

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)

WEB - дизайн и мультимедиа
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка гибридного мобильного приложения для контроля показаний
бытовых счетчиков»

Обучающийся

А.Д. Шаброва
(Инициалы Фамилия)

(личная подпись)

Руководитель

к.т.н., Т.Г. Султанов

(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант

М.М. Бажутина

(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема бакалаврской работы: «Разработка гибридного мобильного приложения для контроля показаний бытовых счетчиков».

Бакалаврская работа посвящена разработке гибридного приложения для контроля показаний бытовых счетчиков.

В ходе выполнения исследований по бакалаврской работе была проанализирована и выбрана технология разработки гибридного мобильного приложения, было разработано гибридное мобильное приложение, выполнена реализация и тестирование мобильного приложения.

Во введении прописываются актуальность темы, написана цель и задачи.

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

Во второй главе разрабатывается архитектура мобильного приложения, описываются сценарии использования, проводится функциональное и логическое моделирование мобильного приложения и разрабатывается дизайн мобильного приложения.

Третья глава содержит выбор инструментов реализации мобильного приложения, программную реализацию требуемых функций и тестирование мобильного приложения.

В заключении представлены результаты выполнения выпускной квалификационной работы.

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

Бакалаврская работа состоит из 40 страниц текста, 14 рисунков, 3 таблиц, 25 источников и 3 листингов.

Abstract

The title of the bachelor's thesis is "Development of a hybrid mobile application for monitoring the readings of household meters".

The research is devoted to developing hybrid application for monitoring the readings of household meters.

When doing a research, the hybrid mobile application development technology was analyzed and selected, a hybrid mobile application was developed, the mobile application was implemented and tested.

The introduction reveals the relevance of the research and gives a brief description of the work done.

The first chapter the subject area of the study is considered, the requirements for a mobile application are formulated, and the analysis of analogues is carried out.

The second chapter the architecture of the mobile application is being developed, usage scenarios are described, functional and logical modeling of the mobile application is carried out and the design of the mobile application is developed.

The third chapter contains the selection of tools for implementing a mobile application, software implementation of the required functions and testing of a mobile application.

In conclusion, the conclusions of the entire work are drawn.

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

The volume of the bachelor's thesis is 40 pages, it also contains 14 figures, 3 tables, 3 code listings, a list of 25 references and 3 listing.

Оглавление

Введение.....	5
Глава 1 Постановка задачи на разработку гибридного мобильного приложения	7
1.1 Постановка задачи.....	7
1.2 Формирование требований к мобильному приложению	8
1.3 Сравнительный анализ используемых аналогов.....	11
Глава 2 Проектирование мобильного приложения	13
2.1 Выбор архитектуры мобильного приложения	13
2.2 Разработка логической модели мобильного приложения.....	15
Глава 3 Реализация гибридного мобильного приложения для контроля показаний бытовых счетчиков.....	20
3.1 Выбор средств разработки мобильного приложения	20
3.2 Реализация гибридного мобильного приложения	23
3.3 Тестирование мобильного приложения.....	26
Заключение	36
Список используемой литературы	38

Введение

Важнейшая обязанность каждого пользователя коммунальных услуг – их своевременная оплата. Управляющие компании, предоставляющие коммунальные ресурсы, начисляют стоимость пользования расходуемыми ресурсами, такими, как газ, электричество, вода, на основании либо общедомовых приборов учёта, либо индивидуальных приборов учёта, расположенных в квартире каждого собственника. Чаще всего используется именно второй вариант: «счётчики» стоят практически в каждой квартире, за редкими исключениями.

Для корректного начисления оплаты каждый собственник жилья должен своевременно передавать в управляющую компанию показания прибора учёта. Осуществляется это различными способами: посредством специальных отрывных бланков, по телефону, через сайт или электронную почту управляющей компании.

В случае наличия у компании сайта, пользователь, как правило, может отслеживать потребление коммунальных услуг и планировать платежи. Однако не все управляющие и ресурсоснабжающие компании предоставляют такой функционал.

Таким образом, актуальность выбранной темы заключается в обеспечении потенциального пользователя универсальным программным средством контроля расхода коммунальных услуг. Использование приложения позволит сократить время, затрачиваемое на фиксацию показаний, и упростит их передачу в управляющую компанию.

Объектом бакалаврской работы являются мобильные приложения для контроля показаний бытовых счетчиков.

Предмет исследования – гибридное мобильное приложение для контроля показаний бытовых счётчиков.

Цель бакалаврской работы – разработка гибридного мобильного приложения для контроля показаний бытовых счётчиков.

Для достижения цели работы необходимо выполнить следующие задачи:

- проанализировать и выбрать технологии разработки гибридного мобильного приложения для контроля показаний бытовых счётчиков;
- разработать гибридное мобильное приложение для контроля показаний бытовых счётчиков;
- выполнить реализацию и тестирование гибридного мобильного приложения.

Методы исследования – технологии разработки мобильных приложений, методы и технологии Web-дизайна.

Практическая значимость бакалаврской работы заключается в разработке гибридного мобильного приложения, которое может быть использовано для контроля показаний бытовых счётчиков.

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

Бакалаврская работа включает 40 страниц текста, 14 рисунков, 3 таблицы и 25 источников.

Глава 1 Постановка задачи на разработку гибридного мобильного приложения

1.1 Постановка задачи

На сегодняшний день практически каждый житель использует в своей квартире, доме или другом жилом объекте бытовые приборы учёта, или, говоря простыми словами, – счётчики. Счётчики предназначены для контроля расхода конечным потребителем коммунальных ресурсов: горячей и холодной воды, электроэнергии, газа.

Гибридное приложение – это программное приложение, которое объединяет в себе два приложения: нативное и Web. Для создания гибридного приложения необходимо загрузить Web-приложение и нативное приложение на устройство.

Преимущества гибридного приложения:

- более низкая цена разработки;
- быстрая реализация проекта;
- автономное обновление;
- кроссплатформенность.

Ресурсоснабжающие организации также выигрывают от установки счетчиков потребителям: расчёты становятся проще и прозрачнее, необходимость в большом штате сотрудников снижается, а передача показаний учёта осуществляется самими потребителями. Разумеется, последний пункт требует периодического контроля со стороны ресурсоснабжающей организации, поскольку потребитель может самостоятельно подать заведомо неверные показания, либо прибор учёта может быть неисправен.

Для передачи показаний бытовых счётчиков чаще всего используются формы в личных кабинетах на сайтах ресурсоснабжающих организаций и управляющих компаний. Пользователь в определённые даты каждого месяца

фиксирует показания бытового прибора учёта, вводит в форму на сайте необходимую информацию, и на основании полученной информации организация-поставщик ресурсов формирует счёт к оплате.

