МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для получателей жилищно-коммунальных услуг»

Обучающийся Калинин Д.Д.
(Инициалы Фамилия) (личная подпись)

Руководитель Крайнова О.А.
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема бакалаврской работы: «Разработка мобильного приложения для получателей жилищно-коммунальных услуг».

Объектом данной бакалаврской работы является изучение приемов создания мобильного приложения, облегчающего передачу показаний ЖКХ.

Предметом исследования является мобильное приложение, которое служит инструментом для получения физическими лицами жилищно-коммунальных услуг.

Актуальность выбранной темы заключается в предложении универсального программного решения, позволяющего пользователям эффективно передавать счета за коммунальные услуги.

Цель бакалаврской работы - разработка мобильного приложения для получателей жилищно-коммунальных услуг.

В первой главе рассматривается предметная область исследования, формулируются требования к мобильному приложению, и проводится анализ аналогов.

Во второй главе разрабатывается архитектура мобильного приложения, выбор инструментов реализации мобильного приложения, описываются сценарии использования, проводятся функциональное и логическое моделирование мобильного приложения.

Третья глава содержит программную реализацию требуемых функций и тестирование мобильного приложения

В заключении подводятся итоги выполнения выпускной квалификационной работы.

Бакалаврской работа состоит из введения, трёх глав, заключения и списка использованной литературы.

Бакалаврская работа состоит из 45 страниц, 20 рисунков, 2 таблиц, 25 источников и 3 листингов.

Abstract

The title of the bachelor’s thesis is “Development of a mobile application for recipients of housing and communal services”.

The object of this bachelor's thesis is to study the techniques of creating a mobile application that facilitates the submission of utility meter readings.

The subject of the research is a mobile application that serves as a tool for individuals to access housing and utilities services.

The relevance of the chosen topic lies in offering a universal software solution that enables users to efficiently submit bills for utility services.

The aim of the bachelor's thesis is to develop a mobile application for recipients of housing and utilities services.

The first chapter examines the subject area of the research, formulates requirements for the mobile application, and conducts an analysis of existing solutions.

The second chapter focuses on the architecture of the mobile application, the selection of implementation tools, the description of usage scenarios, and the functional and logical modeling of the mobile application.

The third chapter involves the programming implementation of the required features and the testing of the mobile application.

The conclusion summarizes the results of the bachelor's thesis.

The bachelor's thesis consists of an introduction, three chapters, a conclusion, and a list of references.

The volume of the bachelor's thesis is 45 pages, it also contains 20 figures, 2 tables, a list of 25 references and 3 listing.

Оглавление

Введение

Каждый владелец квартиры или дома обязан своевременно оплачивать выставленные счета, для избегания пений и штрафов. Коммунальные предприятия взимают плату за такие ресурсы, как газ, электричество и воду, на основе общих или индивидуальных счетчиков. Домовладельцы должны предоставить компании показания счетчиков для точного выставления счетов. Для этих целей используются: формы, телефон, Web-сайт или электронную почту. Некоторые компании предлагают online-отслеживание использования и планирование платежей.

Актуальность выбранной темы заключается в предложении универсального программного решения, позволяющего пользователям эффективно передавать счета за коммунальные услуги. Используя это приложение, пользователи могут сократить время, затрачиваемое на запись показаний, а также упростить процесс передачи в управляющую компанию.

Объектом данной бакалаврской работы является изучение приемов создания мобильного приложения, облегчающего передачу показаний ЖКХ.

Предметом исследования является мобильное приложение, которое может быть использовано для подачи данных счетчиков получателей жилищно-коммунальных услуг.

Цель бакалаврской работы – разработка мобильного приложения для получателей жилищно-коммунальных услуг.

Для достижения заданной цели работы необходимо осуществить следующие задачи:

- провести сравнительный анализ существующих аналогов и сформировать требования к мобильному приложению;
- проанализировать и выбрать технологии разработки мобильного приложения для получателей жилищно-коммунальных услуг;
- разработать мобильное приложение для получателей жилищно-коммунальных услуг;

− протестировать мобильное приложение.

Методами исследования являются технологии разработки мобильных приложений, а также методы и технологии Web-дизайна.

Практическое значение бакалаврской работы заключается в разработке мобильного приложения, которое может быть использовано для удобной передачи данных счетчиков жилищно-коммунальных услуг.

Бакалаврская работа состоит из введения, трех глав, заключения, списка используемой литературы.

Глава 1 Постановка задачи на разработку мобильного приложения

1.1 Анализ средств передачи показаний счетчиков

Потребительские счетчики приносят пользу коммунальным предприятиям, упрощая выставление счетов, сокращая потребность в персонале и полагаясь на потребителей в отношении показаний счетчиков. Тем не менее, периодический контроль необходим из-за возможности неточных показаний или неисправных счетчиков.

Оказание бытовых приборов учета, как правило, передаются с помощью форм на сайтах коммунальных и управляющих компаний. Потребители записывают показания в определенные дни каждого месяца и вводят информацию в форму. Затем коммунальная организация выставляет счет на оплату. Однако, поскольку у каждой организации есть свой Web-сайт и формы, этот метод не так удобен для пользователя, как мог бы быть с помощью мобильного приложения [2].

Задача состоит в том, чтобы создать мобильное приложение, способное передавать показания бытовых счетчиков, таких как счетчики электроэнергии, воды. Приложение должно быть удобным для пользователя и простым в навигации, предоставляя данные о потреблении энергии и затратах в режиме реального времени. Он также должен быть безопасным и способным обрабатывать большие объемы данных с различных счетчиков.

Мобильное приложение – это программное приложение, предназначенное для работы на мобильном устройстве, таком как смартфон или планшет, предоставляющее пользователю определенные функции или услуги.

Преимущества мобильного приложения:

– повышение вовлеченности и лояльности клиентов;

– повышенная доступность и удобство;

– улучшенный пользовательский интерфейс и персонализация;

– более высокая эффективность и производительность;

– улучшение сбора и анализа данных;

– конкурентное преимущество на рынке.

На рисунке 1 представлена функциональная модель для подачи показаний счетчиков.

Рисунок 1 – Функциональная модель

Представим диаграмму «КАК ЕСТЬ» (AS-IS) для авторизации пользователя.

На рисунке 2 представлена диаграммы «КАК ЕСТЬ» (AS-IS).

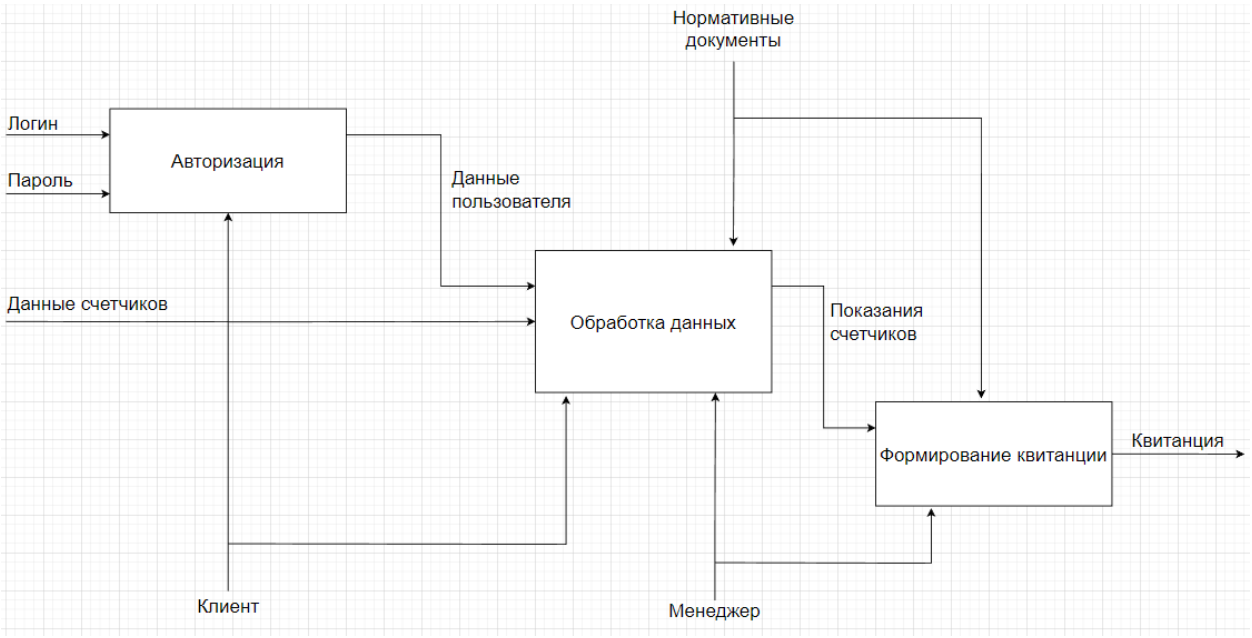

Рисунок 2 – Диаграмма «КАК ЕСТЬ» для авторизации пользователя

Декомпозиция контекстной диаграммы включает в себя следующие процессы:

1. Процесс «Авторизация» включает в себя ввод данных, таких как имя пользователя и пароль.

2. Процесс «Обработка данных» подразумевает уточнение, систематизацию и упорядочение информации о клиенте.

3. Процесс «Формирование квитанции» подразумевает, составление и выдачу готовой квитанции.

Таким образом, был отражен стандартный процесс формирования квитанции в мобильном приложение УК.

После создания диаграммы «КАК ЕСТЬ», видно, что если это новый пользователь, ему необходимо изначально зарегистрироваться. Этого нет в данной диаграмме.

1.2 Формирование требований к мобильному приложению

Мы воспользуемся классификацией FURPS2+ для определения требований к нашему мобильному приложению. Эта классификация широко используется и доказала свою эффективность при разработке приложений и систем [1], [14].

Согласно классификации FURPS+, функциональность – это основа любого приложения или системы, которая включает в себя ее назначение и функции. Диаграммы вариантов использования строятся на основе функциональных требований. На рисунке 2 показана классификация требований FURPS+.

