Министерство образования и науки Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Факультет информатики
Кафедра теоретических основ информатики(ТОИ)

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Руководитель ООП
кандидат техн. наук , доцент
_____А.Л. Фукс
«1» 06 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ИНТЕГРАЦИИ СОЦИАЛЬНЫХ СЕТЕЙ НА ПЛАТФОРМЕ IOS

по основной образовательной программе подготовки бакалавров
02.03.02 – Фундаментальная информатика и информационные технологии

Никулин Семен Викторович

Руководитель ВКР, ассистент
кафедры теоретических основ
информатики
_____ Д.В. Дружинин
подпись

«1» 06 2017 г.

Автор работы,
студент группы 1431
_____ С.В. Никулин
подпись

Томск-2017

Реферат

Бакалаврская работа 48 с., 31 рис., 1 таб., 8 источников, 1 прил.

IOS, SWIFT, XCODE, СОЦИАЛЬНАЯ СЕТЬ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ

Объект исследования – приложение для iOS, архитектура системы.

Цель исследования – разработка мобильного приложения на платформе ios, позволяющего интегрировать несколько социальных сетей для просмотра, создания, редактирования и публикации постов.

Метод исследования – теоретическое исследование и практическая реализация.

Область применения – организации, для распространения в AppStore.

Результаты работы – изучена среда разработки Xcode для устройств от компании Apple; исследована архитектура ОС iOS; проанализированы API социальных сетей; спроектирована структура мобильного приложения; разработано мобильное приложение, доступное через AppStore.

Введение

В современном мире мобильный телефон и социальные сети стали неотъемлемой частью жизни каждого человека. Трафик мобильного интернета растет с каждым годом. Мобильные приложения имеют преимущество над компьютерными, тем, что ими можно пользоваться в любом месте мира: по пути на работу, в кафе на обеде, на прогулке в парке. Мобильное программное обеспечение решает большинство повседневных задач человека, что позволяет свести время потраченное на рутинные дела к нулю.

Социальные сети позволяют добавлять друзей, общаться пользователям текстовыми сообщениями, обмениваться фотографиями и видеозаписями, вступать в группы отличающимися их интересам и быть в курсе последних новостей и событий. В наше время социальные сети могут побороться за внимание с такими крупными медийными аппаратами как новостные порталы, видеохостинги, мессенджеры, развлекательные порталы.

Большинство юридических компаний и индивидуальных предпринимателей ведут свою деятельность в социальных сетях. Социальные сети позволяют продвигать свою бизнес идею, наращивать новых потенциальных клиентов, оповещать свою аудиторию о специальных условиях и предложениях, вести блог о своей деятельности, реагировать на отзывы и вопросы клиентов.

С каждым годом появляется новая социальная сеть. Сейчас насчитывается около 10 популярных социальных сетей. Самые крупные это Facebook, Twitter, Linkedin, Instagram, Pinterest. Работать отдельно с каждой очень хлопотно, так как любая социальная сеть имеет свои особенности, плюсы и минусы для продвижения своей компании. В крупных компаниях имеется несколько аккаунтов для социальной сети, что значительно увеличивает эффективность, но также и увеличивает трудоемкость по

использованию такого подхода.

Чтобы устранить эти проблемы, в рамках данной дипломной работы было решено разработать мобильное приложение на платформе iOS с возможностью управлять множеством социальных сетей и аккаунтами принадлежащим им. Основными задачами данной работы являются:

1) Проанализировать существующие решения
2) Изучить особенности мобильной платформы iOS
3) Сформировать список требований к разрабатываемому приложению
4) Ознакомиться с различными API социальных сетей
5) Спроектировать архитектуру приложения
6) Реализовать приложение
7) Протестировать приложение

1.Обзор существующих решений

В данной предметной области существует не много готовых решений. Рассмотрим несколько самых популярных на данный момент, помогающих работать с социальными сетями на мобильных устройствах

1.1.Facebook Pages Manager

Рис.1

Рис.2

Приложение Facebook Pages Manager предоставляет пользователю управлять аккаунтами социальной сети Facebook. Приложение поддерживает добавление нескольких аккаунтов.

Достоинства:

- Поддержка сбора общей статистики страницы(Рисунок 1)
- Получение информации о каждом посте(лайки, репосты, просмотры)(Рисунок 2)

Недостатки:

- Нет поддержки нескольких социальных сетей
- Нет возможности планировать посты на определенное время

1.2.Buffer

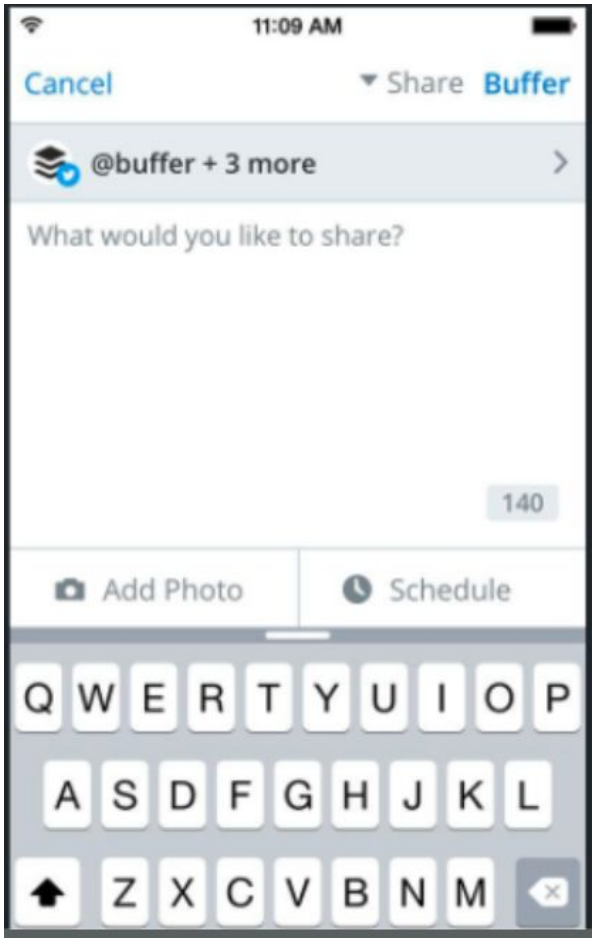


Рис.3

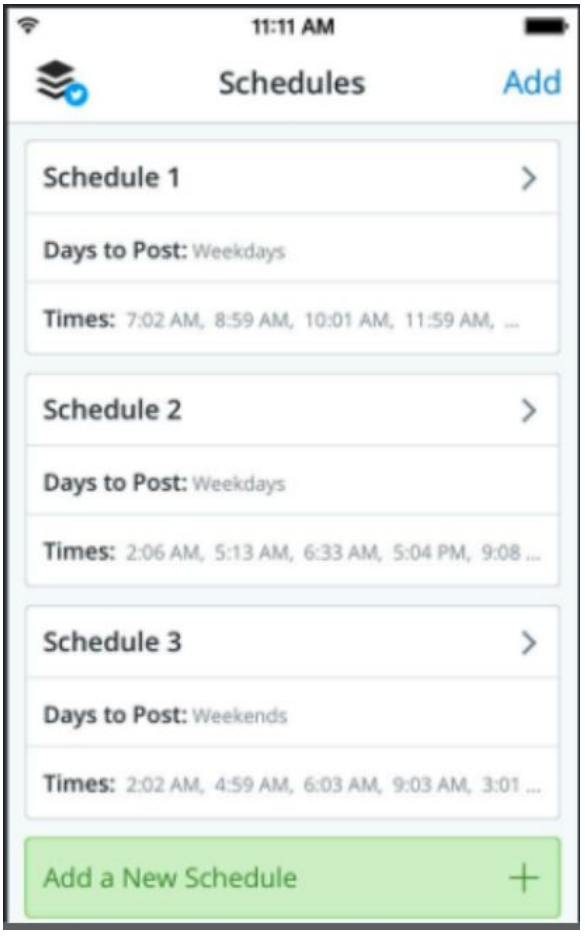


Рис.4

Приложение Buffer объединяет в себе несколько социальных сетей, что позволяет создать один пост для всех социальных сетей за раз. Приложение позволяет редактировать созданные посты ранее, добавлять

фотографии(Рисунок 3).

Достоинства:

- Поддержка нескольких социальных сетей
- Возможность запланировать пост(Рисунок 4)

Недостатки:

- Поддержка одного аккаунта на социальную сеть
- Неудобная система планирования постов, нельзя просто ввести дату предполагаемого времени поста, требуется посчитать через сколько это наступит

1.3.HootSuite

Рис.5

Рис.6

Приложение HootSuite работает с социальной сетью twitter. Пользователь имеет возможность посмотреть подробную статистику по постам и страницы в целом.

Достоинства:

- Поддержка планирования поста на конкретную дату(Рисунок 5)
- Получение сообщений из социальных сетей(Рисунок 6)

Недостатки:

- Нет поддержки добавления различных сетей
- Нет поддержки создания поста с видео

Таким образом, уже существующее программное обеспечение не позволяет полностью охватить весь список требований, что свидетельствует об актуальности этой работы.

2.Функциональность приложения

2.1.Функциональные требования

Система должна представлять собой мобильное приложение работающее на операционной система iOS. Приложение должно иметь доступ в интернет, а также поддерживать различные устройства, такие как iPhone, iPad, iPod. Рассмотрим основные компоненты проектирования, описанные в руководстве[1].

Система должна иметь возможность:

1. Регистрировать пользователя.
2. Восстанавливать пароль.
3. Авторизовать пользователя.
4. Добавить аккаунт социальной сети.
5. Удалять аккаунт социальной сети.
6. Создавать пост.
7. Использовать камеру для добавления фото или видео к посту.
8. Использовать галерею для добавления фото или видео к посту.
9. Редактировать пост.
10. Назначать время автоматического опубликования поста.
11. Опубликовать пост моментально.
12. Просмотреть список запланированных на публикацию постов.
13. Просмотреть список опубликованных постов.
14. Получать и обрабатывать push уведомления.

2.2.Диаграммы вариантов использования

На рисунках 7-9 показаны диаграммы вариантов использования приложения.

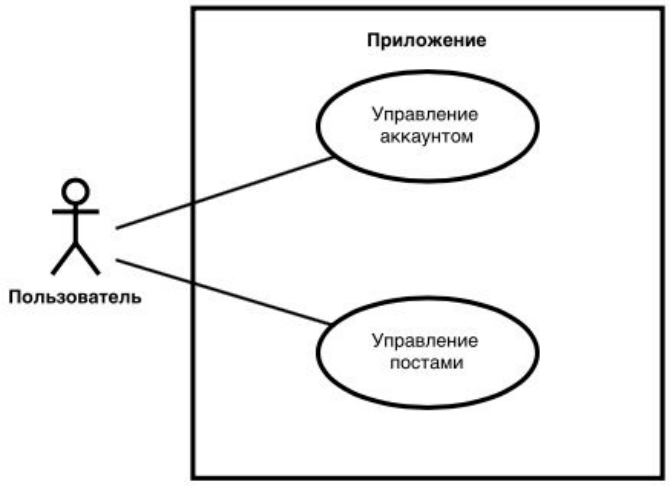
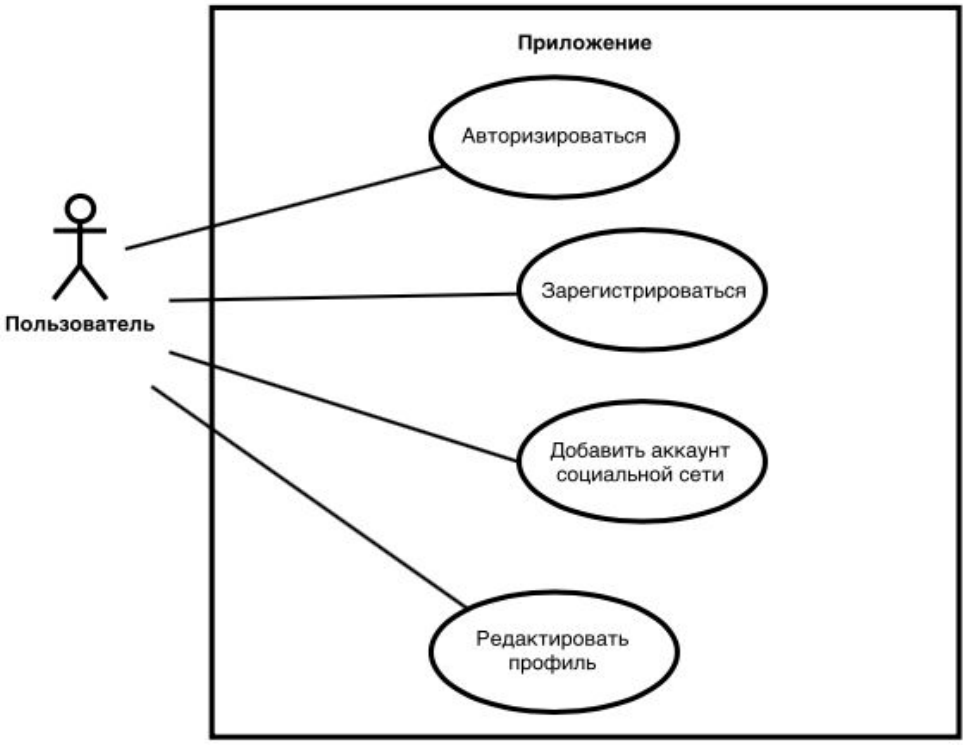


Рисунок 7 - общая диаграмма вариантов использования.



