

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
имени Гумарбека Даукеева

Кафедра «Телекоммуникационные сети и системы»

Специальность: 6М071900 «Радиотехника, электроника и телекоммуникации»

ДОПУЩЕН К ЗАЩИТЕ

Зав. кафедрой

PhD, доцент Темырканова Э.К.

(ученая степень, звание, ФИО)

(подпись)

« _____ » _____ 2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
пояснительная записка

на тему: «Разработка личного кабинета для мобильных устройств на iOS системе»

Магистрант: Смагулов А. Б. _____ группа МРЭТн 18-2
(Ф.И.О.) (подпись)

Руководитель: к.т.н., профессор АУЭС _____ Байкенов А. С.
(ученая степень, звание) (подпись) (Ф.И.О.)

Рецензент _____
(ученая степень, звание) (подпись) (Ф.И.О.)

Консультант по ВТ к.т.н., профессор АУЭС _____ Байкенов А. С.
(ученая степень, звание) (подпись) (Ф.И.О.)

Нормоконтроль: к.т.н., профессор АУЭС _____ Байкенов А. С.
(ученая степень, звание) (подпись) (Ф.И.О.)

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
имени Гумарбека Даукеева

Институт Космической Инженерии и Телекоммуникаций

Специальность: 6М071900 «Радиотехника, электроника и телекоммуникации»

Кафедра: «Телекоммуникационные сети и системы»

ЗАДАНИЕ

на выполнение магистерской диссертации

Магистранту Смагулову Арману Бижанулы

Тема диссертации «Разработка личного кабинета для мобильных устройств на iOS системе»

Утверждена Ученым советом университета №43 от «18» марта 2020г.
Срок сдачи законченной диссертации «25» мая 2020г.

Цель исследования является анализ существующих методов интеграции мобильных приложений с корпоративными системами компаний и разработка модели интеграции, которая обеспечит высокую эффективность и удобство для эксплуатации пользователем.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации:

1. Проанализировать подходы к интеграции мобильных приложений с корпоративными информационными системами (КИС)
2. Проанализировать современные мобильные приложения по образовательным услугам (журнал/дневник)
3. Разработать модель интеграции мобильного приложения по оказанию образовательных услуг (журнал/дневник)
4. Создание базы данных и интеграция в мобильного приложение
5. Разработка мобильного приложения
6. Анализ результатов интеграции приложения

Перечень графического материала (с точным указанием обязательных чертежей)

1. Firebase после авторизации (окно консоли)
2. Создание и добавление необходимых возможностей

3. Установка необходимых библиотек через терминал
4. Проверка наличия API ключа в проекте
5. Реализация интерфейса авторизации
6. Сетевые функции приложения
7. Пример реализации системы предупреждения
8. Пример реализации перехода между окнами

Рекомендуемая основная литература

1. Усов В. Swift. Основы разработки приложений под iOS и macOS. 4-е изд., доп. и перераб. — СПб.: Питер, 2018. — 448 с.

2. Энтони Грей: Swift. Карманный справочник. Программирование в среде iOS и OS X. - Диалектика/Вильямс, 2015. - 224 с.

Г Р А Ф И К подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
1. Информационный обзор согласно теме	05.10.2018	
2. Анализ методов интеграции корпоративных приложений	14.01.2019	
3. Методы разработки и интеграции мобильного приложения	02.05.2019	
4. Модель интеграции мобильного приложения с КИС	18.11.2019	
5. Анализ полученных расчетных данных	10.04.2020	

Дата выдачи задания_30 сентября 2018г. _____

Заведующий кафедрой _____ (Темырканова Э.К.)
(подпись) (Ф.И.О.)

Научный руководитель диссертации _____ (Байкенов А. С.)
(подпись) (Ф.И.О.)

Задание принял к исполнению магистрант _____ (Смагулов А. Б.)
(подпись) (Ф.И.О.)

Аңдатпа

Осы жұмыста тіркеу, авторизациялау, деректерді өңдеу және оларды деректер базасында сақтау функциялары бар мобильді қосымша жасау мүмкіндігі қарастырылады. Болашақта қажет болса мобильді қосымшаны ортақ ақпараттық жүйеге интеграциялау мүмкіндігімен.

Кеңінен таралған мобильді қосымшаларды жасау шешімдері, жасау құрылғылары және сынау жүйелері сарапталған. Қосымшаның құрылғының аппараттық бөліміне түсіретін жүтемесінің есептері келтірілген, желі трафигі нақты кедершілермен сарапталған және аса маңызды қателіктердің болмауына тексерілген.

Аннотация

В данной работе рассмотрены возможность создания мобильного приложения, которая обладает функциями регистрации, авторизации, обработки данных и сохранение их в базе данных. С возможностью в будущем интеграция мобильного приложения в единую информационную систему при необходимости.

Проанализированы популярные решения разработки мобильных приложений, средства разработки и системы тестирования. Приведены расчеты нагрузки приложения на аппаратную часть устройства, проанализирован сетевой трафик с реальными помехами и проверено на отсутствие критических ошибок.

Abstract

In this paper, we consider the possibility of creating a mobile application that has the functions of registration, authorization, data processing and storing them in a database. With the possibility of future integration of the mobile application into a single information system, if necessary.

Popular solutions for developing mobile applications, development tools, and testing systems are analyzed. Calculations of the application load on the hardware of the device are presented, network traffic with real interference is analyzed, and checked for critical errors.

Содержание

Введение.....	6
1 Анализ методов интеграции корпоративных приложений.....	8
1.1 Описание и анализ предметной области.....	8
1.2 Анализ современных способов интеграции информационных систем.....	9
1.3 Анализ современных информационных систем по оказанию услуг личного кабинета.....	18
2 Методы разработки и интеграции мобильного приложения.....	21
2.1 Методы разработки мобильных приложений.....	21
2.2 Мобильные операционные системы.....	23
2.3 Категории мобильной разработки.....	26
2.3.1 Нативные приложения.....	28
2.3.2 Гибридные приложения.....	29
2.3.3 Веб приложения.....	31
2.4 Методы разработки мобильных приложений.....	32
2.5 iOS.....	34
2.6 Метод разработки приложения на iOS.....	36
2.7 Разработка архитектуры мобильного приложения.....	37
3 Модель интеграции мобильного приложения с КИС.....	38
3.1 Разработка модели интеграции мобильного приложения с КИС...	38
3.2 Разработка мобильного приложения.....	43
3.3 Пример окон приложения.....	54
Заключение.....	58
Список литературы.....	60

Введение

В настоящее время растет количество пользователей интернет-услугами и это не обходит стороной почти любую отрасль человеческой деятельности. В нашем случае мы будем рассматривать оказания интернет-услуг для студентов. Многие университеты мира уже предлагают свои занятия и процесс обучения онлайн. Также растет популярность мобильных услуг, когда пользователь может получить необходимую ему услугу через мобильное приложение (записаться на дополнительные курсы или конференции).

Однако степень удобства и скорость получение услуг может быть разной и не все университеты или интернет-сервисы (курсы) по продажам образовательных услуг имеют мобильные приложения.

Актуальностью данной магистерской диссертацией является будущая разработка мобильного приложения, которую можно применить для создания единой информационной системы в высших учебных заведениях Республики Казахстан. Приложение сможет объединить в себе различные системы данных разных ВУЗов страны. Такую разработку сможет инициализировать и профинансировать МОН РК.

Объектом исследования магистерской диссертации является КИС образовательной компании.

Предметом исследования магистерской диссертации является интеграция мобильного клиента образовательных услуг с КИС образовательного учреждения.

Целью исследования является анализ существующих методов интеграции мобильных приложений с информационными системами и разработка модели интеграции, которая обеспечит высокую эффективность и удобство для эксплуатации пользователем.

Для достижения поставленной цели требуется решить следующие задачи:

- 1) проанализировать подходы к интеграции мобильных приложений с корпоративными информационными системами (КИС);
- 2) проанализировать современные мобильные приложения по образовательным услугам (журнал/дневник);
- 3) разработать модель интеграции мобильного приложения по оказанию образовательных услуг (журнал/дневник);
- 4) создание базы данных и интеграция в мобильного приложения;
- 5) разработка мобильного приложения;
- 6) анализ результатов интеграции приложения.

Гипотеза исследования: применение предлагаемой модели интеграции-обеспечить простоту интеграции мобильного приложения для предоставления образовательных услуг с КИС образовательного учреждения.

На защиту выносятся:

Модель интеграции мобильного приложения для предоставления образовательных услуг с КИС образовательного учреждения;

Научная новизна исследования заключается в разработке модели интеграции мобильного приложения с КИС образовательного учреждения.

Практическая значимость работы заключается в разработке мобильного приложения на iOS для предоставления образовательных услуг с КИС образовательного учреждения, которое просто интегрируется.

Методы исследования: методы и модели интеграции компонентов КИС, объектно-ориентированный подход к анализу и проектированию ИС.

Первая глава посвящена анализу архитектуры современных корпоративных информационных систем, методов интеграции с КИС, а также приведен обзор современных информационных систем по предоставлению образовательных услуг.

Во второй главе рассмотрены существующие методы разработки мобильных приложений. Описана архитектура приложения и разработана модель интеграции мобильного клиента с КИС.

Третья глава содержит этапы проектирования и разработки мобильного приложения и приложения для интеграции КИС образовательного учреждения.

В заключении подводятся итоги выполненной работы.

Диссертация состоит из введения, трех глав, заключения, списка литературы и приложения.

Работа изложена на 59 с. и включает 30 рисунков, 4 таблицы.

1 Анализ методов интеграции корпоративных приложений

1.1 Описание и анализ предметной области

Электронный журнал студента не так сильно отличается от физического журнала преподавателя, где он выставляет ему оценки за посещение, выполнение СРС и СРСП студента, а также оценки за расчетно-графические работы и курсовых работ. Разницей может служить только то, что студент видит свои оценки индивидуально, изолированно от результатов других студентов, что только не отвлекает его от поиска своих результатов и попыток сравнивать себя с другими студентами. Преимуществами же является защищенность данных от потери как физического журнала, а также возможность отслеживания даты изменения, а возможность добавления некоторых функциональных возможностей позволяет создавать более гибкую систему под требуемую задачу.

Развитие информационных технологий в Казахстане всегда старается идти вперед и многие государственные проекты, такие как цифровой Казахстан их только ускоряют и дают возможность для роста. Образовательные услуги не стоят в стороне от цифровых технологий и стараются тоже развиваться. Если посмотреть на Закон Республики Казахстан от 27 июля 2007 года № 319-III «Об образовании» на статью 8. “Государственные гарантии в области образования” можно увидеть дополненный пункт 2-1 в соответствии с Законом РК от 04.07.18 г. № 171-VI Государство обеспечивает условия создания информационно-коммуникационной инфраструктуры электронного обучения с использованием информационно-коммуникационных технологий [1].

Стоит учитывать, что разрабатываемая система не ставит перед собой задачу заменить образовательные услуги, а ее цель состоит в том, чтобы обеспечить комфортный процесс обучения и искоренить момент неведения студента о его текущей ситуации в процессе обучения.

1.2 Анализ современных способов интеграции информационных систем

Одним из наиболее выраженных направлений развития современных корпоративных информационных систем (ИС) является их ориентация на интеграцию друг с другом посредством формирования единого информационного пространства (ЕИП) предприятия (рисунок 1.1).

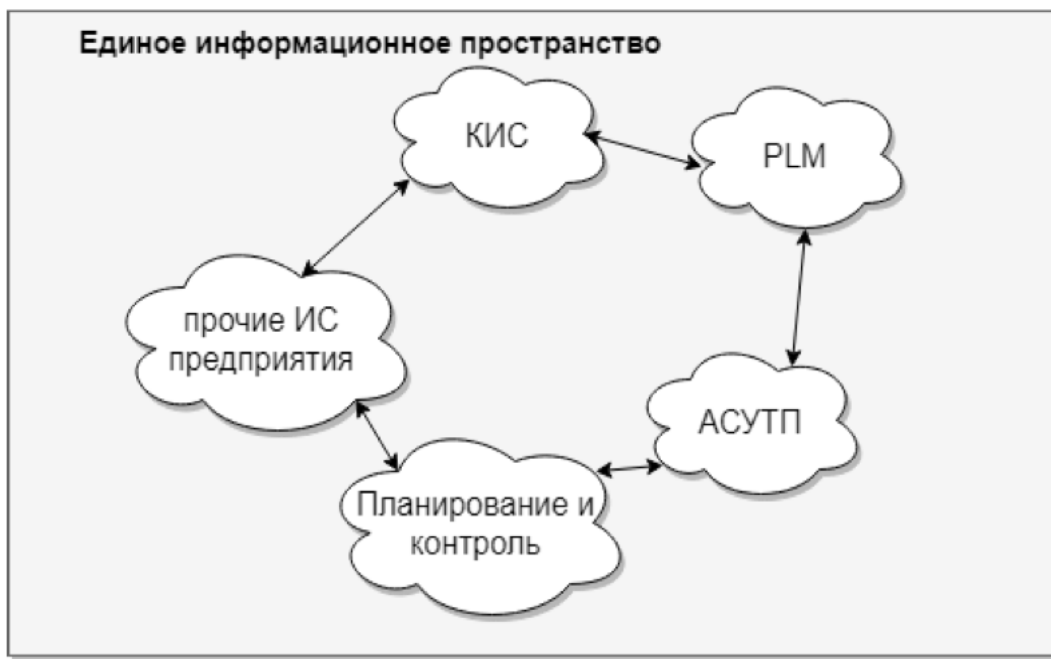


Рисунок 1.1 - Обобщенная схема ЕИП

С точки зрения практической значимости ЕИП призван сохранять целостность данных и возможность их использования различными пользователями в различных информационных системах в соответствии с их профилем деятельности. Используя ЕИП можно построить процессный подход к управлению, который будет целью устранения фрагментации в работе, путем сокращения организационных, бюрократических шагов и информационных недостатков [2]. Организации ЕИП исследуется учеными на протяжении нескольких последних лет, но основная проблема состоит не только в выборе и развитии технологий интеграции ИС, но а также и в создании целостного представления данных. Из этого можно сделать вывод, что новизна исследований, связанных с ЕИП, опирается на исследование методов структурирования информации, получаемой из различных ИС. Для разработки этой концептуальной основы ЕИП использует следующие принципы теории множеств, управления проектами, графов, и т.д.

ЕИП можно организовать путем интеграции всех информационных систем предприятия с целью поддержания целостности данных в любой момент времени, так как сейчас практически на любом предприятии существует мно-

жество информационных и программных систем, и их можно объединить в единое пространство сохраняя согласованность данных между собой.

Обычно, информационная система состоит из следующих компонентов:

- платформа, на которой работают основные компоненты системы, включая аппаратное (железо) и системное программное обеспечение;
- приложения, реализующие бизнес-логику для работы с системой данных. Состоит из таких компонентов как: бизнес-логика, пользовательского интерфейса, компонентов поддержки (интерфейсов) и сервера приложений, который обеспечивает хранение и доступ к компонентам приложения;
- данные, с которыми работает система.на состоит из различных СУБД;
- бизнес-процессы, представляющие из себя сценарии работы пользователя с системой.

Следовательно, интеграция информационных систем - это интеграция одного или нескольких компонентов информационных систем (объектов интеграции):

- платформ;
- данных;
- приложений;
- бизнес-процессов.

Объединением корпоративных платформ пытаются покрыть следующие требования:

- 1) корректная кроссплатформенная коммуникация приложений между собой (например, между RedHat Linux, CentOS, Windows);
- 2) исправная работа написанных кроссплатформенных приложений.

Существует различные подходы для интеграции приложений и в каждом из этих подходов возможно использование различных технологий:

- технология виртуализации;
- удаленный вызов процедур (REST, RPC, Web-сервисы и пр);
- ПО промежуточного слоя (Microsoft.Net, Java Runtime).

Сейчас становится популярным подход виртуализации. Виртуализация операционной системы - это ПО, позволяющие на аппаратном сервере запускать несколько образов операционной системы одновременно. Виртуализация описывает технологию, в которой приложение, гостевая операционная система или хранилище данных отделены от оборудования или программного обеспечения [3]. Основное использование технологии виртуализации - виртуализация серверов, которая использует программный уровень, называемый гипервизором, для эмуляции базового оборудования. Это часто включает память процессора, ввод-вывод и сетевой трафик. Гостевая операционная система взаимодействует с программной эмуляцией этого оборудования, и зачастую гостевая операционная система даже не подозревает, что находится на виртуализированном оборудовании (рисунок 1.2). Стоит отметить, что производительность виртуальной системы не эквивалентна производительности

ОС, работающей на фактическом оборудовании. Концепция виртуализации работает, поскольку большинству гостевых операционных систем и приложений нет необходимости в полноценной мощности аппаратной части реализации. Такой подход обеспечивает большую гибкость, контроль и изоляцию, устраняя зависимость от конкретной аппаратной платформы. Первоначально предназначенная для виртуализации серверов, концепция виртуализации распространилась на приложения, сети, данные и персональные компьютеры. Примеры технологий виртуализации: Microsoft Hyper-V, KVM, OpenShift, Virtuozzo, VMware, Xen и другие.



Рисунок 1.2 – Традиционная архитектура виртуализации

Технологии удаленных вызовов процедур (RPC - аббревиатура для удаленного вызова процедур) - технология меж сервисного взаимодействия, позволяющая компьютерной программе вызывать процедуру в другом адресном пространстве (обычно на другом компьютере или сервере, соединенном сетью). Программисту нет необходимости уделять внимание деталям реализации удаленного взаимодействия: с точки зрения кода, вызов аналогичен вызовам локальной процедуры.

RPC довольно популярная технология для реализации клиент-серверной модели распределенных вычислений. Удаленный запрос инициируется клиентом, отправляющим сообщение на удаленный сервер для выполнения определенного алгоритма, а после ответ возвращается клиенту. Важней разницей между запросами процедур и удаленными сайтами, запрос процедур заключа-

ется в том, что в первом случае запрос может завершиться неудачей из-за проблем в сети. В этом случае даже не гарантируется, что запрос была вызван.

Технология RPC берет свое начало с 1976 года, когда она была описана в RFC 707. Примером ранней коммерческой реализации этой технологии является ее реализация сделанная для Xerox «Cougier» в 1981 году. Первой популярной реализацией для Unix был RPC от Sun (теперь он называется ONC RPC), использующийся в качестве основы сетевой файловой системы, и до сих пор используется на множестве платформ. В настоящий момент широко применяют SOAP и REST протоколы, которые являются основой современных веб-сервисов.

REST обозначает набор архитектурных принципов для разработки веб-сервисов, предназначенных для системных ресурсов, включая способы обработки и передачи состояний ресурсов через HTTP для различных клиентских приложений, написанных на разных языках программирования. За последние несколько лет REST стал преобладающей моделью проектирования веб-сервисов [4]. Таким образом REST оказал огромное влияние на Web, что почти практически полностью вытеснил дизайн интерфейса на основе SOAP и WSDL при помощи значительно упрощенному стилю разработки.

Технология удаленного вызова также содержит и недостатки, главный из которых является необходимостью работоспособности всех сервисов, с которыми происходит коммуникация. Простой пример, имеется какой-то сервис-каталог, содержащий различную информацию, и в определенное время этот сервис синхронизируется с другими приложениями. Предположительно некоторые части системы в это время могут не отвечать на запрос ввиду неработоспособности.

Многие источники и статьи содержат информацию, что использование технологии удаленного вызова, следует использовать только тогда, когда взаимодействия между сервисами инициируется самим пользователем, который сам контролирует результат.

Такой подход имеет неудачную практику реализации, если есть необходимость в автоматической интеграции. Такой способ был разработан для разработки распределённых систем, в которой сервисы одной системы располагаются на различных станциях.

Идея программного обеспечения middle-layer заключается в разработке прикладного программного обеспечения, который не использует службы конкретно взятой операционной системы (например, Windows API), а применяет службы программного обеспечения промежуточного слоя. Промежуточное программное обеспечение – это программный код, который связывает программные компоненты или корпоративные приложения, представленные на рисунке 1.3 продемонстрирована архитектура текущего подхода. Промежуточное программное обеспечение – это программный уровень, который располагается между операционной системой и приложениями на каждом узле распределенной компьютерной сети. Как правило, он обеспечивает работоспособность сложных распределенных приложений для бизнеса.