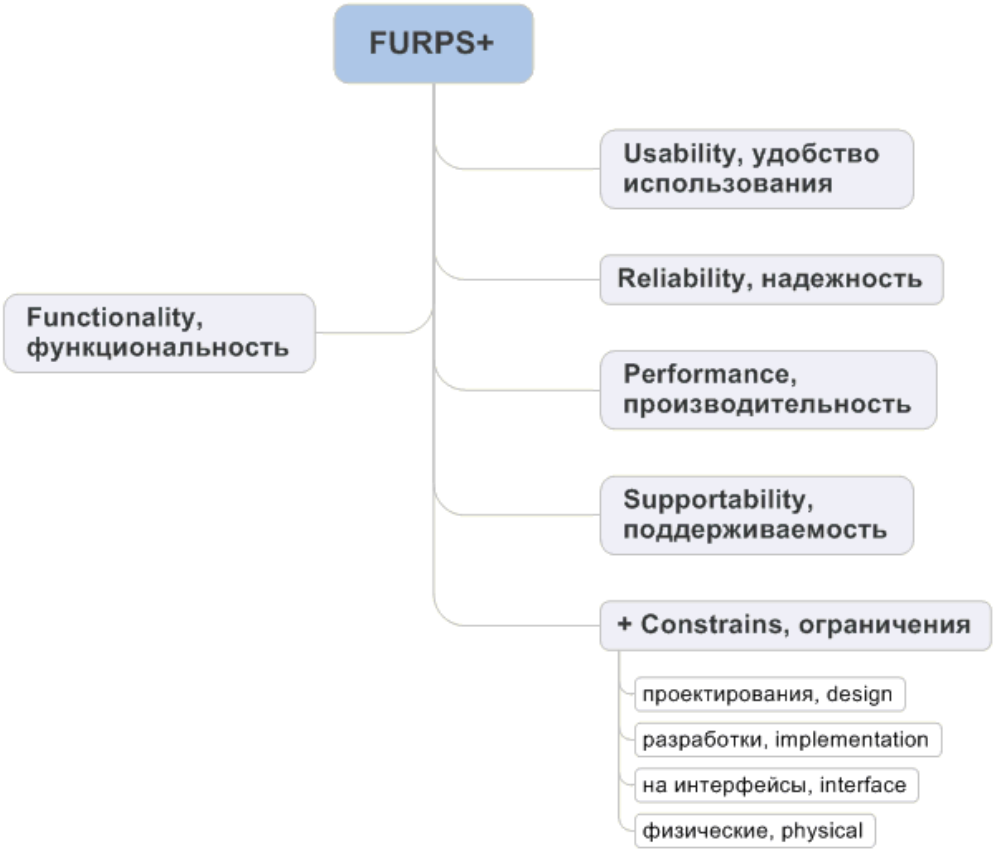

Рисунок 2 – Классификация требований FURPS+

Разрабатываемое приложение имеет 6 функциональных требований, определяющих его основные возможности и приоритеты реализации [24].

Функциональность, в соответствии с классификацией FURPS+, является основой любого приложения или системы. Она определяет цель и реализованные в ней возможности. Диаграммы вариантов использования строятся на основе функциональных требований [3].

Сформулируем и перечислим требования для нашего приложения, которые должны быть учтены при разработке:

- Functionality: передача показаний приборов учёта, история показаний приборов учёта, предварительный расчёт стоимости услуг.
- Usability: удобный и понятный интерфейс, комфортная цветовая гамма приложения.
- Reliability: доступность системы всегда.
- Performance: должна запускаться в течении 3 секунд или менее, а время реакции на пользовательские действия не должно превышать 1 секунду.
- Supportability: поддержка операционной системы iOS и Android, корректная работа функционала на различных устройствах.

Для осознания взаимодействия пользователя с нашей программой используем диаграмму вариантов использования (рисунок 3) [12].

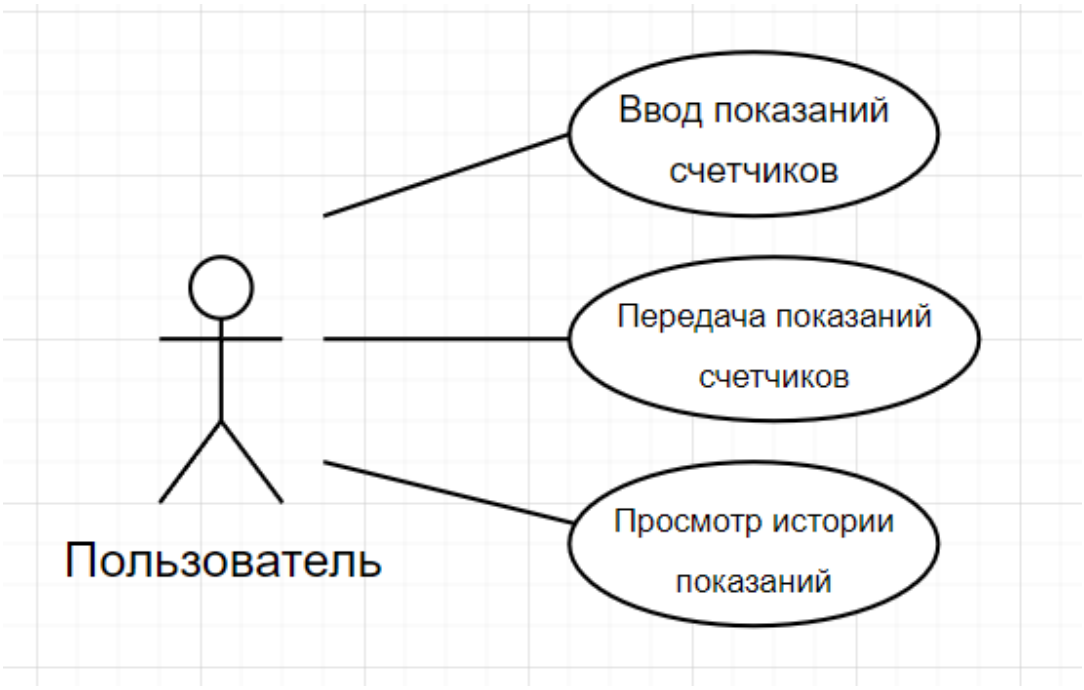

Рисунок 3 – Диаграмма вариантов использования приложения

В результате проектирования были выделены основные потребности пользователя при работе с приложением.

1.3 Сравнительный анализ используемых аналогов

Чтобы ускорить разработку нового приложения, важно оценить, соответствует ли какой-либо из существующих аналогов заданным требованиям. Среди популярных решений для учета показаний счетчиков домохозяйств – «DomoMeter», «Мои счетчики», а также приложения «Фотосчётчик ЖКХ», а также приложения для личных кабинетов отдельных управляющих компаний. Проведение сравнительного анализа этих вариантов на основе установленных критериев поможет определить целесообразность создания пользовательского мобильного приложения (Таблица 1).

Таблица 1 – Сравнительный анализ аналогов

Название / Требования	DomoMeter	Мои счётчики	Фотосчётчик ЖКХ	Приложения управляющих компаний	Разрабатываемое мобильное приложение
Передача показаний приборов учёта в УК	Нет	Нет	Нет	Да	Нет
История показаний приборов учёта	Да	Да	Да	Да	Да
Редактирование приборов учёта	Да	Да	Да	Нет	Да
Несколько ресурсов в одном приложении (вода, газ, электричество)	Да	Да	Да	Нет	Да
Напоминание о подаче показаний счётчиков и оплате квитанций	Да	Да	Да	Да	Да
Поддержка операционной системы Android	Да	Нет	Да	Да	Да
Поддержка операционной системы iOS	Нет	Да	Нет	Да	Да
Контроль факта оплаты счёта	Да	Нет	Да	Да	Да
Ввод показаний по фото	Нет	Нет	Да	Нет	Нет

Цель разработанного приложения – передача показаний напрямую коммунальным организациям. Однако пока эта функция доступна только в отдельных приложениях каждой управляющей компании, что противоречит

идее универсального приложения. Хотя есть и другие приложения, такие как «DomoMeter» и «Фотосчётчик ЖКХ», они поддерживают только одну операционную систему Android и лишены части необходимого функционала. Не смотря на возможность расширить функционал с помощью отдельных модулей, ни одно из этих приложений не предоставляет доступ к исходному коду, что ограничивает возможность внесения дополнительных настроек.

В результате функционально альтернативного приложения, удовлетворяющего указанным выше требованиям, не существует. А значит создание нового мобильного приложения, отвечающего необходимым спецификациям, является подходящим решением.

Предлагается разработать мобильное приложение для своевременной передачи показаний счетчиков, управления оплатой счетов, доступа к истории потребления, а также службы экстренной связи для управляющих организаций (таких как сантехники, электрики и т.д.).

Выводы по главе 1

В первой главе ВКР, выявлены проблемы и предложено решение для мобильного приложения, смоделирована диаграмма «КАК ЕСТЬ» (AS-IS) для авторизации пользователя. Методология FURPS+ использовалась для формирования требований и определения приоритетов реализации. Приложение будет разработано для Android и iOS для увеличения охвата аудитории. Путем сравнительного анализа были выявлены основные характеристики, которые отличают аналоги друг от друга.

Глава 2 Проектирование мобильного приложения

2.1 Выбор архитектуры мобильного приложения

Для выбора архитектуры мы сделаем диаграмму «КАК ДОЛЖНО БЫТЬ» («TO-BE»), которая иллюстрирует предполагаемое будущее состояние предметной области. На рисунке 4 представлена данная диаграмма.