.Рисунок 8 - диаграмма вариантов использования “Управление аккаунтом”.

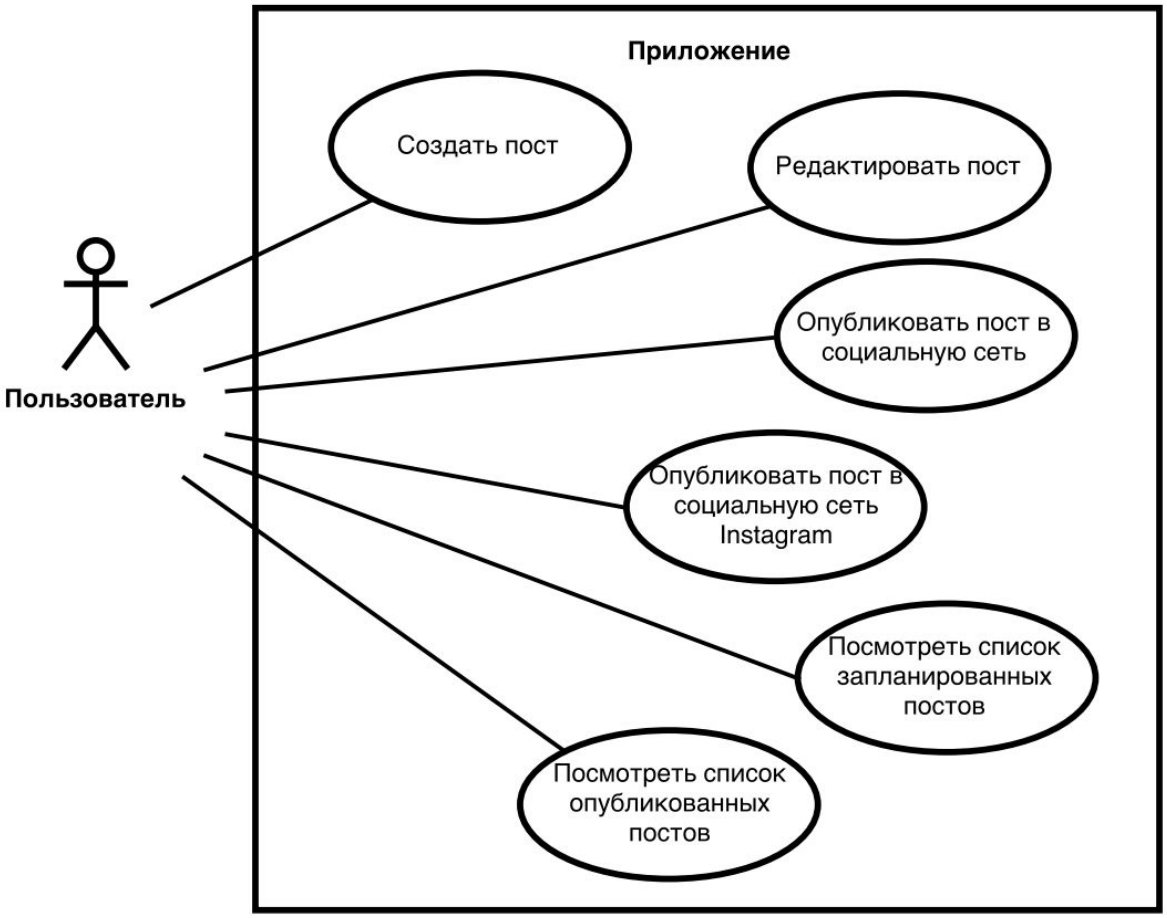


Рисунок 9 - диаграмма вариантов использования “Управление постами”.

2.3.Сценарии вариантов использования

Сценарий варианта использования “Авторизоваться”:

1. Пользователь вводит логин и пароль в текстовые поля на экране авторизация.
2. Система проверяет данные на валидность.
3. Пользователь нажимает на кнопку «Авторизоваться».
4. Система отправляет на сервер введенные данные.
5. Система получает ответ от сервера и показывает ошибку если пользователь ввел некорректные данные или направляет его на главный экран.

Сценарий варианта использования “Зарегистрироваться”:

1. Пользователь вводит username, почту, пароль, проверочный пароль в текстовые поля на экране регистрация.
2. Система проверяет, что пароли совпадают.
3. Пользователь нажимает на кнопку “Зарегистрироваться”.
4. Система отправляет на сервер введенные данные.
5. Система получает ответ, после чего открывает главный экран приложения если авторизация прошла успешно, в другом случае показывает ошибку.

Сценарий варианта использования “Добавить аккаунт социальной сети”:

1. Пользователь нажимает на кнопку “Добавить аккаунт” в левом меню главного экрана.
2. Система показывает экран со списком социальных сетей, которые можно добавить.
3. Пользователь выбирает социальную сеть.
4. Система открывает экран авторизации для выбранной социальной сети.
5. Пользователь вводит данные для авторизации и нажимает кнопку “Авторизоваться”.
6. Система отправляет введенные данные на сервер социальной сети.
7. Если данные были корректные система добавит новый аккаунт социальной сети, в ином случаи покажет ошибку.

Сценарий варианта использования “Редактировать профиль”:

1. Пользователь нажимает на кнопку “Настройки” в левом меню

главного экрана.

2. Система открывает экран настройки.

3. Пользователь может изменить почту и картинку, затем нажать на кнопку “Сохранить”

4. Система отправит данные на сервер и обновит данные пользователя.

Сценарий варианта использования “Создать пост”:

1. Пользователь нажимает на кнопку “Создать пост” на главном экране.

2. Система открывает экран создания поста.

3. Пользователь вводит заголовок поста, выбирает фото или видео для поста.

4. Пользователь нажимает на кнопку “Выбрать аккаунты”.

5. Система открывает экран со списком доступных аккаунтов пользователю.

6. Пользователь выбирает аккаунты и нажимает кнопку “Готово”.

7. Система возвращает пользователя на экран создания поста, а также отображает список выбранных аккаунтов социальных сетей.

8. Пользователь нажимает на кнопку «Выбрать дату».

9. Система показывает экран выбора даты.

10. Пользователь выбирает число на календаре, выставляет часы и минуты, затем нажимает кнопку “Готово”.

11. Система возвращает пользователя на главный экран, где отображается созданный новый пост в списке запланированных постов.

Сценарий варианта использования “Редактировать пост”:

1. Пользователь нажимает кнопку “Редактировать” на конкретном посте, на экране запланированных постов.

2. Система открывает экран для редактирования поста.

3. Пользователь меняет заголовок, фото или видео, выбирает аккаунты социальных сетей и выставляет дату и нажимает кнопку “Готово”.

4. Система отправляет новые данные на сервер и возвращает пользователя на главный экран.

Сценарий варианта использования “Опубликовать пост в социальную сеть”:

1. Пользователь нажимает кнопку “Опубликовать” на конкретном посте, в списке запланированных постов.

2. Система открывает экран с детальной информацией о посте.

3. Пользователь просматривает содержимое поста и нажимает кнопку “Опубликовать”.

4. Система публикует пост в социальную сеть, удаляет его из запланированного списка и добавляет в опубликованный список, возвращает пользователя на главный экран.

Сценарий варианта использования “Опубликовать пост в социальную сеть Instagram”:

1. Пользователь нажимает на кнопку “Опубликовать” на конкретном посте, который принадлежит социальной сети Instagram.

2. Система открывает экран с детальной информацией о посте.

3. Пользователь просматривает содержимое поста и нажимает кнопку “Опубликовать”.

4. Система открывает установленное приложение Instagram на экране создать пост.

5. Пользователь нажимает кнопку “Опубликовать” в приложении Instagram.

6. Система публикует пост в социальную сеть, удаляет его из

запланированного списка и добавляет в опубликованный список, возвращает пользователя на главный экран.

Сценарий варианта использования “Посмотреть список запланированных постов”:

1. Пользователь нажимает на кнопку “Запланированные” в левом меню главного экрана
2. Система открывает экран со списком запланированных постов.

Сценарий варианта использования “Посмотреть список опубликованных постов”:

1. Пользователь нажимает на кнопку “Опубликованные” в левом меню главного экрана
2. Система открывает экран со списком опубликованных постов.

3.Архитектура iOS приложений

3.1Архитектура ОС iOS

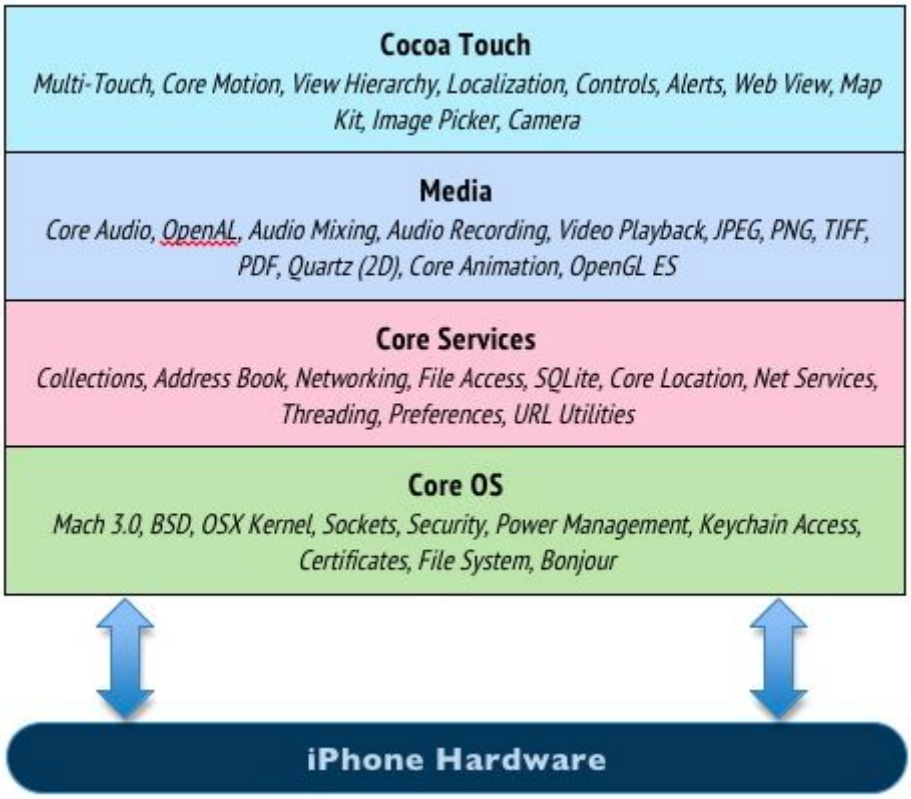


Рисунок 10 – слои ОС iOS.

