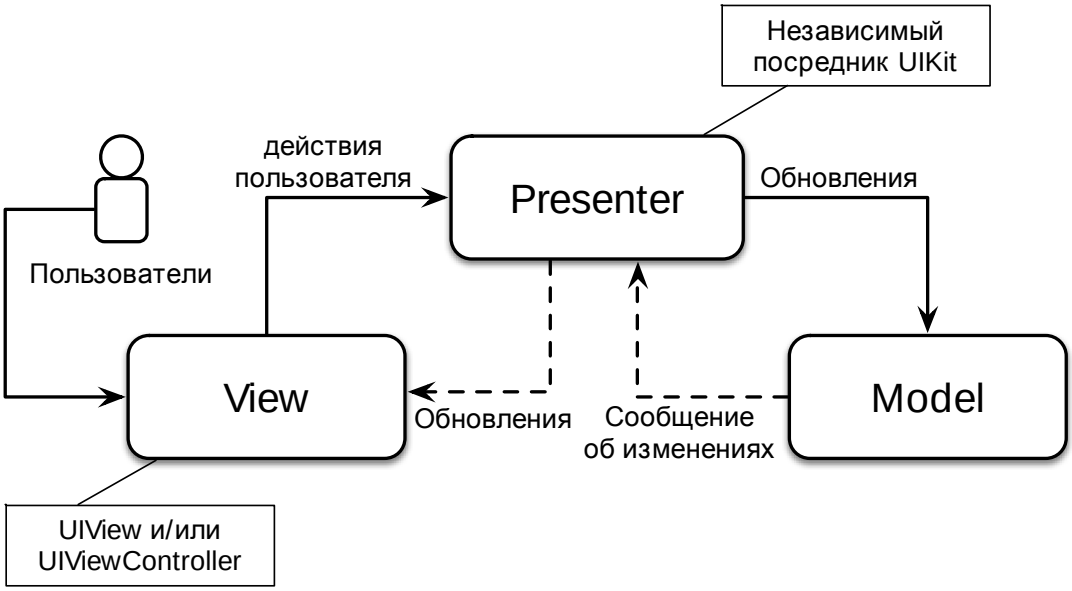

Рисунок 3 – MVP модель

Подклассы UIViewController фактически принадлежат виду (view), а не presenter и это различие обеспечивает превосходную тестируемость.

Рассмотрим вкратце особенности MVP:

– распределение – большая часть обязанностей разделена между presenter и моделью (model);

– тестируемость – можно проверить большую часть бизнес-логики;

– простота использования – количество кода значительно увеличится по сравнению с MVC моделью, но в то же время, идея MVP довольно простая.

В итоге шаблон проектирования MVP означает превосходную тестируемость и большое количество кода.

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.

Рассмотрим 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 4.

Вид (view) и модель (model) уже были разобраны, но в данном случае посредником выступает модуль view model.

View model готовит все данные, которые необходимо отобразить на экране. View model Он никогда не ссылается на вид (view), вместо этого он постоянно следит за любыми изменениями view model, и как только изменения произойдут, они сразу будут отображены на экране.