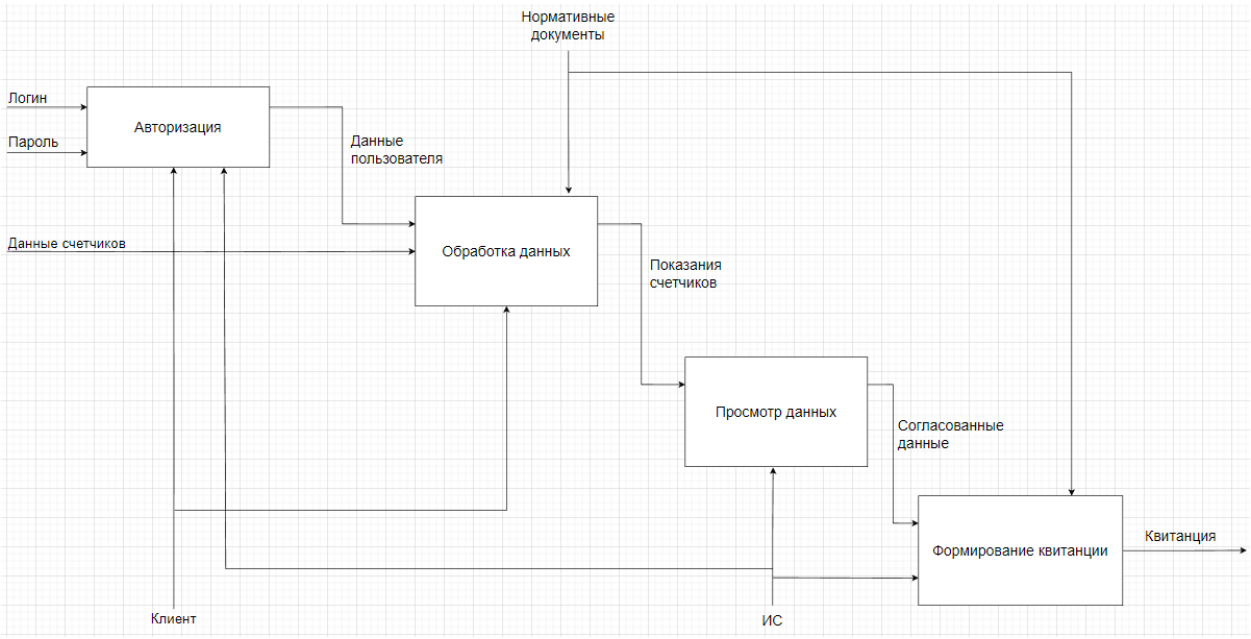

Рисунок 4 – Диаграмма «КАК ДОЛЖНО БЫТЬ» («TO-BE»), регистрация и авторизация

Диаграмма «КАК ДОЛЖНО БЫТЬ» включает в себя следующие процессы:

1. Процесс «Регистрация» подразумевает ввод личных данных для новых пользователей.

2. Процесс «Авторизации» подразумевает ввод логина и пароля для зарегистрированных пользователей.

3. Процесс «Просмотр данных» подразумевает просмотр данных о показаниях счетчиках и сумму платежей.

4. Процесс «Формирование квитанции» подразумевает вывод квитанции для каждого пользователя.

После составления диаграммы «КАК ДОЛЖНО БЫТЬ» можно приступать к проектированию мобильного приложения.

Разрабатываемое мобильное приложение предназначено для выполнения разнообразных функциональных требований благодаря своей модульной архитектуре. Основной задачей приложения является передача показаний счетчиков. Для этого ему необходимо беспрепятственно интегрироваться с внешними Web-приложениями управляющей компании через личные кабинеты. Мы также планируем хранить историю чтения для дальнейшего использования и сделать ее доступной для пользователя, разрешив загрузку на свое устройство. Чтобы обеспечить плавное выполнение всех этих функций, наше приложение должно иметь эффективную и адаптируемую архитектуру, способную удовлетворить множество независимых требований. Мы предлагаем использовать сервер для обработки запросов пользователей для реализации части функционала. Следовательно, мы формируем трехуровневую клиент-серверную архитектуру, которая включает в себя мобильное приложение, сервер обработки запросов, промежуточную базу данных, используемую сервером, а также Web-приложения, предоставляемые поставщиком утилит [5], [7]. Схема архитектуры приложения показана на рис. 5. Эта архитектура позволит нам разработать надежное и эффективное мобильное приложение, удовлетворяющее всем функциональным требованиям [15].

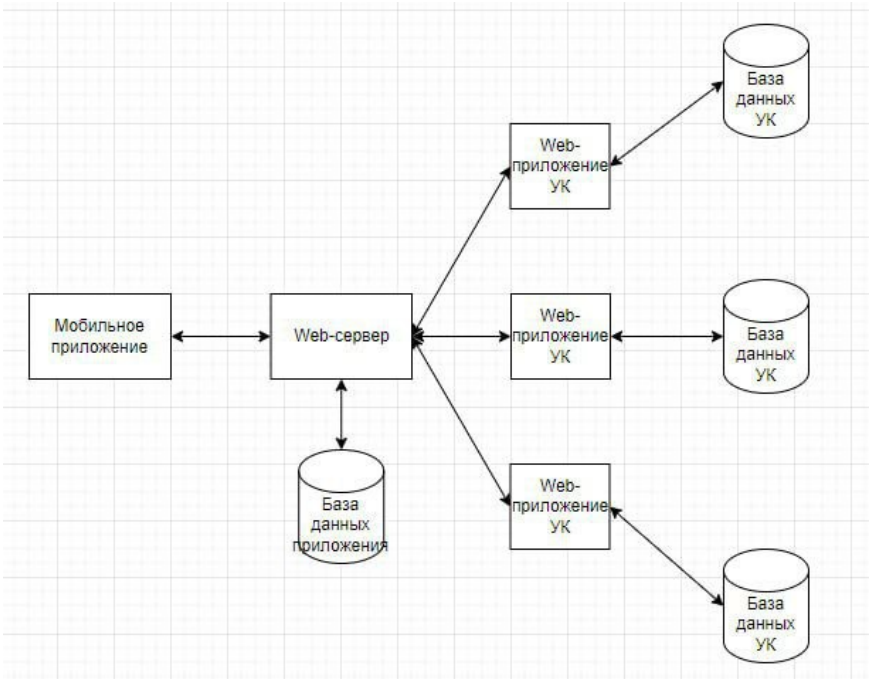

Рисунок 5 – Взаимодействие мобильного приложения с программными средствами УК

В этом случае мобильное приложение взаимодействует с серверами управляющих компаний, в зависимости от принадлежности пользователя той или иной управляющей компании [13], [18].

Также, для нашего приложения необходимо, чтобы оно поддерживалось на операционных системах Android и iOS.

2.2 Выбор средств разработки мобильного приложения

Рассмотрим несколько платформ, предназначенных для разработки мобильных приложений[17].

Xamarin – это кроссплатформенная среда разработки, разработанная Microsoft, которая позволяет разработчикам создавать мобильных приложения для Android, iOS и WindowsPhone с использованием единой

кодовой базы, написанной на C#. Xamarin предлагает разработчикам возможность использовать собственные операционные системы iOS и Android с помощью своих обширных библиотек классов. Фреймворк включает в себя собственную IDE, компиляторы и поддерживает работу через VisualStudio с помощью плагина. Являясь частью платформы .NET, Xamarin совместим с другими продуктами .NET и предлагает бесплатные варианты лицензирования для коммерческого использования. Microsoft утверждает, что использование Xamarin приводит к тому, что до 75-90% общего кода между системами повышает эффективность и экономичность разработчиков, работающих над кроссплатформенными проектами.

ReactNative – это среда разработки мобильных приложений, которая позволяет разработчикам создавать мобильные приложения для Android и iOS с использованием одной и той же кодовой базы. Он был разработан Facebook и выпущен в 2015 году, и с тех пор использовался для разработки некоторых из самых популярных мобильных приложений, включая Facebook, Instagram, Messenger и Skype. Фреймворк основан на библиотеке ReactJavaScript, которая позволяет разработчикам создавать пользовательские интерфейсы с использованием декларативного синтаксиса.

Одним из преимуществ использования ReactNative является то, что он позволяет ускорить циклы разработки, поскольку разработчики могут повторно использовать большую часть своего существующего кода на разных платформах. Кроме того, ReactNativeforWeb – это ответвление фреймворка, позволяющее разработчикам создавать Web-приложения с использованием той же кодовой базы. Это привело к тому, что некоторые популярные Web-приложения, такие как Twitter и Uber, разрабатывались с использованием ReactNative. В целом, ReactNative – это мощный инструмент для разработчиков, стремящихся быстро и эффективно создавать мобильные приложения.

Cordova/PhoneGap – это платформа, используемая для создания мобильныхWeb-приложений. Cordova, изначально разработанная канадским

стартапом Nitobi в 2009 году, была приобретена Adobe в 2011 году. Позднее она была передана Apache Foundation, чтобы продолжить ее разработку. В результате этого передела платформа была переименована в Cordova. Adobe продолжала развивать платформу под названием PhoneGap до 2020 года, когда поддержка была прекращена, и осталась только одна ветвь разработки. Cordova основана на JavaScript и может использоваться для создания как мобильных, так и Web-приложений, но имеет меньше поддержки и инфраструктуры по сравнению с другими платформами. Несмотря на это, Cordova по-прежнему используется в качестве движка для других фреймворков. В целом Cordova/PhoneGap имеет богатую историю и широко используется для создания мобильных приложений.

Ionic – это всеобъемлющий фреймворк, основанный на Cordova, предоставляющий разработчикам отличную среду разработки и широкий спектр полезных инструментов. С Ionic разработчики могут легко изменять код и переносить его между платформами. Одной из его ключевых особенностей является конструктор без кода для приложений пользовательского интерфейса, основанный на HTML. Еще одна выдающаяся функция – возможность обновлять приложения, размещенные в AppStore, с критическими изменениями, такими как исправление ошибок безопасности или работа вне запланированных обновлений. С помощью Ionic разработчики также могут автоматически создавать все значки приложений на основе шаблонов. Однако с точки зрения разработки Web-приложений Ionic не так подходит, как для мобильных приложений. Тем не менее, Ionic – это мощная платформа, которая предоставляет разработчикам множество инструментов и функций для создания высококачественных функциональных мобильных приложений.

Flutter – это платформа для разработки мобильных, настольных и Web-приложений с открытым исходным кодом, созданная и поддерживаемая Google. Его язык программирования Dart предлагает современную и эффективную альтернативу JavaScript. В основе Flutter лежит архитектура на

основе виджетов, позволяющая разработчикам легко создавать интерактивные элементы [11]. В отличие от других платформ, виджеты встроены в само приложение, а не являются отдельной платформой, что позволяет достичь высокой производительности приложений. Flutter также совместим с собственными библиотеками Android и iOS и API-интерфейсами платформы, что позволяет разработчикам с легкостью переносить устаревшие приложения во Flutter. Кроме того, среда разработки Flutter включает функцию горячей перезагрузки, позволяющую вносить изменения в приложение в режиме реального времени без необходимости его перезапуска. Эта функция позволяет разработчикам легко отслеживать и исправлять ошибки в приложении и при необходимости вносить изменения. Проанализировав все доступные нам платформы для разработки мобильного приложения, мы сможем сделать выбор в пользу наиболее подходящей платформы (Таблица 2) [10].

