ПРОЦЕСС ВЫБОРА АРХИТЕКТУРЫ ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

THE PROCESS OF CHOOSING THE ARCHITECTURE FOR A MOBILE APP

Курганова Александра Геннадьевна, магистрант, Московский государственный технический университет имени Н.Э. Баумана, г. Москва

Kurganova A.G. kurganova06.1998@gmail.com

Аннотация

В данной статье рассматривается проблема выбора архитектуры мобильного приложения при его создании. При разработке архитектуры мобильного приложения прежде всего изучают все существующие архитектуры на данным момент, сравнивают их между собой и после выбора определенной начинают разработку. При сравнении архитектур между собой используется метод по Парето и метод ранжирования Борда. Согласно методу Парето лучшим является тот вариант, который по всем показателям был бы не хуже первого, а хотя бы по одному показателю лучше него. Согласно методу Борда варианты ранжируются по каждому показателю в порядке убывания с присвоением им соответствующих значений ранга, затем подсчитывается суммарный ранг по каждой архитектуре. Лучшим результатом станет архитектура с максимальным значением суммарного ранга.

Annotation

This article discusses the problem of choosing the architecture of a mobile application when creating it. When developing the architecture of a mobile application, first of all, they study all the existing architectures at the moment, compare them with each other, and after selecting a specific one, they begin development. When

comparing architectures with each other, the Pareto method and the Board ranking method are used. According to the Pareto method, the best option is the one that would be no worse than the first one in all indicators, but at least one indicator better than it. According to the Board method, the variants are ranked by each indicator in descending order with the corresponding rank values assigned to them, then the total rank for each architecture is calculated. The best result will be the architecture with the maximum value of the total rank.

Ключевые слова: архитектура, мобильное приложение, анализ архитектуры, критерии сравнения

Keywords: architecture, mobile app, architecture analysis, comparison criteria

Введение

В любом мобильном приложении необходимо проектировать архитектуру кода. Плохая архитектура кода ведет к усложнению внесения изменений в него, тестированию, к снижению читаемости, то есть общего понимания работы программы, что в свою очередь увеличивает количество ошибок и приводит к неверной работе программы. Проектировать хорошую архитектуру – задача из непростых, требует творческого подхода, знаний и опыта, но есть готовые решения. Эти решения называются паттернами проектирования, которые будут разобраны в данной статье.

Анализ существующих архитектур

Существует несколько архитектур, которые используются для построения взаимосвязей между компонентами приложения. Это – MVC, MVP, MVVM, VIPER.

Теперь нужно разобраться с базовым набором компонентов при разработке приложения под платформу iOS для того, чтобы далее было возможно анализировать архитектуры мобильных приложений, которые приведены выше.

Приложение состоит из множества экранов (сцен), каждая сцена содержит в себе верстку (view, layout) – отображение элементов на экране и программную реализацию (view controller).

Есть два принципа построения layout:

1. Верстка кодом
2. Верстка с помощью storyboard

Для использования любого из этих принципов, нужно уметь устанавливать взаимосвязи между элементами. Для верстки кодом это – auto layout и layout library (pure layout, flex layout) и другие. Для storyboard – это UIInterfaceBuilder.

Для реализации программной части используется коллекция библиотек Cocoa Touch, которая включает в себя UIKit для построения визуального интерфейса. Программной реализацией экрана, на текущем этапе, будем называть - View Controller.

Из всего вышеперечисленного получаем, что вся логика по взаимодействию пользователя с элементами на экране ложится на View Controller, что включает в себя: хранение моделей данных, запросы по API, работу с внутренний базой данных, а также обработку пользовательских действий. Следовательно, делаем вывод, что View Controller – это объемный модуль, который за отвечает за обработку всего.