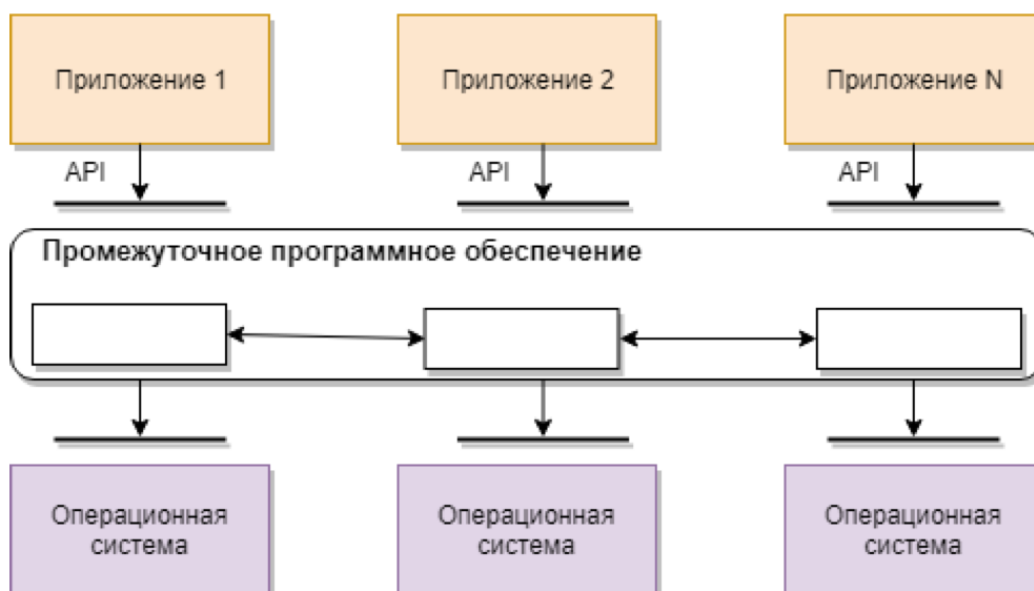


Рисунок 1.3 – архитектура Middleware

Это ПО также принимается как инфраструктура, которая упрощает создание бизнес-приложений и предоставляет основные сервисы, такие как параллелизм, транзакции, многопоточность, обмен сообщениями и инфраструктуру SCA для приложений на основе сервис-ориентированной архитектуры (SOA). Дополнительно обеспечивает безопасность и высокую доступность для предприятия.

Промежуточное ПО является веб-серверы, серверы приложений, системы управления контентом и аналогичные инструменты, поддерживающие разработку и доставку приложений. Оно необходимо для информационных технологий, в основе которых лежит расширяемый язык разметки (XML), протокол простого доступа к объектам (SOAP), веб-службы, инфраструктура SOA, Web 2.0, протокол облегченного доступа к каталогам (LDAP) и другие протоколы и технологии.

Информационные системы работают с различными хранилищами данных. Обычно под хранилищем данных имеют ввиду базу данных (БД), но также встречается практика, когда в обычных файлах (различные таблицы, документы и т.д.) хранят данные. Объединение систем на уровне данных применяет обмен данных из различных систем. Эти данные могут быть интегрированы более простым путем на этом уровне, чем на уровне приложений, так как современные СУБД (система управления базами данных) по умолчанию могут поддерживать различные протоколы передачи информации, что упрощает доступ к данным.

Есть два подхода к интеграции данных:

- хранилище данных;
- универсальный доступ к данным.

Большой вклад в подход универсального доступа к данным осуществила компания Microsoft. OLE DB представляет собой разработанный фирмой

Microsoft набор интерфейсов OLE, обеспечивающих унифицированный доступ приложений к данным из разнообразных источников, включая текстовые файлы, файлы электронной почты, электронные таблицы, данные мультимедиа и прочие данные.

Благодаря OLE DB и ADO универсальный доступ к данным обеспечивает высокопроизводительный доступ к различным источникам информации, включая реляционные и нереляционные источники, и простой в использовании интерфейс программирования, который не зависит от инструмента и языка. Эти технологии дают возможность клиентам интегрировать разнообразные источники данных, создавать простые в обслуживании решения и использовать различные инструменты, приложения и платформы (рисунок 1.4).

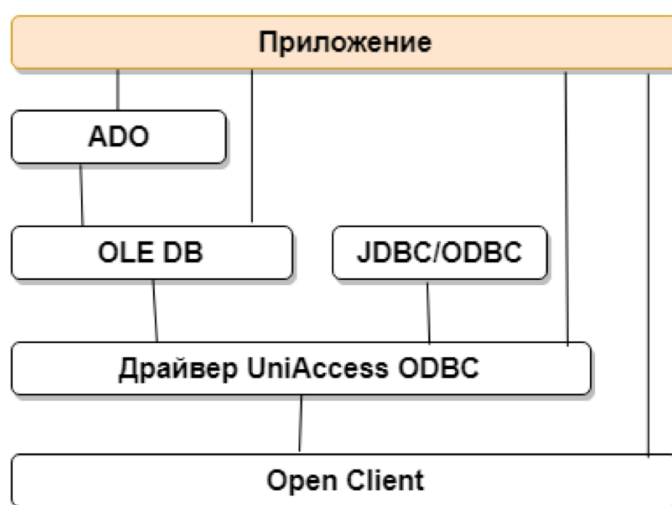


Рисунок 1.4 – Концепция универсального доступа к данным

Для универсального доступа к данным не требуется от пользователя переносить данные из одного хранилища данных, что является дорогостоящим и требует много времени. Универсальный доступ к данным основан на открытых спецификациях с широкой поддержкой и работает со всеми основными платформами баз данных. Универсальный доступ к данным является эволюционным шагом по сравнению с современными стандартными интерфейсами, включая ODBC, RDO и DAO; и расширяет функциональность этих известных и проверенных технологий. Доступ к данным основан на способности OLE DB получать доступ к данным всех типов, и он полагается на ADO, чтобы предоставить модель программирования, которую разработчики приложений будут использовать.

Концепция хранилища данных содержит в себе идеи создания корпоративного хранилища данных, которая содержит в себе объединение данных из нескольких различных источников и хранит информацию с использованием различных технологий при этом обеспечивая единое представление данных. Интеграция данных становится все более важной в случае слияния двух компаний или сведения приложений в одной компании для обеспечения единого представления информации данных.

Вероятно, наиболее известной реализацией интеграции данных является создание warehouse (хранилища данных предприятия). Это позволяет компании проводить анализы на основе данных в хранилище данных. Это было бы невозможно сделать с данными, доступными только в исходной системе. Причина в том, что исходные системы могут не содержать соответствующих данных из других систем, и даже если данные имеют одинаковые имена, они могут ссылаться на разные объекты.

Интеграция на уровне Интеграция приложений (или интеграция корпоративных приложений) – это разделение процессов и данных между различными приложениями. Как для небольших, так и для крупных организаций стало критически важным приоритетом подключение разрозненных приложений и использование совместной работы приложений в масштабах всего предприятия для повышения общей эффективности бизнеса, повышения масштабируемости и снижения затрат на ИТ.

Существуют следующие подходы к интеграции приложений:

- интерфейсы;
- обмен сообщениями (корпоративная сервисная шина);
- Service oriented architecture (сервис-ориентированная архитектура);
- интеграция юзер-интерфейсов (на уровне представления).

Программный интерфейс приложения (API – application programming interface) - является набором инструментов, определений и протоколов для создания и интеграции прикладного программного обеспечения. Позволяет ПО или сервису взаимодействовать с другими ПО и услугами, не зная, как они реализованы.

API упрощает разработку приложений, позволяя тем самым уменьшить время на реализацию проекта и стоимость его. Когда разрабатываются новые инструменты и продукты, или же управляются существующие, интерфейсы предоставляют гибкость, позволяя упростить дизайн, администрирование и использование.

Сервис-ориентированная архитектура (SOA) - это модульный подход к разработке ПО, построенный на практике распределённых, слабо связанных заменяемых компонентов, имеющих стандартизированные интерфейсы для взаимодействия с использованием стандартизированных протоколов. SOA дает возможность сочетать большое количество средств из существующих сервисов для формирования приложений. Стоит отметить так же то, что он включает в себя набор принципов проектирования, который структурирует разработку системы и предоставляет средства для интеграции компонентов в целостную и децентрализованную систему.

SOA занимается объединением всех взаимодействующих сервисов, которые необходимо интегрировать в самые разнообразные программные системы, принадлежащие отдельным поставленным задачам.

Сервис-ориентированная архитектура SOA содержит две важные роли.

Сервис провайдер – является сервисом, который включает в себя функционал и сервисы для использования другими.

Сервис консьюмер – сервис, который содержит в себе мета-данные и содержит в себе все необходимые клиентские компоненты, которые могут быть использованы [5].

При таком походе приложение состоит из нескольких уровней. Эту структуру можно видеть на рисунке 1.5.

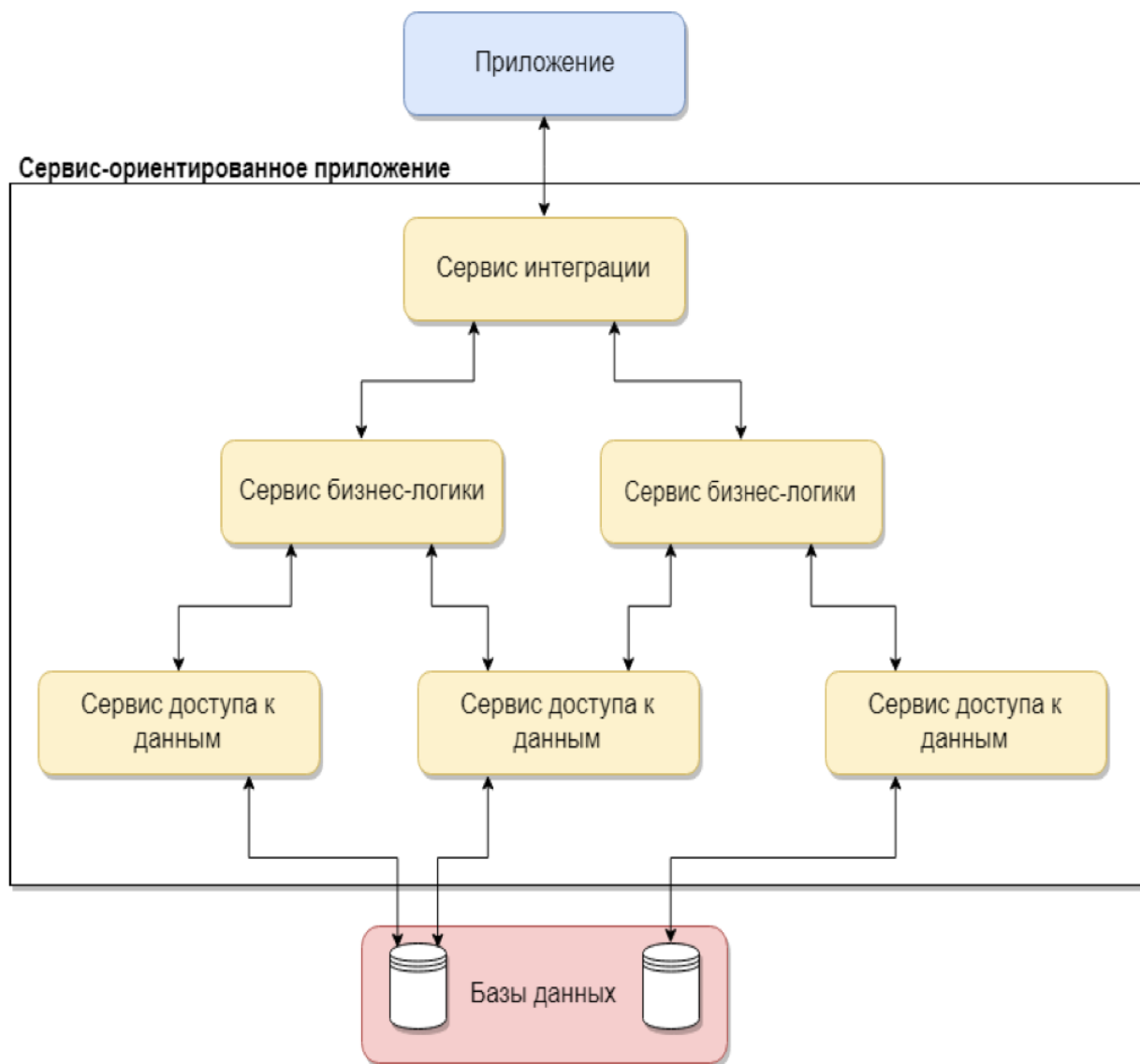


Рисунок 1.5 – Уровни сервис-ориентированного приложения

Самый верхний уровень содержит одну или одну из нескольких служб интеграции, каждая из которых управляет потоками запросов. Каждый сервис интеграции ссылается на один или несколько бизнес-сервисов.

Второй уровень состоит из сервисов, каждый из которых выполняет бизнес-задачу относительно низкого уровня. Сервис интеграции может вызывать серию бизнес-сервисов.

Третий уровень состоит из систем управления базами данных, каждая из которых выполняет относительно техническую задачу чтения и записи в области хранения данных, такие как базы данных и очереди сообщений. Служба доступа к данным чаще всего вызывается из бизнес-уровня, но упрощенный доступ к службам дает возможность применять её по-разному.

Основными направлениями SOA можно назвать:

- публикация функционала корпоративных приложений в виде Веб-сервисов;
- создание новых приложений на основе веб-сервисов путем их объединения.

Цена проектирования новых приложений на основе ранее созданных веб-сервисов будет значительно ниже, чем разработка абсолютно нового приложения или обширная интеграция с другими системами.

На уровне представления интеграция достигается путем объединения нескольких приложений в качестве единого решения с обобщенным пользовательским интерфейсом (UI). Интеграция на уровне представления ранее применялась для интеграции приложений, принцип работы которых был устроен по иной схеме, но технология интеграции приложений не стоит на месте и в данный момент является очень сложной.

Наиболее целостный подход к системной интеграции - это интеграция на уровне бизнес-процессов. На этом уровне интеграции происходит интеграция приложений, интеграция данных и интеграция пользователей [6]. У предприятий есть возможность применять интеграцию приложений для осуществления взаимодействия отдельных взятых приложений с целью автоматизации бизнес-процессов, что дает такие преимущества как быстрая доставка товаров, услуг для клиентов, снижение вероятности человеческих ошибок и снижение эксплуатационных расходов.

Идеи, которые лежат в базовом понимании интеграции бизнес-процессов, довольно просты:

- применяется сценарий бизнес-процесса, который существует в организации, в нём описываются все операции взаимодействия, возникающие между пользователями, систем и между самими системами. Таким образом, бизнес-процессом является элемент логически интегрирующий и включающий в себя различные системы. Бизнес-сценарий появляется при помощи специального ПО, который будет дальше осуществлять контроль этого бизнес-процесса в соответствии с алгоритмом;
- операции взаимодействия с системами в рамках бизнес-процесса подробно описаны в терминах обмена информацией: события, используемые службы, приложения, правила, форматы обмена, и т.п.;
- объединяемые системы, которые участвуют в бизнес-процессе подсоединяются через адаптеры к интегрирующему ПО, описывающему алгоритм бизнес-процесса. Таким образом появляется возможность автоматического обмена данным между системами;

- созданный процесс добавляется на «панель управления» менеджера, откуда он сможет осуществлять управление имеющимися бизнес-процессами, отслеживать их статус, принимать решения по операциям процессов, требующие участия пользователя и т.д. При отсутствии взаимодействия пользователя с системами, это выполняется путем автоматически [7].

На сегодняшний день одной из ключевых тенденций ИТ-отрасли выступают облачные технологии, реализация потенциала которых возможна за счет высокого уровня развития ИКТ, повсеместного распространения широкополосного и мобильного интернета, наличием не только персональных компьютеров, но и мобильных устройств у подавляющего большинства обучающихся. Это стало возможным лишь в последнее десятилетие XX века (в следствие появления высокотехнологичных интернет-технологий и мейнфреймов): вплоть до 1999 года пропускная способность интернета и слабая аппаратная мощность компьютеров в значительной степени сдерживали развитие информационных технологий [8].

Облачные технологии понимают как отдельную отрасль вычислительных технологий, задача которой состоит в том, чтобы обеспечить требования пользователя к удаленному доступу к большому набору вычислительных ресурсов (сервисы, приложения, хранение данных), расположенных в интернете. Доступ к облачным технологиям осуществляется посредством облачных сервисов, которые могут быть бесплатными, условно бесплатными или же полностью платными. На практике облачные сервисы не требовательны к высокопроизводительных устройствах пользователя, им необходимо стабильный и высокоскоростной выход в сеть интернета у пользователя [23].

Первым примером реализации облачной парадигмы является разработка облачного сервиса «cloudweb-service» от компании Amazon в 2002 году. Сервис позволял хранить разнообразную информацию и производить вычисления. По прошествии 5 лет Amazon запустило новый сервис “Elastic Compute Cloud”, позволяющий запускать собственные приложения уже на вычислительной мощности облачного сервиса.

Такой подход компании создал новую тенденцию на рынке в целом, такие компании как Google, Apple, Яндекс, MailGroup и другие начали предлагать свои услуги в сфере облачных технологий.

Технология облачных технологий развивается и обретает новые сервисы, которые упрощают как и интеграцию, так и работу с ней в целом. кроссплатформенность и защищенность данных, удаленный доступ и увеличение вычислительных возможностей клиента только дают положительный опыт как для разработчиков, так и для конечного пользователя.

1.3 Анализ современных информационных систем по оказанию услуг личного кабинета

Чтобы иметь более полноценное представление необходимого функционала и критерии к проектируемому приложению, необходимо провести ана-

лиз уже существующих информационных систем, а затем сравнить их между собой.

Сразу стоит отметить, что количество предложений на рынке может казаться на много больше, но не все обладают достаточным количеством сервисных услуг или же имеют какие-либо другие ограничения. Например: региональные ограничения на использование продукта или же приложение рассчитано строго под определенного заказчика, а находятся они в свободном доступе для скачивания клиентом. Как примером такого продукта может выступать мобильное приложение Северо-Казахстанского государственного университета им. М. Козыбаева под названием “NKZU Student”, изображенного на следующей иллюстрации (рисунок 1.6).

Рассмотреть данное приложение не возможно, ввиду закрытого доступа для посторонних пользователей.

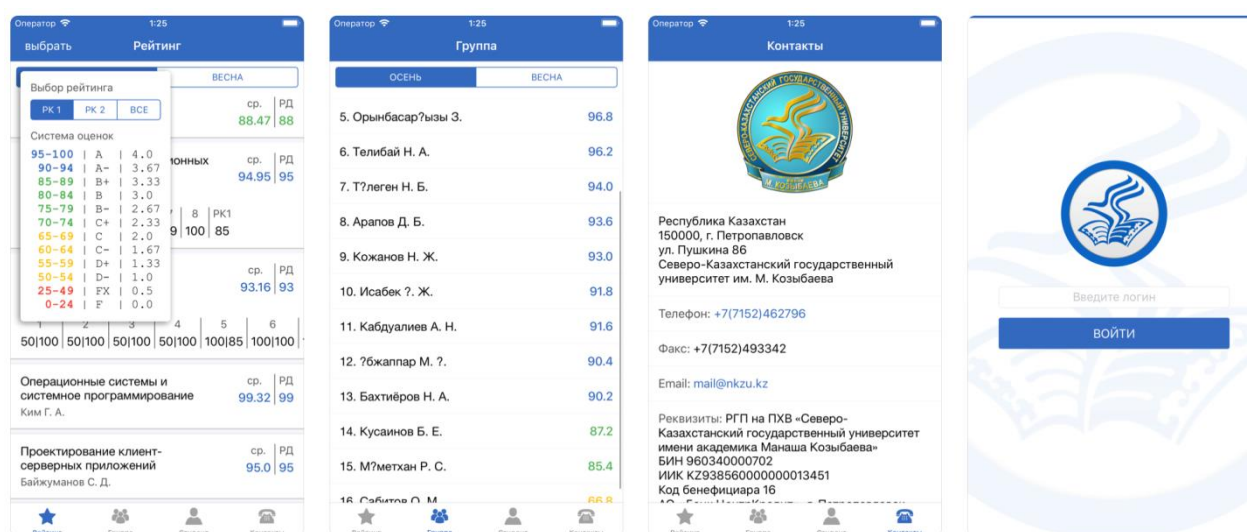


Рисунок 1.6 - интерфейс “NKZUStudent”

MyStudyLife - кросс-платформенный планировщик для студентов, преподавателей и лекторов, призванный облегчить управление вашей учебной жизнью. Приложение позволяет хранить уроки, домашние задания и экзамены в облаке, делая их доступными на любом устройстве. На следующем рисунке можно увидеть окна приложения (рисунок 1.7).