Архитектуру iOS можно разделить на 4 слоя(Рисунок 10):

1) Уровень ядра(Core OS) - работает с файловой системой, контролирует действительность различных сертификатов, принадлежащие приложениям[2]. Также отвечает за безопасность всей системы вцелом. Содержит низкий уровень доступа к элементам устройства.

2) Уровень сервисов ядра(Core Service) - предоставляет доступ к таким сервисам как

- Встроенная база данных(SQLite), позволяющая создавать модель данных, а также осуществлять SQL запросы
- Файловый диспетчер(File Accesses), сервис имеющий высокий

уровень интерфейса для доступа к различным файлам принадлежащим приложению.

- Локация(Core Location) - сервис позволяющий отследить текущее местоположение устройства, а также отслеживать его длительные перемещения.
- Интернет сервис(Net Service) - сервис позволяющий получать и передавать данные через сеть интернет.

3) Уровень медиа(Media) - содержит инструменты позволяющие обрабатывать большинство форматов медиаданных, например:

- Аудио(Core Audio) - позволяет записывать, прослушивать , а также и редактировать различные аудио дорожки. Возможность достать из видео, определенную дорожку.
- Видео(Video playback) - сервис позволяющий воспроизводить, перематывать, останавливать видео. Редактирование видео подразумевает изменение расширения, длины или качества.
- Анимации(Core Animation) - библиотека позволяющая создавать анимации. При создании анимации может использоваться как показ картинок последовательно, так и изменение различных параметров элементов интерфейса, таких как высота, ширина, местоположение, видимость и т.д.

4) Уровень интерфейса(Cocoa Touch) - имеет множество элементов для создания мобильных интерфейсов, а а также предоставляет остальным слоям информацию входящую от пользователя. В него входят такие элементы как:

- Обработчик множественных нажатий(Multi-Touch) - позволяющий обрабатывать несколько нажатий на экран.
- Обработчик жестов(Core Motion) - позволяющий обработать различные жесты производящиеся на экране устройства, такие как сдвиг вправо, вниз и т.д.

- Галерея(ImagePickers) - предоставляет доступ к фотографиям и видео лежащих на устройстве. С помощью него приложения могут получить доступ и загрузить к себе файлы из галереи.
- Камера(Camera) - используется для снимком и последующим загрузкой данных в приложение.

3.1.1.Жизненный цикл iOS приложения

Приложения имеют несколько состояний, чтобы узнать как они меняются, рассмотрим жизненный цикл(Рисунок 11).

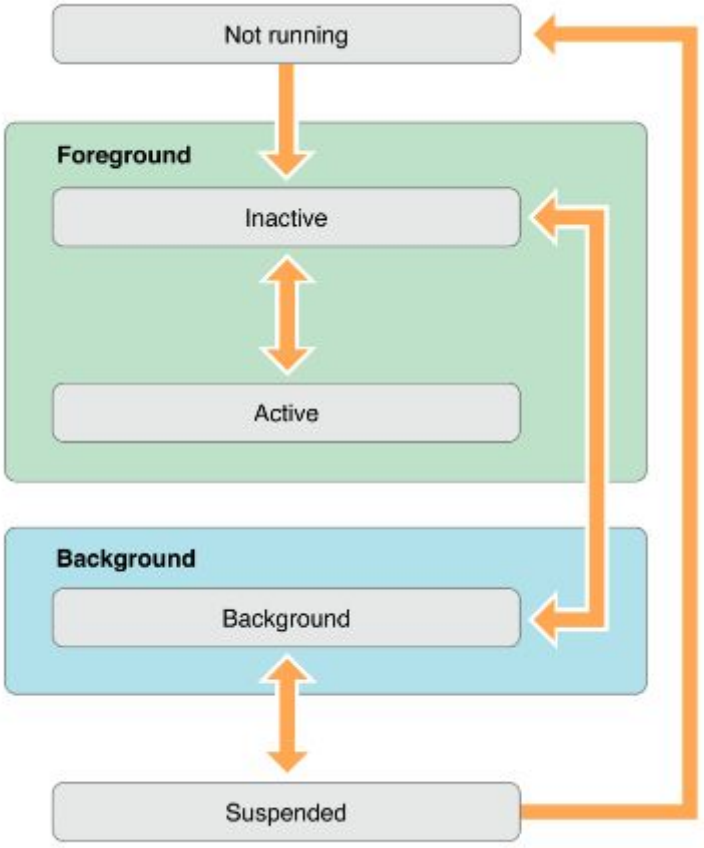


Рисунок 11 – состояния iOS приложения.

Состояния зависят друг от друга и меняются строго своей иерархией. Каждое приложение начинает свою работу после нажатия иконки приложения[3]. Рассмотрим состояния подробнее:

1) Not running - приложение не запущено или система удалила его из памяти.

2) Inactive - приложение запущено на переднем плане, но все еще не принимает события. Обычно приложение остается в этом состоянии только не надолго, поскольку переходит в другое состояние.

3) Active - обычный режим при котором приложение работает на переднем плане и принимает события.

4) Background - приложение находится в фоновом режиме и выполняет какой-либо код. Большинство приложений вводят это состояние не надолго, после чего идет приостановка приложения. Однако если требуется больше времени для работы в фоновом режиме, приложение может запросить его у системы и остаться в этом состоянии на некоторое время.

5) Suspended - приложение находится в фоновом режиме и не выполняет код. Система автоматически переводит приложения в это состояние и не уведомляет их об этом. Приложение приостановлено, но остается в памяти. При возникновении ситуации, когда кончается память на устройстве, система может удалить приостановленные приложения без уведомления, чтобы освободить место для следующего приложения.

Приложения должны быть готовы к прекращению работы в любое время и не должны ждать, чтобы сохранить пользовательские данные и выполнить другие важные задачи.

Пользователь также как и система способен удалить приложение из памяти используя многозадачный интерфейс, в этом случае приложение не получит никаких уведомлений об этом.

3.1.2.Потоки iOS приложения

Приложение создает основной поток вашего приложения и при

необходимости можно создавать дополнительные потоки для выполнения других задач[4]. Для iOS приложений предпочтительным методом является использование Grand Central Dispatch(GCD), операционных объектов и других асинхронных интерфейсов программирования, а не создание и управление потоками самостоятельно. Такие технологии, как GCD, позволяют определять работу, которую вы хотите сделать, и порядок, в котором вы хотите это сделать, а также позволять системе решать, как лучше всего выполнить эту работу на доступных CPU(Рисунок 12).

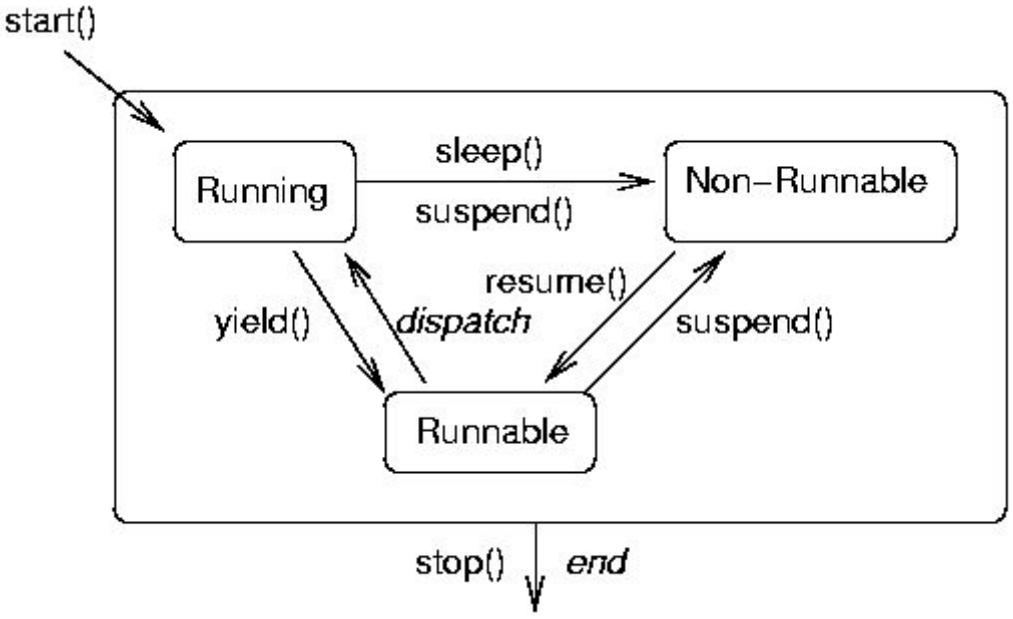


Рисунок 12 – работа потоков в iOS приложении.

Предоставление системе управления потоками упрощает код, который вы должны писать, упрощает обеспечение правильности этого кода и обеспечивает более высокую общую производительность.

Рассмотрим основные подходы работы с потоками:

1) Работа с представлениями, основной анимацией и многими другими классами UIKit обычно должна происходить в главном потоке приложения. Существуют некоторые исключения из этого правила -

например, манипуляции на основе изображений часто могут выполняться в фоновом потоке, но если есть какие-то затруднения с выполнением, тогда следует запускать эту задачу в главном потоке.

2) Длинные задачи, всегда должны выполняться в фоновом потоке.

3) Любые задачи, связанные с сетевым доступом, доступом к файлам или большими объемами обработки данных, должны выполняться асинхронно с использованием объектов GCD или операций.

Во время запуска ваше приложение должно использовать доступное время для установки своего пользовательского интерфейса как можно быстрее. В главном потоке должны выполняться только те задачи, которые способствуют настройке пользовательского интерфейса. Все остальные задачи должны выполняться асинхронно, результаты которых должны показываться пользователю, как только они будут готовы.

3.1.3.Используемые инструменты разработки

Среда разработки.

Xcode - интегрированная среда разработки под macOS и iOS, разработанная компанией Apple, позволяющая создавать приложения для iPhone, iPad, Apple Watch, Mac и Apple TV. Первую версию анонсировали в 2001 году. Программное обеспечение является бесплатным. Xcode содержит в себе большую часть документации разработчиков от Apple. На момент написания использовался Xcode 8.

Основные преимущества Xcode:

1) Текстовый редактор(Source Editor) - инструмент позволяющий расширение кода, свертывание кода, осуществляет подсветку синтаксиса,

отображает предупреждения, ошибки и другую контекстно-зависимую информацию, встроенную в ваш код.

2) Каталог картинок(Asset Catalog) - инструмент позволяющий управлять изображениями вашего приложениями, группируя разрешения одного и того же ресурса.

3) Редактор версии(Version Editor) - инструмент позволяющий отображать текущую временную шкалу коммитов, помогает определить причину сбоя и позволяет возвращать файлы к прошлым коммитам для сравнения с исходным файлом.

4) Симулятор(Simulator) - с помощью iOS SDK XCode создавать, устанавливать, запускать и отлаживать написанные приложения в симуляторе на базе Mac, что оптимизирует рабочий процесс разработки.

5) Встроенный интерфейс билдер(Interface Builder) - инструмент позволяющий создавать и тестировать интерфейс без написания строки кода.

Язык программирования.

Swift - открытый компилируемый язык программирования используемый в первую очередь для написания приложений под iOS и Mac. Язык совместим с основной кодовой базой, написанной на Objective-C.

Преимущества языка:

1) Более легкий для чтения и устойчивый к ошибкам программиста язык, нежели предшествовавший ему Objective-C.

2) Более безопасный. Swift содержит множество инструментов для работы с опциональными типами, что дает программисту уверенность в написанном им коде.

3) Swift намного быстрее, чем его предшественник Objective-C.