Таблица 2 – Сравнение сред для разработки мобильного приложения

Платформа разработки	Язык программирования	Мобильные платформы	Удобство интерфейса	Работа с базами данных
Xamarin	C#	Android, iOSи WindowsPhone	+	-
ReactNative	JavaScript	Android и iOS	+	+
Cordova/PhoneGap	JavaScript	Android и iOS	-	+
Ionic	HTML	Android, iOSи WindowsPhone	-	+
Flutter	Dart	Android и iOS	+	-
Итого	JavaScript	Android и iOS	+	+

Наиболее популярными решениями, согласно динамике поисковых запросов, являются ReactNative и Flutter (Рисунок 6). Популярность в контексте выбора среды разработки означает число реализованных проектов, классов и шаблонов, а также сообщество программистов, готовых помочь с возникающими трудностями и разрабатывающих руководства по реализации тех или иных функций в выбранной SDK [4].

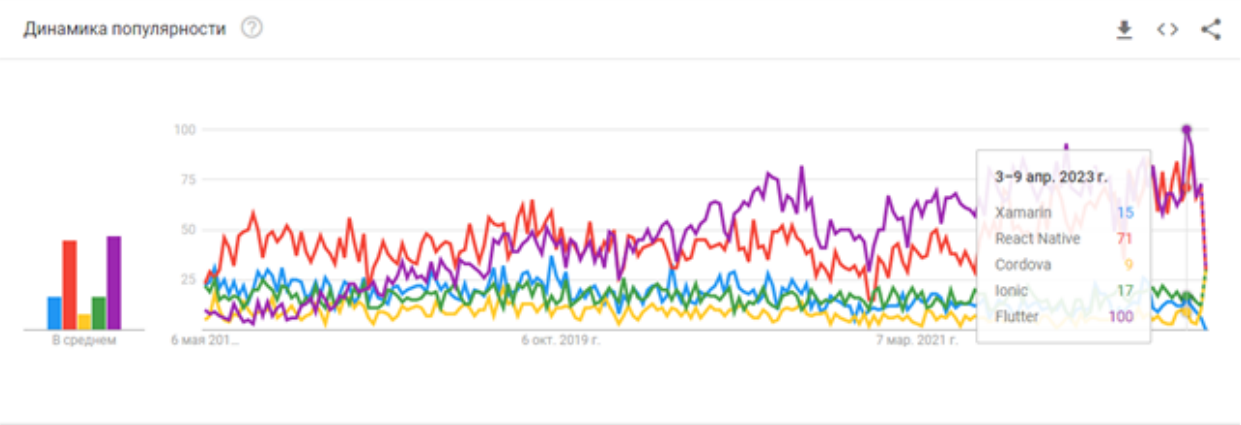

Рисунок 6 – Динамика популярности средств разработки мобильных приложений в GoogleTrends на 2023 год

Исходя из популярности, следовало бы выбрать Flutter, однако отсутствие опыта написания кода на Dart увеличивает трудоёмкость разработки. Поэтому было решено использовать платформу ReactNative для разработки мобильного приложения.

React Native был выбран для мобильного приложения из-за его экономичности, удобного пользовательского интерфейса и доступности обучающих материалов. Программный код написан на JavaScript и выполняется на мобильных платформах или в браузере. Из-за ограничений Apple для Android доступно больше сторонних реализаций. JSengine обрабатывает весь код JavaScript, в то время как основной поток отображает элементы интерфейса приложения, а отдельные потоки обрабатывают нативные компоненты, такие как клавиатура [20].

2.3 Разработка логической модели мобильного приложения

Логическая модель приложения представляет собой графическое представление, которое иллюстрирует взаимосвязи между компонентами приложения, отображая их взаимоотношения и возможные сценарии

взаимодействия. Она также определяет различные типы объектов, которые присутствуют в приложении.

Взаимодействие с приложением осуществляется только одним пользователем, и все доступные функции активируются либо им самим, либо автоматически приложением по расписанию или при наступлении определенных событий. Схема логической работы пользователя с мобильным приложением представлена на рисунке 7.

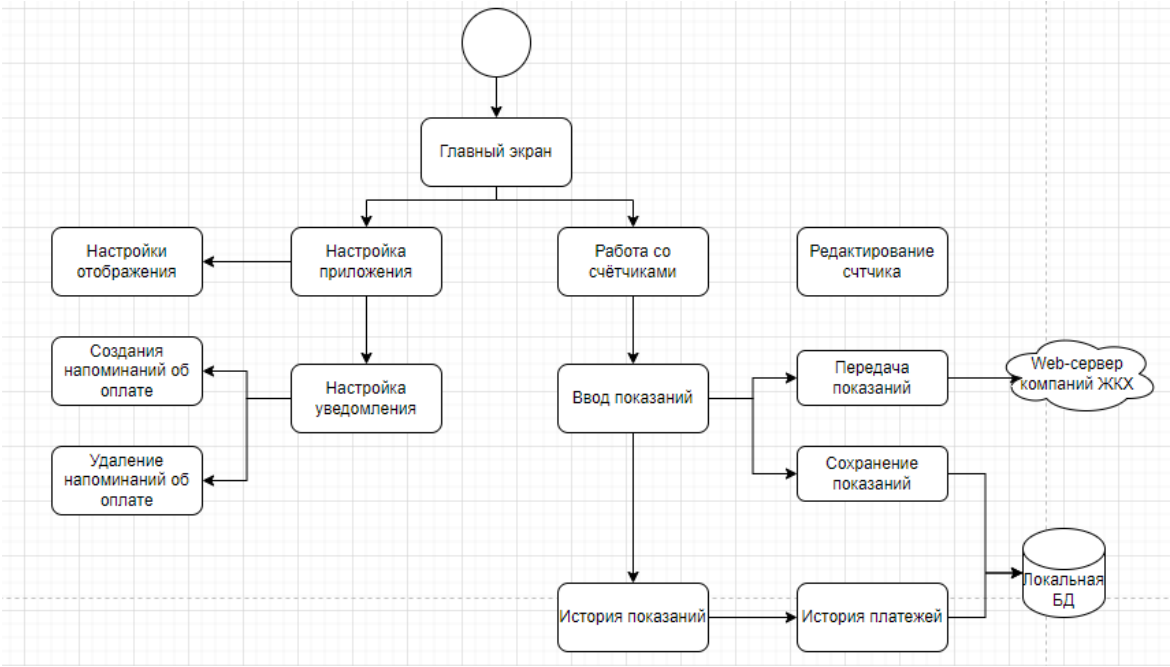

Рисунок 7 – Логическое взаимодействие компонентов

Все возможные функции в приложении инициируются одним пользователем, который с ним взаимодействует [6]. Подробное описание сценариев, представленных на диаграмме, показано выше. На рисунке 8 показана диаграмма классов UML мобильных приложений, разработанная на основе диаграммы вариантов использования приложения.

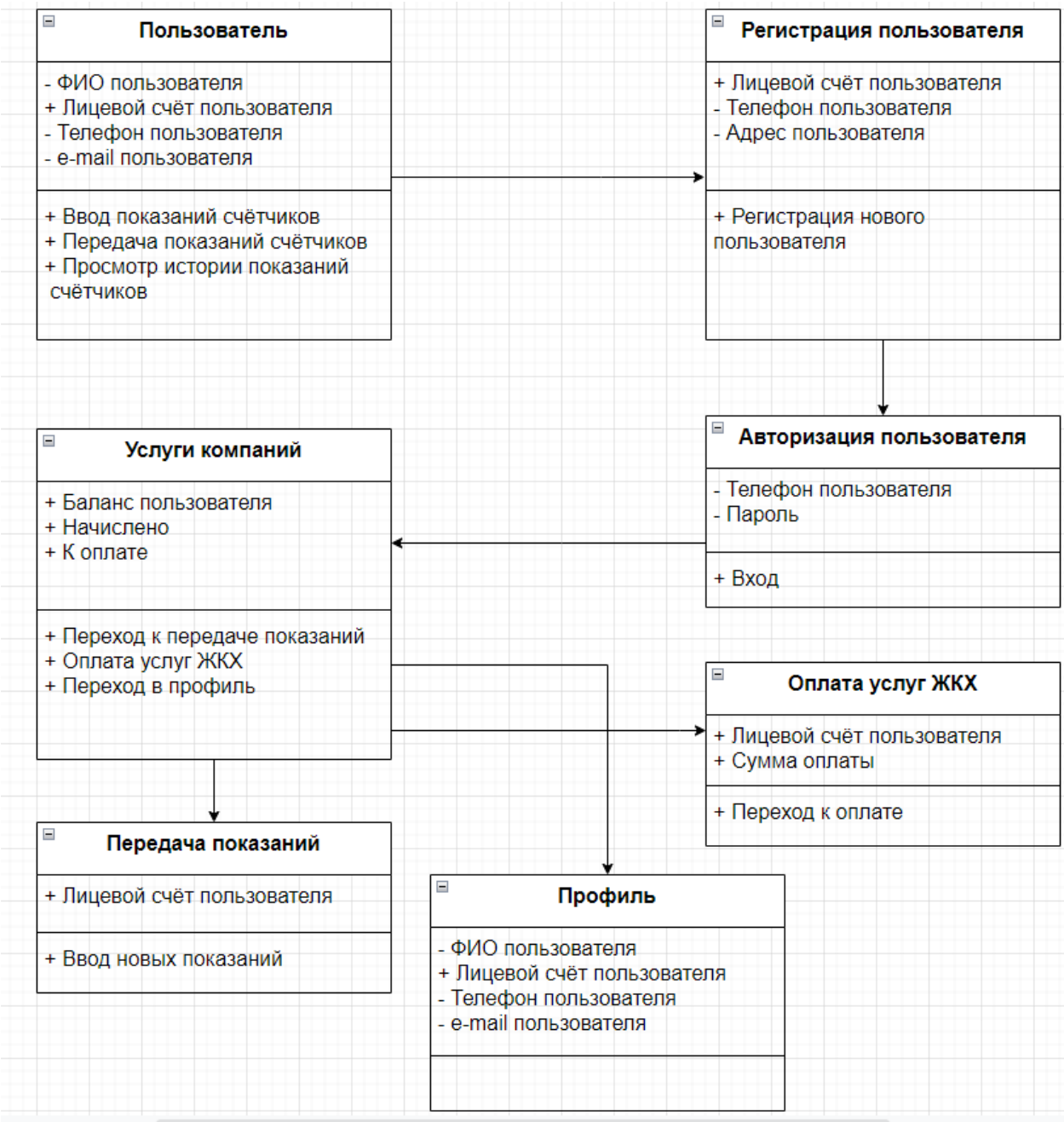

Рисунок 8 – Диаграмма классов UML пользователя

На рисунках 9 и 10 показаны диаграммы компонентов, которые отображают соотношение компонентов программного обеспечения, зависимости и интерфейсы. Эти диаграммы помогают в разработке архитектуры будущей системы.