Поскольку каждая организация использует свой собственный сайт с отдельными формами и учётными записями, для потребителя такой способ хоть и удобнее, чем отправление показаний в «бумажном» виде, но недостаточно комфортен в плане упрощения взаимодействия с коммунальными службами [2].

Для решения проблемы своевременной передачи показаний приборов учёта предлагается разработать мобильное приложение, с помощью которого пользователь сможет отправлять показания приборов учета, контролировать оплату счетов, иметь доступ к истории потребления и при необходимости возможность связаться с экстренными службами управляющих организаций (сантехники, электрики и т.д.).

1.2 Формирование требований к мобильному приложению

Для формирования требований к мобильному приложению мы будем использовать классификацию FURPS+.

Данная классификация требований разработана и успешно применяется для создания приложений и информационных систем в настоящее время [1], [14].

Основа любого приложения или системы, согласно классификации FURPS+, это её функционал, то есть то, для чего предназначена система, и какие возможности в ней реализованы.

На основании функциональных требований строятся диаграммы вариантов использования.

Требования к проектируемому приложению представлены в таблице 1.

Приоритет реализации конкретных требований приложения представлен значениями от 1 до 5, где 1 – наивысший приоритет, 5 – наименьший [3].

Таблица 1 – Требования к мобильному приложению

№	«Требование	Приоритет	Описание
Functionality			
1	Передача показаний приборов учёта	1	Пользователь должен иметь возможность ввода показаний за коммунальные услуги (холодное водоснабжение, горячее водоснабжение, электроэнергия, газ, отопление) и передачу их в соответствующие управляющие компании
2	История показаний приборов учёта	1	Пользователь должен иметь возможность просмотра ранее внесённых показаний приборов учета
3	Редактирование имеющихся приборов учёта	1	Пользователь должен иметь возможность добавления новых приборов учёта, редактирования имеющихся и удаления неактуальных счётчиков
4	Контроль оплаты счетов	1	Пользователь должен иметь возможность контроля факта оплаты счетов по услугам за каждый месяц
5	Контроль тарифов услуг	2	Пользователь должен иметь возможность ввода и редактирования тарифов на коммунальные ресурсы для предварительного расчёта их стоимости за месяц
6	Предварительный расчёт стоимости услуг	2	Пользователь должен иметь возможность рассчитать предполагаемую стоимость услуг за текущий месяц на базе тарифов управляющей компании» [14]

Продолжение таблицы 1

Usability			
1	Удобный и понятный интерфейс	1	Каждое действие должно быть интуитивно понятным, используются стандартные жесты для управления приложением и логичные пункты в меню
2	Комфортная цветовая гамма приложения	4	Цветовая гамма должна быть контрастной, но не напрягать глаза пользователя при длительной работе
3	Читабельные и понятные шрифты	5	Весь текст должен быть удобен для чтения пользователем
Reliability			
1	Доступность системы 24/7	2	Приложение (хотя бы частично) должно быть доступно клиенту вне зависимости от внешних факторов (работа сервера, баз данных, интернет-соединение)
Performance			
1	Время запуска приложения не более 3-х секунд	3	-
2	Время отклика приложения на действия пользователя не более 1 секунды	2	-
Supportability			
1	Адаптивная вёрстка приложения для корректного отображения на различных устройствах	2	Элементы интерфейса должны одинаково располагаться и быть доступными на различных устройствах вне зависимости от разрешения и соотношения сторон экрана
2	Корректная работа функционала на различных устройствах	1	Приложение должно стабильно работать на любых устройствах, соответствующих минимальным системным требованиям
3	Поддержка операционной системы iOS	1	Приложение должно корректно работать на операционной системе iOS
4	Поддержка операционной системы Android	1	Приложение должно корректно работать на операционной системе Android

Продолжение таблицы 1

Дополнительные условия			
1	Система push-уведомлений	4	Приложение должно уведомлять пользователя о необходимости подачи показаний и оплате имеющихся счетов

Таким образом, разрабатываемое приложение имеет 6 функциональных требований, которые определяют список основных возможностей приложения и приоритеты в их реализации [24].

1.3 Сравнительный анализ используемых аналогов

Для целесообразности разработки нового приложения необходимо проверить, если ли среди используемых аналогов то, что подходит под сформулированные требования. Наиболее популярными решениями в области контроля показаний приборов бытового учёта являются приложения «DomoMeter», «Мои Счётчики», «Фотосчетчик ЖКХ».

Ключевая функция разрабатываемого приложения – передача показаний напрямую в компании ЖКХ – реализована только в отдельных приложениях каждой отдельной управляющей компании, что идёт вразрез с концепцией приложения «всё в одном». Хорошими альтернативами разрабатываемому приложению могли бы стать приложения «DomoMeter» и «Фотосчетчик ЖКХ», но они поддерживают только одну операционную систему и не обладают частью функционала. Необходимый функционал можно было бы добавить отдельными модулями, однако ни одно из указанных приложений не предоставляет открытый исходный код для внесения дополнительных изменений.

Проведём сравнительный анализ указанных приложений для выявления целесообразности разработки собственного мобильного приложения (таблица

2). Сравнительный анализ конкретных требований приложения представлен значениями от 0 до 5, где 0 – полное несоответствие требованиям, 5 – полное соответствие требованиям.

Таблица 2 – Сравнительный анализ аналогов

Название Требования	DomoMeter	Мои Счётчики	Фотосчётчик ЖКХ
Передача показаний приборов учёта в УК	0	2	1
История показаний приборов учёта	4	5	3
Редактирование приборов учёта	5	4	5
Несколько ресурсов в одном приложении (вода, газ, электричество)	4	3	5

Таким образом, не существует функционального альтернативного приложения, соответствующего вышеуказанным требованиям. В таком случае разработка собственного мобильного приложения имеет актуальность и целесообразность.

Выводы по главе 1

Для проектируемого приложения сформированы требования по методологии FURPS+, указан приоритет реализации каждого требования. Для увеличения охвата аудитории пользователей мобильного приложения было решено разработать мобильное приложение, работающее на операционных системах Android и iOS.

Также был проведён анализ аналогов, показавший отсутствие на рынке готового приложения, соответствующего предъявляемым требованиям. В ходе анализа были выявлены ключевые функции приложения, отличающие его от аналогов.

Глава 2 Проектирование мобильного приложения

2.1 Выбор архитектуры мобильного приложения

Исходя из обозначенных требований, разработаем архитектуру мобильного приложения.

Функционал приложения подразумевает использование модульной архитектуры, поскольку приложение имеет множество функциональных требований, не зависящих друг от друга.

Основная функция приложения – передача показаний приборов учёта.

То есть, проектируемое мобильное приложение должно корректно взаимодействовать с внешними Web-приложениями управляющих компаний (личными кабинетами) [13], [18].

Для реализации учёта истории показаний необходимо также где-то хранить эти данные и загружать на устройство пользователя при необходимости [5], [7].