4) Поддерживает динамические библиотеки.

5) Упрощает наследование классов.

3.2.Архитектура iOS приложений

Для начала нужно познакомиться с двумя основными компонентами iOS приложения это View и ViewController(Рисунок 13).

Рисунок 13 – взаимосвязь главных компонентов.

3.2.1.View.

В приложениях на платформе iOS за пользовательский интерфейс отвечает один или несколько компонентов представлений(view). Общий интерфейс состоит из множества компонентов(subviews) унаследованные от общего класса View(Рисунок 14).

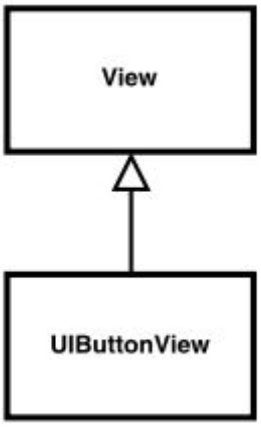


Рисунок 14 – наследование UIButtonView от View.

К таким доступным элементам интерфейса относятся: текстовые поля,

кнопки, таблицы, контейнеры для картинок и прочее.

Компоненты способные группировать между собой элементы:

1) TableView - группирует элементы TableViewCell в виде горизонтального списка, где в одной строке будет находиться только один элемент.

2) CollectionView - группирует элементы CollectionViewCell, как в горизонтальный, так и вертикальный список, в одной строке может быть несколько элементов.

3) Horizontal Stack View и Vertical Stack View - стек элементов, позволяющий с легкостью настроить общие размеры, отступы элементов.

4) ScrollView - контейнер, может содержать в себе любые элементы, а также расширяется в зависимости от количества элементов в нем.

Существует несколько способов создания view:

1) Создавать и настраивать view программно.

2) Использовать интерфейс билдер(Рисунок 15)

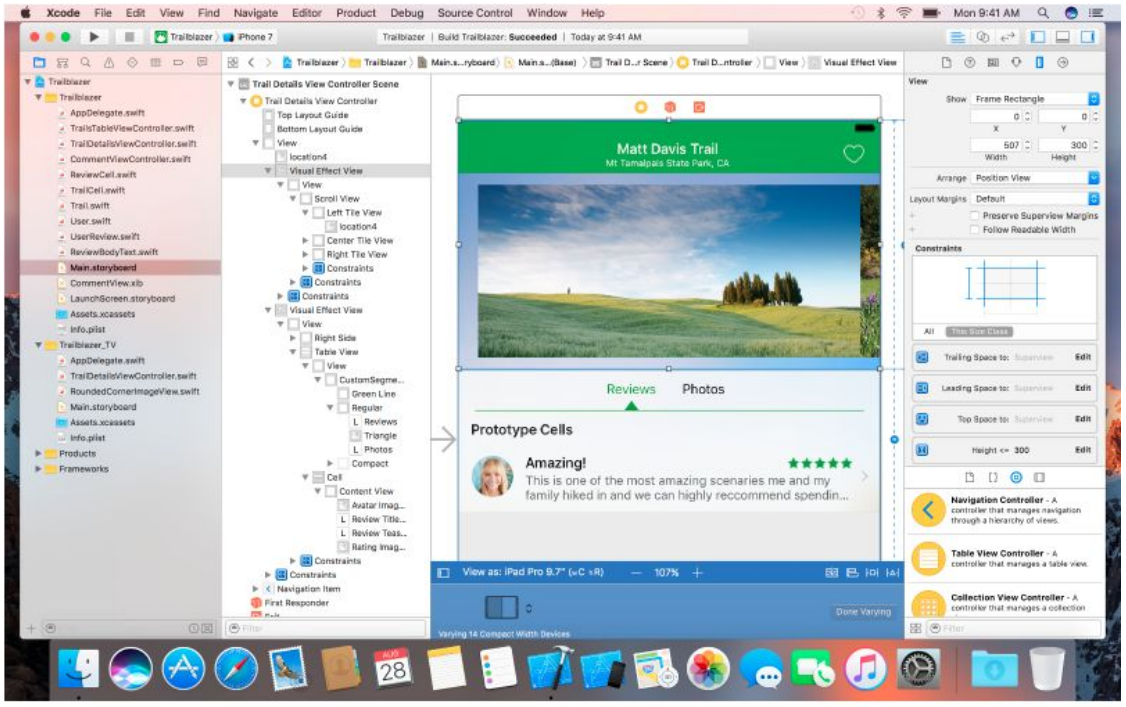


Рисунок 15 – интерфейс билдер.

При создание своей view в дальнейшем с ее помощью можно строить интерфейсы различных экранов.

Основные требования к дизайну интерфейса мобильных приложений:

1) Минимизация количества элементов на экране. Не следует перегружать небольшое пространство дисплея мобильного устройства большим количеством объектов. Необходимо постараться уместить максимум функционала в лаконичный и дружелюбный интерфейс.

2) Управление современными смартфонами с сенсорными экранами осуществляется при помощи пальца. Ввиду того, что площадь касания пальца гораздо больше размеров указателя компьютерной мыши, интерфейс не должен содержать мелких элементов. Если же например кнопка имеет маленькие размеры, что усложняет попадание по ней пальцем, то следует увеличить пространство вокруг кнопки, которое также обрабатывает прикосновения.

3) Размер текста в полях приложения должен быть достаточно крупный, чтобы была возможность с легкость его читать на удаленном расстояние от устройства.

3.2.2.ViewController

ViewControlers - компоненты iOS приложения, отвечающие за управление набором view и занимающиеся навигацией по приложению[5]. Обычно компоненты создаются изолированно друг от друга, имеющие различные представления[6]. Так например один контроллер может показывать список элементов, а в то время другой контроллер будет отображать выбранный элемент из этого списка(Рисунок 16).

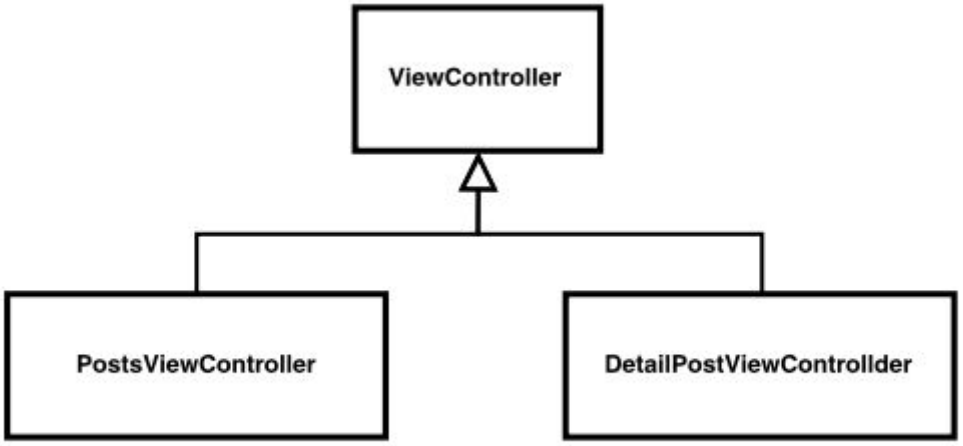


Рисунок 16 – наследование контроллеров.

ViewController выполняет такие задачи как:

1) Обновляет содержание view, обычно при обновлении каких либо данных, обновляется и представление.

2) Реагирует на взаимодействие пользователя с интерфейсом.

3) Меняет размеры и перерисовывает представления в зависимости от полученных событий: переворот экрана, добавления нового элемента и прочие.

3.3.MVVM и Coordinators в iOS приложениях.

3.3.1.MVVM.

MVVM - шаблон применяют для проектирования архитектуры приложения, ориентированный на современные платформы разработки[7]. Реализация паттерна для iOS архитектуры содержит 4 элемента: ViewController, View, Model, ViewModel(Рисунок 17).

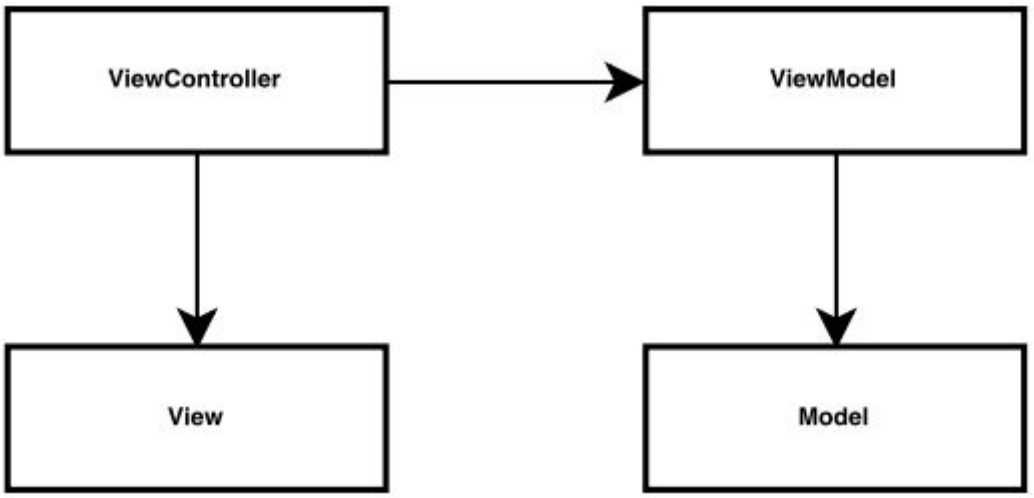


Рисунок 17 – модель MVVP в iOS приложении.

MVVM применяется для разделения модели и её представления, что необходимо для изменения их отдельно друг от друга. Например, разработчик работает с логикой данных, в то время дизайнер соответственно разрабатывает пользовательский интерфейс. MVVM удобно использовать вместо классического MVC и ему подобных в тех случаях, когда в платформе, на которой ведется разработка, присутствует явная связанность между пользовательским интерфейсом и бизнес логикой.

3.3.2 Coordinator.

Coordinator(Координатор) - паттерн позволяющий инкапсулировать всю навигацию по приложению[8]. Обычно ответственный за переход на следующий контроллер был текущий контроллер. Объект координатор позволяет взять все эти обязанности на себя. Он возьмет на себя управление переходами между контроллерами, что способствует увеличению слабых зависимостей и правильного переиспользования кода. Координатор может управлять также координаторами, добавленными к главному родительскому координатору. Рассмотрим основной интерфейс координатора(Рисунок 18)

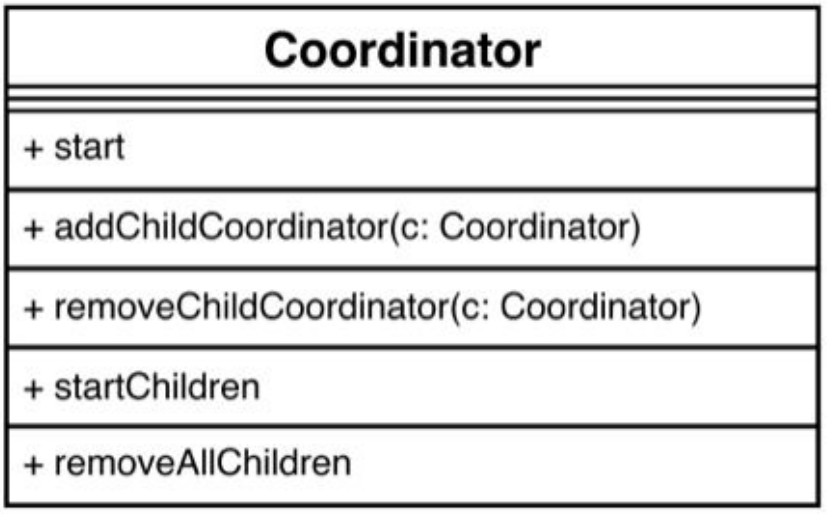


Рисунок 18 – интерфейс координатора.