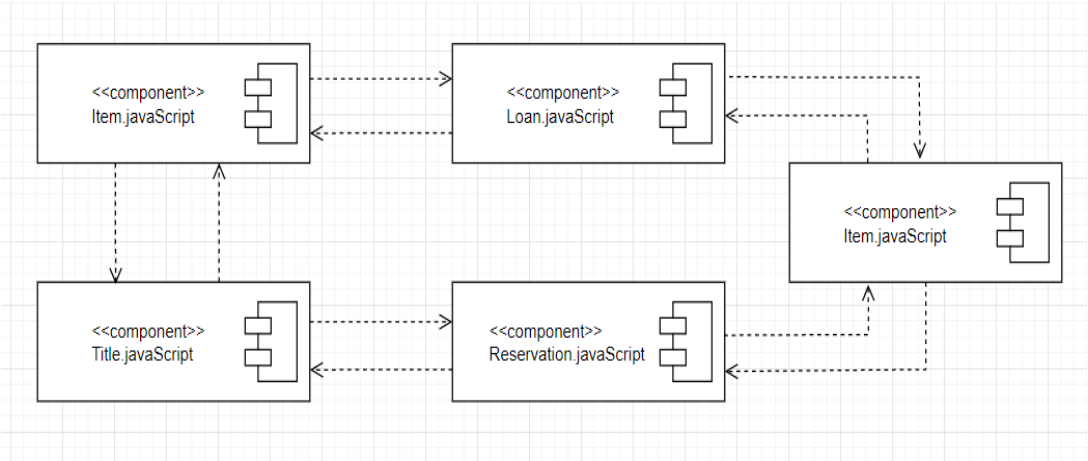

Рисунок 9 – Компонентная модель кода для JavaScript

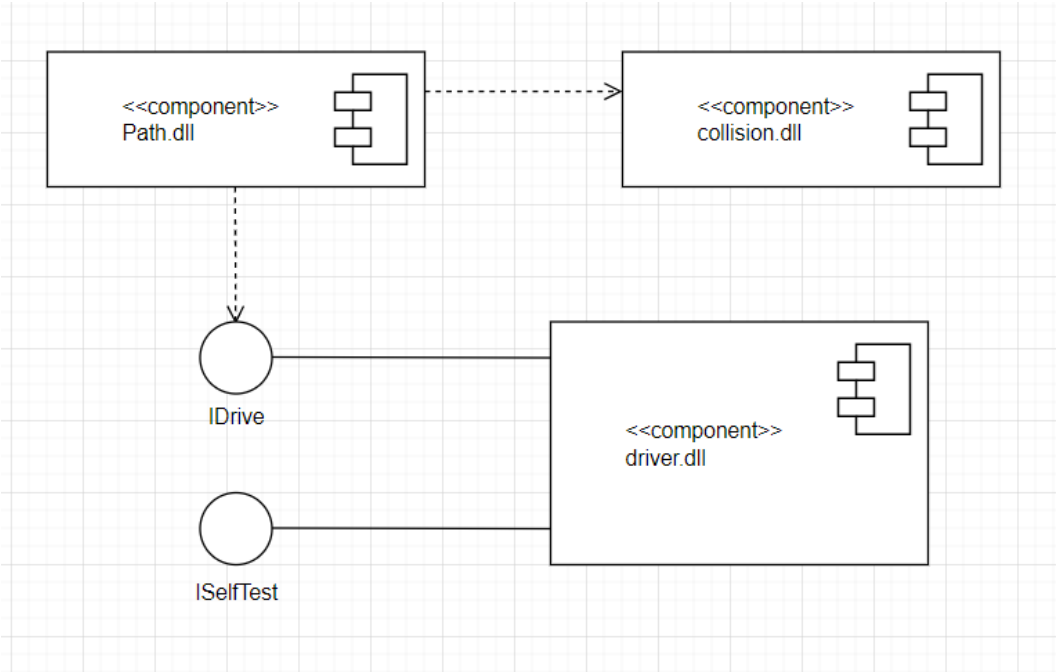

Рисунок 10 – Компонентная модель исполняемой версии

Важнейшими графическими элементами диаграмм являются компоненты, а также связывающие их интерфейсы [8], [9], [19].

2.4 Разработка физической модели данных мобильного приложения

Физическая модель данных является завершающим этапом проектирования базы данных для программного продукта. Она визуально отображает фактическое физическое представление, основанное на предварительно определенной логической модели, и фактически создает базу данных в текущий момент времени. В этой модели физического объекта реляционные таблицы, а их экземпляры отражаются в виде строк таблицы, которое представлено на рисунке 11 [21], [22].

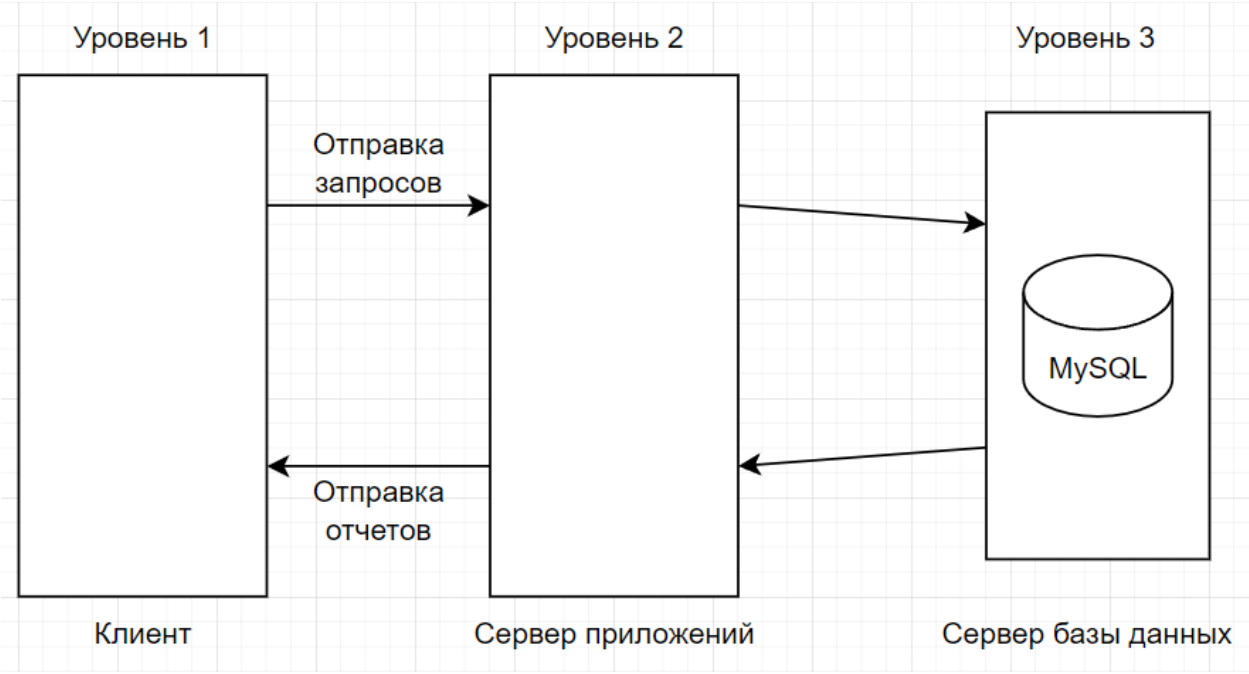

Рисунок 11 – Архитектура СУБД MySQL

MySQL – это система управления реляционными базами данных (RDBMS) с открытым исходным кодом, использующая архитектуру клиент-сервер. Он предназначен для эффективного управления большими объемами данных и позволяет нескольким пользователям одновременно получать доступ к данным и управлять ими. Архитектура СУБД MySQL

состоит из нескольких уровней, включая уровень языка запросов, уровень SQL, уровень механизма хранения и уровень операционной системы. Уровень языка запросов интерпретирует запросы SQL и отправляет их на уровень SQL, который обрабатывает запросы и выполняет их на уровне механизма хранения. Уровень механизма хранения отвечает за хранение и извлечение данных с диска, а уровень операционной системы управляет аппаратными ресурсами и предоставляет интерфейс для взаимодействия с сервером MySQL. В целом, архитектура СУБД MySQL предназначена для предоставления масштабируемой, высокопроизводительной и надежной системы управления базами данных.

В трехуровневой архитектуре несколько серверов обрабатывают клиентский запрос одновременно, что может эффективно снизить нагрузку на сервер за счет распределения операций. Однако этот подход может быть не таким надежным, как двухуровневая архитектура [16].

Диаграмма развертывания – это диаграмма на унифицированном языке моделирования (UML), которая показывает, как аппаратные и программные компоненты системы развертываются на различных узлах и как они взаимодействуют друг с другом. На рисунке 12 представлена диаграмма развертывания.