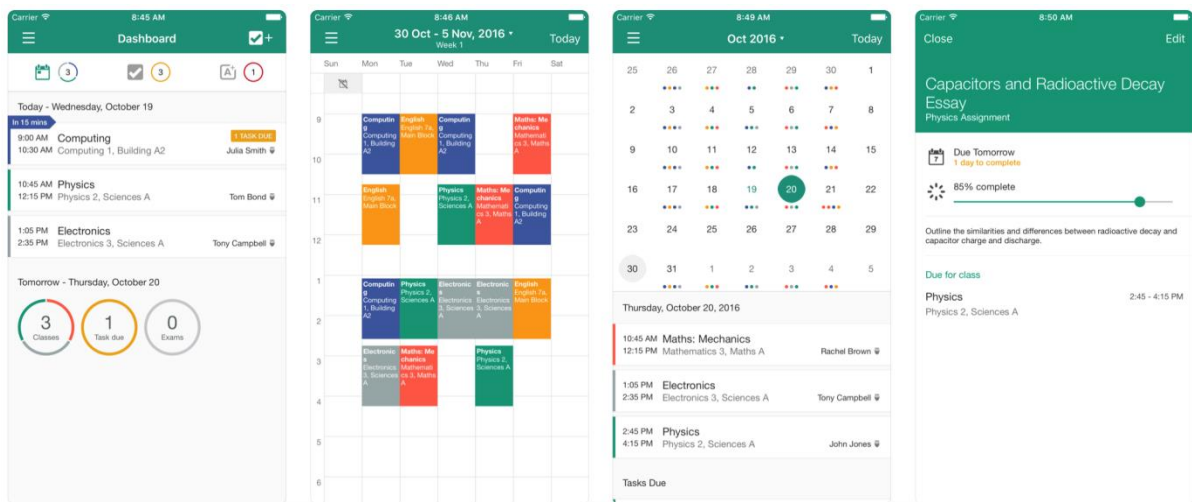


Рисунок 1.7 - MyStudyLife интерфейс

Список возможностей приложения:

- отслеживайте свои задачи - домашние задания, экзамены, напоминания о будущих экзаменах;
- учебный календарь - сохранение предметов в семестре, составление учебного расписания;
- уведомления - напоминания о незавершенных заданиях, предстоящих экзаменах и занятиях еще до их начала.

iStudiezProLegendaryPlanner пожалуй один из самых удобных планёров для студентов в процессе обучения, обладает огромным функционалом и имеет кроссплатформенность. Обладает облачным хранилищем и синхронизацией между устройствами. Представление можно построить по следующей иллюстрации (рисунок 1.8)

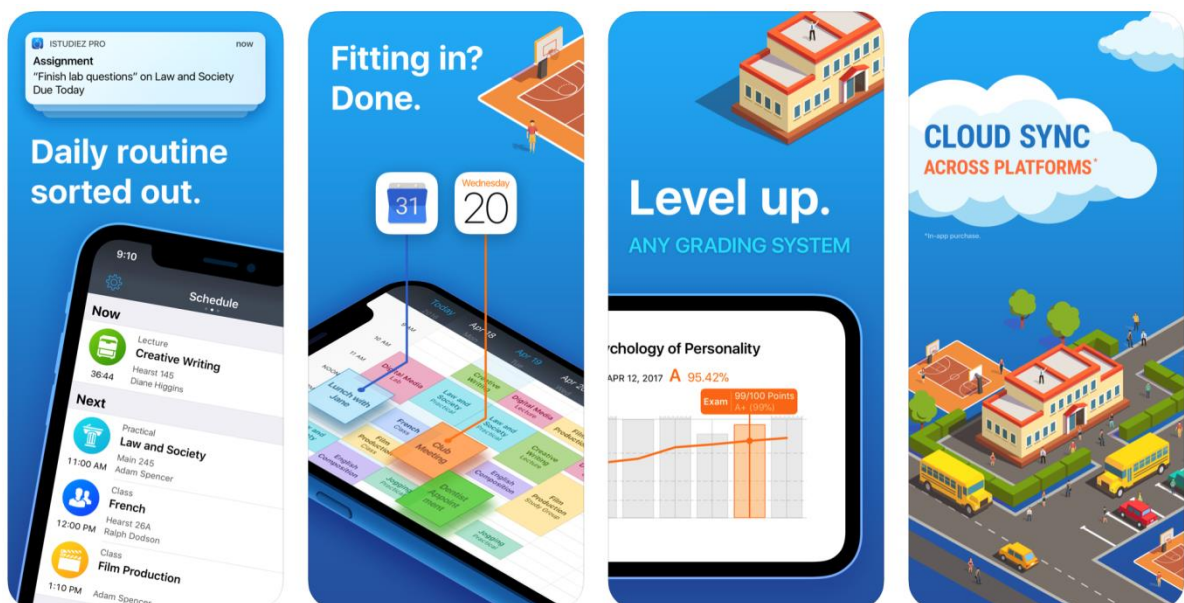


Рисунок 1.8 - демонстрация интерфейса и возможностей

Можно выделить следующие функции:

- учебный календарь - сохранение предметов в семестре, составление учебного расписания;
- интеграция календаря - возможность интегрировать календарь из приложения в родной календарь устройства;
- отслеживайте свои задачи - домашние задания, экзамены, напоминания о будущих экзаменах;
- уведомления - напоминания о незавершенных заданиях, предстоящих экзаменах и занятиях еще до их начала (пользователь может гибко настроить уведомления как ему удобно);
- контактная книжка - можно записывать данные о преподавателях и добавлять важную информацию, такую как звание, кафедра, электронный адрес или ссылки на социальные страницы;
- калькуляция - приложение имеет калькулятор, который считает баллы за пройденные экзамены и домашние задания и предварительно оценивает конечный балл за предмет.

Эти приложения обладают прекрасным функционалом, но вся проблема заключается в том, что пользователь должен самостоятельно заполнять учебный календарь, записывать домашние работы и пополнять данные о преподавателях, что только добавляет больше работы в процессе обучения и может отталкивать их. А калькуляция системы оценивания может меняться в зависимости от ВУЗа и его подхода к расчету оценок.

Краткий вывод по вышеизложенному:

- 1) был проведен анализ современных интеграционных способов интеграции информационных систем: технологии виртуализации, облачных технологий, удаленный вызов процедур (например, REST, SOAPи т.д.), программное обеспечение промежуточного слоя. На основе данного сравнения было выведено, что разрабатываемой модели интеграции необходимо уметь осуществлять работу с различными интерфейсами корпоративных информационных систем;
- 2) проанализированы современные информационные системы по оказанию услуг личного кабинета. Были рассмотрены популярные варианты мобильных приложений;
- 3) сформулированы основные критерии, которые должны быть в разрабатываемом приложении: регистрация, авторизация, аутентификация, хранение данных, демонстрация материала.

2 Методы разработки и интеграции мобильного приложения

2.1 Методы разработки мобильных приложений

Информационные ресурсы, системы и технологии представляют собой неотъемлемые, быстро развивающиеся элементы современной человеческой деятельности. Свидетельством этого может выступать статистика, опыт и раз-

личные исследования зарубежных и отечественных компаний. В 1997 году рынку сотовой связи была представлена технология WAP, которая позволяла осуществлять установку программы на мобильные телефоны напрямую из Интернета не используя кабель, подключенного к компьютеру. Данное событие можно считать точкой отсчета начала процесса «мобилизации». Промышленный комплекс по производству мобильных устройств с большими сенсорными экранами в начале 2000-х годов, позволил улучшить технологии создания новых мобильных приложений, что является качественным прорывом в разработке мобильных приложений.

За тридцать с небольшим лет мобильные операционные системы (ОС) прошли путь от скромной вычислительной среды для работы простых программ-помощников (контакты, записная книжка, калькулятор и др.) до мощного комплекса обеспечения, который обеспечивает полноценную работу специализированных бизнес-приложений под конкретные цели и развлекательных сервисов. На текущий момент мобильные ОС дают возможность производить фото/видео запись, редактировать и просматривать видео в высоком разрешении, играть в 3D-игры, применять искусственный интеллект и технологии дополненной реальности.

Изначально приложения служили для поддержки доступа и взаимодействия с сервисами почты, календаря и контактной книжки, то есть базовые потребности пользователей. К некоторым приложениям пользователи акцентировали на них больше внимания, от некоторых вообще отказывались в виду отсутствия необходимого функционала или же невостребованностью. Спрос пользователей к приложениям с различным функционалом начинал только появляться и производители пытались предложить свои решения. Такая реализация оказалась очень сложна, требует много времени и для обновления текущего ПО до последней версии было необходимо обратиться в сервисный центр, что не всегда было практично для конечного потребителя. Среднестатистический пользователь предпочитал к покупке устройство порой только ради некоторых игр.

Хоть и игровая индустрия оказывает очень сильное влияние на индустрию потребительского рынка, также стоит не забывать о других возможных сферах применения мобильных устройств. Например, корпоративные решения, где в зависимости от направления и сферы деятельности компании, необходим особые функциональные возможности под конкретные цели, не говоря уже о внутренних технологических процессов компании. Также глобальная популяризация мобильных устройств и огромное количество производителей только ускорило рост всей индустрии в целом, а конкуренция между ними за конечного потребителя заставляла улучшать устройства как аппаратно, так и программно в ускоренном темпе.

Если посмотреть на рынок на текущий момент, то можно увидеть, что развитие аппаратной части начало замедленно прогрессировать в виду новых сложностей и отсутствия возможности перейти на новый технический этап (оптическая полупроводниковая среда передачи как пример будущего разви-

тия технологии), но в то же время основное внимание стало уделяться программным продуктам и сервисным услугам, что только повышает требования к качеству программного обеспечения. Выбирая устройство, покупатели нередко сталкиваются с дилеммой между новыми желанными спецификациями и разумной ценой. В среднесрочной перспективе не намечается каких-то прорывных технологических новшеств, которые бы серьезно побудили людей массово переходить на новые аппараты, одновременно обесмысливая приобретение устаревших устройств на вторичном рынке.

В 2019 году объем мирового рынка смартфонов снизился на 2% относительно 2018-го и составил 1,52 млрд штук. Это первый спад с 2008 года, сообщают в исследовательской компании Gartner [9].

2019 год был непростым для производителей смартфонов, в первую очередь из-за переизбытка поставок дорогих устройств на развитых рынках и увеличения цикла обновления телефонов, - говорит вице-президент по исследованиям Gartner Аннетт Циммерманн (Annette Zimmermann). - Однако в 2020 году ожидается восстановление рынка благодаря запуску 5G-сетей в новых странах. Кроме того, скажется то, что пользователи, которые, возможно, отложили свои покупки смартфонов до 2020 года в ожидании падающих цен, снова начнут приобретать трубки.

Рост рынка мобильных приложений исследователи связывают с увеличением расходов потребителей на игры, которые являются наиболее прибыльным источником доходов для разработчиков.

В 2018 году были популярны такие игры, как Fortnite, PUBG и Roblox, создатели которых использовали растущую производительность смартфонов и кроссплатформенный подход. Аналитики предсказывают еще больше популярности в 2019 году, поскольку гаджеты начинают поддерживать все более сложные многопользовательские игры, уровень которых соответствует некоторым домашним игровым консолям.

Еще одним драйвером рынка мобильных приложений стала система подписки, когда пользователи ежемесячно платят за доступ к функционалу и/или контенту. Данная бизнес-модель, как считают эксперты, будет способствовать росту потребительских расходов в магазинах приложений в 2019 году, в результате чего продажи ПО для смартфонов и планшетов увеличатся до \$122 млрд.

Все вышеописанные требования возникли на основании предоставления доступа к качественному контенту разнообразного характера.

Доступ к интернету, социальные сети, мультимедиа и развлечения стали неотъемлемым требованием к устройству, так как без всего этого устройство представляет из себя коробку с электронными схемами.

2.2 Мобильные операционные системы

В 2008-м году на рынок вышла ОС Android. Данная ОС обеспечивала работу смартфонов, планшетов, цифровых плееров, электронных книг, смарт-

часов и браслетов, компьютеров, игровых приставок, телевизоров, роботов и других устройств. В основе ОС лежит ядро Linux одноименной компанией, приобретенной затем Google. Следующим этапом развития Android занимается альянс Open Handset Alliance (ОНА), в состав которых входят более 100 производителей. Фактически Android является безоговорочным лидером среди ОС для мобильных устройств - занимает 87% рынка, по [данным](#) IDC (III квартал 2019-го) за счет большого количества устройств самой разной ценовой категории, использования новейших технологий. На рост Android повлияло то, что при поддержке тачскрин-устройства были дешевле «айфонов», выпускались многими производителями, привлекали новыми возможностями, которые не давали консервативные iOS-устройства (например, поддержка 2 сим-карт появилась в айфонах только в 2018 году с выходом iOS 12.1)[15].

iPhone OS, впоследствии переименованная в iOS, появилась в 2007 году как операционная система для смартфонов, планшетов и умных часов. Разработана на основе OS X. Своей популярностью iOS обязана революционному интерфейсу, в котором разработчики отказались от управления и ввода текста пером (стилусом) и создали систему управления пальцами («мультитач»). Удачная маркетинговая политика, харизма Стива Джобса, привлекательный дизайн способствовали росту продаж смартфонов iPhone вплоть до 2010 года. Кроме того, смартфоны iPhone и планшеты iPad входящие в экосистему Apple, отлично взаимодействуют с ее компьютерной техникой.

Немаловажное значение в распространении iOS и Android сыграло большое количество приложений, легко доступных в онлайн-магазинах AppStore и Google Play. Установка приложений стала возможным по нажатию одной кнопки. до этого было необходимо поиск программы в многочисленных каталогах или на сайтах разработчиков, осуществить покупку каким-то образом, загрузку на компьютер и записать в мобильное устройство, подключив кабель, и только потом установить ее, запустив инсталлятор. Данный алгоритм действий был сложен и для не которых пользователей был не возможен ввиду отсутствия необходимых знаний или аппаратного обеспечения. Сейчас же дошкольники легко справляются с этой задачей.

Сегодня рынок мобильных операционных систем поделен между Android (87%) и iOS (12,3%). По количеству загрузок приложений безоговорочно лидирует Google Play: согласно [отчету](#) аналитической компании SensorTower, в 2019 году на устройства с ОС Android установлено 84,3 млрд приложений против 30,6 млрд скачиваний в App Store. При этом популярность Android растет: количество скачиваний увеличилось на 11,7% по сравнению с 2018 годом, тогда как число загрузок из App Store выросло только на 2,7%. Однако по доходности с большим отрывом лидирует iOS: пользователи «яблочных» устройств потратили \$54,2 млрд в 2019 году против \$29,3 млрд, израсходованных пользователями Android.[17]

Что же касается продаж устройств, то тут оказывается совсем другая картина. Самым популярным смартфоном в мире в 2019 году стала модель

iPhoneXR, которая разошлась в количестве 46,3 млн штук, свидетельствуют данные аналитической компании Omdia.

Ещё один смартфон от Apple - iPhone 11 - расположился на втором месте рейтинга с результатом в 37,3 млн проданных устройств. Тройку лидеров замкнул Samsung A10, продажи которого в 2019 году оказались равными 30,3 млн штук. []

Половина из десяти самых востребованных смартфонов в глобальном масштабе по итогам 2019 года пришлось на продукцию Apple. Устройства Samsung заняли четыре места в этом списке. Xiaomi вошла в рейтинг с одним устройством RedmiNote 7. В 2018 году разные модели iPhone получили шесть позиций в топ-10, телефоны Samsung - три, а Huawei - одну. График изображен на рисунке 2.1.

По подсчетам Omdia, самым популярным смартфоном за 2018 год оказался iPhone 8, который был продан тиражом 31,5 млн экземпляров. На втором и третьем местах расположились iPhone X и iPhone 8 Plus соответственно.

По словам директора по исследованиям и анализу рынка смартфонов в Omdia Джуси Хонг (JusyHong), смартфоны Apple оказываются самыми популярными в мире уже пять лет подряд.



Рисунок 2.1 -График популярных смартфонов

В истории развития мобильных устройств остались и другие операционные системы, но их доля настолько мала на рынке или же они прекратили свое существование, что нет особого смысла их рассматривать для актуальной разработки.

2.3 Категории мобильной разработки

Обычно приложения делят по категориям, исходя из того, для каких целей оно применимо. Каждой категории присущи свои характеристики пользовательской лояльности.

Создание качественного мобильного приложения в настоящее время становится обязательным для любого интернет-проекта. Мобильные приложения, доступные для покупки и установки на смартфон, имеют тематический

контент, четко структурированы по специальным рубрикам и ориентированы на информационные потребности определенных категорий пользователей.

Приложения пользующиеся наибольшим спросом можно вынести в следующие категории:

- 1) развлечения (игровые приложения, мультимедиа, музыка, заказ билетов в театр, кино и т.п.);
- 2) путешествия (заказ отеля, аренда авто, услуги гида, сервис он-лайн-переводчика и т.п.);
- 3) бизнес (финансовые приложения, планирование, торговля, приложения для города, поиск работы и т.п.);
- 4) социальные приложения (социальные сети, глобальные брендовые сети, специализированные (клубные) сети и т.п.);
- 5) еда (заказ и доставка еды, геолокация заведения питания, рецепты);
- 6) спорт (спортивные новости, покупка билетов на спортивные мероприятия, игровые симуляторы);
- 7) образование (обучающие программы, интерактивные курсы и т.п.);
- 8) новости (дайджесты, ленты, рейтинги).

Следует отметить, что представленный список категорий постоянно развивается и пополняется. Более того, происходит трансформация мобильных приложений в новые виды сервисов, включающие 3Э-изображения, дополненную реальность, трекеры физического состояния человека, носимые гаджеты, мобильных агентов в системах безопасности, интеллектуальных агентов в гибридных сетях и когнитивных системах и многое другое.

В процессе работы над приложением в диссертации будет подразумеваться разработка приложения имеющего совокупность некоторых категорий - приложением службой.

Также необходимо отметить, что мобильные приложения по способам разработки делятся на следующие виды:

- 1) нативные (создаются поставщиками платформ и загружаются через их магазины приложений);
- 2) кросс-платформенные (HTML5, веб-сайт и веб-приложение, оптимизированные под мобильное устройство);
- 3) стандартные (гибридные) приложения (содержат в себе некоторые функции нативных и веб-приложений) [16].

В таком случае выбор приложения представляет интерес для разработчиков с точки зрения быстродействия, независимости от поставщика, безопасности работы и ряда других факторов, приведенные в таблице 2.1.

Таблица 2.1 - Критерии сравнения различных типов мобильных приложений по разработке

р	Доступ к функционалу устройства	Скорость работы	Стоимость разработки	Распространение через магазин	Процесс одобрения
Нативное	Полный	Высокая	Высокая	Доступно	Обязательный
Гибридное	Частичный	Средняя (зависит от скорости интернета)	Доступная	Доступно	Обязательный
Веб	Отсутствует	Средняя (зависит от скорости интернета)	Доступная	Недоступно	Отсутствует

2.3.1 Нативные приложения

Название данного типа приложений показывает, что они используют родной язык программирования (с англ. native – родной) для конкретно выбранной платформы разработки. Для Android этим языком является Java (но на текущий момент на замену ему развивается язык Kotlin), тогда как для iOS – objective-C или Swift.

Нативные приложения находятся на самом устройстве, доступ к которым можно получить, нажав на иконку, которые можно установить через магазин приложений (PlayMarket на Android, App Store на iOS и др.).

Разработка ведется специально для конкретной платформы и ввиду этого могут применяться все аппаратные и программные части устройства – камеру, GPS-датчик, акселерометр, компас, список контактов и всё остальное. Также они могут распознавать стандартные жесты, предустановленные операционной системой или совершенно новые жесты, которые используются в конкретном приложении.

В силу того, что нативные приложения оптимизированы под конкретную ОС, они органично вписываются в любой смартфон, под который они были написаны, отличаясь высокой скоростью работы и производительностью.

Нативные приложения могут получить доступ к системе оповещений устройства, а также, в зависимости от предназначения нативного приложения, оно может всецело или частично обходиться без наличия интернет-

соединения. Сравнение положительных и отрицательных моментов нативных приложений выведены в таблицу 2.2.