Обычно приложение стартует с главного координатора, который в зависимости от состояния покажет требуемый приложению координатор. Объединив координатор с паттерном MVVM мы получим следующую структуру(Рисунок 19)

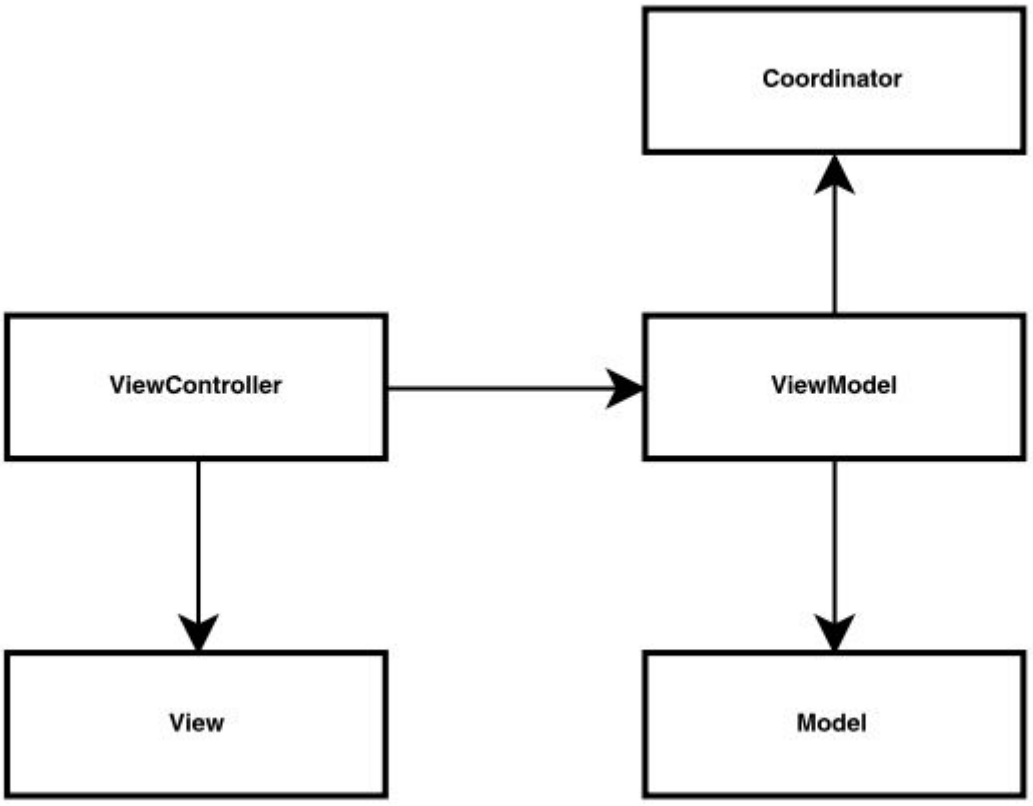


Рисунок 19 – модель MVVM и Coordinator.

Рассмотрим обязанности каждого элемента архитектуры подробнее:

1) View(Представление) - отображает пользовательский интерфейс.

2) Model(Модель) - содержит в себе пользовательские данные, так же производит с ними манипуляции и сообщает обо всех изменениях ViewModel.

3) ViewModel(Модель представления) - получает данные от модели и преобразует их в нужный вид. Получает сообщения от контроллера и передает их координатору, для корректной навигации в приложении.

4) ViewController(Контроллер) - запрашивает данные у ViewModel и используя их формирует представление нужным образом. Обрабатывает и передает события совершенные пользователем ViewModel.

5) Coordinator(Координатор) - обрабатывает сообщения полученные от ViewModel, тем самым совершает навигацию по приложению.

Избавление контроллеов от обязоности работы с моделью данных и навигацие по приложению, делает их намного читабельнее и позволяет использовать их в других проектах.

4.Разработка приложения.

4.1Архитектура системы.

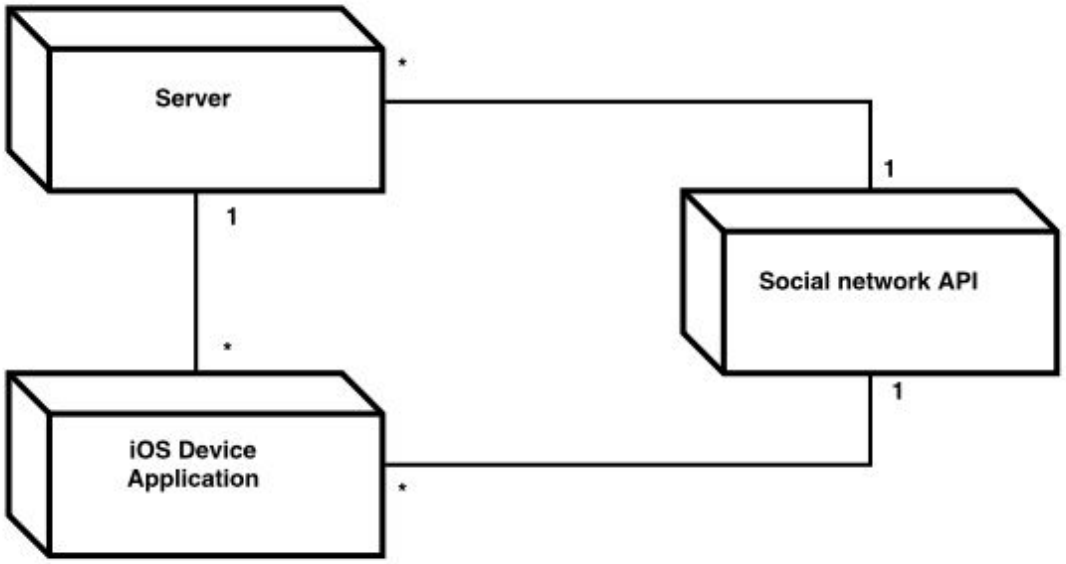


Рисунок 20 - диаграмма развертывания системы.

Рассмотрим систему развертывания на рисунок 20. Система состоит из 3 частей:

1) Приложения для платформы iOS.
2) WebService
3) Группа серверов социальных сетей, предоставляющих API.

В данной дипломной работе сервер не рассматривается, так как он реализован другим разработчиком.

Приложение взаимодействует с сервером по https протоколу. Приложение отправляет на сервер post, get, upload запросы. Сервер и приложение обмениваются информацией в виде текстового формата json.

Преимущества использование json формата:

1) Легко читается людьми(Рисунок 21)

2) Более подходящий для сериализации сложных структур, чем XML.

3) В среде iOS разработке существует множество библиотек, упрощающих сериализацию .


```
{
    "firstName": "Иван",
    "lastName": "Иванов",
    "address": {
        "streetAddress": "Московское ш., 101, кв.101",
        "city": "Ленинград",
        "postalCode": 101101
    },
    "phoneNumbers": [
        "812 123-1234",
        "916 123-4567"
    ]
}
```


Рисунок 21 – пример json ответа.

Приложение также взаимодействует с серверами социальных сетей. Некоторые социальные сети такие как Facebook, Google+ предоставляют SDK встраиваемые в проект. SDK направлены на интеграцию приложения с социальной сетью, что существенно упрощает разработку. Но у большинства социальных сетей нет своего SDK, тогда требуется использовать их API и создавать свои веб запросы.

4.2.Структура приложения.

Приложение начинает работу с вызова applicationDidFinishLaunchingWithOptions в *AppDelegate*. В этом методе мы создадим AppCoordinator и выполним метод s(Рисунок 22).

```
func application(_ application: UIApplication,
                 didFinishLaunchingWithOptions launchOptions:
                     [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
      let keyWindow = UIWindow(frame: UIScreen.main.bounds)
      window = keyWindow
      appCoordinator = AppCoordinator(window: keyWindow)
      appCoordinator?.start()
      return true
}
```

Рисунок 22 – запуск главного координатора.

AppCoordinator в свою очередь содержит и управляет 3 координаторами(Рисунок 23).

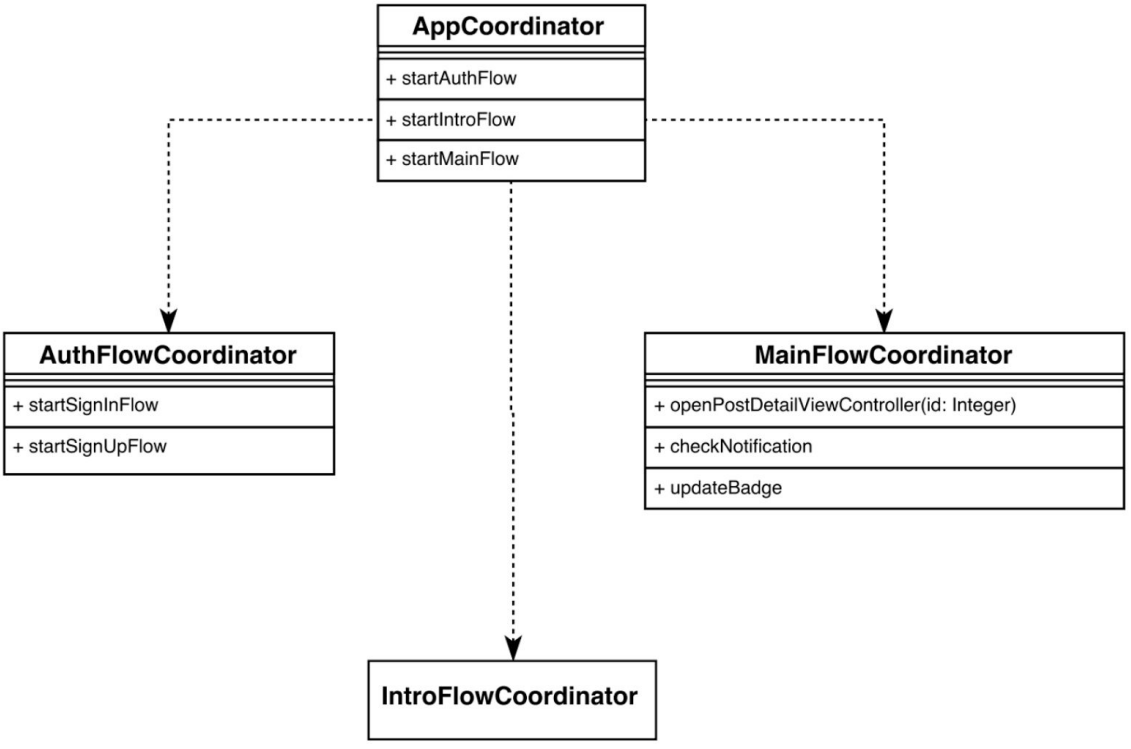


Рисунок 23 – диаграмма классов, показывающая зависимости между координаторами.

В зависимости от состояния главный координатор решает, какой показать.

Если пользователь первый раз запустил приложение, то AppCoordinator выполнит метод startIntroFlow.Это в свою очередь запустит работу IntroFlowCoordinator, который отвечает за показ обучающих слайдов.

Если пользователь не авторизован в системе, то AppCoordinator выполнит метод startAuthFlow, тем самым запустит работу AuthFlowCoordinator. Этот координатор отвечает за авторизацию и регистрацию пользователя в приложении.

Если пользователь уже авторизован в приложении , то AppCoordinator запустит MainFlowCoordinator, который отвечает за основной функционал приложения.

4.2.1AuthFlowCoordinator.

AuthFlowCoordinator - координатор приложения отвечающий за навигацию при авторизации и регистрации в приложении. Рассмотрим его структуру подробнее(Рисунок 24).