Для реализации части функционала предлагается использовать сервер, обрабатывающий запросы пользователей.

Таким образом, мы формируем трёхзвенную архитектуру «клиент-сервер», состоящую из мобильного приложения, сервера-обработчика запросов, промежуточной базы данных, используемой сервером и конечных Web-приложений поставщиков коммунальных услуг [15].

Схема архитектуры мобильного приложения представлена на рисунке 1.

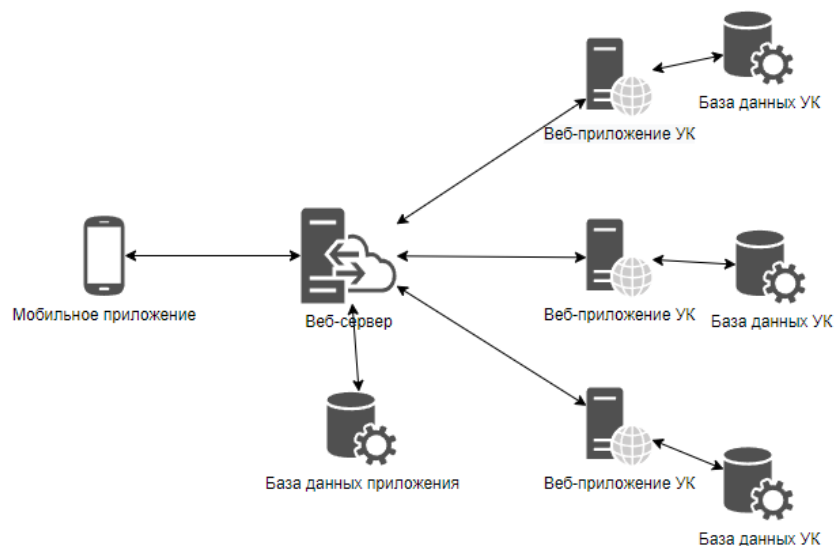


Рисунок 1 – Архитектура мобильного приложения

Ещё одним важным требованием к мобильному приложению является поддержка на устройствах под управлением операционных систем Android и iOS [21], [22]. Выборка по использованию ОС для планшетов и смартфонов за последние 10 лет [4] представлена на рисунке 2.

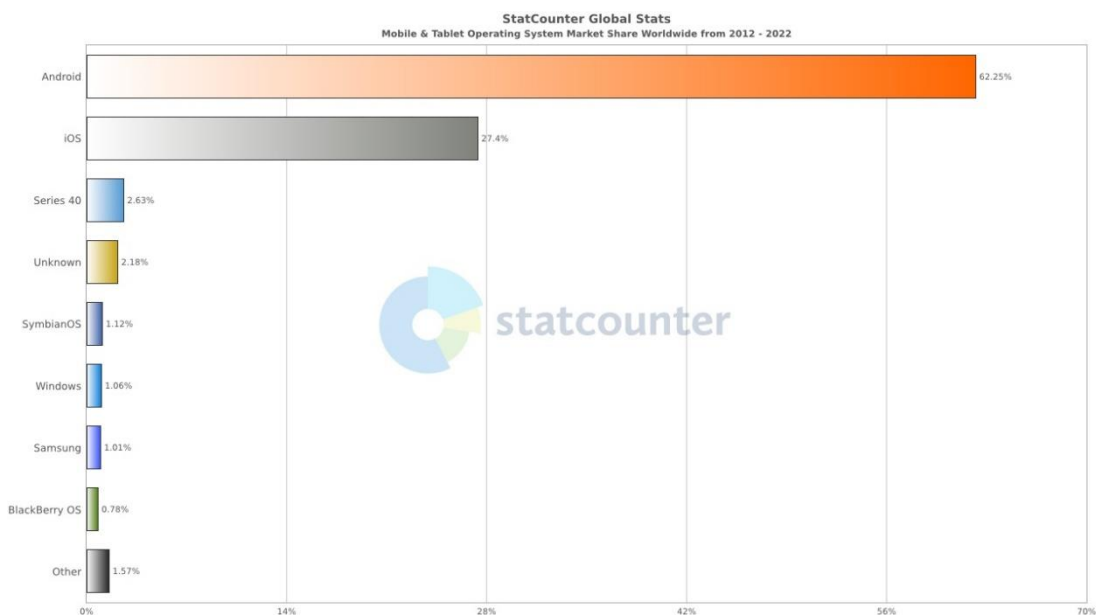


Рисунок 2 – Выборка по использованию ОС для мобильных приложений

Реализация этого требования возможна несколькими вариантами:

- разработка двух независимых нативных приложений под каждую операционную систему;
- реализация Web-приложения, работа которого не зависит от используемой ОС, а только от браузера;
- разработать гибридное приложение для корректного взаимодействия пользователя с сервисом вне зависимости от используемой операционной системы.

Гибридное приложение объединяет достоинства нативного подхода и Web-приложения. Часть функционала приложения реализуется средствами устройства, а часть – Web-приложением на сервере, и отображается на устройстве пользователя через формы WebView [16].

Таким образом, Web-сервер в проектируемой трёхзвенной архитектуре будет выполнять не только роль переадресации запросов и доступа к базе данных, но и участвовать в формировании отображаемых сущностей, с которыми взаимодействует пользователь.

Мобильное приложение обеспечивает вызов нативных функций, кэширование информации и обмен данными между устройством пользователя и Web-сервером [25].

2.2 Разработка логической модели мобильного приложения

Логическая модель приложения – это иллюстрация логической взаимосвязи всех компонентов приложения, показывает отношения, а также возможные варианты взаимодействия компонентов. Кроме этого, она определяет типы объектов, существующие в приложении [12].

Для создания логической модели и понимания взаимодействия пользователя с приложением необходимо построить диаграмму вариантов использования (рисунок 3).