Таблица 2.2 - Положительные и отрицательные стороны разработки нативных приложений

Плюсы нативных приложений	Минусы нативных приложений
скорость работы и производительность	охват платформ
высокая степень безопасности	длительные сроки разработки
расширенный интерфейс	относительно высокая стоимость разработки
максимально возможная функциональность	необходимость выпускать обновление в косметических целях
способность работать без Интернета	
удобство для конечного пользователя	

Примеры нативных приложений на рынке:

Приложение “Shazam”, осуществляющее определение и поиск информации об играющей на другом устройстве песне:

- устанавливается из магазина приложений;
- для работы необходим доступ в Интернет;
- использует диктофон телефона.

Приложение “Instagram”:

- устанавливается из магазина приложений;
- для работы также необходим доступ в Интернет;
- использует ПО смартфона: камера, геолокация, адресная книга;
- можно включить получение push-уведомлений.

2.3.2 Гибридные приложения

В основе гибридных приложений лежит сочетание веб и нативных приложений. Их особенностью является кроссплатформенность и доступ к функционалу смартфона. Такие приложения могут быть загружены исключительно из маркетов вроде GooglePlay и App Store. Вместе с тем они располагают опцией автономного обновления информации, а для их работы необходимо интернет-подключение. Без наличия последнего веб-функции попросту не работают.

Практика многих компаний показывает, что выбор чаще всего оказывается в пользу разработки гибридных приложений. Это объясняется тем, что гибридные приложения способны соединять достоинства нативных с технологичной актуальностью, которая обеспечивается последними веб-технологиями. Однако, в отличие от нативных, стоимость создания гибридных на порядок ниже, а его скорость – выше. Родство гибридных приложений с веб-приложениями, в свою очередь, даёт плоды в виде того, что в них можно легко и оперативно вносить коррективы. То есть разработчикам не приходится, как в случае с нативным типом приложений, повторно размещать приложение в магазине ради устранения ошибок предыдущей версии или адаптации под малое обновление ОС.

Разработка гибридного приложения может казаться перспективной ещё и потому, что она подразумевает создание сразу под две платформы, то есть кроссплатформенное. В следствии этого упрощается процесс разработки на две платформы сразу. Важным фактором является то, что качество и возможности гибридных приложений зависят от цели которой преследует разработчик при этом используя различные типы фреймворков. Стоит уделить внимание факторам, которые делают гибридные приложения предпочтительным вариантом на фоне остальных.:

- необходимость сэкономить в бюджетном плане;
- необходимо создать относительно несложное приложение с простой анимацией;
- имеется задача оперативной разработки приложения как минимум на 2 платформы.

В таблице 2.3 приведен анализ особенностей разработки гибридного мобильного приложения.

Таблица 2.3 - Положительные и отрицательные стороны разработки гибридных приложений

Плюсы гибридных приложений	Минусы гибридных приложений
стоимость и скорость разработки	некорректная работа при отсутствии интернет-соединения
количество разработчиков	средняя скорость работы на фоне нативных
кроссплатформенность	минимализм в отношении визуальных элементов
опция автономного обновления	

Примеры гибридных приложений:

Приложение “HeartCamera” для iOS, позволяющее украсить фотографию рисованными сердцами и т.п.

- загружается из магазина;
- использует камеру телефона;
- необходимо подключение к Интернету при желании поделиться результатом своей работы;
- можно настроить push-уведомления.

Приложение “TripCase” – органайзер для планирования путешествий.

- загружается из магазина;
- может использовать геолокацию;
- необходимо подключение к Интернету;
- может использовать сотовую сеть;
- можно настроить push-уведомления.

2.3.3 Веб приложения

Мобильные веб-приложения не представляют из себя приложениями по своей сути. Так как веб-приложения являют собой адаптацию и оптимизацию веб-сайта под мобильное устройство. Для того, чтобы воспользоваться им, достаточно иметь на устройстве браузер, знать нужный адрес в сети интернет и иметь выход в сеть интернет (благодаря ему происходит загрузка информации в данном виде приложений).

При запуске мобильного веб-приложения, пользователь активирует все те действия, которые выполняются при переходе на любой веб-сайт, а также появляется возможность установки их на свой рабочий стол, путем создания иконки закладки страницы веб-сайта.

Веб-приложения отличаются кроссплатформенностью. Преимуществом является то, что оно работает без использования ПО. А по причине того, что являются мобильной версией сайта с расширенным интерактивом, веб-приложения не занимают свободное место в памяти смартфона.

Веб-приложения стал широко известен в то время, когда начал развиваться HTML5 и разработчики осознали, что могут получить доступ к имитации множеству функций нативных приложений, просто зайдя на веб-сайт через обычный браузер. На текущий момент сложно сказать, что нету четкого различия между веб-приложениями и обычными веб-страницами, поскольку функционал HTML5 растёт с каждым днём и всё больше сайтов начинают его использовать.

Разрабатываются веб-приложения с помощью инструментов и фреймворков, которые стали традиционными. Вследствие чего процесс их разработки в последнее время существенно ускорился. Специалистов по их разработке, ко всему прочему, так и вовсе предостаточно.

В то же время недостатком веб-приложений является неспособность работать с ними без активного выхода в сеть интернета. В следствии этого мож-

но выделить еще один недостаток – производительность, которая находится на среднем уровне, в сравнении с другими типами мобильных приложений. Более того, она зависит и от возможностей интернет-соединения провайдера услуг и его ограничений. Некоторые сервисные услуги подразумевают наличие высокоскоростного выхода в сеть интернет для оказания своих услуг, например таких как, стриминг видео в реальном времени, предоставление высококачественного контента или же введение видео общения. Об особенностях разработки веб-приложений можно посмотреть в таблицу 2.4.

Таблица 2.4 - Положительные и отрицательные стороны разработки веб-приложений

Плюсы мобильных веб-приложений:	Минусы мобильных веб-приложений:
полный охват платформ	обязательное подключение к Интернету
простой и быстрый процесс разработки	скудный интерфейс приложения
количество компетентных разработчиков	невозможность отправить push-уведомления
отсутствие необходимости загрузки из магазина приложений	производительность и скорость работы
	неудовлетворительный уровень безопасности

Примеры мобильных веб-приложений:

- last.fm считается веб-приложением, хотя, по сути, это в то же время и веб-сайт.

- google.com/maps – веб-сайт, но в то же время это и веб-приложение.

2.4 Методы разработки мобильных приложений

На сегодняшний день сфера мобильной разработки предоставляет различные инструменты и платформы, которые упрощают создание мобильного приложения.

Одной из таких платформ является Appcelerator Titanium. С ее помощью вы можете разрабатывать как десктопные приложения, так и мобильные приложения. На данный момент на этой платформе вы можете создавать приложения для основных мобильных операционных систем. Чтобы написания приложения необходимо знать язык JavaScript [18].

Программа, разработанная на этой платформе, состоит из объектов, которые имеют уникальные свойства и методы. Большой набор объектов из коробки позволяет по максимуму использовать все функции операционной системы.

В приложении, JavaScriptобщается с Appcelerator Titanium API. Фреймворк позволяет использовать различные компоненты пользовательского интерфейса для построения таких элементов:

- 1) текстовые поля;
- 2) кнопки;
- 3) списки.

Элементы управления мобильной платформы поддерживают точность представления этих объектов. В большинстве возможных случаях код, написанный для одной платформы, может работать на других платформах без каких-либо изменений. Однако не все графические элементы конкретной платформы могут правильно отображаться. В этом случае для каждой конкретной платформы необходимо написать собственный сегмент кода.

Используясь этой платформой, разработчик получает преимущество в скорости по сравнению с написанием приложения на Java или Swift, но производительность приложения снижается, поскольку скорость native- программ выше. Таким образом можно сделать вывод что программист должен выбрать, что скорость или скорость важнее в конкретной задаче.

Платформа Xamarin может быть использована для разработки кроссплатформенных мобильных приложений. Xamarin использует язык C#. Используя эту платформу, вы можете писать одну логику приложения для нескольких платформах одновременно - Android, iOS, Windows Mobile. Xamarin состоит из нескольких частей:

- 1) Xamarin.iOS;
- 2)Xamarin Studio;
- 3)Xamarin.Android;
- 4) Компиляторы для iOS и Android;
- 5) Плагины для Visual Studio.

Эти части играют важную роль - с их помощью приложения могут отправлять запросы к интерфейсам приложений на устройствах с операционной системой Android или iOS.

Используя эти платформы, разработчик может создавать как отдельные приложения для конкретной операционной системы, так и используя единую логику - кроссплатформенное приложение. Благодаря этому может быть создан визуальный интерфейс, и вы можете связать с ним код, написанный на C#. Это приложение будет работать на всех основных мобильных платформах Android, iOS и Windows Phone.

Это достигается путем использования технологии Xamarin.Forms. Использование Xamarin дает преимущества в следующих случаях:

- 1) приложение содержит большой кусок кроссплатформенного кода;
- 2) необходимо создать приложение за короткий срок, поддерживая несколько платформ;
- 3) необходимо прототипировать приложение.

Не следует использовать Xamarin в следующих случаях:

- 1) разработка приложения идет для конкретной платформы;

- 2) GUI приложение;
- 3) должны быть удовлетворены определенные / особые требования стабильности.

Еще одним инструментом для разработки кроссплатформенных мобильных приложений является Corona SDK. Разнообразный и удобный инструментальный движок, с помощью которого возможно за короткое время создать игру или приложение. Для того чтобы разрабатывать на этом фреймворке необходимо знать язык Lua, что добавляет нагрузку на процесс разработки [20].

В результате вы получите приложение, которое вы сможете запустить на платформах iOS и Android. В Corona SDK есть практически все необходимое для разработки пользовательского интерфейса, например, ползунки, кнопки и т.д. Приложения на этой платформе производительны и быстры, но все же уступают нативным приложениям.

Можно выделить следующие основные недостатки этой платформы:

- 1) документация преимущественно на иностранном языке;
- 2) приложение компилируется на сервере;
- 3) сложная отладка приложения.

Тот или иной подход к созданию мобильных приложений имеет свои преимущества и недостатки, вследствие этого разработчик должен выбирать технологии на основе потребностей и задач, которые должно выполнять это приложение.

В качестве платформы для которой будет разрабатываться приложение была выбрана платформа iOS, потому как показывает статистика – 13,5% рынка мобильных платформ принадлежит iOS. Такой процент удерживается уже долгое время и пользователи данной системы не спешат переходить на Android. А приложения крупных проектов до сих пор выходят на рынок этой платформы и некоторые из них остаются эксклюзивными. Также покупательная способность клиентов iOS выше, чем у Android клиентов. Хотя и Apple накладывает ограничение на среду разработки, это не является большой проблемой, так как инструментальный для разработки всегда актуален и обновляется.

2.5 iOS

iOS (ранее iPhone OS) - это мобильная операционная система, созданная и разработанная Apple Inc. исключительно для своего оборудования. Это операционная система в настоящее время работает на многих мобильных устройствах компании, таких как iPhone и iPod Touch. Также работала на iPad до появления iPadOS в 2019 году. Это вторая по популярности мобильная операционная система в мире после Android. Построена она на XNU (акроним англ. X is Not Unix) [21].

XNU - ядро компьютерных операционных систем, разрабатываемое компанией Apple и используемое в ОС семейства OS X. Исходные коды ядра были опубликованы под открытой лицензией (APSL 2.0) как часть ОС

Darwin. Изначально ядро разрабатывалось компанией NeXT для ОС NeXTSTEP. Архитектурно являлось гибридным ядром на базе микроядра Mach версии 2.5 (разработано в Carnegie Mellon University), компонентов от 4.3BSD и объектно-ориентированного интерфейса драйверов Driver Kit.

После приобретения NeXT компанией Apple микроядро Mach было обновлено до версии 3.0, компоненты ядра BSD были обновлены с использованием наработок проекта FreeBSD, а Driver Kit был заменён на C++ API для драйверов под названием I/O Kit.

Архитектура iOS схожа с базовой архитектурой операционной системы Mac OS X. На самом высоком уровне, iOS представляет собой промежуточный слой между оборудованием устройства и приложениями, которые отображаются на экране. Очень редко приходится создавать приложения, которые будут обращаться к оборудованию напрямую. Вместо этого, приложения взаимодействуют с оборудованием через набор четко-определённых системных интерфейсов, которые защищают приложения от изменений оборудования. Эта абстракция позволяет очень легко писать приложения, которые корректно работают на устройствах с различными аппаратными возможностями.

Хотя приложения в целом защищены от изменений оборудования, вам все равно приходится учитывать различия между устройствами при написании кода. Например, у некоторых устройств есть камера, а у некоторых ее нет. Если приложение умеет работать при наличии или отсутствии какой-то функции, то используя интерфейсы соответствующей библиотеки можно определить, доступна эта функция или нет. Приложения, которым требуется наличие определенного оборудования, должны декларировать это требование в файле со списком свойств приложения (Info.plist)

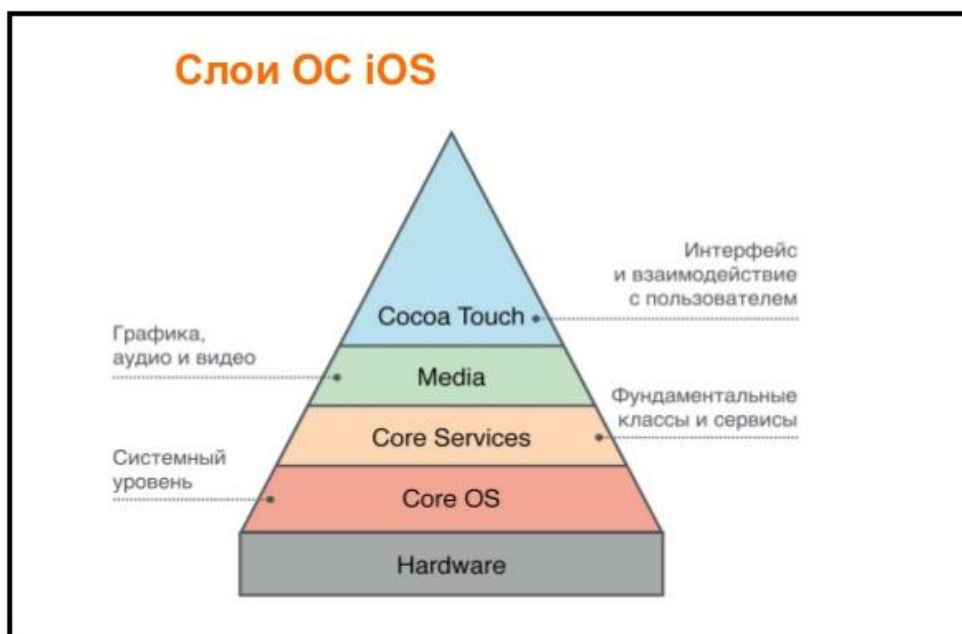


Рисунок 2.2 - Слой ОС iOS

Реализация технологий iOS может быть представлена в виде набора слоев, которые показаны на иллюстрации выше (рисунок 2.2). Фундаментальный слой операционной системы имеет основные службы и технологии, от которых зависят все приложения; на более высоких уровнях находятся более сложные службы и технологии. Библиотеки более высокого уровня написаны для того, чтобы обеспечить объектно-ориентированные абстракции для низкоуровневых структур. Эти абстракции существенно облегчают написание кода, потому что они уменьшают объем кода, который необходимо написать и скрывают достаточно сложные функции, такие как сокеты и потоки. И хотя они скрывают низкоуровневые функции, эти функции по-прежнему доступны для разработчиков.

2.6 Метод разработки приложения на iOS

Выбран тип нативного мобильного приложения для iOS. Необходимо разработать способ регистрации новых пользователей, систему авторизации и аутентификации пользователей и создать переход в рабочее окно.

Необходимо интегрировать базу данных (БД) или же файловое хранилище, откуда возможно загружать и извлекать данные для блоков представления интерфейса.

В качестве сервиса БД будет использоваться “FireBase” от компании Google. Это компания поставщик облачных услуг, которая позволяет разработчикам хранить и синхронизировать данные между пользователями.

Ключевые компоненты средств разработки на iOS включают:

Среда разработки Xcode - представляющая из себя набор инструментов для разработки iOS приложений, в которые входят следующие ключевые компоненты [22]:

Xcode - интегрированная среда разработки, которая управляет проектами приложений и позволяет редактировать, компилировать, выполнять и отлаживать ваш код. В Xcode интегрировано много других инструментов, но он является главным приложением, используемым во время разработки.

InterfaceBuilder - инструмент, для графического создания пользовательского интерфейса. Создаваемые объекты интерфейса сохраняются в файле ресурсов и загружаются приложением во время выполнения.

Instruments - утилита для анализа производительности и отладки приложений во время выполнения. Вы можете использовать эту программу для сбора информации о поведении вашего приложения во время выполнения и для поиска потенциальных проблем.

Эмулятор iOS - это приложения для Mac OS X, которое эмулирует работу операционной системы iOS, позволяя вам тестировать ваши iOS приложения локально на компьютере Macintosh с процессором Intel.

Библиотека Разработчика iOS - справочная документация, в которой находится информация о технологиях и процессе разработки приложений iOS.

Хотя вы можете запускать приложения в эмуляторе iOS, инструменты Xcode также позволяют запускать и отлаживать приложения непосредственно на присоединенном устройстве. Эмулятор идеально подходит для сборки и быстрого тестирования приложений, но он не является заменой для тестирования на реальном устройстве. Разработка на реальном устройстве требует подписки в платной программе от Apple “iOS Developer Program” и конфигурирования устройства для разработки. Но в последние годы можно производить проверку собственных приложений на собственном устройстве бесплатно.

2.7 Разработка архитектуры мобильного приложения

Выбор подхода и архитектуры мобильного приложения играет важную роль, так как от этого будет напрямую зависеть успех приложения, его стоимость, стоимость дальнейшей поддержки и легкость внедрения нового функционала.

В ходе исследовательской работы разработана следующая архитектура мобильного приложения.

В качестве платформы для которой будет разрабатываться приложение была выбрана платформа iOS, потому как показывает статистика – хотя аудитория Android более 85% рынка мобильных платформ, в то же время продажи App Store выше и устройства на iOS более популярны на рынке.

Логику приложения и её визуальное представление находятся в коде самого приложения, интегрированная БД является в первую очередь хранилищем данных с которым взаимодействует приложение. Коммуникация между iOS-приложением и сервером будет происходить по протоколу HTTP. Такой подход позволит доставлять до пользователя новый функционал за максимально короткий период, все будет сводиться к изменениям на стороне сервера и минимальным изменениям со стороны мобильного приложения. Концепция архитектуры можно видеть на рисунке 2.3

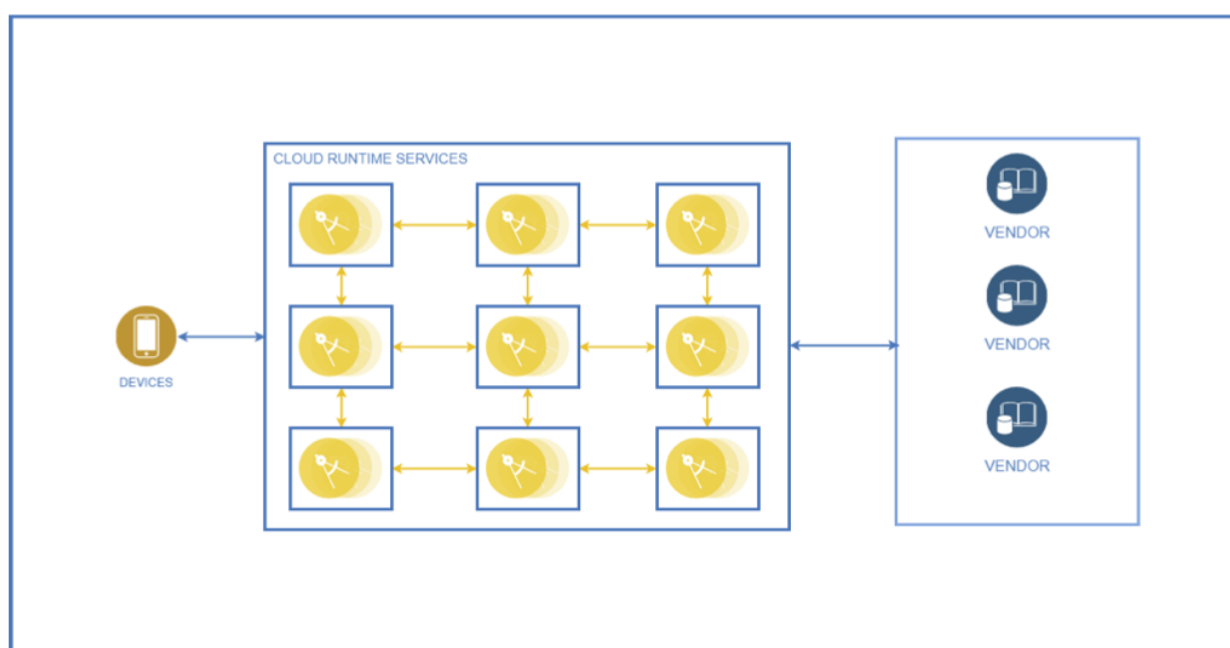


Рисунок 2.3 - Концептуальная архитектура приложения

Для реализации серверной части приложения будут использоваться облачные технологии, так как сейчас они набирают популярность и позволяют абстрагироваться от железной составляющей сервера. Облачные системы - это модели для обеспечения удобного и непрерывного сетевого доступа к набору настраиваемых вычислительных ресурсов (таких как системы хранения, приложения, ресурсы памяти, сети, и службы), которые могут быть выделены для той или иной задачи и запущены с минимальными усилиями по управлению и необходимостью взаимодействия с провайдером.