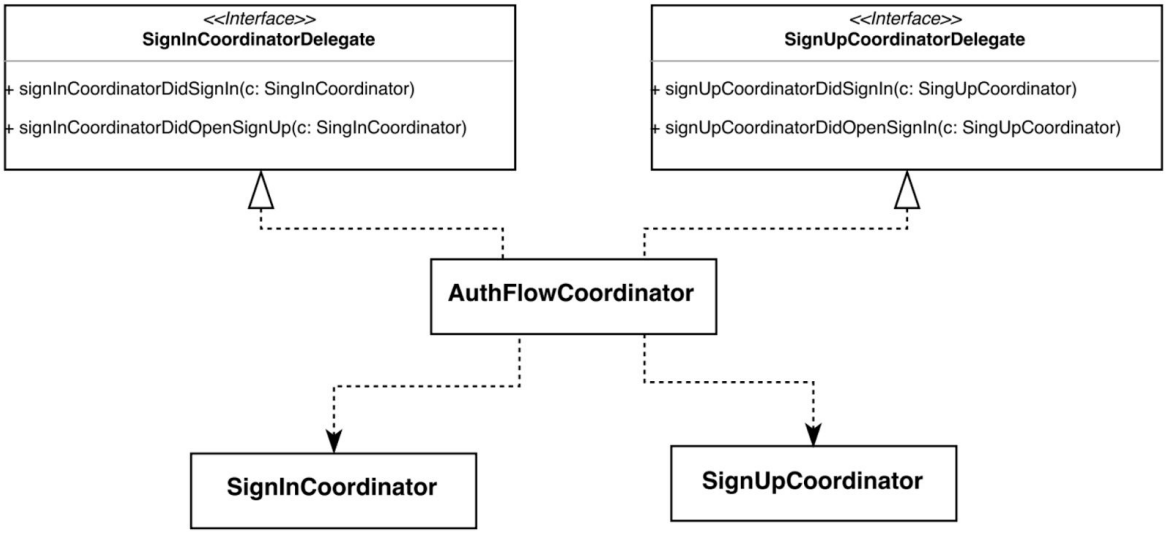


Рисунок 24 – диаграмма классов, показывающая зависимости AuthFlowCoordinator.

Как мы можем заметить, AuthFlowCoordinator содержит и управляет 2

координаторами, а также реализует два интерфейса. Главный координатор при необходимости замещает один координатор другим. На рисунках 25 - 26 продемонстрированы контроллеры, связанные с этим координаторами.

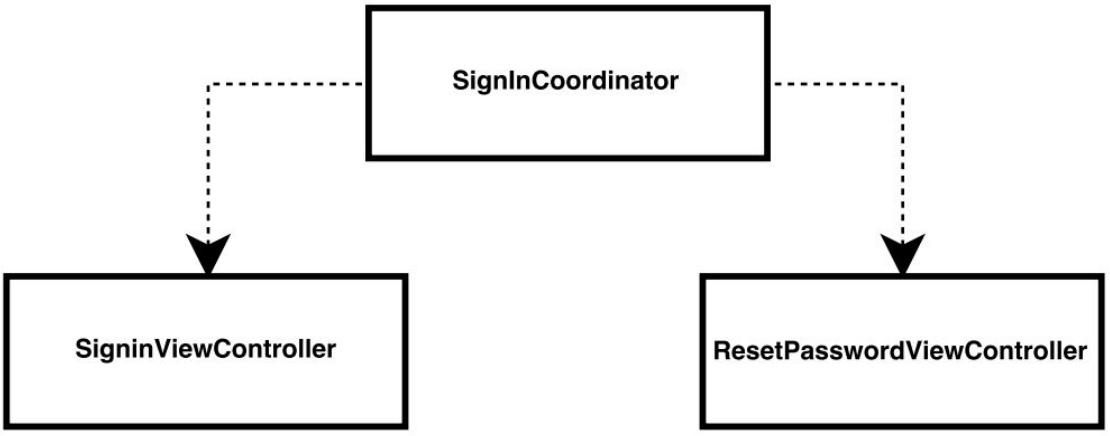


Рисунок 25 – диаграмма классов, показывающая зависимости SignInCoordinator.

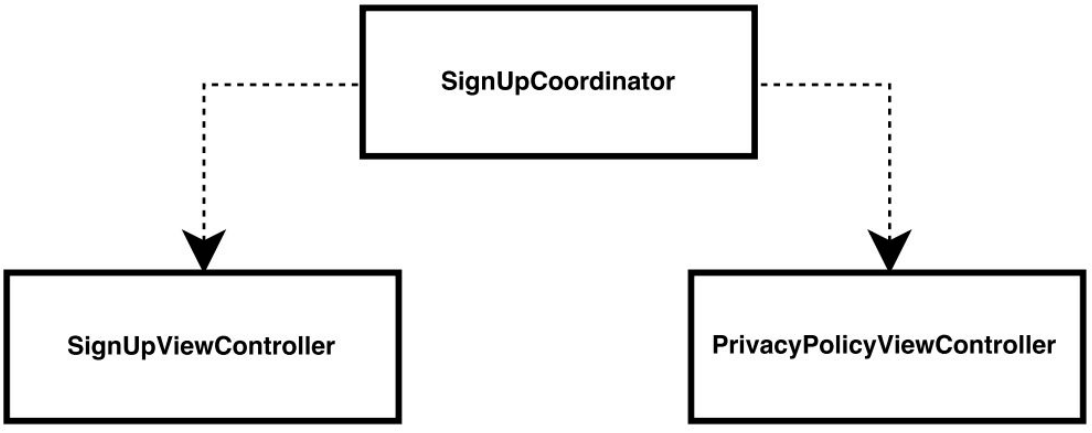


Рисунок 26 – диаграмма классов, показывающая зависимости SignUpCoordinator с контроллерами.

ViewModel каждого контроллера имеет ссылку на свой координатор. Рассмотрим каждый контроллер подробнее:

1) SignInViewController: необходим для реализации самой авторизации.
2) ResetPasswordViewController: необходим для восстановления пароля.
3) SignUpViewController: необходим для самой регистрации в

приложении.

4) PrivacyPolicyViewController**:** необходим для просмотра политики конфиденциальности приложения.

После успешной авторизации или регистрации, соответствующий координатор сообщает об этом AuthFlowCorrdinator, который в свою очередь сообщает это главному координатору, которые уже решает показывать ли главный функционал приложения.

4.2.2.IntroFlowCoordinator.

IntroFlowCoordinator - отвечает за навигацию между экранами в приложении, помогающие новому пользователи с легкостью разобраться с основным функционалом.

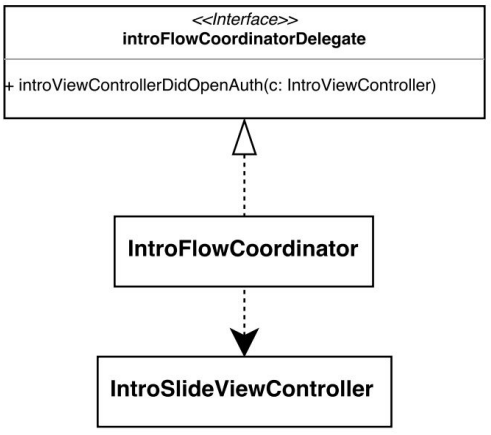


Рисунок 27 – диаграмма классов, показывающая зависимости SignUpCoordinator.

Координатор управляет IntroSlideViewController и реализует интерфейс IntroFlowCoordinatorDelegate. IntroSlideViewController: переключает view, реагируя на жесты пользователя(Рисунок 27). Можно перемещаться влево и вправо. На последней view осуществляется переход на координатор AuthFlowCoordinator.

4.2.3.MainFlowCoordinator.

MainFlowCoordinator - отвечает за навигацию между основными экранами приложения.Координатор принимает несколько интерфейсов и имеет метод openPostDetailViewController, который позволяет просматривать пост пришедший в push уведомление(Рисунок 28). Вне зависимости на каком экране мы будем находиться, контроллер PostDetailViewController будет отображаться выше всех в иерархии контроллеров.

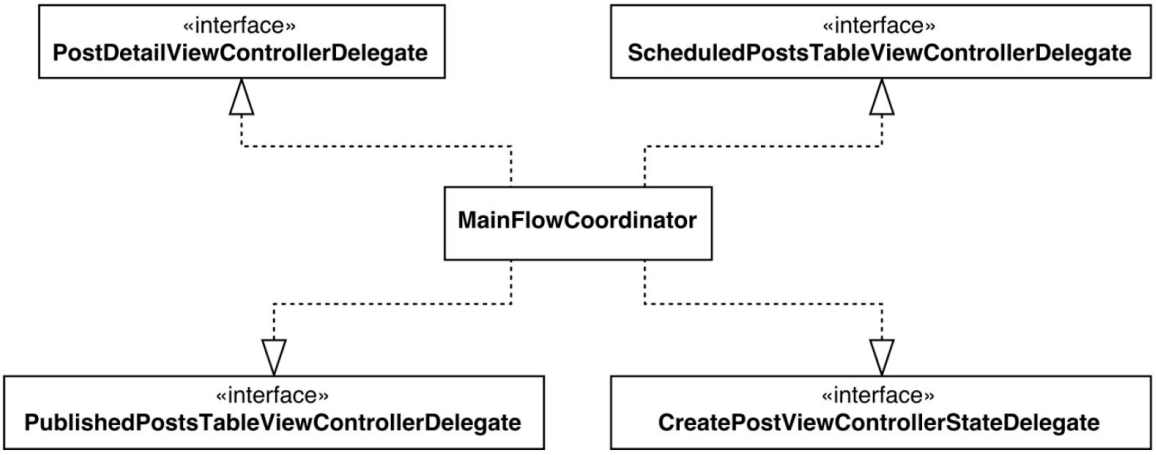


Рисунок 28 – диаграмма классов, показывающая зависимости MainFlowCoordinator.

MainFlowCoordinator взаимодействует с несколькими контроллерами, которые обмениваются сообщениями друг с другом через делегаты. Пользователь управляет навигацией через выдвижное меню слева, за это отвечает SlideMenuViewController(Рисунок 26).

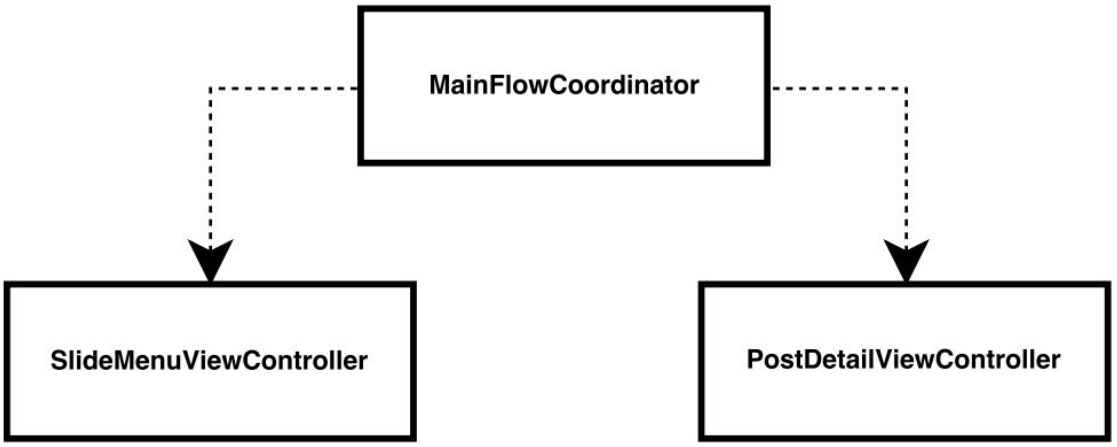


Рисунок 29 – диаграмма классов, показывающая зависимости SignUpCoordinator с котроллерами.

SlideMenuViewController содержит и управляет множеством отдельных контроллеров(Рисунок 30).