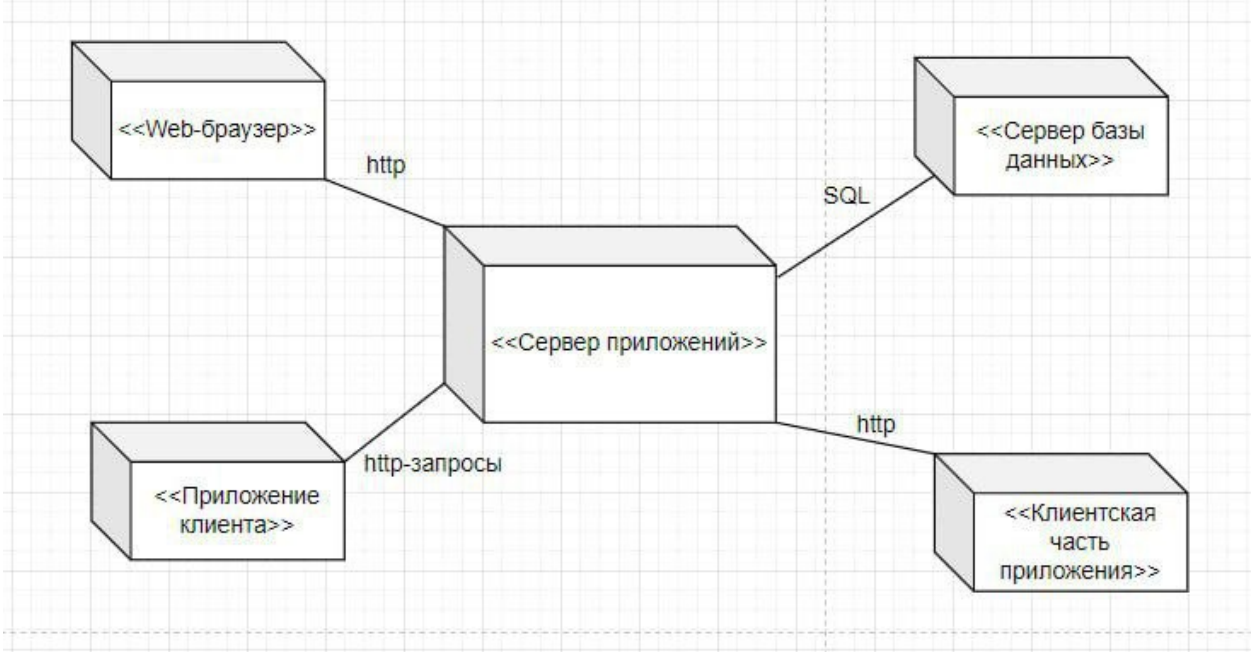

Рисунок 12 – Диаграмма развертывания

На рисунке 13 будет показана физическая модель базы данных, предназначенная для мобильного приложения, которое использует систему управления базами данных MySQL.

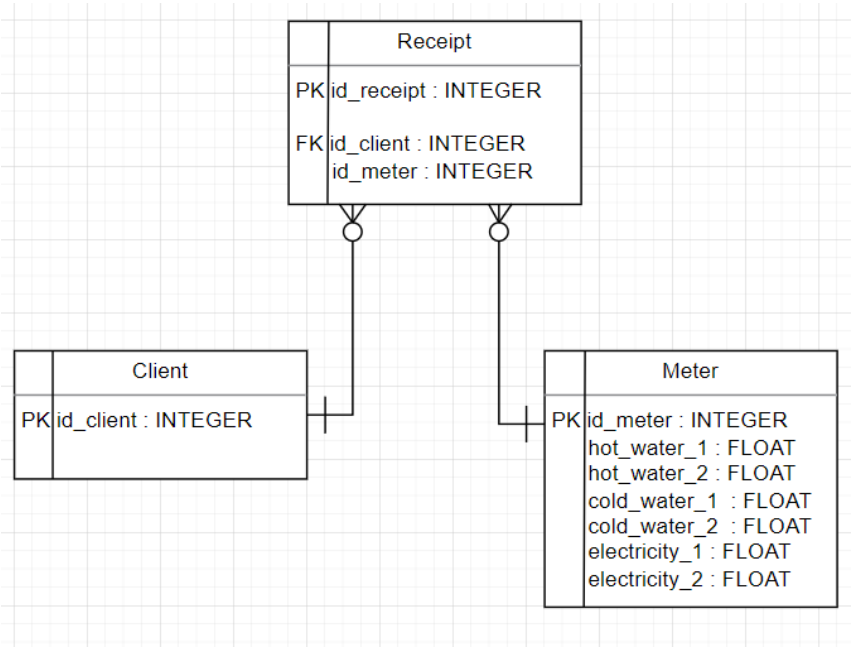

Рисунок 13 – Физическая модель данных

Этот шаг включает в себя организацию файлов, индексов и установление базовых отношений для обеспечения осмысленного доступа к данным, а также реализацию необходимых мер защиты и ограничений целостности.

В физической модели объекты представлены в виде таблиц, где каждый экземпляр представляет собой строку в таблице. Атрибуты соответствуют столбцам в этих таблицах. Кроме того, физическая модель данных требует, чтобы объекты содержали типы данных, специфичные для выбранной СУБД.

На основе логической модели данных успешно завершена разработка физической модели базы данных для мобильного приложения. Это можно считать завершающим этапом проектирования базы данных мобильного приложения.

Выводы по главе 2

Вторая глава ВКР посвящена разработке мобильного приложения.

Для выбора архитектуры, сделали диаграмму «КАК ДОЛЖНО БЫТЬ» («TO-BE»), которая отображает будущее предполагаемое состояние предметной области.

Была спроектирована трехуровневую клиент-серверная архитектура и проведен анализ средств разработки мобильных приложений. В конечном итоге мы решили, что фреймворк React Native был наиболее подходящим выбором.

Также создали функциональную диаграмму использования для отображения взаимодействия пользователя с приложением, а также логическую диаграмму с подробным описанием взаимодействия различных компонентов.

Глава 3 Реализация и тестирование мобильного приложения для получателей жилищно-коммунальных услуг

3.1 Реализация мобильного приложения

ReactNative использует Node.js, среду выполнения JavaScript, для генерации кода JavaScript. Начнем с установки Homebrew, а затем Node.js, следуя этим инструкциям в окне терминала:

```
brew install node
```

После этого установите watchman, службу, которая отслеживает изменения и находит файлы с помощью brew:

```
brew install watchman
```

React Native использует сторож для отслеживания изменений кода и внесения соответствующих корректировок.

Установим интерфейс командной строки React Native (CLI):

```
npm install -g react-native-cli
```

С помощью NodePackageManager для глобального вызова и установки инструмента CLI, который поставляется с Node.js.

Затем используем инструмент CLI для создания начального проекта, включающего все необходимые элементы для разработки и запуска приложения ReactNative:

```
react-native init PropertyFinder
```

В папке node_modules расположен фреймворк ReactNative, а файл index.ios.js представляет является сгенерированным инструментом CLI макет приложения.

Далее внесем код, представленный в листинге 1.

Листинг 1 – Добавление единого стиля

```
varstyles = React.StyleSheet.create({
text: {
color: 'black',
backgroundColor: 'white',
fontSize: 30,
margin: 80
}
});
```

Этот код задает единый стиль.

Добавим следующий код прямо под переменной со стилями:

```
class PropertyFinderApp extends React.Component {
render() {
return React.createElement(React.Text, {style: styles.text}, "Авторизация")};
}
```

Вступление классов произошло с выпуском ECMAScript 6 (ES6), и разработчикам следует соблюдать определенные ограничения, чтобы обеспечить совместимость с более старыми системами и браузерами. Наконец, добавим в конец файла эту строку:

```
React.AppRegistry.registerComponent('PropertyFinder', function() { return PropertyFinderApp });
```

AppRegistry определяет точку входа в приложение и предоставляет корневой компонент. Добавим поле ввода и кнопки, отредактировав файл SearchPage.js (листинг 2).

Листинг 2 – Добавление полей ввода и кнопки

```
React.AppRegistry.registerComponent('PropertyFinder', function() { return PropertyFinderApp });

 <Viewstyle={styles.flowRight}>
<TextInput
style={styles.searchInput}
placeholder='+7 888 888 88 88'/>
<TouchableHighlight style={styles.button}
underlayColor='#3c9630'>
<Text style={styles.buttonText}>Go</Text>
</TouchableHighlight>
</View>
<TouchableHighlight style={styles.button}
underlayColor='#3c9630'>
<Text style={styles.buttonText}>Войти</Text>
</TouchableHighlight>
```

Мы добавили два поля ввода и одну кнопку.

Чтобы добавить следующий код под закрывающим тегом компонента TouchableHighlight, которые отвечает за кнопку «Войти», вернемся к файлу SearchPage.js:

```
<Image source={require('./Resources/house.png')} style={styles.image}/>
```

Чтобы реализовать поиск, обрабатываем нажатие кнопки «Вход», создаем требуемый запрос API и сообщаем пользователю, что запрос обрабатывается.

Обновим начальное состояние внутри конструктора файла SearchPage.js:

```
 this.state = {
searchString: 'london',
isLoading: false
};
```

Чтобы добавить индикатор загрузки на страницу, вставим {spinner} под изображением в JSX, который управляет интерфейсом поиска

Затем добавим данные методы в класс SearchPage (листинг 3):

Листинг 3 – Добавление сообщений в консоль

```
_executeQuery(query) {
console.log(query);
this.setState({ isLoading: true });
}
onSearchPressed() {
var query = urlForQueryAndPage('place_name', this.state.searchString, 1);
this._executeQuery(query);
}
```

Упомянутый выше код формирует основу для реализации мобильного приложения.

3.2 Тестирование мобильного приложения

Для тестирования приложения использован метод функционального тестирования.

Функциональное тестирование – это метод тестирования программного обеспечения, который проверяет функциональность приложения или системы на соответствие заданным требованиям. Он включает в себя тестирование функций и функций программного обеспечения, чтобы убедиться, что оно работает должным образом и соответствует потребностям пользователя [23], [25].

Цель функционального тестирования – убедиться, что программная система выполняет все функции, которые она должна выполнять, и не выполняет функции, для которых она не предназначена. Обычно его проводят тестировщики программного обеспечения, которые хорошо разбираются в требованиях и спецификациях программного обеспечения.

Мы будем оценивать производительность нашего приложения на основе следующих критериев, используя этот подход:

- запуск приложения;
- функциональность основных функций приложения;
- авторизация номера телефона;
- регистрация по номеру телефона или в личном кабинете;
- проверка обязательных полей;
- возможность изменять информацию профиля пользователя.

Для получения доступа к системе мы можем следовать этим шагам: запустить приложение, ввести свой номер телефона и пароль для входа в систему, после чего нажать кнопку «Войти», которая показана на рисунке 14.