3 Модель интеграции мобильного приложения с КИС

3.1 Разработка модели интеграции мобильного приложения с КИС

Архитектура разрабатываемой информационной системы может быть расширена, но на текущий момент имеется возможность использовать Firebase.

Firebase - это BaaS, единое решение для реализации всех сервисов в мобильном приложении, можно сказать, что это попытка компании Google объединить всех разработчиков сервис-ориентированных приложений. Backend-as-a-service (MBaaS, BaaS) - является моделью обеспечения разработчиков разными инфраструктурными возможностями, такими как облачное хранилище, интеграция с социальными сетями, пуш-уведомления, управление пользователями и многое другое. BaaS значительно упрощает создание приложений, предлагая уже готовые функции и созданную серверную сторону приложения самой компанией [10].

По заявлению Google, Firebase помогает создавать лучшие мобильные приложения и развивать бизнес использующий информационные системы. Платформа предоставляет нам многочисленные сервисы: аутентификацию, хранилище, базу данных, аналитику, тестирование приложений и еще ряд функций, который они расширяют со временем или будущими тенденциями.

Для того, чтобы подключиться к сервисным услугам Firebase необходимо зарегистрировать электронный адрес в Google, после этого мы можем пройти процесс авторизации в системе Firebase (рисунок 3.1). После того, как мы авторизуемся в системе, мы можем создать новый проект.

Создание нового проекта необходимо для генерации ключа для подключения и создания БД будущего или текущего приложения. Когда создается новый проект, Firebase регистрирует приложение на своих серверах и создает уникальный ключ API, который необходим для интеграции приложения с сервисом (рисунок 3.2). Такой способ связи нам предлагает сам сервис. Так как сервис использует технологию API, при необходимости в будущем можно расширить список подключаемых технологий, таких как web или же Androidk существующему проекту.

Когда сервис закончит его генерировать будет создан файл, этот файл необходимо загрузить в папку проекта Xcode и далее с помощью терминала с применением технологии CocoaPods инициализировать Podсфайл сервиса перед этим указав какие функции сервиса будут необходимы для использования в проекте (рисунок 3.3).

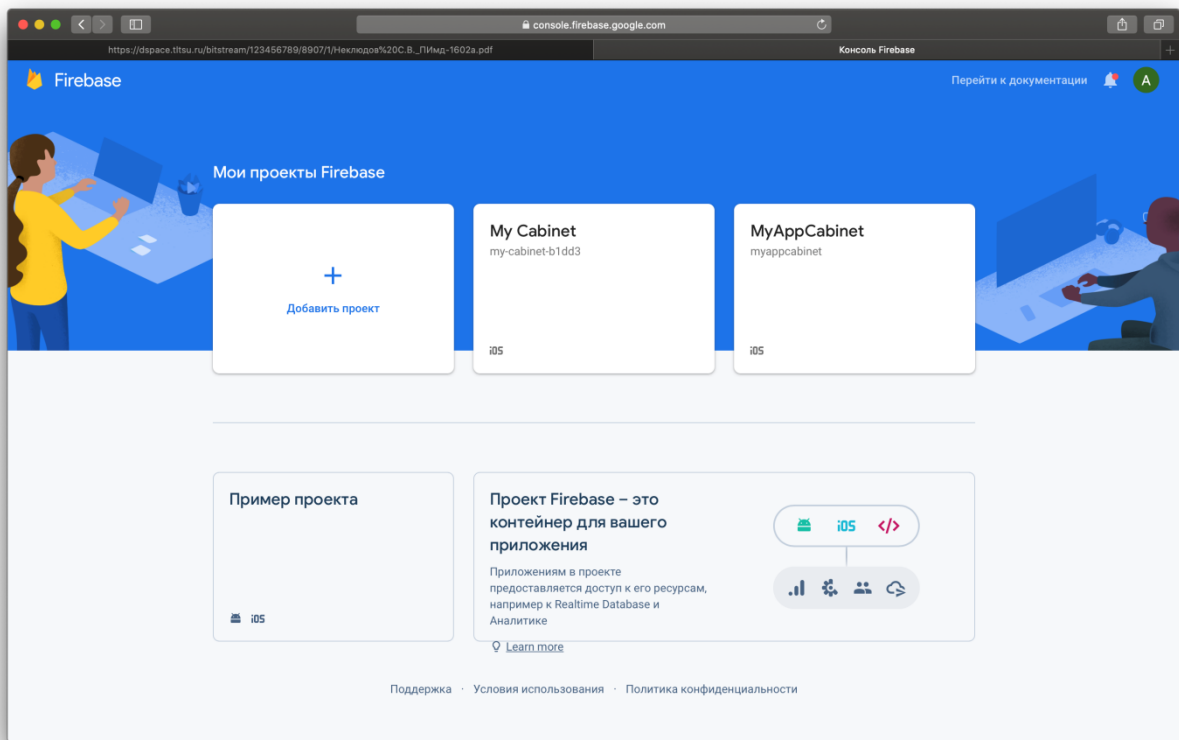


Рисунок 3.1 - Firebase после авторизации (окно консоли)

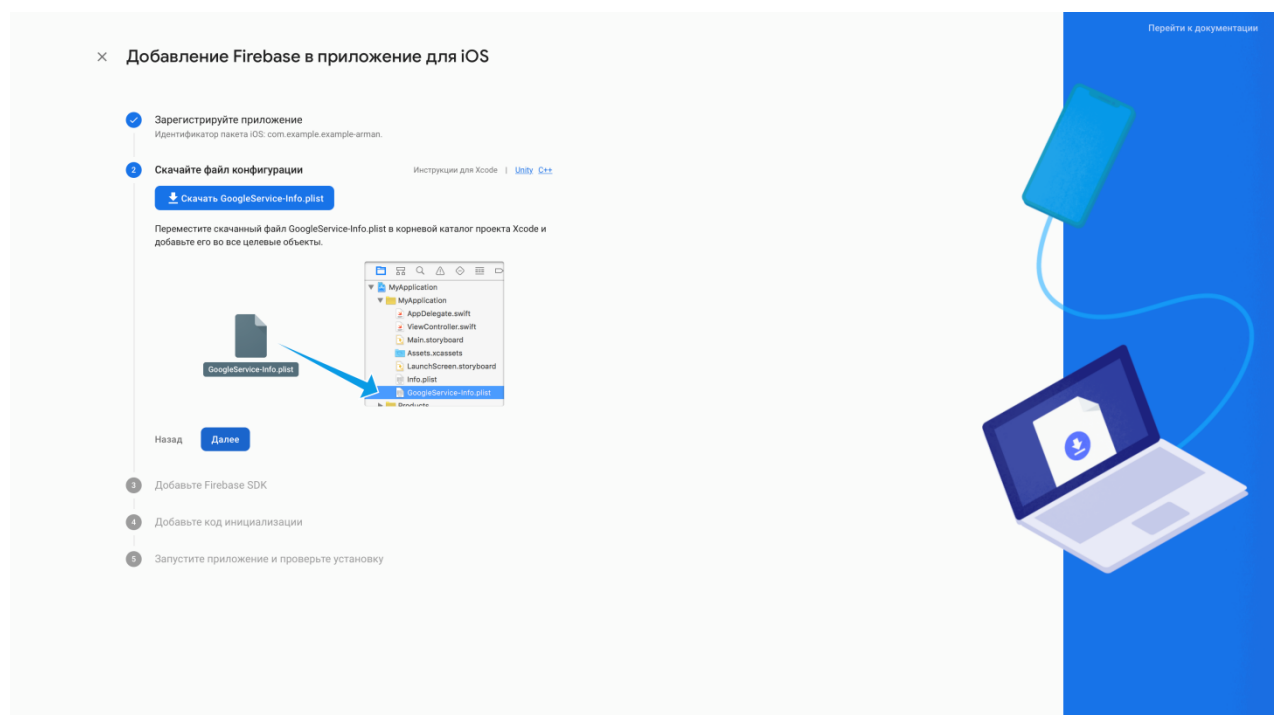
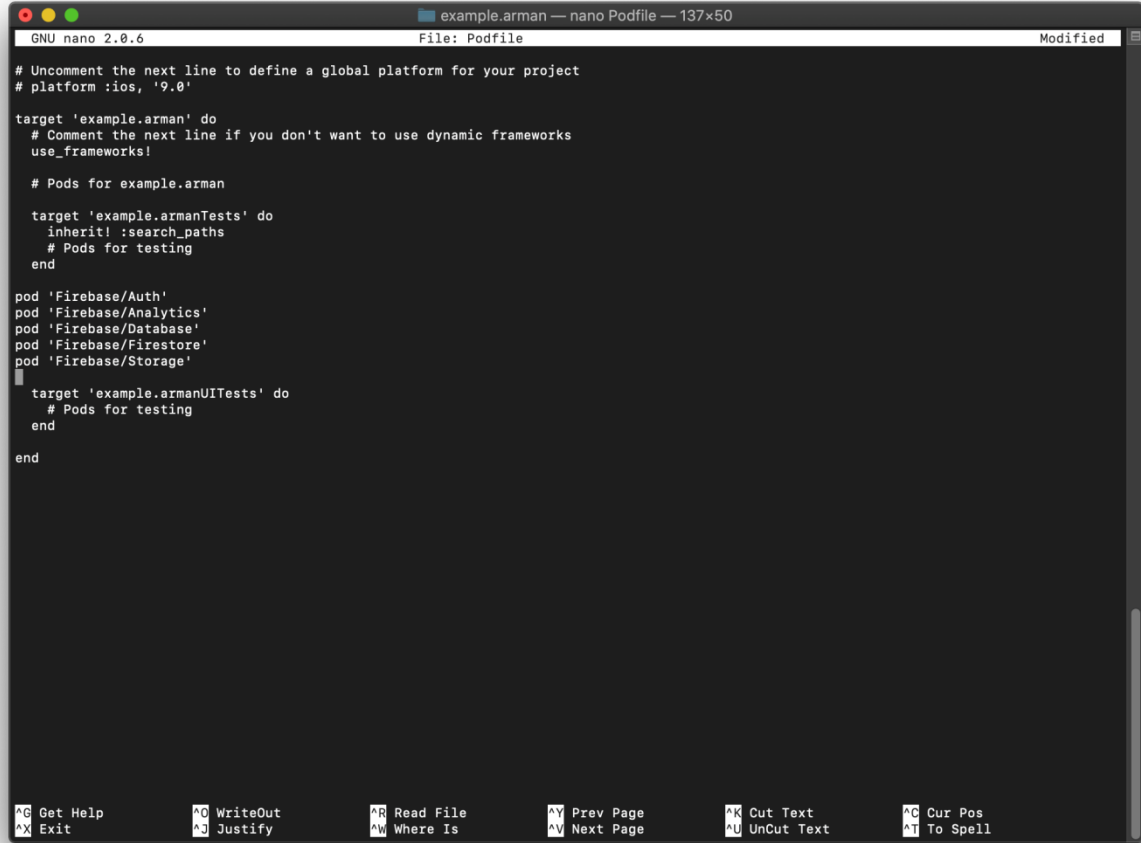


Рисунок 3.2 - Создание podфайла для генерации ключа

Далее установка будет соответствовать инструкции сервиса, которую можно посмотреть на официальном сайте в разделе документации[11].

Непосредственно когда мы подключим наш проект необходимо будет подключить к нему необходимые сервисные услуги, такие как авторизация, база данных в реальном времени или же классическую базу данных (для хранения более статических данных, таких как изображения, документы или другие типы файлов). Перечень функций, которые можно подключить к проекту можно посмотреть в документации Firebase [12]. В рамках проекта будет подключена система авторизации, аналитики, хранения данных, хранилище для меда файлов. Для этого необходимо ввести команду `podintv` терминале находясь в директории проекта и после отредактировать созданный файл введя туда необходимые функции в проекте. На скриншоте терминала (рисунок 3.3) ниже производится редактирование файла с помощью команды `nano`.



```
GNU nano 2.0.6                               File: Podfile                               Modified
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'

target 'example.arman' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for example.arman

  target 'example.armanTests' do
    inherit! :search_paths
    # Pods for testing
  end

  pod 'Firebase/Auth'
  pod 'Firebase/Analytics'
  pod 'Firebase/Database'
  pod 'Firebase/Firestore'
  pod 'Firebase/Storage'

  target 'example.armanUITests' do
    # Pods for testing
  end

end
```

⌘ Get Help ⌘ WriteOut ⌘ Read File ⌘ Prev Page ⌘ Cut Text ⌘ Cur Pos
⌘ Exit ⌘ Justify ⌘ Where Is ⌘ Next Page ⌘ UnCut Text ⌘ To Spell

Рисунок 3.3 - Создание и добавление необходимых возможностей

После этого необходимо будет инициализировать установку прописанных функций в наш проект, для этого выполняется команда `podinstall`. На скриншоте терминала (рисунок 3.4) ниже показан процесс установки файлов и его завершение. После этого необходимо будет открыть проект уже с новым расширением workspace все работы уже проводить в нем.

Важный момент установки в том, что без использования технологии CocoaPods нельзя инициализировать установку. Он устанавливается отдельно и представляет из себя Ruby-проект, воплощающий в себе средства управления зависимостями Cocoa-библиотек. На практике представляет упрощенный способ подключения существующих библиотек, утилит и инструментов.

```
armankastan@MBP-Arman example.arman % ls
GoogleService-Info-3.plist      example.armanTests
example.arman                   example.armanUITests
example.arman.xcodeproj
armankastan@MBP-Arman example.arman % pod init
armankastan@MBP-Arman example.arman % nano Podfile
armankastan@MBP-Arman example.arman % pod install
Analyzing dependencies
Downloading dependencies
Installing BoringSSL-GRPC (0.0.3)
Installing Firebase (6.22.0)
Installing FirebaseAnalytics (6.4.1)
Installing FirebaseAuth (6.5.1)
Installing FirebaseAuthInterop (1.1.0)
Installing FirebaseCore (6.6.6)
Installing FirebaseCoreDiagnostics (1.2.3)
Installing FirebaseCoreDiagnosticsInterop (1.2.0)
Installing FirebaseDatabase (6.1.4)
Installing FirebaseFirestore (1.12.0)
Installing FirebaseInstallations (1.1.1)
Installing FirebaseStorage (3.6.1)
Installing GTMSessionFetcher (1.3.1)
Installing GoogleAppMeasurement (6.4.1)
Installing GoogleDataTransport (5.1.1)
Installing GoogleDataTransportCCTSupport (2.0.2)
Installing GoogleUtilities (6.5.2)
Installing PromisesObjC (1.2.8)
Installing abseil (0.20190808)
Installing gRPC-C++ (0.0.9)
Installing gRPC-Core (1.21.0)
Installing leveldb-library (1.22)
Installing nanopb (0.3.9011)
Generating Pods project
Integrating client project

[!] Please close any current Xcode sessions and use `example.arman.xcworkspace` for this project from now on.
Pod installation complete! There are 5 dependencies from the Podfile and 23 total pods installed.
```

Рисунок 3.4 - Установка необходимых библиотек через терминал

Открыв новое расширение файла стоит сразу перейти в библиотеку файлов и проверить наличие записи API-ключа к Firebase. Бывают случаи когда происходит сбой в процессе установки проекта и возможна ошибка и проект не будет связан, в такой ситуации скорее всего придется производить алгоритм интеграции с нуля. На скриншоте программы (рисунок 3.5) ниже можно увидеть, что после установки библиотек у нас генерировался ключ и он записан.

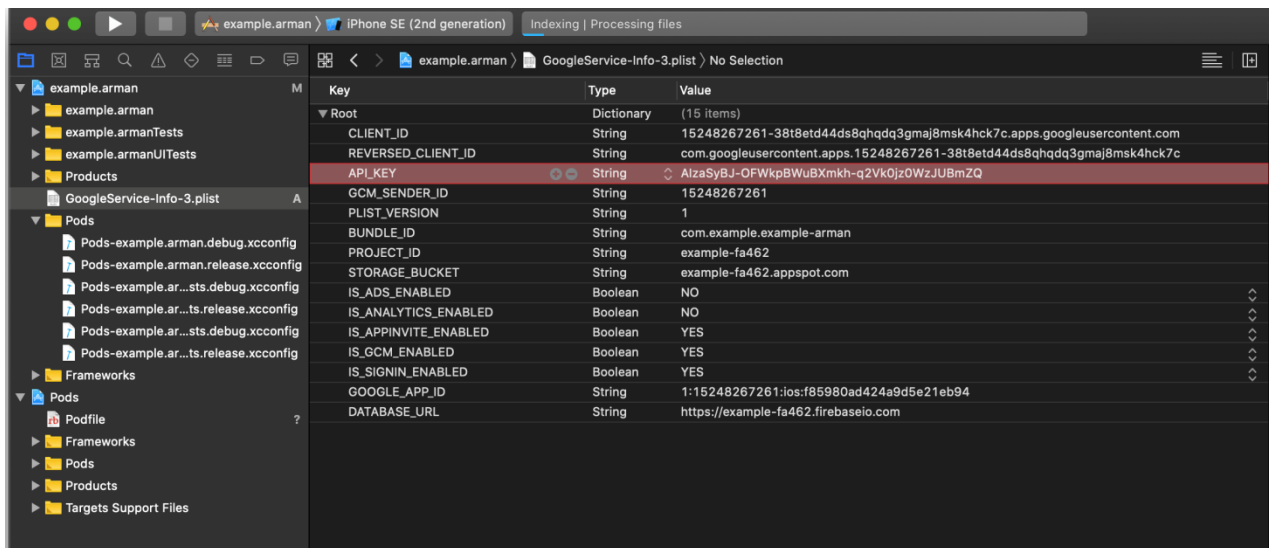


Рисунок 3.5 - Проверка наличия APIключа в проекте

При успешной установке можно зайти в терминал Firebase и включить необходимые функции уже в самой консоли находящейся в панели инструментов слева. В них можно прописать необходимые правила для проекта, ограничивать запись или чтение БД. Консоль проекта можно посмотреть на рисунке 3.6.