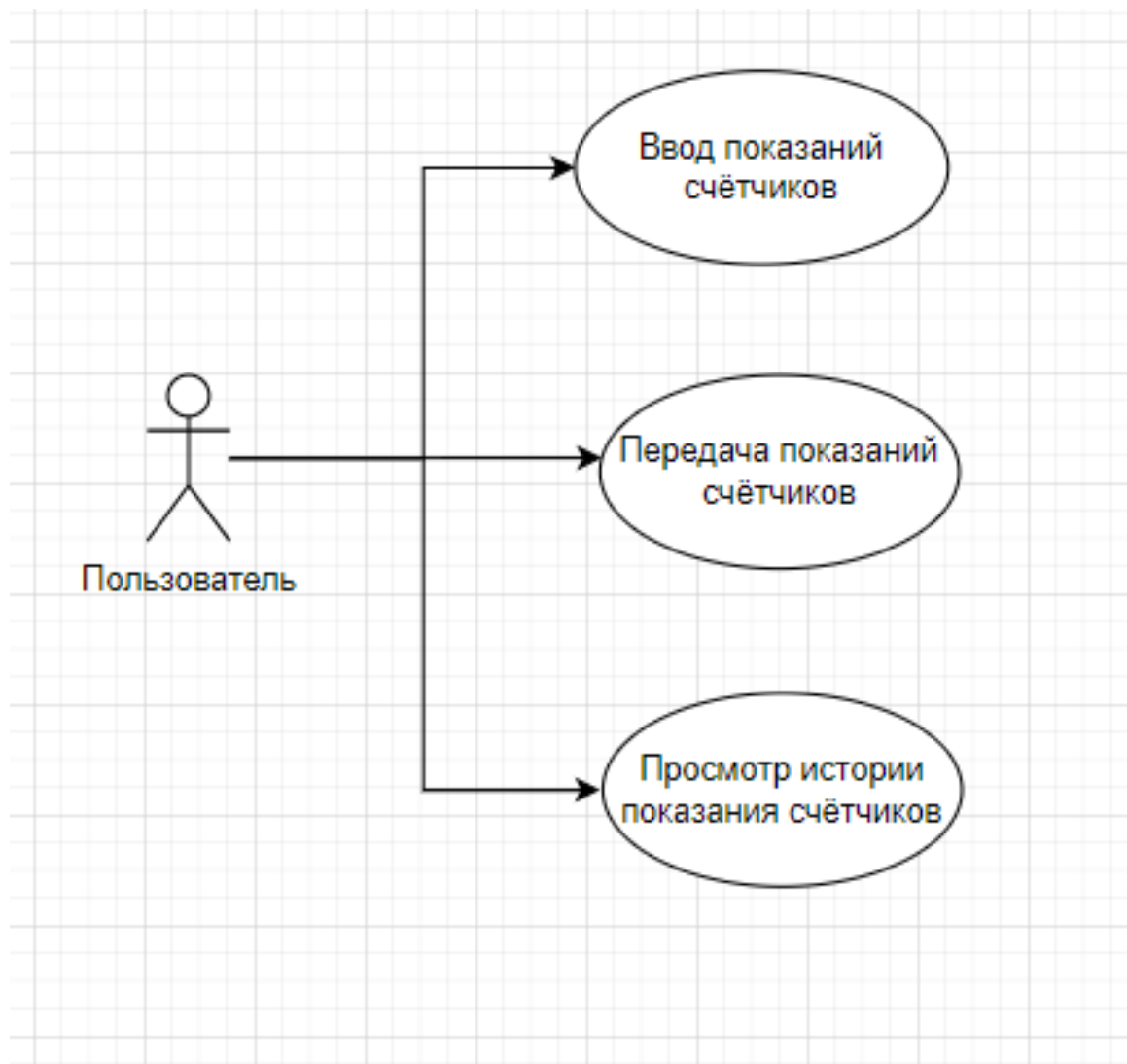


Рисунок 3 – Диаграмма вариантов использования приложения

С приложением взаимодействует единственный пользователь, соответственно, все возможные функции инициируются им. Описание сценариев, указанных на схеме, подробно представлено в предыдущем параграфе [6]. Опираясь на диаграмму вариантов, была создана диаграмма классов мобильного приложения, которая представлена на рисунке 4.

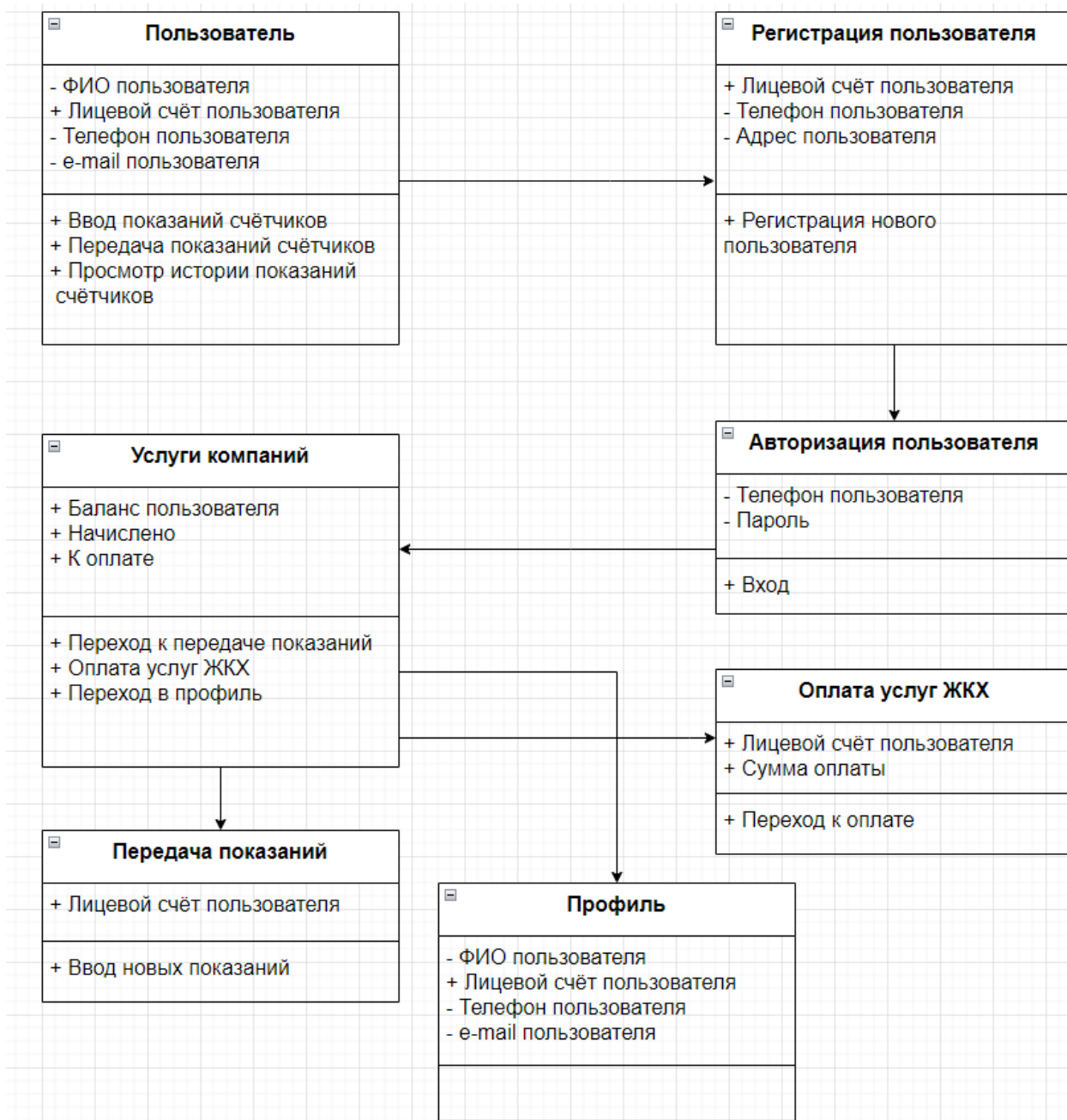


Рисунок 4 – Диаграмма классов UML пользователя

Главными графическими элементами диаграммы являются: интерфейсы зависимости между ними, а также компоненты [8], [9], [19]

Разработанные диаграммы компонентов представлены на рисунках 5, 6.