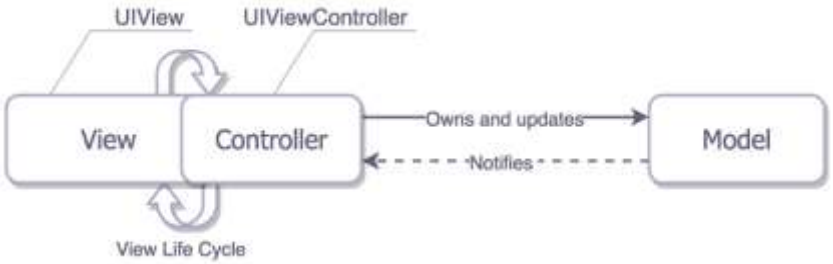
Теперь возникает проблема, что программа содержит в себе один или несколько огромных модулей View Controller, где находится абсолютно все, что затрудняет разработку и ориентирование в коде.

Данная проблема называется – Massive View Controller, то есть это – котроллер, который отвечает за все.

Чтобы решить эту проблему – необходимо выбрать правильную архитектуру в рамках проекта. В каждой архитектуре предложены способы разделения ответственности между компонентами приложения, чтобы решить проблему Massive View Controller.

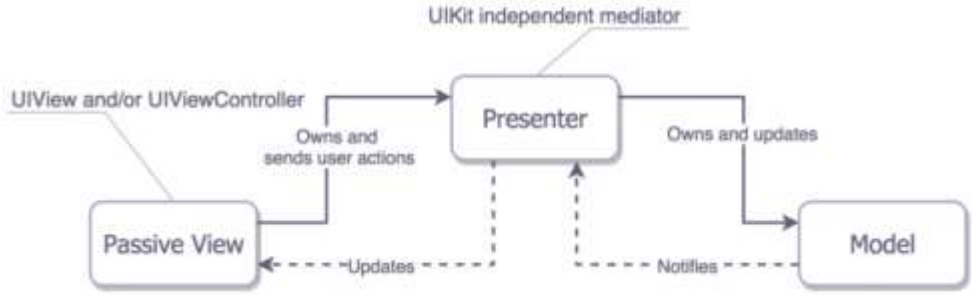
Рассмотрим несколько базовых архитектур:

MVC – Model View Controller



В этой архитектуре применяется вынесение задачи хранения моделей из контроллера в отдельный компонент – Model, соответственно, каждая модель находится в своем файле. При этом, в контроллере все еще находится бизнес-логика, связанная с обращением в методы API, работой с базой данных и обработкой действий пользователя.

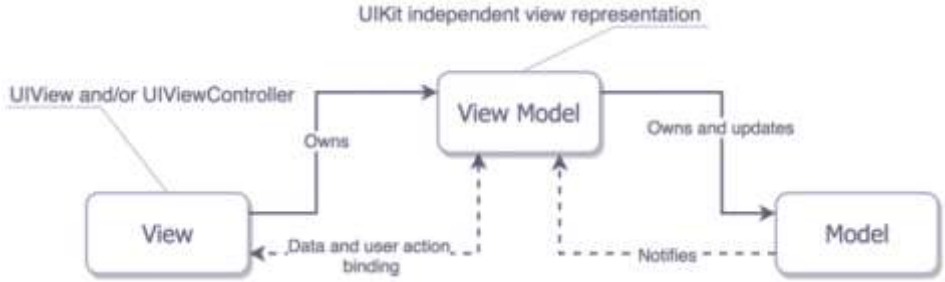
Архитектура MVP – Model View Presenter



В архитектуре MVP, логику работы с пользовательским интерфейсом вынесли в отдельный компонент – Presenter, из-за чего повышается уровень разделения ответственности. В данной архитектуре заложено «честное» разделение слоев, где есть 3, действительно, отдельных слоя.

Соответственно, мы получаем хорошую распределенность между компонентами, отличную тестируемость, так как большая часть бизнес логики вынесена в модуль и простоту использования.

Архтектура MVVM – Model View View-Model



Архитектура MVVM наследует все положительные характеристики от архитектуры MVP: в ней нет прямой связи между View и Model, так как View выделена как независимый слой.

Таким образом слой View-Model является связкой между двумя уже независимыми слоями View и Model, а вся бизнес-логика приложения принадлежит этому слою. Благодаря слою View–Model появляется возможность четко разделить работу с данными, бизнес-логикой и визуальными элементами.