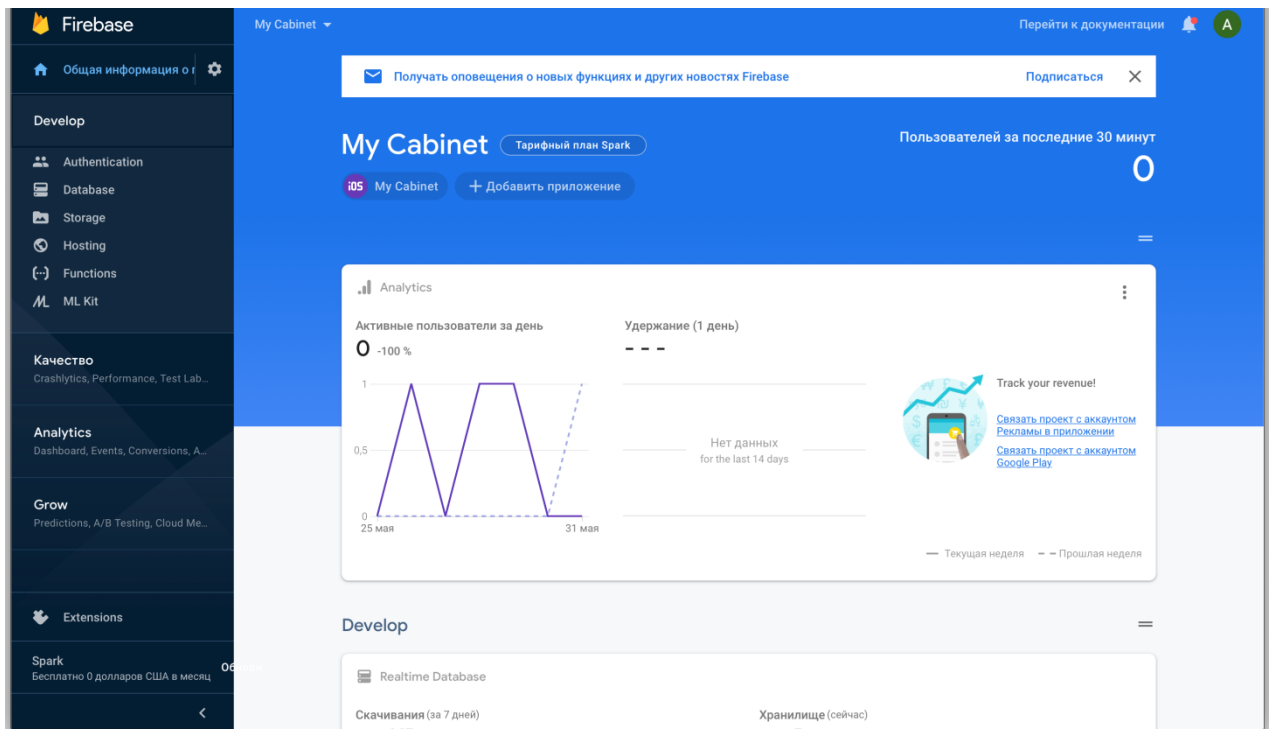


Рисунок 3.6 - Консоль проекта (панель инструментов слева)

С помощью консоли и сервиса FirebaseAnalytics можно смотреть как часто пользуются приложением, выводить статистику и многое другое. На рисунке 3.6 выше видно, что уже построен график активности приложения по

данным, когда производилась авторизация в приложении для отладки и контроля процесса разработки.

3.2 Разработка мобильного приложения

В процессе разработки мобильного приложения необходимо учитывать уже ранее разработанные проекты, так как многие пользователи привыкают к некоторым функциональным или визуальным решениям уже реализованных проектов. Это необходимо для создания более дружелюбного интерфейса для конечного потребителя.

Чтобы иметь более наглядное представление о дизайне приложения можно воспользоваться программой “Sketch” (рисунок 3.7), она представляет из себя минималистичный визуальный редактор, позволяющий дизайнерам или клиентам спроектировать дизайн окон и переходов в проекта, а также многие дизайнеры предоставляют свое представление удобного дизайна на интернет платформах. Благодаря таким шаблонам можно быстро построить свое представление о будущем функционале проекта и редактировать представление о функциональных возможностях. Такой подход имеет экономическое обоснование, нарисовать приложение быстрее и проще, чем создавать сразу функциональные возможности и сервисы на ощупь без общего представления команды разработчиков.

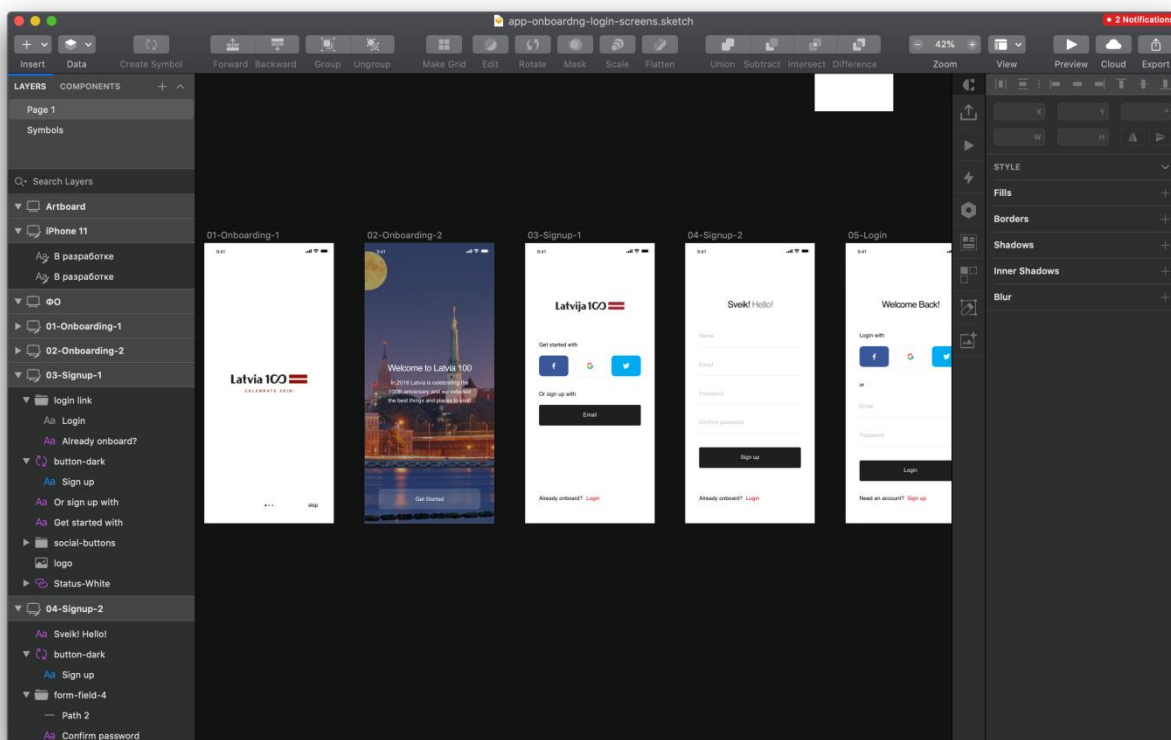


Рисунок 3.7 - Окно программы “Sketch” с примером использованного шаблона

Интернет ресурс “sketchappsources” представляет большую базу самых различных существующих или проектируемых приложений в бесплатном до-

ступе. Разработчику просто необходимо перерисовать элементы интерфейса по их шаблону и уже написать непосредственно код на эти элементы интерфейса [24].

Разработка нативного мобильного приложения осуществляется предоставляемым ПО “Xcode” компанией Apple. Среда разработки обладает обширным набором инструментов и сервисами, которые необходимы процессе создания проекта, если их не хватает, то их можно добавить. Программа эмулятора, входящая в среду разработки поможет отслеживать работоспособность программы, а консоль Firebaseотслеживать связь приложения. Также в процессе разработки происходил и запуск на реальном устройстве, так как эмуляция не всегда дает возможность иметь ясное представление масштабирования интерфейса и комфортность работы с приложением.

Для того, чтобы осуществлять проверку работоспособности приложения на реальном устройстве будет необходимо зарегистрировать AppleIDна сайте developer.apple.com, где произвести подтверждение и получить доступ к форуму разработчиков. Там находится вся документация по библиотекам, протоколам и другой необходимой литературе в процессе разработки.

На следующем скриншоте экрана (рисунок 3.8) можно увидеть окно программы Xcode с открытым проектом.

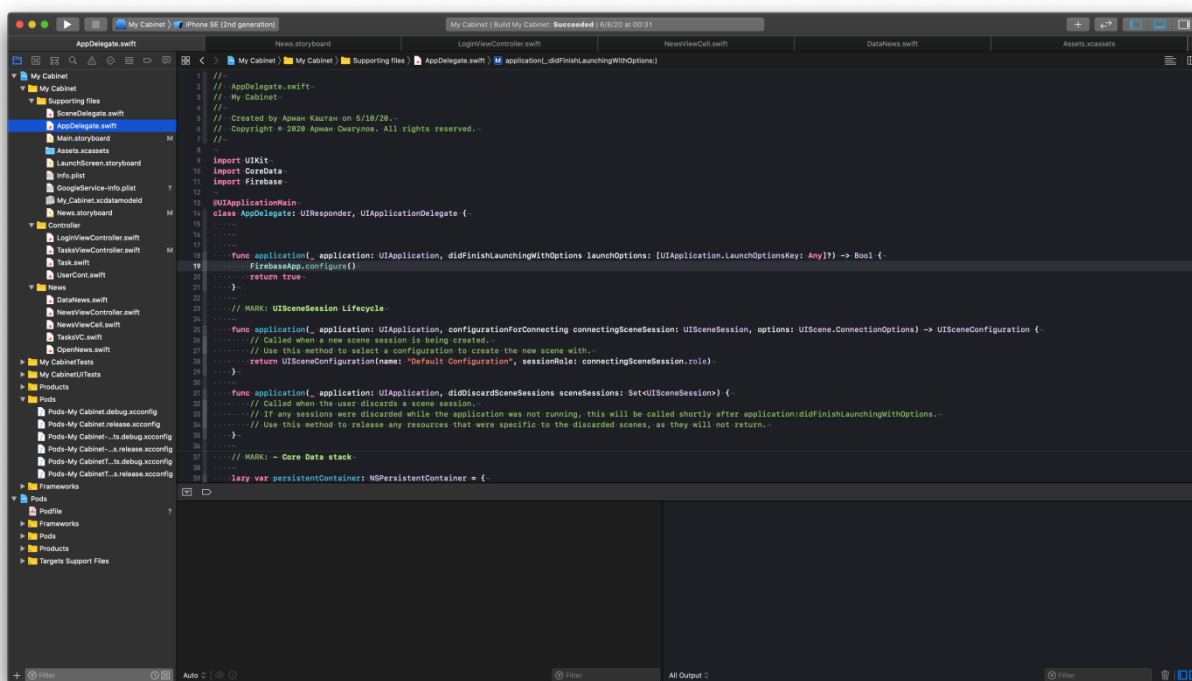


Рисунок 3.8 - Программа Xcode

Рассмотрим окно авторизации, его проектирование и интеграция в само приложение.

Для построения простого принципа авторизации было решено оставить регистрацию по электронному адресу и паролю. Для выполнения этой задачи

было необходимо использовать только два текстового поля для ввода информации. Особенное условие для поля было оставлено для пароля, необходимо его настроить так, чтобы при вводе данных они визуально были сокрыты другим символом, что было и сделано. Ниже были добавлены две функциональные кнопки: авторизация и регистрация.

Для авторизации пользователя достаточно ввести верные данные, логин и пароль, если они не будут таковыми, то необходимо предупредить пользователя, что он вводит не верные данные, иначе если система не будет откликаться, то возможно формирование представления неработоспособности приложения. Для этого будет добавлен сокрытая строчка, которая будет появляться на короткий временной интервал для отображения ошибки. Добавленные элементы можно посмотреть на скриншоте экрана ниже (рисунок 3.9).

Реализовав интерфейс окна визуально с использованием Storyboard надо создать связь всех созданных элементов непосредственно в контроллере нашего представления, а самому представлению дать идентификатор файла контроллера.

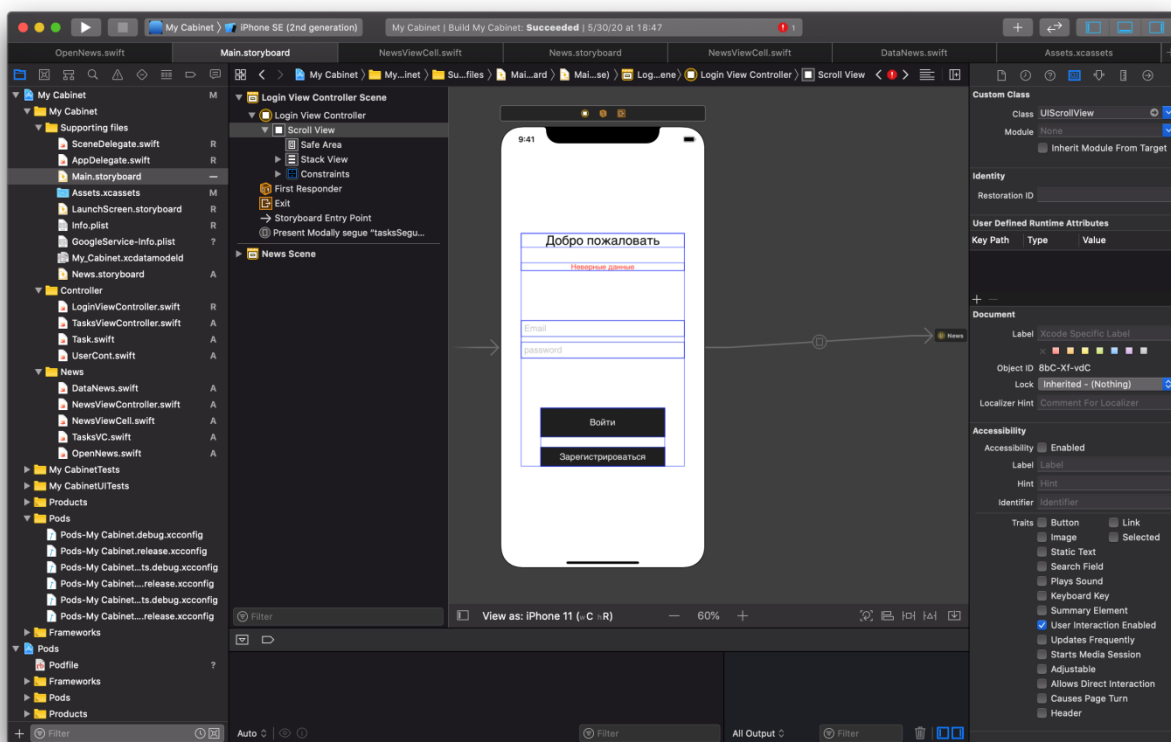


Рисунок 3.9 - Реализация интерфейса авторизации

При создании связи между элементами и контроллером у нас будет выходить диалоговое окно, выбрав необходимые параметры в нем у нас будет отображаться соответствующий код в самом контроллере, который в будущем можно будет изменить, если он не будет соответствовать новым требованиям. Все созданные элементы будут иметь приставку @IB, что можно разобрать

как ссылку на элемент, а аббревиатура IBрасшифровывается как Interface-Builder. Важным моментом является название элементов, так как они не должны повторяться между собой. Такая созданная связь помогает взаимодействовать с элементами в коде непосредственно вызывая их по мере написания только по их имени. На скриншоте экрана ниже (рисунок 3.10) можно увидеть контроллер окна авторизации и его часть кода.

Сама регистрация имеет упрощенный вид, необходимо заполнить выше описанные поля и нажать на кнопку регистрации. Стоит отметить, что пользователь не может многократно зарегистрировать одного и того же пользователя в базу данных. Сам фреймворк Firebase, который используется для регистрации имеет прописанные условия от разработчика, что можно создать пользователя только при условии того, что будет использоваться электронный адрес почты (обязательное использование символа @ в поле логин), а пароль должен содержать в себе не менее 6 символов, иначе регистрация просто не осуществится.

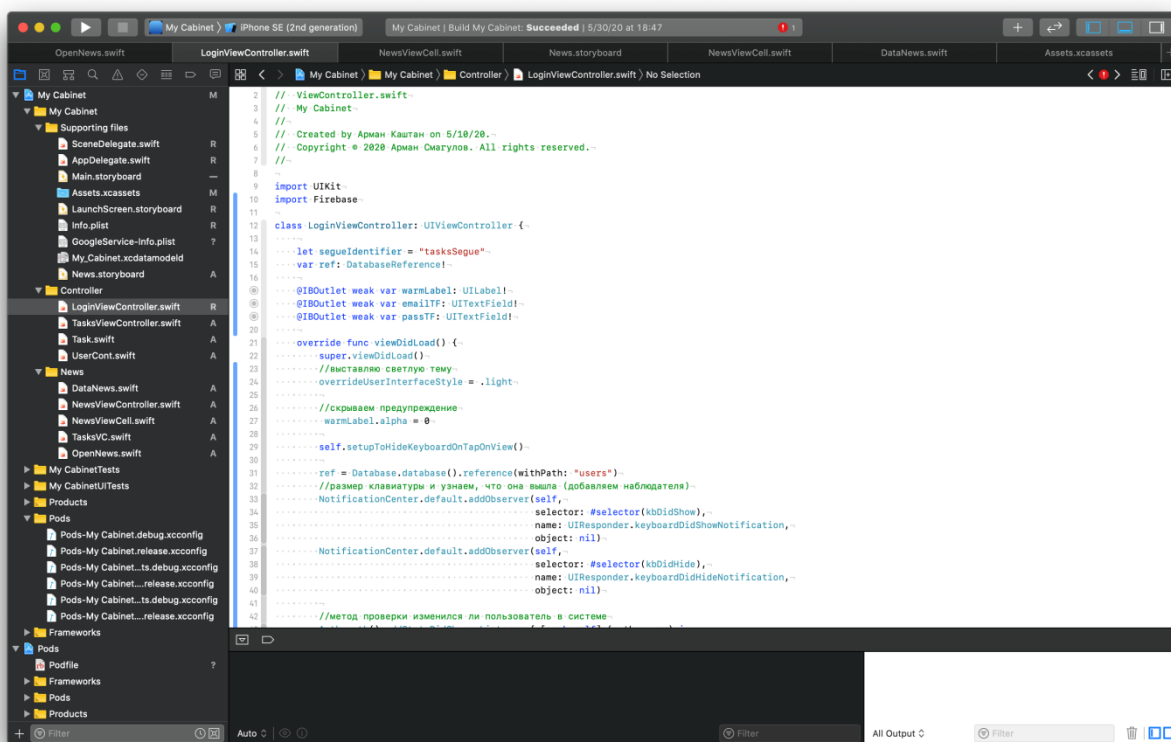


Рисунок 3.10 - Контроллер окна авторизации/регистрации

После регистрации или если у пользователя уже есть аккаунт, ему откроется доступ к главному экрану приложения, также если пользователь будет закрывать приложение, то система его не забудет и при следующем запуске он сразу перейти к основному окну без повторной авторизации. Для этого будет необходимо поставить наблюдателя, который будет следить за тем, что выходил ли пользователь из системы или нет. Firebaseв своей библиотеке предо-

ставляет метод проверки условия выхода. Для большинства пользователей будет достаточно один раз войти и не выходить, но выход из системы должен быть, если будет необходимость авторизоваться другому пользователю через чужое устройство.

Рассматривая отрывок кода авторизации ниже можно увидеть, что синтаксис языка Swift очень похож на Object-Cи все языки семейства языка C. Сам язык Swift еще молодой и построен на семействе Cс заимствованиями готовых решений других языков. Язык не требует особенно строгой пунктуации, а человечность языка (легкость восприятия кода человеком) у него на высоком уровне [19].

Тут приведен код, изображенный на скриншоте экрана (рисунок 3.10) выше. Он совмещает в себе реализацию кода двух библиотек, UIKitи Firebase.

Тут наглядно видно, что созданные элементы на Storyboard, получившие свои имена и связь соединили с контроллером и имеют функциональный код. Цветовое же разделение текста помогает быстро и четко ориентироваться среди большого количества кода.

```
@IBAction func registerTapped(_ sender: UIButton) {
    guard let email = emailTF.text, let password = passTF.text, email != "", password != ""
    else { displayWL(withText: "Пустое поле")
    return
    }
    Auth.auth().createUser(withEmail: email, password: password, completion: {
    [weak self] (authResult, error) in

    guard error == nil, let user = authResult?.user else {

    print(error!.localizedDescription)
    return
    }

    //      let userRef = self?.ref.child(user.uid)
    //      userRef?.setValue(user.email, forKey: "email")
    let userRef = self?.ref.child(user.uid)
        userRef?.setValue(["email": user.email])
    })
    }
}
```

Выше описанный код имеет разные типы данных, подключенные элементы, условия сравнения и назначение новых значений.

IBAction говорит нам о том, что весь ниже описанный код будет работать с кнопкой, под названием registerTapped при ее нажатии (можно со-

здать определенные условия при каком типе взаимодействия). Структура `guard` является по своей функциональности очень похожей логической структурой `if-else`, которая позволяет проверять условия и принимать решение. В целом этот код выполняет следующие важные задачи [25]:

Производит проверку верности ввода данных в поля ввода для пользователей, а именно на то, пустые они или частично заполненные. Структура `Guard` проверяет условия, если поля с названием `emailTF` и `passTF` будут пустыми, то необходимо предупредить пользователя об этом. Структура при пустых полях передает значение системному ярлыку дать текст “Пустое поле” и отобразить его на время (за его отображение и сокрытие отвечает написанная функция `displayWL`). Если же все условия соблюдаются, то будет уже выполняться сам алгоритм процесса регистрации, где работает код фреймворка `Firebase`. Для этого необходимо обратиться к функции `Auth.auth().createUser`. Особенность построения кода на каждом фреймворке может отличаться и чтобы узнать какие команды у него есть, необходимо обращаться к документации.