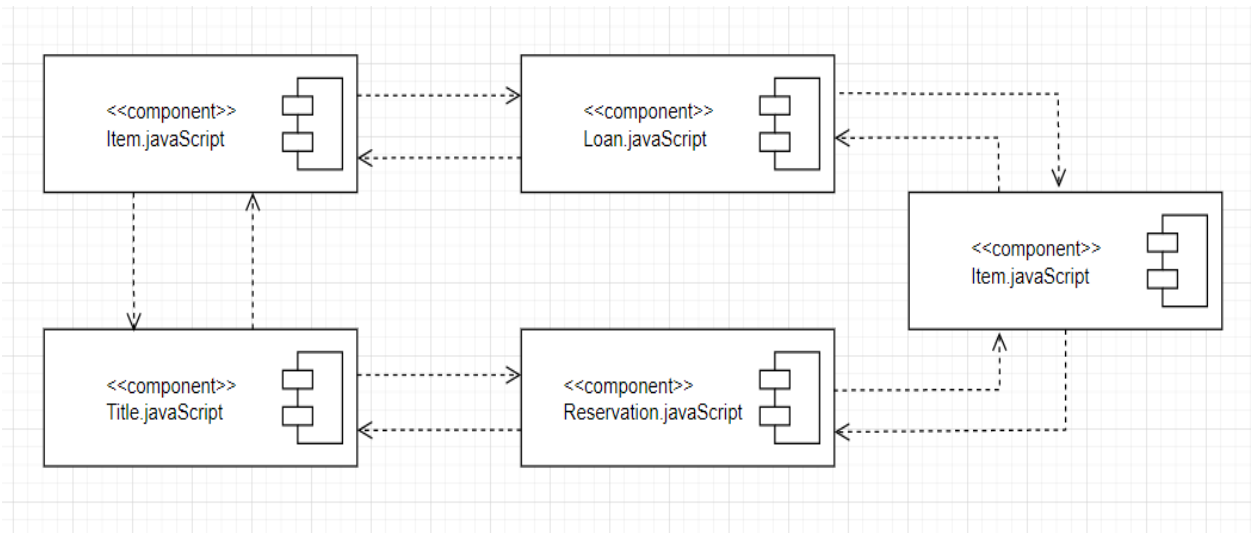


Рисунок 5 – Компонентная модель кода на JavaScript

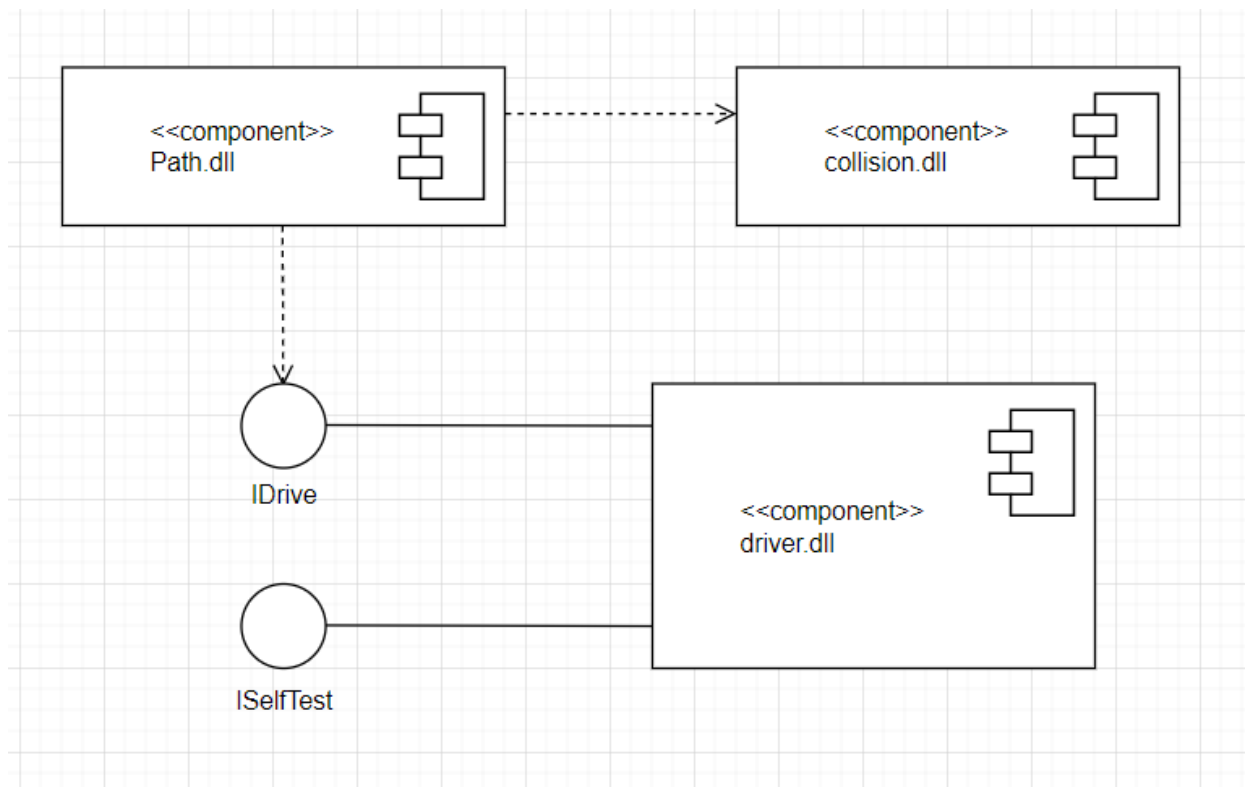


Рисунок 6 – Компонентная модель исполняемой версии приложения

Диаграмма компонентов дает возможность сформировать архитектуру будущей системы, обозначив соотношения программных компонентов.

Выводы по главе 2

Вторая глава ВКР посвящена описанию архитектуры и моделированию разрабатываемого мобильного приложения.

Разработана логическая модель приложения, которая является иллюстрацией логической взаимосвязи всех компонентов приложения, показывает отношения, а также возможные варианты взаимодействия компонентов.

В ходе работы была разработана трёхзвенная архитектура «клиент-сервер», выбран вариант реализации поддержки нескольких операционных систем в форме гибридного приложения, описаны сценарии использования приложения.

Также были разработаны функциональная диаграмма использования, описывающая взаимодействие пользователя с приложением и логическая схема взаимодействия компонентов.