Критерии сравнения для архитектур мобильных приложений

После первичного анализа каждой из архитектур необходимо сравнить их между собой по следующим критериям:

- Простота использования
- Принцип разделения ответственности
- Сложность внедрения нового функционала
- Тестируемость
- Документация

Дадим кодовые названия для архитектур в таблице 4:

Таблица 4. Кодовые названия архитектур.

Архитектура	Кодовое название
MVC – Model View Controller	B1
MVP – Model View Presenter	B2
MVVM – Model View View-Model	B3

Таблица 5. Описание критериев для сравнения архитектур.

Критерий	Код критерия	Описание
Простота использования	K1	Понятие концепции архитектуры и того, как ее применить к приложение
Принцип разделения ответственности	K2	На сколько слои архитектуры тесно связаны между собой

Сложность внедрения функционала	К3	На сколько долго будет делаться расширение приложения
Тестируемость	К4	Оценивание стабильности работы
Степень решения MVC	К5	MVC – Massive View Controller [2]. Проблема, при которой один контроллер отвечает за все

Для возможности сравнения значения качественных критериев необходимо их перевести в количественные показатели. Шкала количественных показателей для каждого из критериев приведены ниже в таблицах 6,7,8,9,10.

Таблица 6. Вербально-числовая шкала для критерия К1.

Критерий	Код критерия
Очень простая концепция архитектура, быстро встраиваемая в приложение	1,0
Простая концепция архитектуры, быстро встраиваемая в приложение	0,9
Средняя сложность концепции архитектуры, требует времени для внедрения	0,8
Сложная концепция архитектуры, требует очень много времени для внедрения	0,7

Таблица 7. Вербально-числовая шкала для критерия К2.

Критерий	Код критерия
Максимально полное разделение ответственности	1,0
Полное разделение ответственности	0,9
Частичное разделение ответственности	0,8
Слабое разделение ответственности	0,7

Таблица 8. Вербально-числовая шкала для критерия К3.

Критерий	Код критерия
Очень быстрое внедрение нового функционала	1,0
Быстрое внедрение нового функционала	0,9
Требуется время для внедрения нового функционала	0,8
Очень долгое внедрение нового функционала	0,7

Таблица 9. Вербально-числовая шкала для критерия К4.

Критерий	Код критерия
Максимально полная тестируемость	1,0
Полная тестируемость	0,9
Частичная тестируемость	0,8
Очень плохая тестируемость	0,7

Таблица 10. Вербально-числовая шкала для критерия К5.

Критерий	Код критерия
Полное решение проблемы MVC	1,0
Близкое к полному решению проблемы MVC	0,9
Частичное решение проблемы MVC	0,8
Близкое к частичному решению проблемы MVC	0,7

Таблица 11. Качественные и количественные показатели критериев.

Код критерия	Критерий	Значение критериев вариантов архитектуры		
		B1	B2	B3
K1	Простота использования	1,0	0,85	0,8
K2	Принцип разделения ответственности	0,7	0,8	1,0

К3	Сложность внедрения функционала	0,7	0,8	0,85
К4	Тестируемость	0,8	0,9	0,9
К5	Документация	0,8	0,85	0,85

Оценивание архитектур по методу Парето

Далее необходимо оценить варианты на Парето-оптимальность (таблица 12).

Таблица 12. Сравнение вариантов на Парето-оптимальность

Вариант архитектуры	Вариант архитектуры		
	В1	В2	В3
В1	0	0	0
В2	0	0	0
В3	0	0	0
Результат сравнения	0	0	0
Парето-оптимальность варианта	Да	Да	Да

Все варианты являются Парето-оптимальными.

Сравнение архитектур методом ранжирования Борда

Для выбора лучшего варианта архитектуры будет использоваться метод ранжирования Борда. Для этого составим таблицу рангов, в которой каждому критерию присваивается ранг в зависимости от других критериев (таблица 13).