По выполнении процесса регистрации создается `RefID`, который создает уникальный `ID` для каждого зарегистрированного пользователя и через него можно будет понимать от кого именно приходит запрос или кому именно необходимо предоставить определенные услуги.

Так как `Firebase` по своей сути не является полноценным собственным сервером для разработчика, имеется представление той информации, которую посчитал `Google` стоит демонстрировать им. По этому необходимо можно произвести проверку от конечного пользователя с помощью инструментов среды разработки `Xcode`. С помощью мониторинга трафика был получена информация за временной интервал в размере одной минуты, о процессах авторизации, аутентификации, создания трех записей и выхода из системы.

Ниже будут представлены скриншоты мониторинга трафика для конечного устройства с текущего местоположения (рисунок 3.11) и для города Лондон (рисунок 3.12).

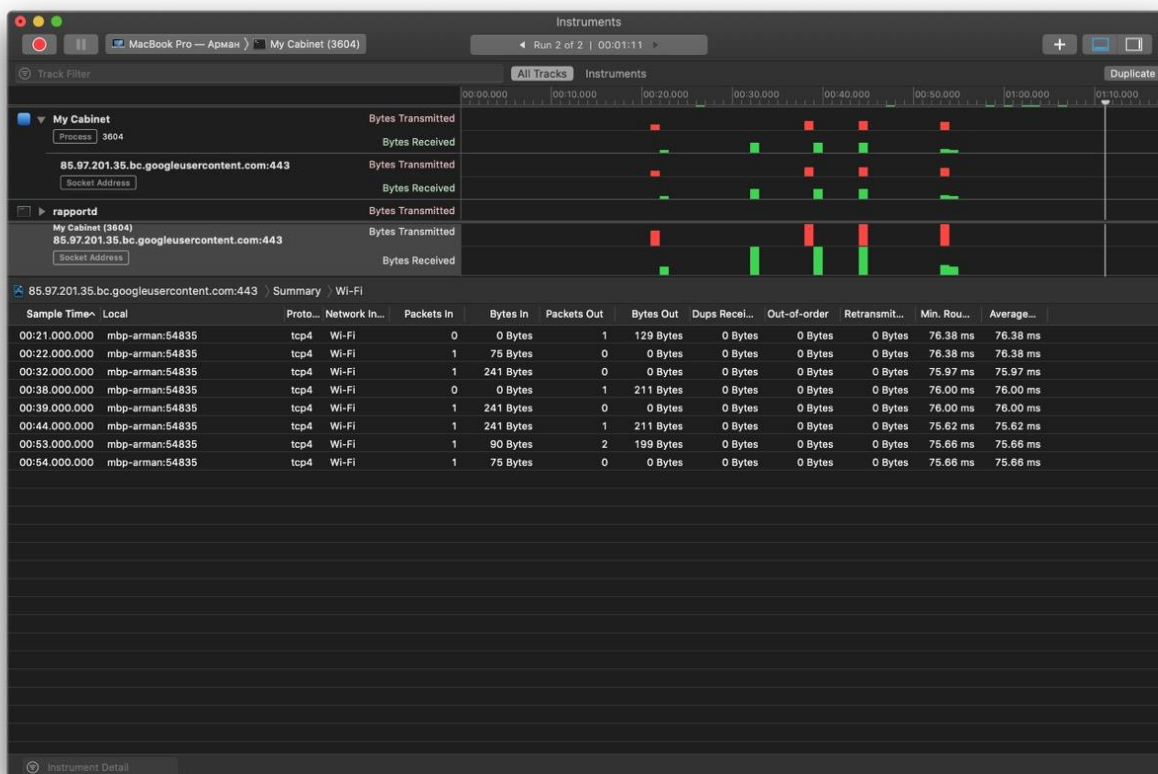


Рисунок 3.11 - Проверка связи эмуляции приложения с БД (Алматы)

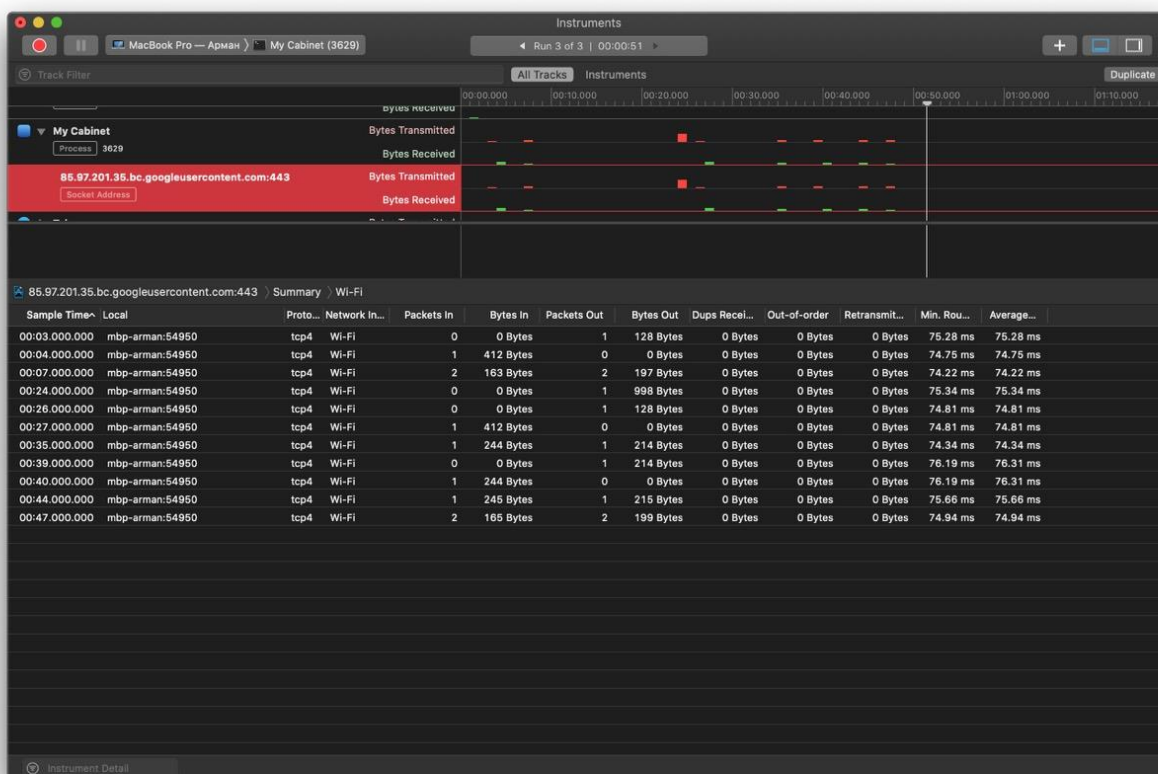


Рисунок 3.12 - Проверка связи эмуляции приложения с БД (Лондон)

Данные числа не являются абсолютными и нагрузка может измениться с введением новых функций или же появлением большого количества пользователей, а также нагрузка переменна, так как в реальности количество запросов не одинаково.

Также нагрузкой может послужить регистрация пользователей с подтверждением электронного адреса или отправкой на сотовый телефон специального кода подтверждения пользователя. Все это трафик, который стоит учитывать при проектировании приложения, так как расширения возможностей может стоить больших денег на содержание проекта.

На текущий момент приложение имеет следующие показатели аппаратной нагрузки:

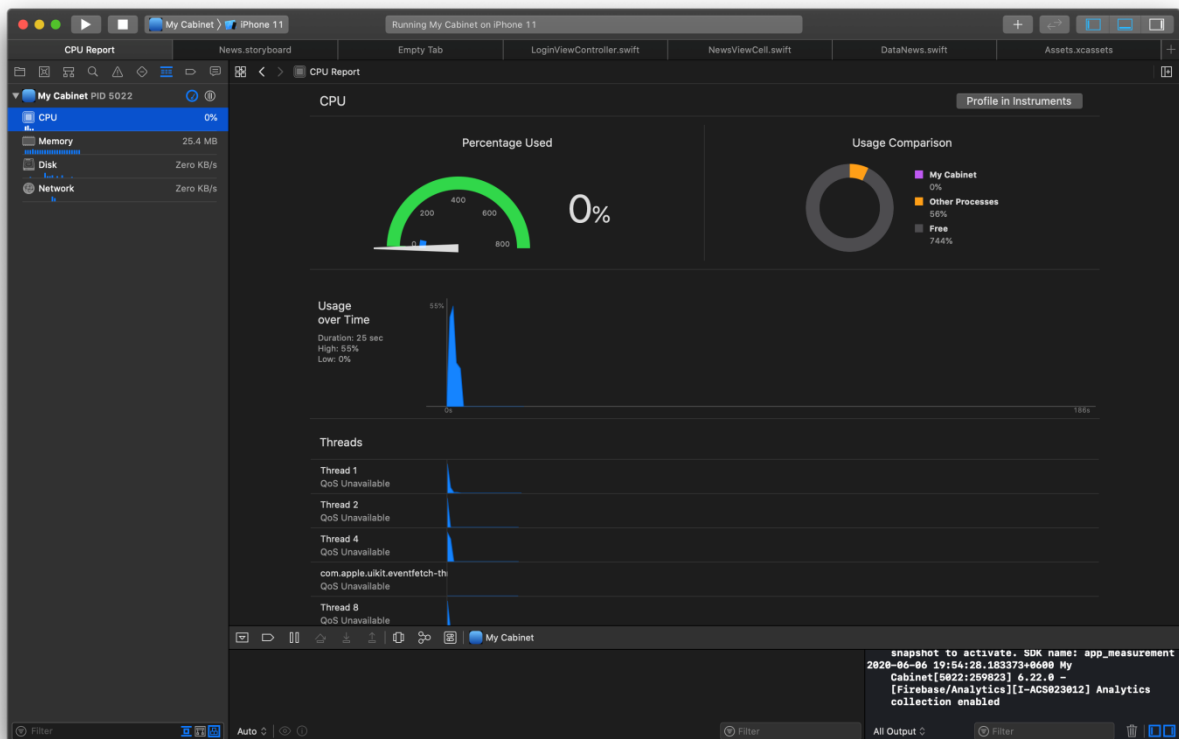


Рисунок 3.12 - Загрузка центрального процессора

Центральный процессор является сердцем электронного устройства, оно отвечает за скорость и качество обработки кода нашей программы. Как можно видеть из графика выше (рисунок 3.12) нагрузка процессора происходит только в момент запуска самого приложения, в дальнейшем при процессе эксплуатации не происходит повышенной нагрузки, приложение не имеет закликивания или сложной анимации и приложение на текущий момент может подойти для любого устройства iOS по этим параметрам. На основе этого графика можно иметь условное представление о работе приложения. Если возникнет

ситуация зависания приложения, то скорей всего это будет связано с плохим качеством соединения к сети Интернет.

Принцип работы приложения заключается в начальной предзагрузке приложения и в последующем выгрузке его из памяти, а обработка может совершаться при использовании анимации или загрузки новых данных из БД, где процессору будет необходимо обрабатывать уже новые данные.

Второй тест идет на использование оперативной памяти. Она отвечает за хранение выполняемого кода (приложения), а также входные/выходные и промежуточные данные, которые в свою очередь обрабатываются процессором. Приложение имеет нагрузку на оперативную память только в процессе работы с приложением и имеет небольшое потребление памяти при его закрытом состоянии. Загрузку оперативной памяти приложением можно посмотреть на графике ниже (рисунок 3.13).

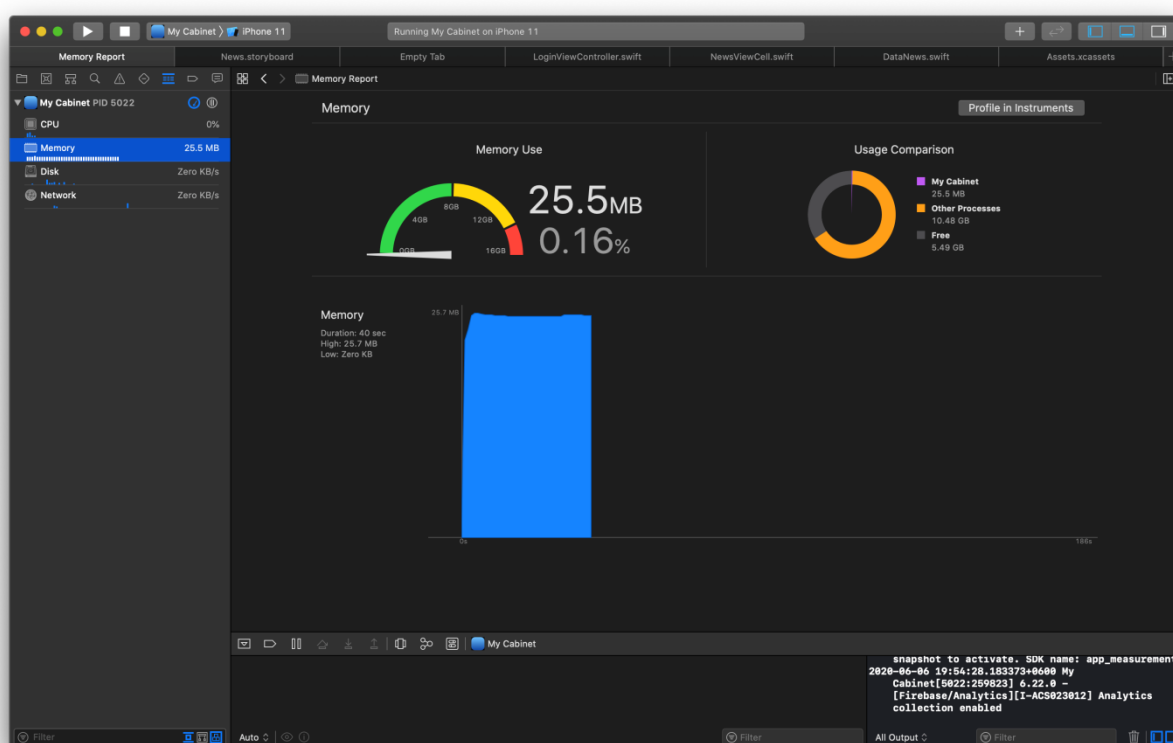


Рисунок 3.13 - Загрузка оперативной памяти

Данный график имеет крайне низкую нагрузку ввиду небольшого функционала и отсутствие большого количества анимаций. Высокую нагрузку на оперативную память можно увидеть в приложениях, где наблюдается большая обработка данных (фото/видео редакторы как яркий пример), так как приложение работает с демонстрацией текстовой информации, то объем занимаемой оперативной памяти остается довольно на низком уровне. Так как проверка происходила с использованием эмулятора, то мы можем видеть большой объем оперативной памяти для мобильного устройства, но на самом деле

это память компьютера. В зависимости от мобильного устройства, имеется 2 ГБ оперативной памяти, которые не стоит использовать до предела, так как смартфон параллельно поддерживает работу и других мобильных приложений. На практике реализации других продуктов можно сделать, что нагрузка на оперативную память бывает переменной и если наблюдается её повышение, то пользователя ограничивают на какое-то время от эксплуатации устройства, для уменьшения вероятности появления ошибки, повреждения данных или кратковременного вывода самого устройства из работоспособности.

Третий тест основан на использовании памяти устройства во время эксплуатации. Apple использует в своих устройствах твердотельные накопители (SSD). SSD - это технология постоянного хранения данных. Доступ к данным из SSD намного медленнее, чем из оперативной памяти и это стоит учитывать. Приложения, которые активно используют дисковое хранилище для доступа к своим данным, рискуют снизить производительность и время отклика приложения [13].

Чередование записи и чтения с диска увеличивает задержку операций чтения. Запись на SSD является более медленной операцией, чем чтение, и диск должен завершить операции записи, прежде чем он сможет выполнить невыполненные запросы на чтение данных. Накопитель ставит в очередь запросы на чтение данных, ожидающих записи, что увеличивает время, в течение которого приложение ожидает данные. iOS блокирует запрашивающий поток, пока он ожидает данных.

iOS может записывать в одну и ту же область SSD ограниченное количество раз до того, как диск изнашивается. Чем больше записей делает приложение, тем быстрее оно изнашивает ячейку памяти и в целом память устройства.

Данный проект имеет крайне низкое потребление памяти и не имеет свойств перезаписи одной и той же информации, тем самым засорения памяти одной и той же информацией.

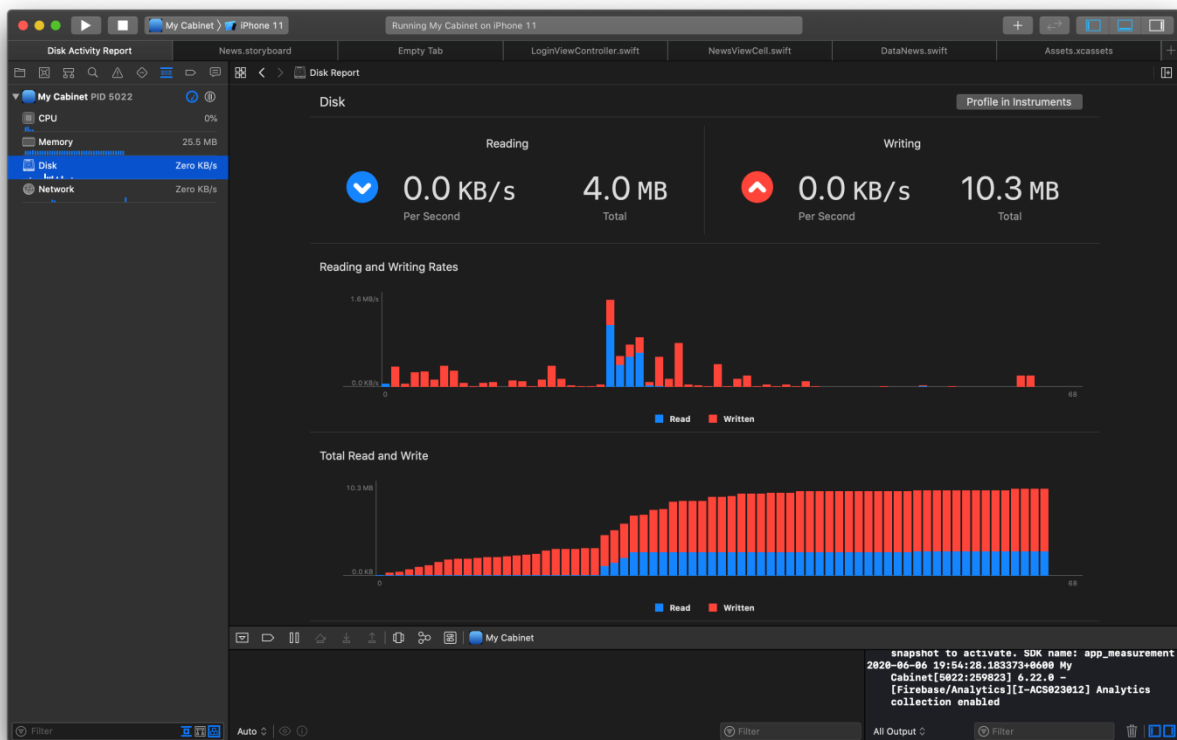


Рисунок 3.14 - Загрузка памяти

При разработке приложений рекомендуется помнить, что никогда нельзя предугадать, с какими сетевыми условиями будут сталкиваться пользователи при использовании приложения. Некоторые пользователи могут использовать Wi-Fi, некоторые 4G - LTE, но, скорее всего, многие из пользователей находятся в условиях, подобных 3G или сети EDGE, которая может быть намного медленнее, чем Wi-Fi или 4G. В этих случаях важно протестировать приложение, чтобы убедиться, что приложение сможет обеспечить работу в неблагоприятных условиях сети с некоторыми допущениями. Плохая практика использования приложения выражается в слишком большом количестве повторных попыток соединения или малое время ожидания клиента, что в некоторых случаях может привести к тому, что пользователи будут отталкиваться или вовсе отказываться от приложения. Одним из способов реальной проверки является использование смартфона как модема. При наличии плохого сигнала связи или отдалении от компьютера с эмуляции можно добиться плохого соединения [14].