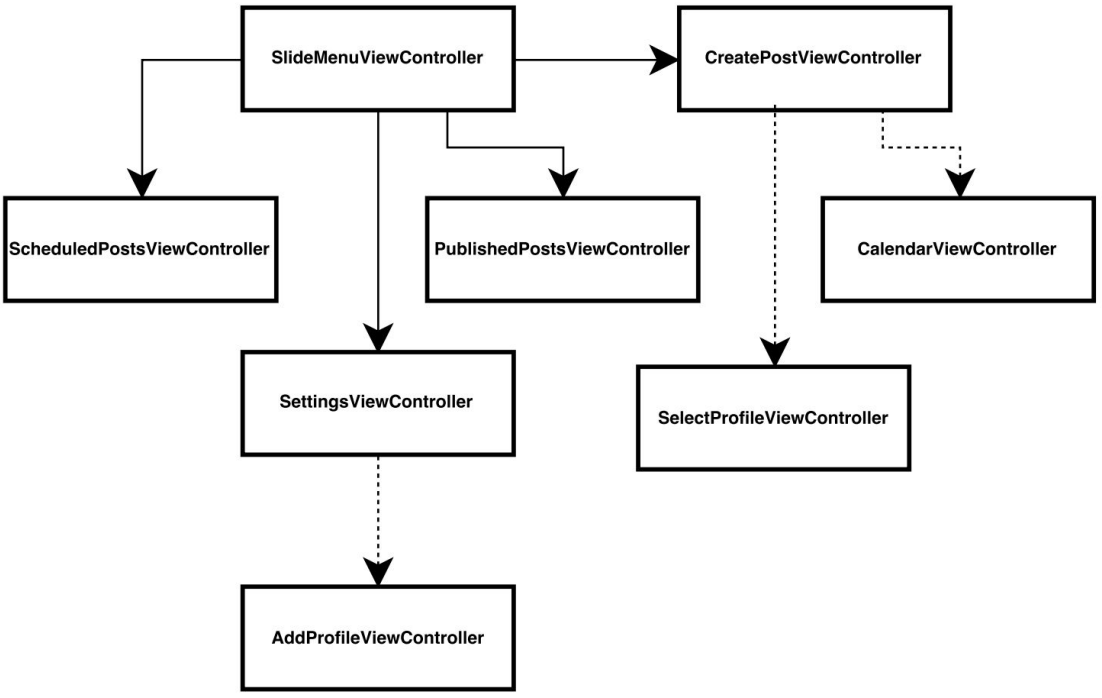


Рисунок 30 – диаграмма классов, показывающая зависимости SlideMenuViewController.

Рассмотрим каждый контроллер подробнее:

- ScheduledPostsViewController: необходим для отображения списка запланированных постов, отображается в виде таблице, где каждая ячейка является постом.
- PublishedPostsViewController: необходим для отображения списка опубликованных постов.
- CreatePostViewController: контроллер позволяющий создавать новые посты.
- SelectProfileViewController: позволяет выбрать один или несколько аккаунтов социальных сетей, что требуется при создании поста.
- CalendarViewController: контроллер который позволяет выбрать день и выставить время для поста, во время его создания.
- SettingsViewController: позволяет менять профиль пользовательского аккаунта.
- AddProfileViewController: позволяет добавить новый аккаунт для социальных сетей.

4.3. Интеграция социальных сетей.

Социальные сети имеет различные технологии авторизации и публикации постов. Требуется научится взаимодействовать с различными API и SDK. Для интеграции социальных сетей, приведем функционал к общему классу SocialNetwork. Для каждой социальной сети создадим унаследованный класс от SocialNetwork и реализуем каждый метод в зависимости от технологии(Рисунок 31). Хранить все объекты будем в упорядоченном массиве. Как только требуется доступ к определенной социальной сети, из массива достается заранее сконфигурированный объект.

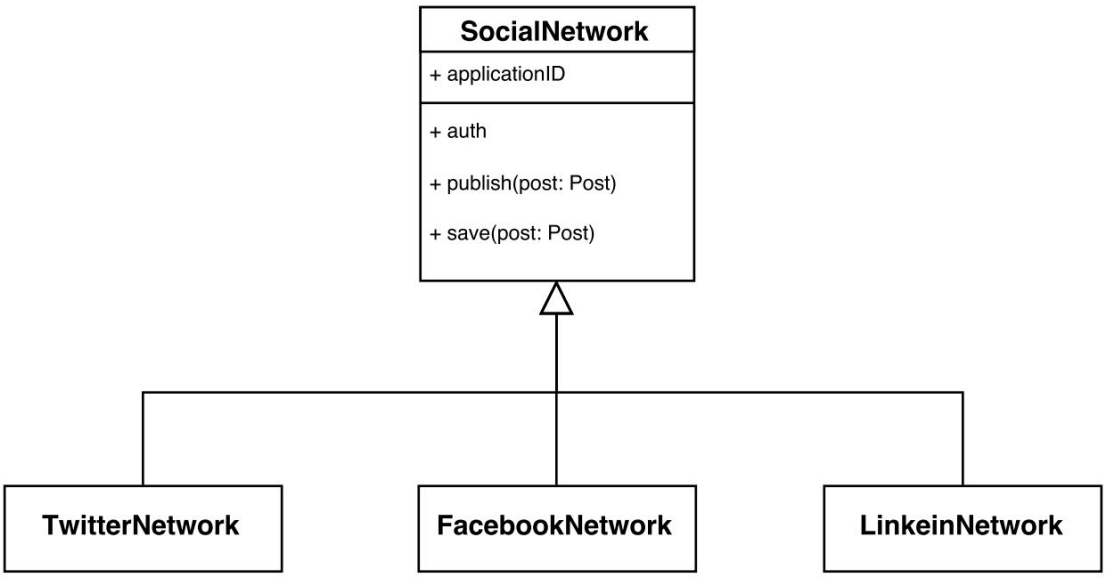


Рисунок 31 - диаграмма классов социальных сетей.

В социальной сети Instagram нет возможности публикации постов через API. Единственный способ, это отправлять содержание поста заранее установленному приложению Instagram и используя его публиковать.

5.Тестирование

5.1.Тестирование разработанного приложения

Тестирование приложения было произведено на устройствах с различными характеристиками(Таблица 1).

Таблица 1: Тестируемые устройства

Устройство	Разрешение	Диагональ	Версия iOS
iPhone 4s	640 × 960	3.5”	9.1
iPhone 5s	640 × 1136	4.0”	9.2
iPhone 6s	750 × 1334	4,7”	10
iPhone 7 plus	1920×1080	5,5”	10.1
iPhone 7	750 × 1334	4,7”	10.2
iPad Air 2	2048 ×1536	9.7”	9.3
iPod touch	640 x 960	3.5”	10

При тестировании уделялись вниманию характеристики: диагональ экрана, разрешение экрана, версия ОС iOS.

Во время процесса тестирования были выявлены и своевременно устранены следующие ошибки:

1. Некорректное отображение пользовательского интерфейса: из-за большой диагонали на устройствах iPad некоторые элементы интерфейса располагались в неправильном месте,а также растягивались.
2. Различия в версиях операционной системы повлияло на push уведомления, в 9 и 10 версиях iOS SDK предоставляет различную обработку приходящих уведомлений, что требовало реализации двух вариантов.

3. Некоректное отображение анимаций перехода между экранами
4. Ошибка при конвертации видео и фото

Заключение

В проделанной работе:

1) Проанализированы существующие решения
2) Изучены особенности мобильной платформы iOS
3) Сформированы требования к разрабатываемому приложению
4) Изучено взаимодействие с различными API социальных сетей
5) Спроектирована архитектура приложения
6) Реализовано приложение
7) Протестировано приложение

В работе разработано iOS-приложение позволяющее интегрировать несколько социальных сетей для просмотра, создания, редактирования и публикации постов, т.е. поставленная цель работы была выполнена. Кроме того, в рамках данной работы были проанализированы технологии взаимодействия с социальными сетями. В данный момент в приложение интегрировано 5 социальных сетей. Однако реализован простой пример добавления новых сетей, что позволит в будущем расширить функциональность.

Список источников

1. Фаулер М., Архитектура корпоративных программных приложений // М.Фаулер. — М.: Издательский дом "Вильяме", 2006. — 544 с.
2. iOS Developer Library URL: http://developer.apple.com/library/ios/navigation/ (дата обращения: 10.11.2016).
3. Life Cycle [Электронный ресурс] // Apple Inc. — Электрон. дан. — [Б.м.], 2017 — URL: https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle.html
4. Thread [Электронный ресурс] // Google Inc. — Электрон. дан. — [Б.м.], 2016 — URL: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/Multithreading/(дата обращения: 24.05.2016)
5. ViewController [Электронный ресурс] // Apple Inc. — Электрон. дан. — [Б.м.], 2016 — URL: https://developer.apple.com/library/content/featuredarticles/ViewController/ (дата обращения: 10.11.2016)
6. Гамма Э., Приемы объектно-ориентированного проектирования // Э.Гамма, Р.Хелм, Р.Джонсон. — СпБ: Питер, 2001. — 368 с.
7. Introduction to MVVM [Электронный ресурс] – Электрон. дан. – Режим доступа: https://www.objc.io/issues/13-architecture/mvvm/, свободный.
8. An iOS Coordinator Pattern [Электронный ресурс] – Электрон. дан. – Режим доступа: https://will.townsend.io/2016/an-ios-coordinator-pattern, свободный.