Далее необходимо подсчитать суммы рангов используя формулу 1:

$$\square_{\square} = \sum_{\square=1}^{\square} \square_{\square\square} \quad (1)$$

Таблица 13. Ранжирование вариантов

Код критерия	Ранг локальных критериев		
	B1	B2	B3
K1	1	2	3
K2	3	2	1
K3	3	2	1
K4	3	1,5	1,5
K5	3	1,5	1,5
Итого	13	9	8

Для нахождения лучшего варианта воспользуемся формулой ниже (формула 2):

$$\square_{\square} = \square\square\square_{\square}\ \square_{\square} \qquad (2)$$

В результате подсчетов получаем:

$$\square_{\square} = \square_{3} = 8$$

Ранжирование вариантов по методу Борда показывает, что архитектуры следует расположить в следующем порядке: $\square 3 \succ \square 2 \succ \square 1$. При этом архитектура MVVM под кодовым названием B3 является лучшей среди сравниваемых, а соответственно именно данная архитектура будет использована для разрабатываемого мобильного приложения.

Заключение

Процесс выбора и разработки архитектуры мобильного приложения довольно долгий и ресурсозатратный. Хороший разработчик всегда стремится нивелировать количество ошибок, уменьшить потраченное время на внесение нового функционала, при этом не ухудшив качество кода. Признаками плохой архитектуры являются повторение компонентов кода, сильная связанность

между ними. Поэтому для построения хорошей и подходящей архитектуры мобильного приложения, необходимо проанализировать все существующие решения и сравнить их между собой по определенным параметрам, которые в той или иной степени соответствуют функционалу конечного разрабатываемого мобильного приложения.

Литература

1. Сергей Петраков. Введение в проблематику iOS приложений [Электронный ресурс] / Сергей Петраков. – Электронные данные. – Москва: [б.и], 2019. – Режим доступа: https://medium.com/@petrachkovsergey/введение-в-проблематику-архитектуры-ios-приложений-da766ae5f74c, свободный.
2. Блог компании Badoo. Архитектурные паттерны в iOS [Электронный ресурс] / Блог компании Badoo. – Электронные данные. – Москва: [б.и], 2016. – Режим доступа: https://habr.com/ru/company/badoo/blog/281162/, свободный.
3. Электронный ресурс удаленного доступа (https://developer.apple.com/xcode/interface-builder/)
4. Архитектурный дизайн мобильных приложений. Часть 1. [Электронный ресурс] / Блог компании red_mad_robot. – Электронные данные. – Москва: [б.и], 2014. – Режим доступа: https://habr.com/ru/company/redmadrobot/blog/246551/, свободный.
5. Архитектурный дизайн мобильных приложений: часть 2. [Электронный ресурс] / Блог компании red_mad_robot. – Электронные данные. – Москва: [б.и], 2015. – Режим доступа: https://habr.com/ru/company/redmadrobot/blog/251337/, свободный.
6. Чистая архитектура на Android и iOS. [Электронный ресурс] / Анна Гуляева. – Электронные данные. – Москва: [б.и], 2020. – Режим доступа: https://apptractor.ru/develop/chistaya-arhitektura-na-android-i-ios.html, свободный.

Literature

1. Sergey Petrakov. Introduction to the problems of iOS applications [Electronic resource] / Sergey Petrakov. - Electronic data. - Moscow: [b. i], 2019. - Access mode: https://medium.com/@petrachkovsergey/введение-в-проблематику-архитектуры-ios-приложений-da766ae5f74c, free.
2. Badoo company blog. Architectural patterns in iOS [Electronic resource] / Blog of the Badoo company. - Electronic data. - Moscow: [b. i], 2016. - Access mode: https://habr.com/ru/company/badoo/blog/281162/, free.
3. Electronic resource of remote access (https://developer.apple.com/xcode/interface-builder/)
4. Architectural design of mobile applications. Part 1. [Electronic resource] / Blog of the company red_mad_robot. - Electronic data. - Moscow: [b. i], 2014. - Access mode: https://habr.com/ru/company/redmadrobot/blog/246551/, free.
5. Architectural design of mobile applications: part 2. [Electronic resource] / Blog of the company red_mad_robot. - Electronic data. - Moscow: [b. i], 2015. - Access mode: https://habr.com/ru/company/redmadrobot/blog/251337/, free.
6. Clean architecture on Android and iOS. [Electronic resource] / Anna Gulyaeva. - Electronic data. - Moscow: [b. i], 2020. - Access mode: https://apptractor.ru/develop/chistaya-arhitektura-na-android-i-ios.html, free.