Приложение на практике оказалось устойчивым, связано это с малым объемом реальных пакетов данных, но если пользователю потребуется получить медиа контент, то с этим могут возникнуть проблемы. Связаны они в первую очередь с большим весом самих исходных файлов, которые необходимо пользователю загрузить на устройство. Тут помогает загрузка данных приложения в момент, когда пользователь имеет доступ к высокоскоростному

интернету и в последующем при необходимости их выгружать из памяти, а при изменении данных, дополнять, а не производить полную замену.

На графике снизу (рисунок 3.15) видно, что запросы/ответы не имеют постоянной формы, они дискретны ввиду того, что необходимость в постоянной загрузке данных отсутствует.

Приложение отправляет запросы БД на проверку наличия изменений в ее структуре и если их нет, то нет необходимости прогружать данные из нее, так как актуальность текущих данных на устройстве сохраняется.

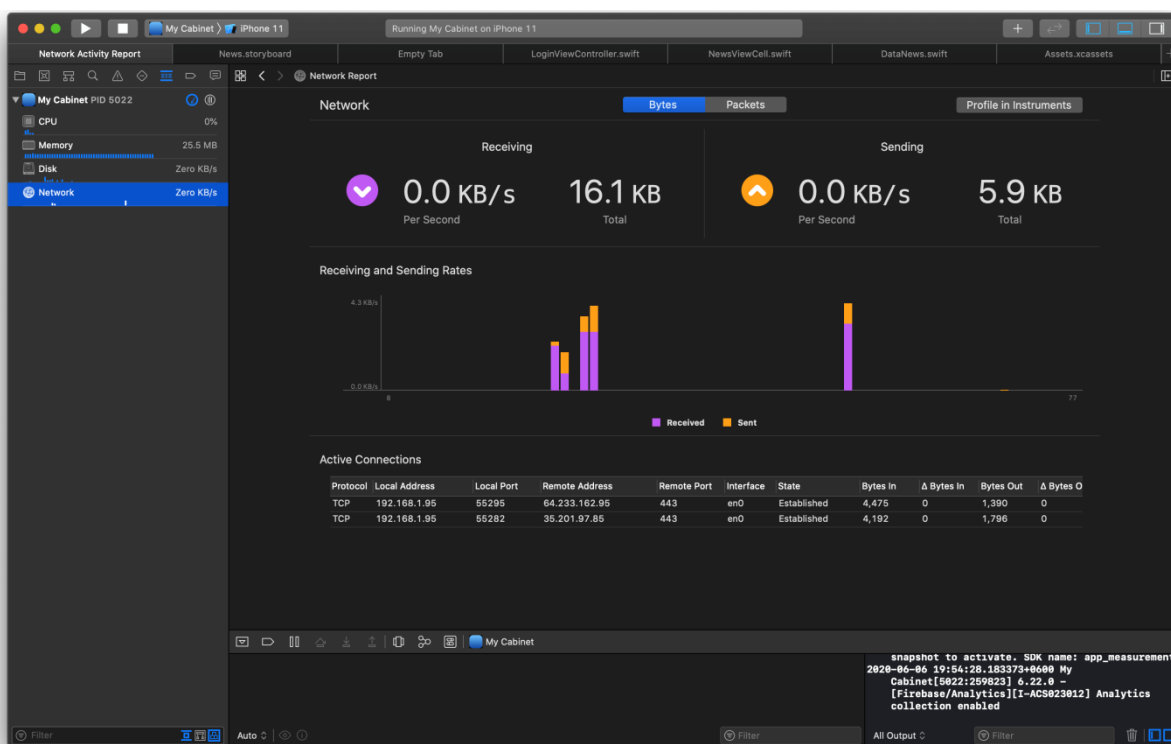
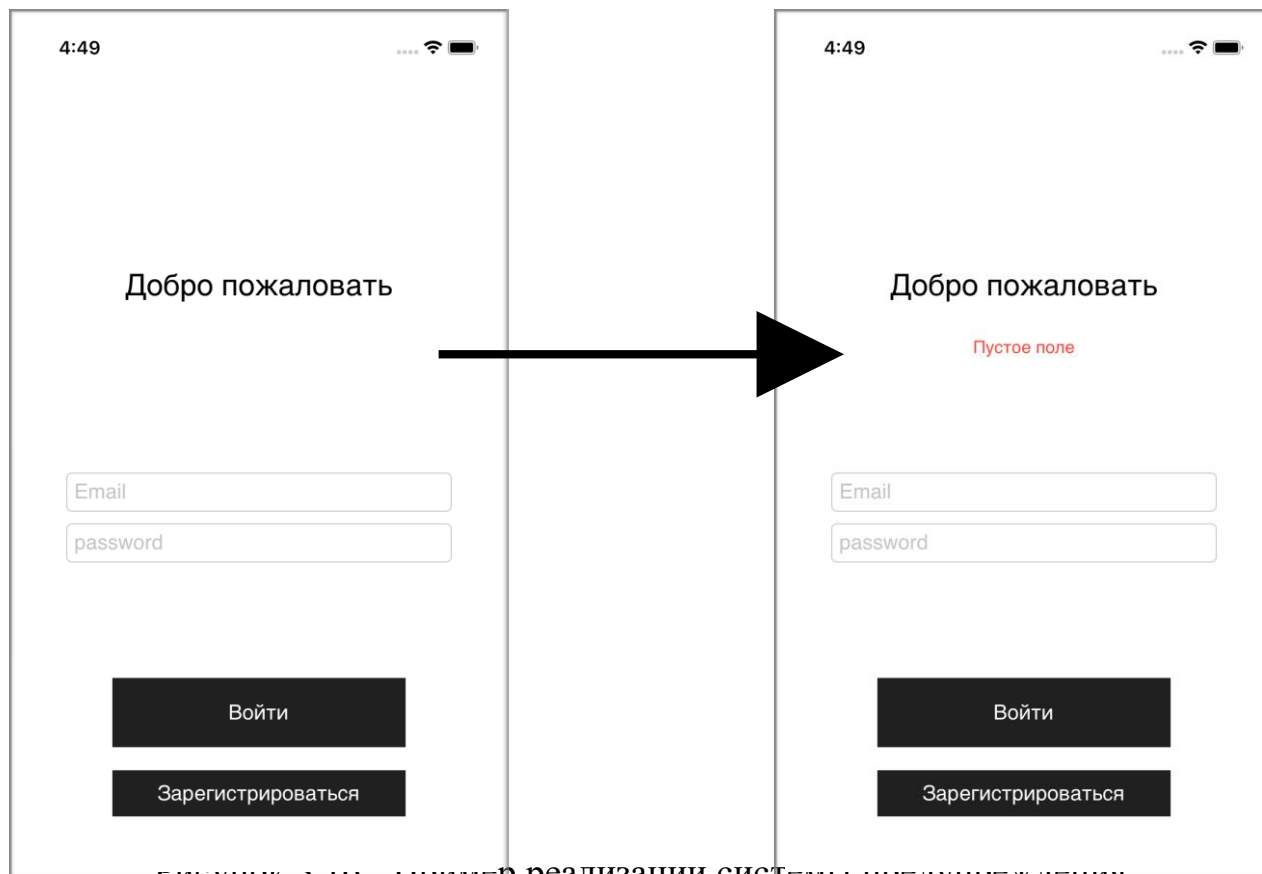


Рисунок 3.15 - Сетевые функции приложения

3.3 Пример окон приложения

Пользователь увидит следующий экран (рисунок 3.16), который скрывает описанный код в работе. В процессе эмуляции приложение можно наблюдать исполнение системы проверки условия ввода (рисунок 3.17) данных и назначение новых параметров для элементов интерфейса.



рисунк 3.18 – пример реализации системы предупреждения

Для удобства использования приложения пользователями в будущем можно будет добавить регистрацию с помощью социальных сетей с подтверждением внутри них, мобильного телефона или же добавить подтверждение электронной почты путем отправки письма.

Если данные введены верно, то система по нажатию кнопки войти совершает переход в систему (рисунк 3.18). Данный принцип перехода не подвязывается к конкретному окну, что позволяет при изменении структуры приложения в будущем совершать переход уже в совершенно другое назначенное окно без потери функционала авторизации.

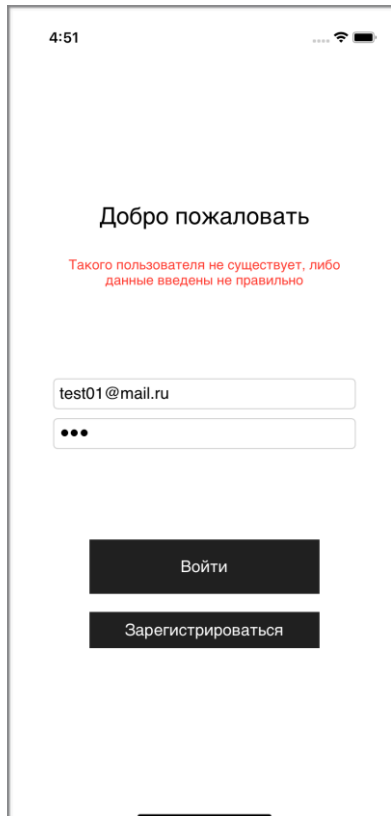


Рисунок 3.17 - Система предупреждения на верность данных

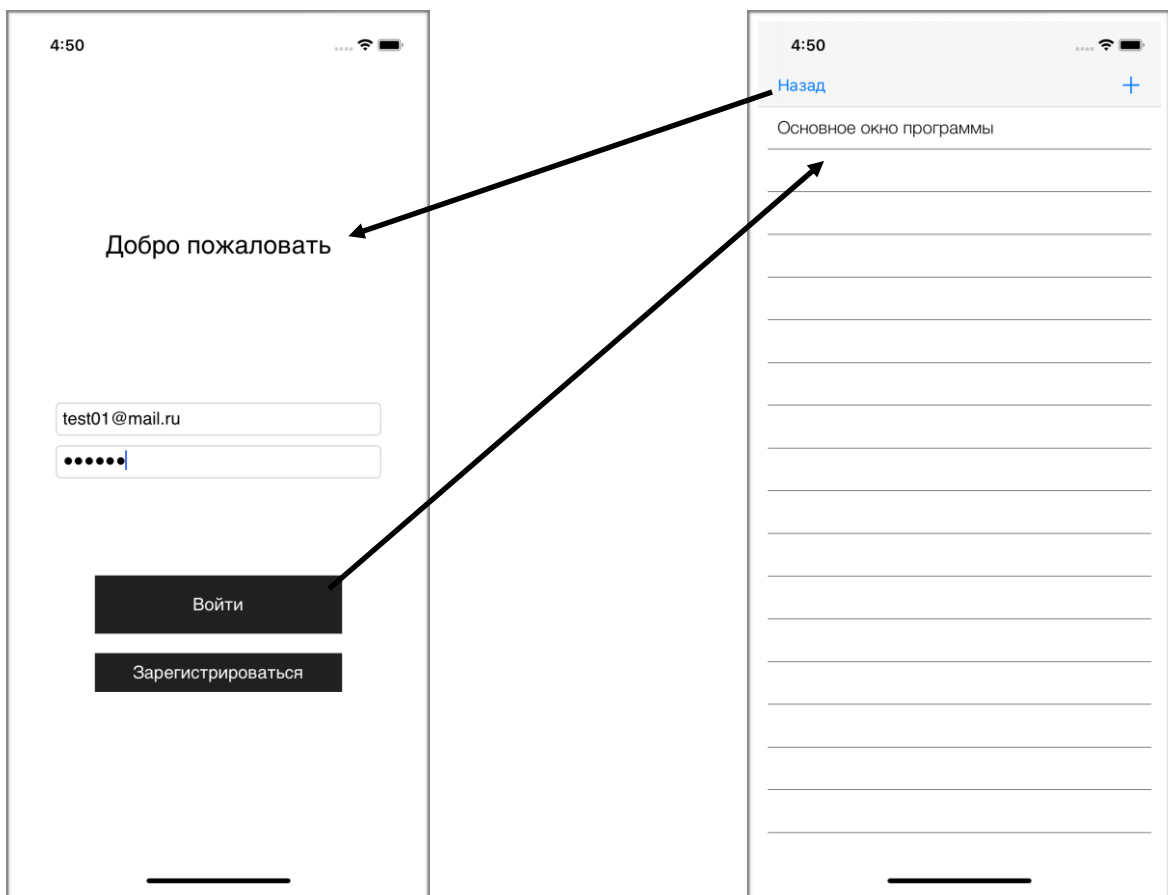


Рисунок 3.18 - Пример реализации перехода между окнами

Как было ранее описано приложение исполняет написанные задачи и ошибок не возникает, перегрузки аппаратной части приложения не возникает и нет заикливания приложения. Сетевой трафик не обладает большими пакетами, которые могли бы повышать требования к типу сетевого подключения и потребление сетевого трафика крайне низко.

Стоит отметить, что параметры не устойчивы к каким-либо изменениям в приложении, стоит всегда следить за нагрузкой процессора, так как некоторые логические операции потребуются изменить при повышении количества обрабатываемой информации. Не стоит использовать сильные ссылки в процессе разработки, так как они могут вызвать возникновение бесконечных циклов. Также при разработке дизайна приложения стараться использовать не сильно большое количество элементов, так как можно перегрузить контроллер или нельзя будет отображать все элементы интерфейса правильно для устройств с меньшим размером экрана.

При разработке важно придерживаться одного из типов паттерна написания приложения. В данном проекте придерживались разработки по паттерну MVC, который рекомендует компания Apple. Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения.

Заключение

В течении 15 лет у нас формируется новый технический уклад в мышлении и он не перестает формироваться. Он приходит к нам и связан с инженерией и мобильных технологии, которые больше развиваются к удаленной работе.

Проанализированные современные подходы к интеграции мобильных приложений с корпоративными информационными системами (КИС). Разработана модель интеграции мобильного приложения к БД при помощи API технологии и подключена система регистрации и хранения данных пользователей. Написан код мобильного приложения и дизайн интерфейса. Проведены проверки работоспособности приложения.

Мобильная разработка будет только повышать интерес аудитории к себе и в ближайшем будущем возможно ожидание большого спроса на их написание. Такая технология отлично работает с последними тенденциями и направлениями развития технологий, так как облачные технологии, интернет-вещей и систем умного дома. Можно сказать, что смартфоны становятся ключом идентификации пользователя уже в физическом мире, где без него уже в будущем будет крайне сложно обходиться.

Уже сейчас можно наблюдать большой интерес к мобильной разработке во многих крупных компаниях, которые предлагают рынку свои услуги для уменьшения количества персонала по работе с клиентами, а сотрудники которые остаются уже имеют более конкретизированный вопрос, так как приложения помогают составить категорию запроса самим клиентом.

Многие государственные структуры в самых различных странах начинают применять мобильные приложения для своих продуктов, например армия Израиля создала собственное приложение для солдат, где они могут видеть сколько им осталось служить, когда они могут отпускные и сколько им полагается выплата.

В Казахстане же с глобальной программой цифровизации это служит хорошим инструментом для фактического представления результатов для населения. Последняя практика применения ботов 42500 в социальном мессенджере телеграмм, открытие сайтов дало возможность разгрузить сотрудников по работе с населением в период пандемии. Система мобильного приложения в больнице помогает пациентам ориентироваться в здании здравоохранения и быстро оформлять необходимые документы для процедур, дистанционно записываться на прием к врачу и получать результаты анализов.

Развитие же отрасли показывает тенденцию за использование простых мобильных устройств, которые опять же будут обладать программным продуктом для совершения различных вычислительных операциях уже на серьезных серверных мощностях. Такой подход можно увидеть в компании Sony, которые уже несколько лет рекламируют возможности стриминга игрового процесса на мобильные устройства и играть в игры на слабых компьютерах,

которые нужны только для получения, отображения и отправки команд на сервера.

Сами же мобильные устройства скорее всего будут только увеличивать вычислительную мощность для повышения быстродействия системы, поддержки нового протокола 5G мобильного интернета и повышать автономность системы.

В целом индустрия мобильной разработки еще молода и за свою небольшую практику не имела сильного падения рынка спроса, что может только привлекать новых инвесторов для финансирования разнообразных проектов. Ее ограниченность заключается только в представлении ее реализации.

Список литературы

1. Статья 8. Государственные гарантии в области образования. URL: https://online.zakon.kz/document/?doc_id=39633866#pos=230;-54 (дата обращения: 09.01.2019).
2. Методы и средства интеграции информационных систем в рамках единого информационного пространства. URL: <http://lab18.ipu.ru/projects/conf2012/1/7.htm> (дата обращения: 15.01.2019).
3. Дьяченко Д.Г. Унификация системного программного обеспечения на основе технологии виртуализации / Д.Г. Дьяченко // Известия ТулГУ. Технические науки. 2012. №5. [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/unifikatsiya-sistemnogo-programmnogo-obespecheniya-na-osnove-tehnologii-virtualizatsii> (дата обращения 18.01.2019).
4. Интеграция бизнес-процессов и сервис-ориентированная архитектура // Решения и технологии [Электронный ресурс]. – Режим доступа: <https://www.osp.ru/data/134/081/1237/soa2.pdf> (дата обращения 23.01.2019).
5. Интеграция бизнес-процессов и сервис-ориентированная архитектура // Решения и технологии [Электронный ресурс]. – URL: <https://www.osp.ru/data/134/081/1237/soa2.pdf> (дата обращения 27.01.2019).
6. Интеграция бизнес-процессов и сервис-ориентированная архитектура // Решения и технологии [Электронный ресурс]. – Режим доступа: <https://www.osp.ru/data/134/081/1237/soa2.pdf> (дата обращения 30.01.2019).
7. История мобильного интернета // Хабр [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/yota/blog/352450/> (дата обращения 14.02.2019).
8. Облачные интегрированные среды веб-разработки. URL: <https://moluch.ru/archive/113/29191/> (дата обращения: 18.02.2019).
9. Cloud computing URL: https://en.wikipedia.org/wiki/Cloud_computing (дата обращения: 24.02.2019).
10. Смартфоны (мировой рынок). URL: [http://www.tadviser.ru/index.php/Статья:Смартфоны_\(мировой_рынок\)](http://www.tadviser.ru/index.php/Статья:Смартфоны_(мировой_рынок)) (дата обращения: 03.03. 2019).
11. Smartphone Market Share. URL: <https://www.idc.com/promo/smartphone-market-share/os> (дата обращения: 15.03.2019).
12. Consumer Spending In Mobile Apps Grew 17% in 2019 to Exceed \$83 Billion Globally. URL: <https://sensortower.com/blog/app-revenue-and-downloads-2019> (дата обращения: 25.03.2019).
13. What Are the Different Types of Mobile Apps? And How Do You Choose? URL: <https://clevertap.com/blog/types-of-mobile-apps/> (дата обращения: 04.04.2019).
14. Native apps. Mobile APIs. Real-time analytics. One Platform. URL: <https://www.appcelerator.com/mobile-app-development-products/> (дата обращения: 14.04.2019).

15. 2Дигровой движок URL:<https://ru.coronalabs.com/product/> (дата обращения: дд.мм.2019).
- 16.iOS URL:<https://en.wikipedia.org/wiki/IOS>(дата обращения: 19.04.2019).
- 17.Xcode 11 URL:<https://developer.apple.com/xcode/> (дата обращения: 25.04.2019).
- 18.What is BaaS? URL:
<https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/>(дата обращения: 02.05.2019).
- 19.Add Firebase to your iOS project. URL:
<https://firebase.google.com/docs/ios/setup>(дата обращения: дд.мм.2019).
- 20.Available pods. URL:<https://firebase.google.com/docs/ios/setup#available-pods>(дата обращения: 09.05.2019).
- 21.iOS UI Kits and iOS GUI Wireframes for Prototypes free resources for Sketch URL: <https://www.sketchappsources.com/tag/ios.html> (дата обращения: 13.05.2019).
- 22.Swift. URL: <https://developer.apple.com/swift/> (дата обращения: 19.05.2019).
23. Усов В. Swift. Основы разработки приложений под iOS и macOS. 4-е изд., доп. и перераб. — СПб.: Питер, 2018. — 448 с.
- 24.Reducing Disk Writes. URL:
https://developer.apple.com/documentation/xcode/improving_your_app_s_performance/reducing_disk_writes (дата обращения: 21.05.2019).
- 25.How to test with the Network Link Conditioner in Xcode 11. URL:
<https://www.agnosticdev.com/content/how-test-network-link-conditioner-xcode-11> (дата обращения: 25.05.2020).