Глава 3 Реализация гибридного мобильного приложения для контроля показаний бытовых счетчиков

3.1 Выбор средств разработки мобильного приложения

Гибридное приложение сочетает в себе функции нативного приложения под определённую платформу и возможности Web-приложений, используя комбинации нативных элементов и областей WebView. Существуют различные платформы для разработки гибридных приложений, рассмотрим некоторые из них [17].

Xamarin. Фреймворк, разработанный компанией Microsoft для разработки гибридных приложений под Android, iOS и Windows Phone. Основным языком платформы – C#. В своём составе фреймворк имеет обширные библиотеки классов для использования нативных элементов операционных систем iOS и Android, компиляторы, собственную IDE. Также поддерживает работу через Visual Studio через плагин. Является частью платформы .NET, что находит отражение как в совместимости с другими продуктами платформы, так и в особенностях лицензирования – использование Xamarin бесплатно, в том числе и для коммерческого использования. Согласно заявлениям Microsoft, Xamarin позволяет использовать до 75-90% общего кода между системами.

React Native. Ядром платформы является библиотека React для JavaScript. Использование фреймворка позволяет создавать приложения для Android и iOS на основе одной и той же кодовой базы. React Native выпущен компанией Facebook в 2015 году, на его базе в дальнейшем были разработаны приложения Facebook на мобильные устройства, а также другие популярные приложения Meta: Skype, Instagram, Messenger. Кроме этого, существует ответвление фреймворка под названием React Native for Web, которое позволяет запускать приложения, разработанные с использованием React

Native, как Web-приложения. Примеры использования последнего решения – Twitter и Uber.

Cordova/PhoneGap. Чтобы разобраться в том, что из себя представляют эти платформы, необходимо обратиться к истории их создания. PhoneGap был разработан в 2009 году канадским стартапом Nitobi, как open-source платформа для создания гибридных приложений на базе Web-технологий. Затем в 2011 году компания Nitobi вместе с её проектами была приобретена Adobe. Платформу передали Apache Foundation для дальнейшей разработки, в результате чего она была переименована в Cordova. Таким образом, разработка разделилась на две ветки, имеющие общее происхождение. Среда разработки Apache называется Cordova, а Adobe продолжили развитие под названием PhoneGap. Однако, в 2020 году поддержка PhoneGap была прекращена, оставив только одну ветку разработки. Основывается Cordova также на JavaScript, позволяет создавать мобильные и Web-приложения, но имеет меньшую поддержку и инфраструктуру, чем аналоги, что не мешает использовать Cordova в качестве движка для других фреймворков.

ionic. Данный фреймворк как раз основывается на базе Cordova и предлагает уже полноценную среду разработки со своим набором инструментов, облегчающих внесение изменений в код и перенос между платформами. Из особенностей можно выделить no-code конструктор для UI приложения (на базе HTML), функции обновления приложения, уже размещённого в AppStore (позволяет вносить критические изменения вроде устранения ошибок безопасности или работы вне плановых обновлений), автоматическая генерация всех иконок приложения на базе шаблонов. Однако Ionic больше подходит именно для разработки мобильных приложений, Web-приложения на нём писать сложнее.

Flutter. Платформа разработки мобильных, десктопных и Web-приложений, разработанная и поддерживаемая Google. Базируется Flutter на языке программирования Dart, который позиционируется как замена устаревшему JavaScript [11]. Основу Flutter составляют виджеты и

взаимодействие с ними. Причём все виджеты являются частью приложения, а не платформы, что обеспечивает высокую производительность в работе приложения. Flutter имеет обратную совместимость с нативными библиотеками и платформенными API Android и iOS, что добавляет гибкости в разработке приложений, позволяя перевести любое legacy-приложение на Flutter в короткие сроки. Также среда разработки обладает возможностями внесения изменений в приложение «на лету» благодаря функции HotReload. Такой подход позволяет мгновенно следить за изменениями в приложении и устранять ошибки без перезапуска самого приложения (таблица 3) [10].

Таблица 3 – Сравнение сред для разработки мобильного приложения

Платформа разработки	Язык программирования	Мобильные платформы	Удобство интерфейса	Работа с базами данных
Xamarin	C#	Android, iOS и Windows Phone	+	-
React Native	JavaScript	Android и iOS	+	+
Cordova/PhoneGap	JavaScript	Android и iOS	-	+
Ionic	HTML	Android, iOS и Windows Phone	-	+
Flutter	Dart	Android и iOS	+	-
Иторо	JavaScript	Android и iOS	+	+

Исходя из проведенного сравнения, более рентабельно использовать платформу React Native. В первую очередь, потому что у неё самый удобный пользовательский интерфейс, много доступного материала для обучения, удобный язык программирования. Поэтому для реализации гибридного мобильного приложения была выбрана платформа React Native.

В React Native программный код пишется на JavaScript и выполняется либо средствами мобильной платформы (iOS/Android), либо непосредственно в браузере. Для Android доступно больше сторонних реализаций и интерфейсов, поскольку Apple ограничивает использование внешних

движков. Этот факт необходимо учитывать при написании кода, поскольку некоторых сторонних элементов под iOS может не быть. Весь JavaScript-код приложения обрабатывается с помощью JSengine. Элементы интерфейса приложения выводятся на экран через mainthread. Нативные компоненты приложения, например клавиатура, обрабатываются при помощи отдельных потоков [20].

3.2 Реализация гибридного мобильного приложения

Для создания JavaScript-кода React Native использует Node.js, среду выполнения JavaScript. Сначала установим Homebrew, а затем – Node.js, выполнив в окне терминала следующее:

```
brew install node.
```

Затем с помощью homebrew установим watchman – сервис для отслеживания изменения и поиска файлов:

```
brew install watchman.
```

React Native использует его, чтобы отслеживать изменения кода и делать соответствующие правки.

Далее установим React Native Command Line Interface (CLI), используя npm:

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

Он использует Node Package Manager, чтобы вызвать и глобально установить CLI-инструмент; npm поставляется вместе с Node.js.

Дальше нужно перейти к папке, в которой мы хотим сохранить проект, и воспользуемся CLI-инструментом для его создания:

```
react-native init PropertyFinder
```

Эта строка создает начальный проект, в котором содержится всё необходимое для разработки и запуска приложения на React Native.

Взглянув на созданные папки и файлы, мы обнаружим папку `node_modules`, в которой находится фреймворк React Native. Файл `index.ios.js` – это макет приложения, созданный CLI-инструментом.

Далее добавим код, представленный в листинге 1.

Листинг 1 – Добавление единого стиля

```
var styles = React.StyleSheet.create({
text: {
  color: 'black',
  backgroundColor: 'blue',
  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");
}
}
```

Классы были добавлены в ECMAScript 6 (ES6). Поскольку JavaScript постоянно развивается, разработчики вынуждены ограничивать себя в используемых средствах ради сохранения совместимости со старыми системами или 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.