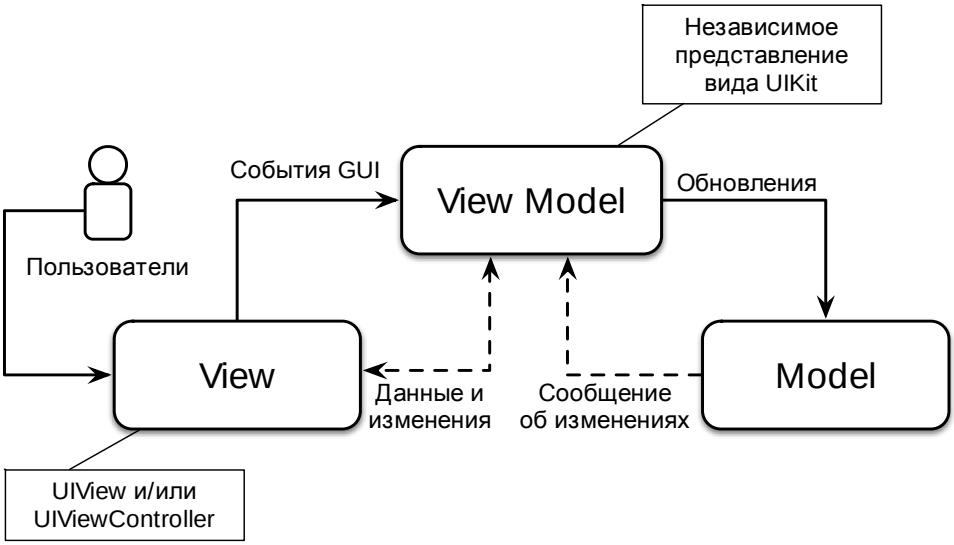

Рисунок 4 – MVVM модель

И снова вернемся к оценке шаблона проектирования:

– распределение – по сути, вид (view) MVVM имеет больше обязанностей, чем вид (view) MVP, поскольку первый шаблон обновляет состояние системы с точки зрения модели (model), когда второй направляет все события к Presenter и не обновляет себя;

– тестируемость – полное распределение классов позволяет легко проводить модульные тесты;

– простота в использовании – объем кода тоже больше, чем в MVC. Но по сравнению с MVP код более лаконичный так, как не нужно направить все события из вида (view) к presenter и обновлять его вручную.

Поэтому 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.

Все шаблоны проектирования являются актуальными на данный момент. Выделим некоторые моменты для выбора шаблона проектирования:

– MVVM используется в ситуации, когда возможно связывание данных без необходимости ввода специальных интерфейсов;

– MVP используется в ситуации, когда невозможно связывание данных;

– MVC используется в ситуации, когда связь между видом (view) и другими частями приложения невозможна.

Выбор определенного паттерна остается за разработчиком, который определяет более подходящий шаблон для конкретного случая. Мобильный технологии в бизнесе является логическим продолжением электронного документооборота и часто понимается как его подмножество. Обычно под мобильным бизнесом понимают традиционные услуги электронного бизнеса, только предоставляемые

через беспроводные сети. Однако мобильный бизнес обеспечивает и принципиально новые сервисы, которые не могут быть реализованы с помощью обычной сети, поэтому он выходит за его границы. Следовательно, актуальность разработки мобильного приложения для управления бизнес-процессами на предприятии не вызывает сомнения.

Список литературы

1. Масленникова О.Е., Попова И.В. Основы искусственного интеллекта Учебное пособие. Магнитогорск : МаГУ, 2008, 282 с.
2. Нойбург М. Программирование для iOS 7. Основы Objective-C, Xcode и Cocoa. Вильямс, 2014, 384 с.
3. Далримпл М., Кнастер С. Objective-C 2.0 и программирование для Mac. Учебник и примеры. Вильямс, 2009, 320 с.
4. Великанова Л.О., Гайвук А.Р. Применение мобильных технологий для автоматизации бизнес-процессов на торговом предприятии. Экономическое прогнозирование: модели и методы. Материалы XII международной научно-практической конференции 17-19 ноября 2016 г. Воронеж, 2016, 314 с.

References

1. Maslennikova O.E., Popova I.V. Osnovy iskusstvennogo intellekta Uchebnoe posobie. Magnitogorsk : MaGU, 2008, 282 c.
2. Nojburg M. Programmirovanie dlja iOS 7. Osnovy Objective-C, Xcode i Cocoa. Vil'jams, 2014, 384 s.
3. Dalrimpl M., Knaster S. Objective-C 2.0 i programmirovanie dlja Mac. Uchebnik i primery. Vil'jams, 2009, 320 s.
4. Velikanova L.O., Gajvuk A.R. Primenenie mobil'nyh tehnologij dlja avtomatizacii biznes-processov na torgovom predprijatii. Jekonomicheskoe prognozirovanie: modeli i metody. Materialy HII mezhdunarodnoj nauchno-prakticheskoj konferencii 17-19 nojabrja 2016 g. Voronezh, 2016, 314 s.