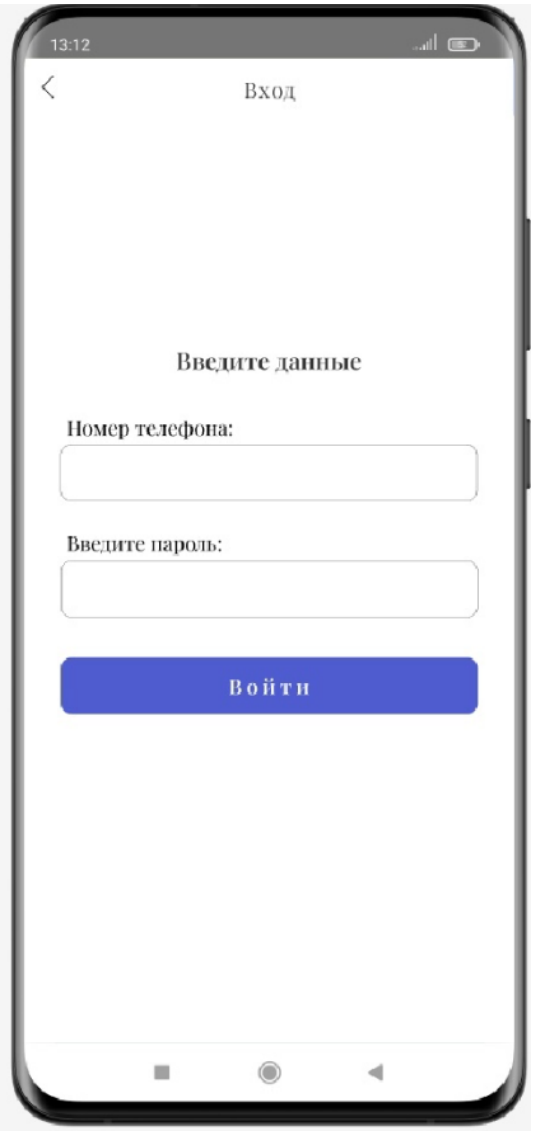

Рисунок 14 – Авторизация

Для завершения процесса регистрации пользователь должен нажать на кнопку, помеченную как «Регистрация». Появится окно, в котором нужно ввести свои данные, а именно: номер телефона, адрес, e-mail, ФИО и пароль (как показано на рисунке 15).

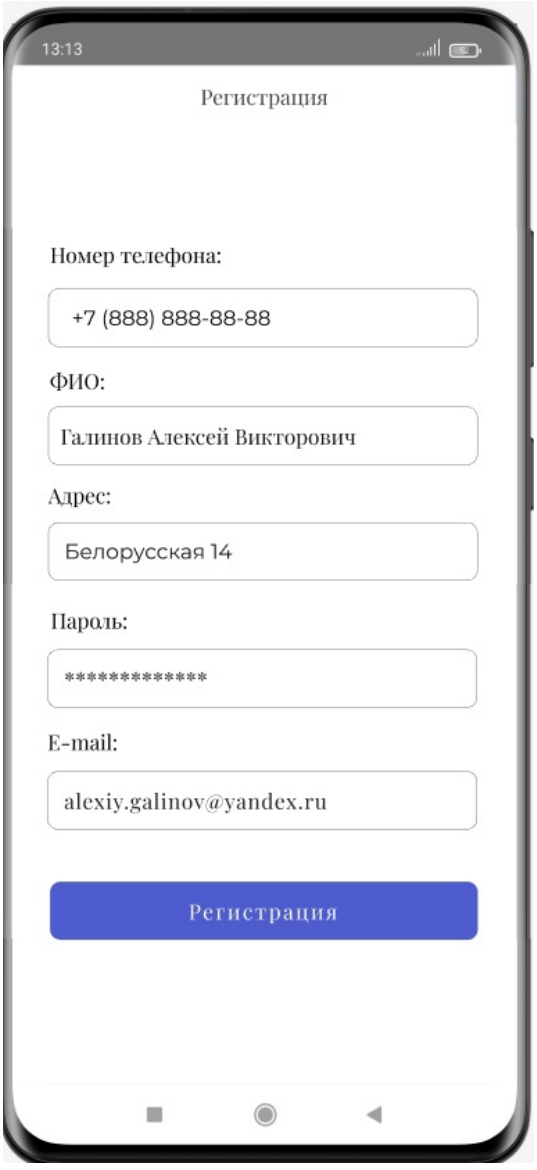

Рисунок 15 – Регистрация

После того, как вы ввели необходимые данные, нажмите на кнопку «Зарегистрироваться». Это действие вызовет появление окна, где нужно указать пароль повторно. После того, как вы успешно создали пароль, вы будете перенаправлены на вкладку «Авторизация». Здесь вам нужно будет ввести свой номер телефона и вновь созданный пароль.

Мы попадаем на домашнюю страницу мобильного приложения, где мы можем увидеть различные услуги, предлагаемые компанией, в том числе холодное водоснабжение, горячее водоснабжение и электричество. В каждом сервисном блоке отображается текущий баланс, начисленная сумма и оставшаяся сумма к оплате. При нажатии на кнопку «Перейти к передаче показаний», мы переходим на новую страницу, где есть возможность ввести новые показания счетчика (как показано на рисунках 16 и 17).

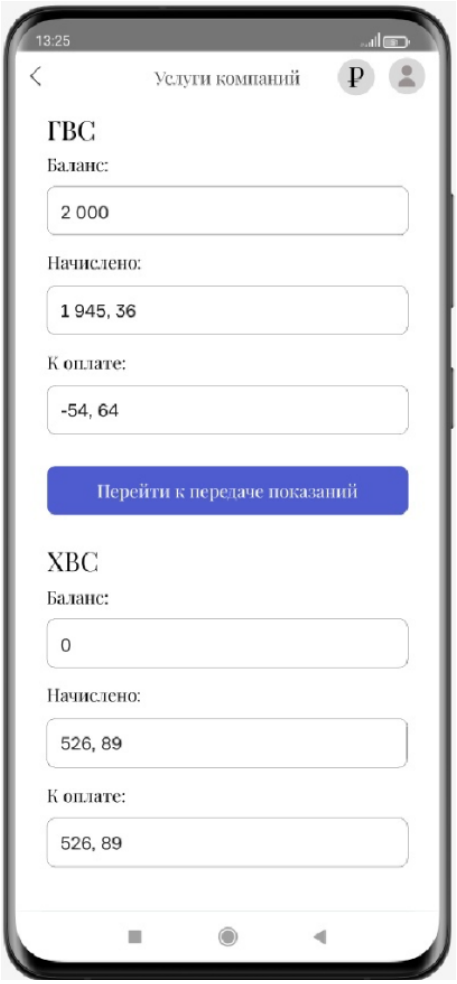

Рисунок 16 – Услуги компании ГСВ и ХСВ

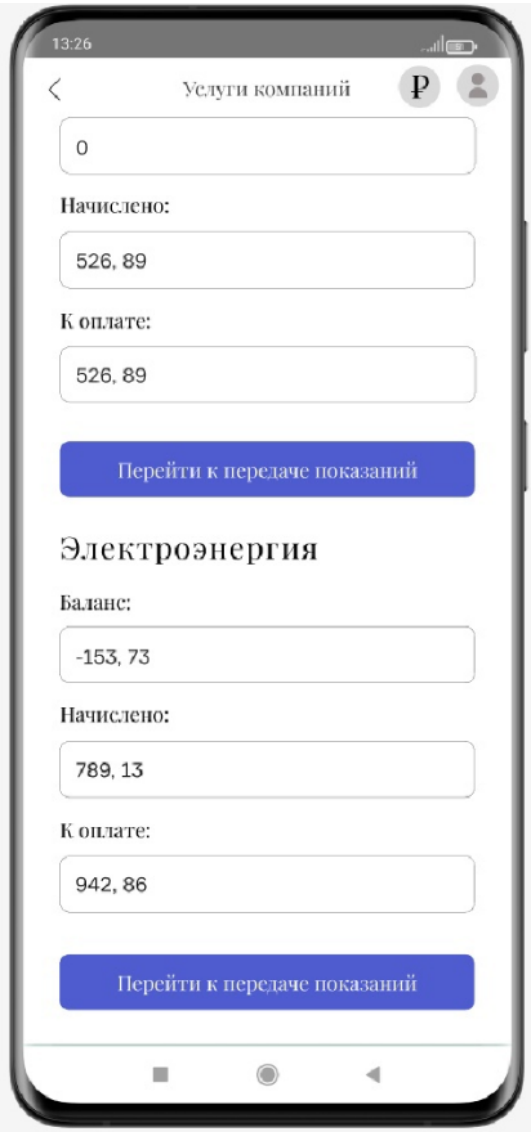

Рисунок 17 – Услуги компании Электроэнергия

Вверху экрана есть две кнопки «Оплата коммунальных услуг» и «Профиль». При выборе «Оплата коммунальных услуг» вам будет предложено ввести номер лицевого счета и сумму к оплате. Приложение также рассчитает комиссию. После того, как мы ввели необходимую информацию, мы можем нажать на кнопку «Перейти к оплате». Это приведет нас к использованию мобильного приложения банка, где мы сможем

выполнить транзакцию по оплате счетов за коммунальные услуги (рисунок 18).

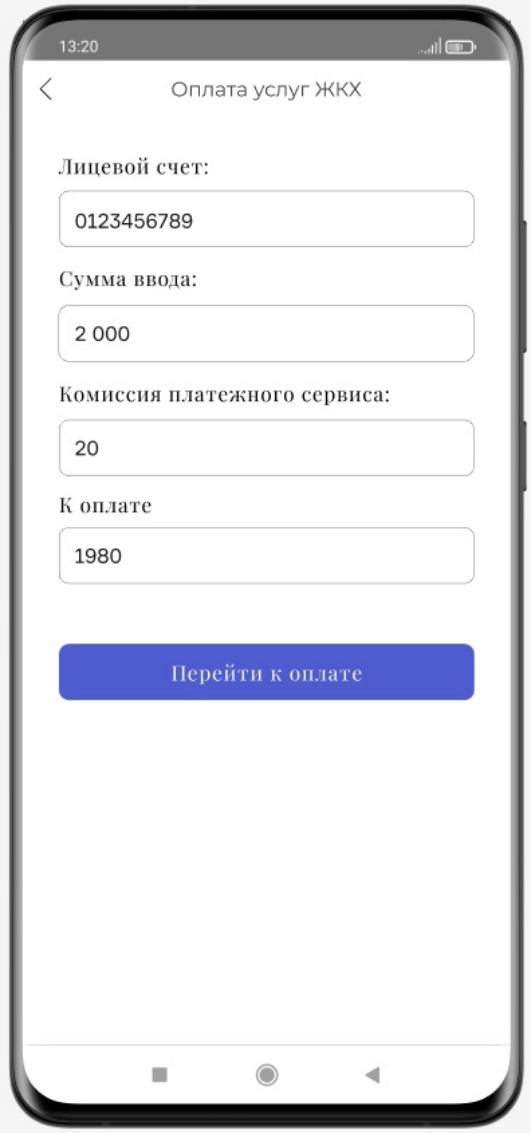

Рисунок 18 – Оплата услуг ЖКХ

Нажав на вторую кнопку, расположенную в правом верхнем углу главного меню, мы переходим к странице «Профиль» (рисунок 19). На этой странице отображается наша личная информация, такая как: ФИО, адрес, номер телефона и адрес электронной почты.