Теперь добавим поле ввода и кнопки.

Откроем файл SearchPage.js и введем следующий код сразу после закрывающего тега второго элемента Text (листинг 2).

Листинг 2 – Добавление полей ввода и кнопки

```
<View style={styles.flowRight}>
  <TextInput
    style={styles.searchInput}
    placeholder='+7 999 999 99 99' />
  <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.

Вернемся к файлу SearchPage.js и добавим строку под закрывающим тегом компонента TouchableHighlight, отвечающего за кнопку «Войти»:

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

Чтобы реализовать поиск, нужно обработать нажатие кнопки «Войти», создать необходимый API-запрос и предоставить пользователю визуальное подтверждение того, что запрос обрабатывается.

Откроем файл SearchPage.js, найдем constructor и обновим внутри него начальное состояние:

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

Чтобы добавить на страницу индикатор загрузки, перейдем к JSX, отвечающему за интерфейс поиска в return, и вставим под Image эту строку:

```
{spinner}
```

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);  
}
```

Представленный выше программный код является основой для реализации гибридного мобильного приложения.

3.3 Тестирование мобильного приложения

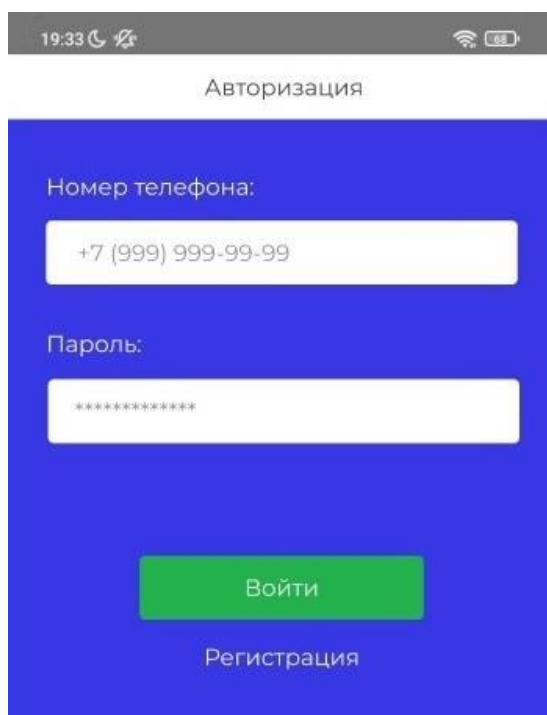
Для тестирования приложения использован метод функционального тестирования.

Функционального тестирования – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям [23].

С помощью данного метода мы проверим работоспособность нашего приложения по следующим пунктам:

- Запуск приложения;
- Работоспособность основного функционала приложения;
- Авторизация (по номеру телефона);
- Регистрация (по номеру телефона, лицевому счёту);
- Валидация обязательных полей;
- Редактирование данных в профиле пользователя.

Запустим приложение. Для входа в систему необходимо авторизоваться. Введем номер телефона, пароль и нажмем кнопку «Войти», как показано на рисунке 7.



19:33

Авторизация

Номер телефона:

+7 (999) 999-99-99

Пароль:

Войти

Регистрация

Рисунок 7 – Авторизация пользователя

Если пользователь входит в систему в первый раз, ему необходимо зарегистрироваться. Для этого необходимо перейти по кнопке «Регистрация» и в появившемся окне ввести: лицевой счет, номер телефона, адрес (рисунок 8).

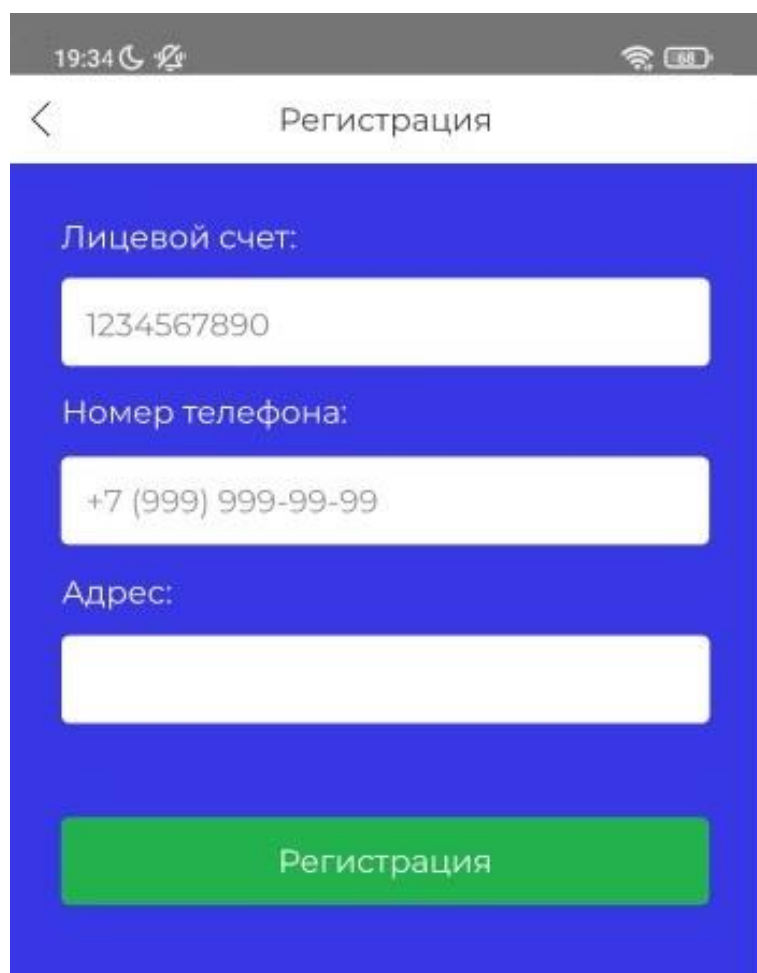
The image shows a mobile application interface for user registration. At the top, there is a status bar with the time 19:34, a location icon, and battery level indicators. Below the status bar is a navigation bar with a back arrow on the left and the title 'Регистрация' in the center. The main content area has a blue background and contains three white input fields. The first field is labeled 'Лицевой счет:' and contains the number '1234567890'. The second field is labeled 'Номер телефона:' and contains '+7 (999) 999-99-99'. The third field is labeled 'Адрес:' and is currently empty. At the bottom of the form is a large green button with the text 'Регистрация' in white.

Рисунок 8 – Регистрация пользователя

После ввода данных, необходимо нажать на кнопку «Регистрация» и у нас высветится окно с созданием пароля, где нужно указать пароль и повторить его. После создания пароля попадаем на вкладку «Авторизация», вводим номер телефона и созданный пароль (рисунок 9).