Приложение А. Руководство пользователя

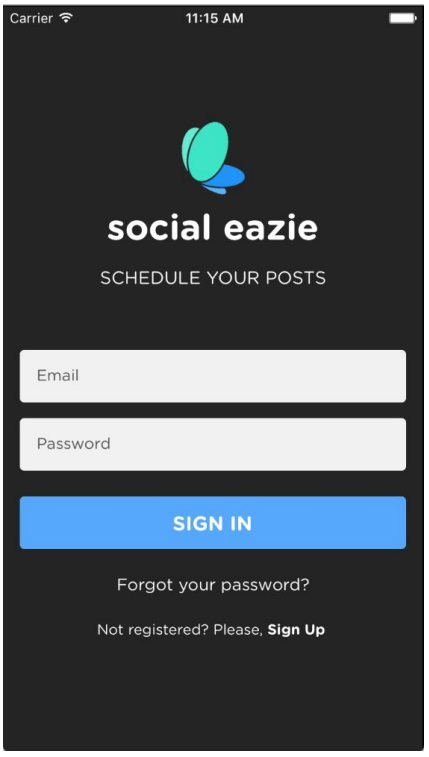


Рисунок А1 - авторизация

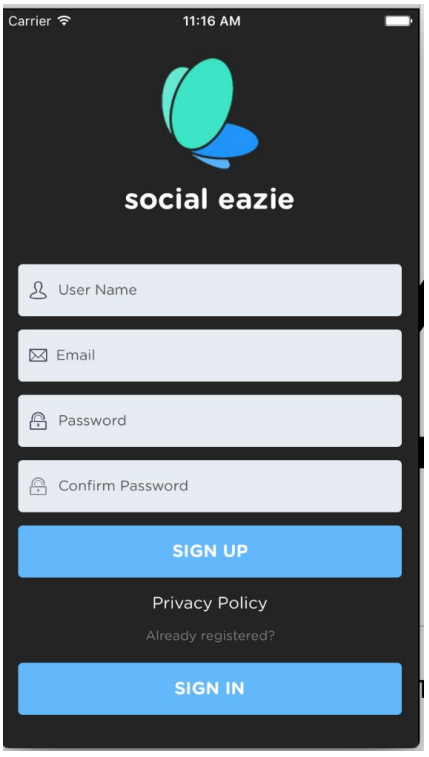


Рисунок А2 - регистрация

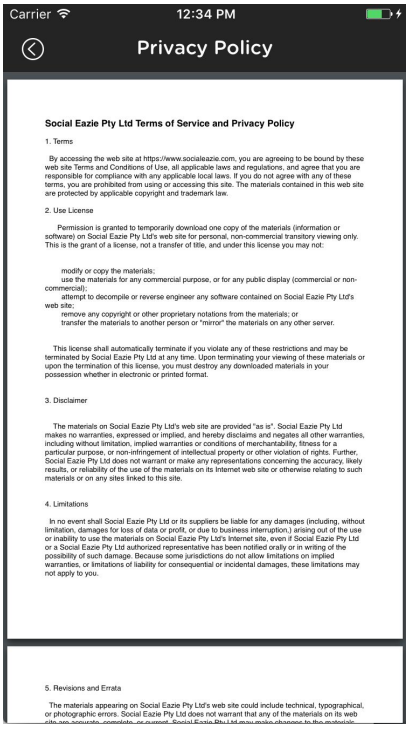


Рисунок А3 - privacy policy

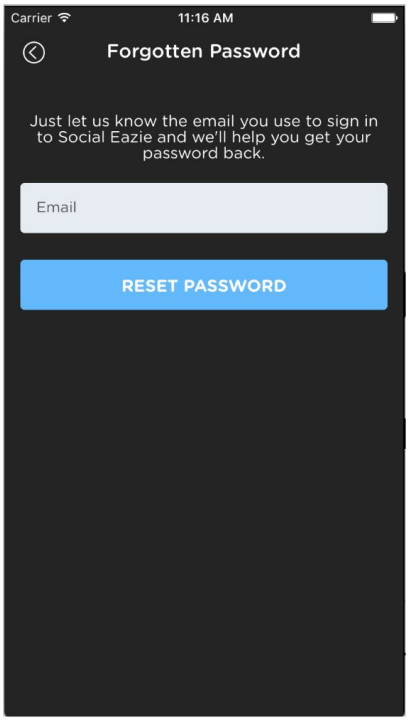


Рисунок А4 - восстановление пароля

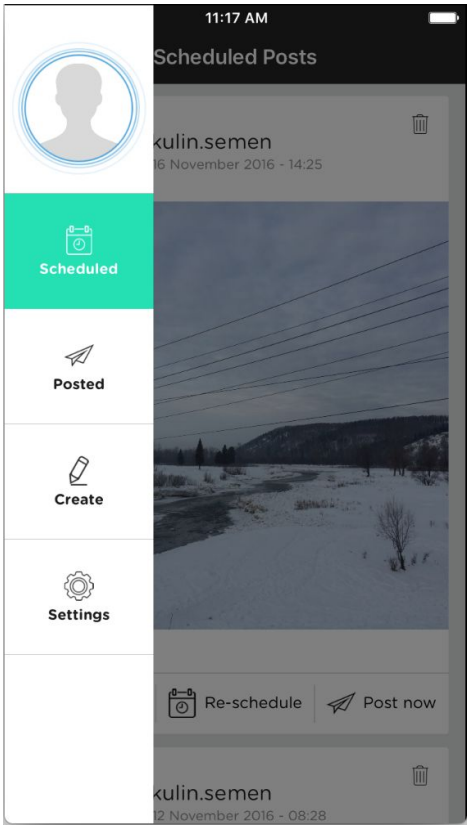


Рисунок А5 - боковое меню

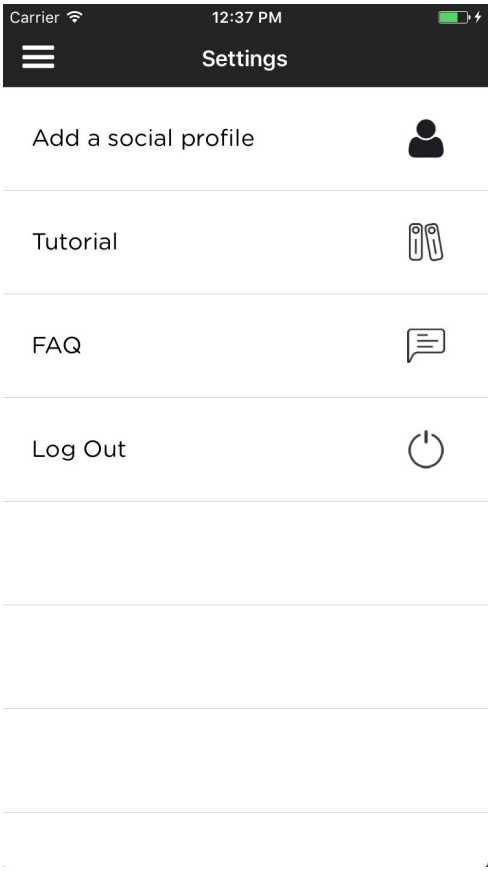


Рисунок А6 - настройки

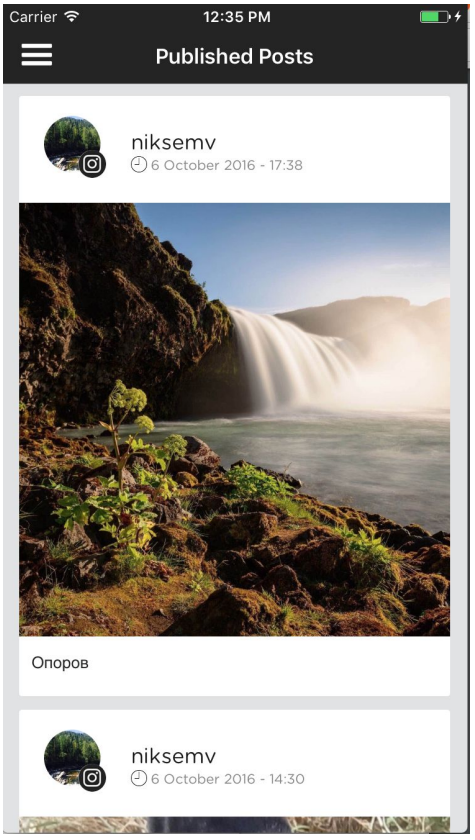


Рисунок А7, А8 - запланированные и опубликованные посты

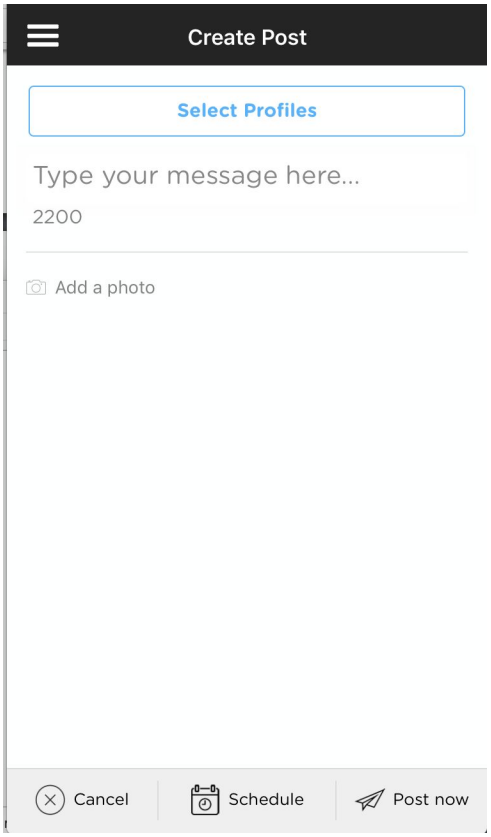


Рисунок А9 - создание поста

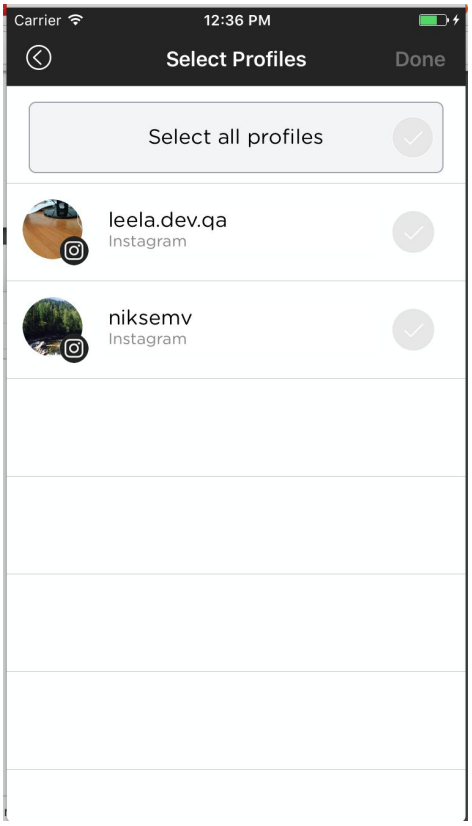


Рисунок А10 - выбор профиля

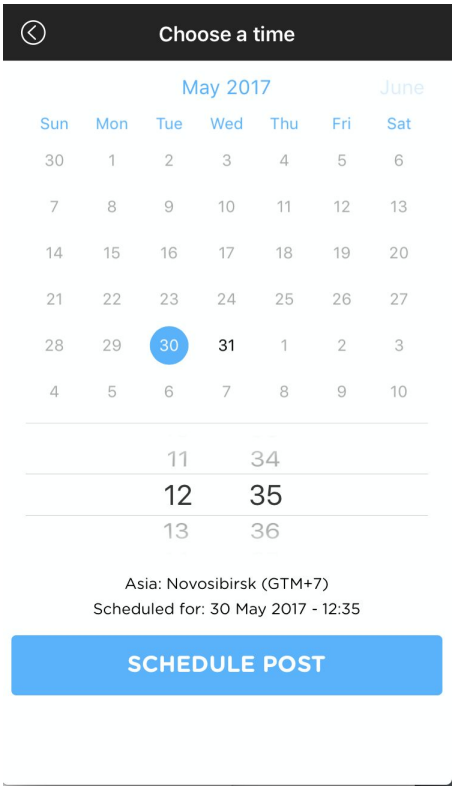


Рисунок А11 - выбор даты

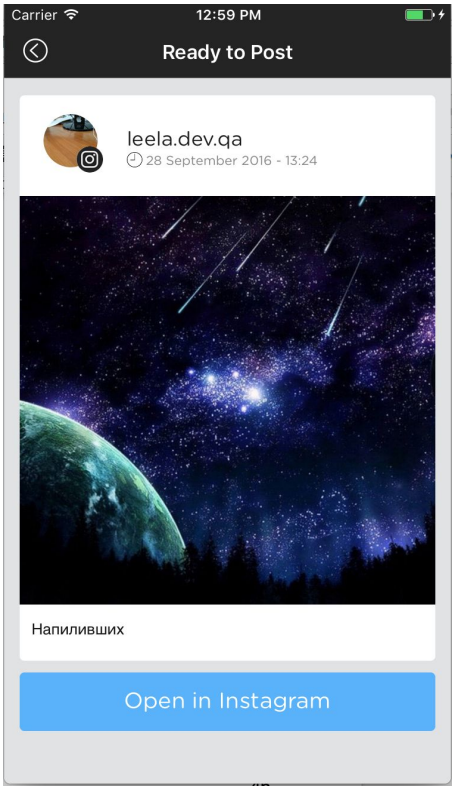


Рисунок А12 - просмотр профиля

Отчет о проверке № 1

ФИО: gtf jgg
дата выгрузки: 09.06.2017 05:52:43
пользователь: labworktsu@yandex.ru / ID: 4563176
отчет предоставлен сервисом «Антиплагиат»
на сайте http://www.antiplagiat.ru

Информация о документе

№ документа: 7
Имя исходного файла: Nikulin_Diplom.docx
Размер текста: 5695 кБ
Тип документа: Не указано
Символов в тексте: 36189
Слов в тексте: 4192
Число предложений: 305

Информация об отчете

Дата: Отчет от 09.06.2017 05:52:43 - Последний готовый отчет
Комментарии: не указано
Оценка оригинальности: 99.45%
Заимствования: 0.55%
Цитирование: 0%

Оригинальность: 99.45%
Заимствования: 0.55%
Цитирование: 0%

Источники

Доля в тексте	Источник	Ссылка	Дата	Найдено в
0.55%	[1] Model-View-ViewModel	http://ru.wikipedia.org	раньше 2011 года	Модуль поиска Интернет
0.55%	[2] Программа для учета сотрудников студенческой организации и их HR-оценок	http://knowledge.allbest.ru	раньше 2011 года	Модуль поиска Интернет