Рисунок 19 – Окно «Профиль»

На главной странице есть раздел «ГВС», «ХСВ» и «Электроэнергия». Нажав на кнопку «Перейти к передаче показаний» в этом разделе, мы перейдете на страницу «Показания приборов учета». На этой странице нам будет предложено ввести данные нашей личной учетной записи. После этого приложение отобразит информацию о наших счетчиках. Отсюда мы можем ввести новые показания наших счетчиков (как показано на рис. 20).

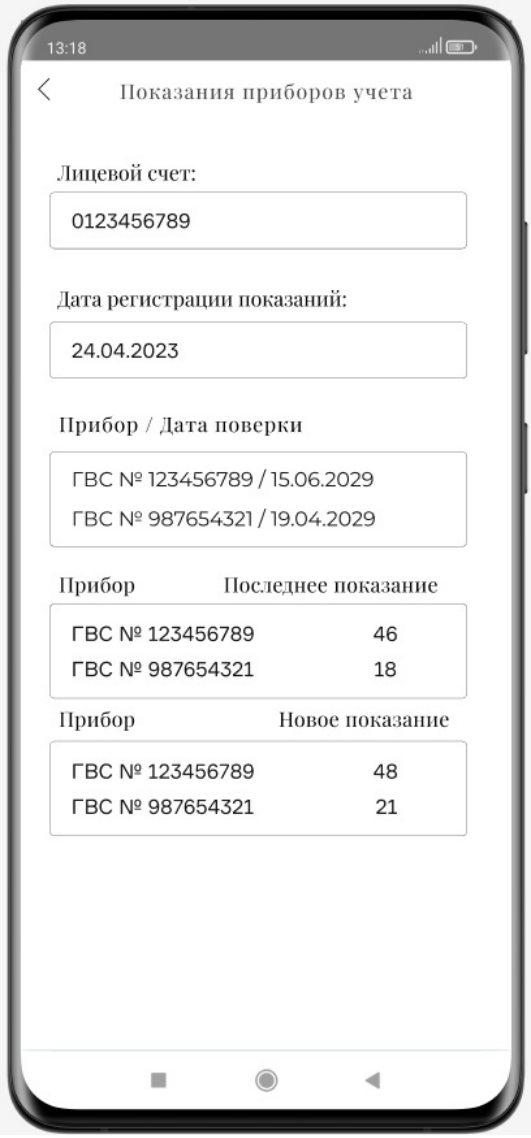

Рисунок 20 – Регистрация показаний приборов учета

Также с помощью функционального тестирования были произведена проверка работоспособности мобильного приложения, а именно вход в личный кабинет.

Проверка. Вход в личный аккаунт.

При авторизации пользователь вводит свой номер телефона и пароль в соответствующие поля в окне, а затем нажимает на кнопку «Войти». Если введенные данные верны, то пользователь переходит в главное меню, иначе

мобильное приложение уведомляет пользователя о неправильности введенных данных.

Результаты проведенного нами функционального тестирования подтверждают, что приложение функционирует в соответствии с выбранными критериями.

Выводы по главе 3

Используя определенный инструментарий, мы смогли реализовать различные функции мобильного приложения, такие как передача новых показаний и оплата ЖКХ. Реализация мобильного приложения была выполнена с помощью кода JavaScript, при этом для его разработки использовался фреймворк React Native. Тестирование проводилось с помощью функционального тестирования, которое подтвердило работоспособность приложения для выбранных элементов. Поэтому можно считать, что мобильное приложение успешно прошло функциональное тестирование.

Заключение

Главной целью выпускной квалификационной работы являлась разработка мобильного приложения для получателей жилищно-коммунальных услуг. Для достижения поставленной цели был проведен анализ предметной области для выявления проблем и предложения решения в виде мобильного приложения.

По методике FURPS+ определены приоритеты требований к приложению, и принято решение разработать мобильное приложение для операционных систем Android и iOS для расширения охвата аудитории. Также был проведен анализ существующих приложений, который показал потребность в мобильном приложении, отвечающем конкретным требованиям данного проекта.

В качестве системы управления базами данных выбрана MySQL. Данная система лучше всего подходит для работы с клиент-серверным мобильным приложением и обеспечивает безопасность запросов пользователя.

Для приложения была разработана «клиент-серверная» архитектура, а в качестве варианта реализации выбрано мобильное приложение для поддержки нескольких операционных систем. Были описаны варианты использования и создана функциональная схема, иллюстрирующая, как пользователи взаимодействуют с приложением и его компонентами.

После анализа различных инструментов мобильной разработки был выбран фреймворк React Native как наиболее подходящий вариант для данного проекта. С помощью языка программирования JavaScript разработана клиентская часть мобильного приложения, позволяющая пользователям передавать новые показания и оплачивать коммунальные услуги. Приложение протестировано с помощью функционального тестирования, и оно соответствует выбранным критериям.

В целом, бакалаврская работа достигла поставленной цели по разработке мобильного приложения для получателей жилищно-коммунальных услуг. В процессе выполнения проекта были приобретены ценные практические навыки в области проектирования и разработки приложений с использованием фреймворка React Native. Полученное мобильное приложение, созданное в рамках данного проекта, может быть дополнительно развито и использовано в качестве альтернативного метода передачи показаний счетчиков в сфере ЖКХ.

Список используемой литературы

1. Аксенов К. В. Обзор современных средств для разработки мобильных приложений /К.В. Аксенов // Новые информационные технологии в автоматизированных системах. 2014. №17. С. 10-11.
2. Бейли Л. Изучаем SQL. СПБ.: Питер, 2012. 573 с.
3. Вигерс К. Разработка требований к программному обеспечению. 3-е изд., дополнительное / К. Вигерс, Д. Битти М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2014. 736 с.
4. Гагарина Л. Г., Киселев Д. В., Федотова Е. Л. Разработка и эксплуатация автоматизированных информационных систем М.: ИД «ФОРУМ»; ИНФРА-М, 2012. 384 с.
5. Голицина О.Л., Максимов Н.В., Попов И.И. Базы данных: Учебное пособие. М.: Формум: ИНФРА-М, 2013. 352 с.
6. Грофф Д. SQL: Полное руководство / Д. Грофф, П. Вайнберг. / К.: BHV, 2001. 816 с.
7. Дейт Д. Введение в системы баз данных. М.: Издательский дом "Вильяме", 2005. 1328 с.
8. Емельянова Н.З. Проектирование информационных систем: учебное пособие / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. М.: Форум, 2014. 432 с.
9. Заботина Н. Н. Проектирование информационных систем М.: ДРОФА, 2013. 336 с.
10. Колесов Ю. Б. Моделирование систем. Объектно-ориентированный подход: учебное пособие / Ю. Б. Колесов, Ю. Б. Сениченков. СПб.: БХВ-Петербург, 2012. 192 с.
11. Конноли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Конноли, К. Бегг, А. Стратчан. М.: Вильямс, 2000. 1093 с.

12. Корнеев И. К. Информационные технологии: учебное пособие. М: Проспект, 2007. 224 с.

13. Куроуз Д. Компьютерные сети. Многоуровневая архитектура Интернета: 2-е издание / Д. Куроуз, К. Росс. СПб.: Питер, 2004. 765 с.

14. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Издание второе, дополненное и переработанное. М., 2012. 672 с.

15. Чистов Д. В. Проектирование информационных систем. Учебник и практикум / Д. В. Чистов, П. П. Мельников, А.В. Золотарюк, Н.Б. Ничепорук. М.: Юрайт, 2016. 260 с.

16. Шейн Ч. Разработка гибридных Web-приложений, способных использовать аппаратные средства мобильных устройств. // журнал MSDN Magazine, 2012. 14 с.

17. Ихтиар В.Ф. Сравнение кроссплатформенных фреймворков // Вестник магистратуры. 2018. №1-3 (76). 25 с. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/razrabotka-kross-platformennyh-prilozheniy-na-ya (дата обращения: 01.03.2023).

18. Калиневич Н., Гильванов Р.Г. Разработка кроссплатформенных приложений на языке Dart при помощи фреймворка Flutter // Интеллектуальные технологии на транспорте. 2021. №4 (28). 7 с. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/razrabotka-kross-platformennyh-prilozheniy-na-y (дата обращения: 19.03.2023).

19. Новожилова А. Азбука клиента. Мобильные приложения: нативные, html5, // CMSMagazine. 2020. №1 (15). 97 с. (дата обращения 10.12.2022)

20. Русанова И.В. Анализ платформ для разработки гибридного мобильного приложения для систем iOS и Android // Актуальные проблемы авиации и космонавтики. 2017. №13. 326 с. [Электронный ресурс] (дата обращения: 09.02.2023).

21. Dawn Griffiths. Head First Android Development: A Brain-Friendly Guide / Dawn Griffiths, David Griffiths - O'Reilly Media, 2015 - 734 p.

22. Helder Vasconcelos. Asynchronous Android Programming / Helder Vasconcelos - Packt Publishing, 2016. - 394 p.

23. Kaner, Falk, Nguyen. Testing Computer Software. – USA: Wiley Computer Publishing, 1999. 42 p.

24. Reto Meier. Professional Android 4 Application Development. / Reto Meier - Wrox, 2012. 864 p.

25. Shoutem - Make an App - Build Apps with Easy Application Creator, 2014. 368 p.