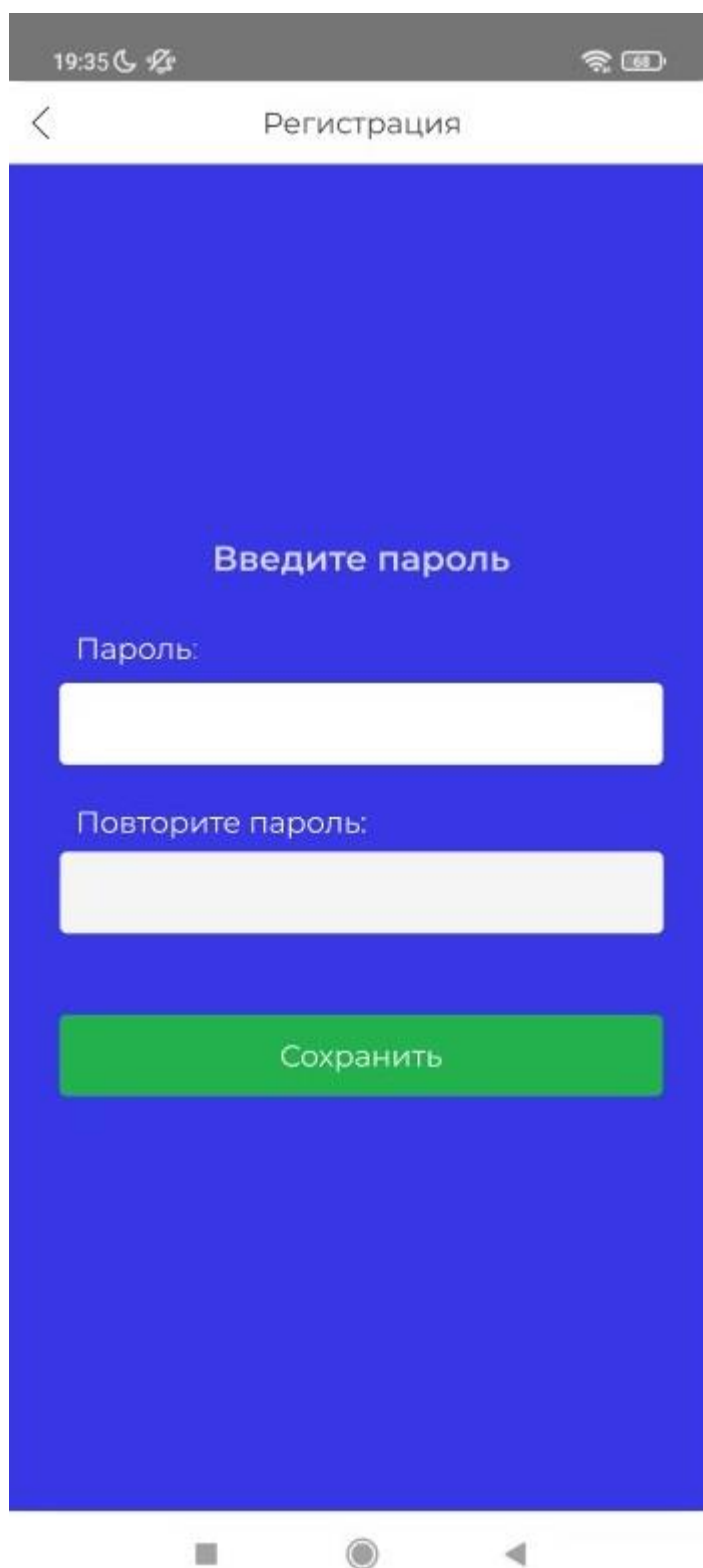


Рисунок 9 – Создание пароля

Попадаем на главную страницу, где показаны услуги компании, а именно: холодное водоснабжение, горячее водоснабжение и электроэнергия.

По каждому блоку мы можем увидеть: баланс, сколько начислено и сколько необходимо оплатить, а также по кнопке «Перейти к передаче показаний», мы переходим на новую страницу, где указываем новые показания (рисунки 10, 11).

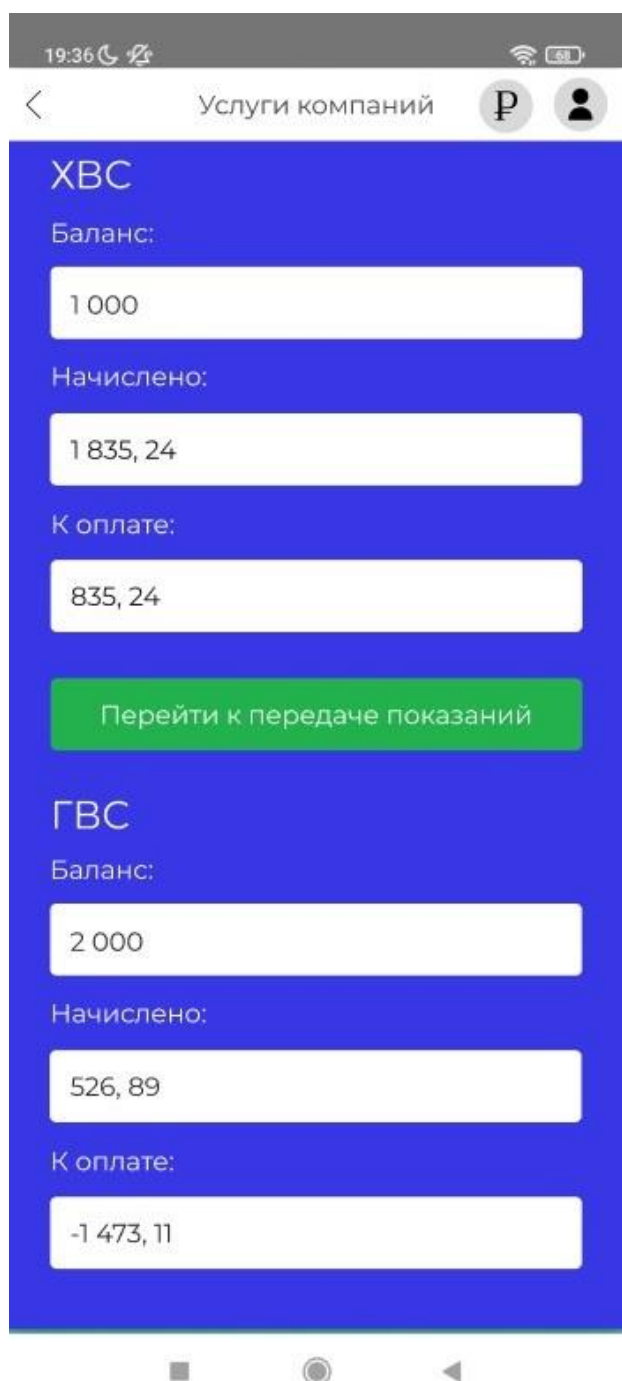


Рисунок 10 – Услуги компаний

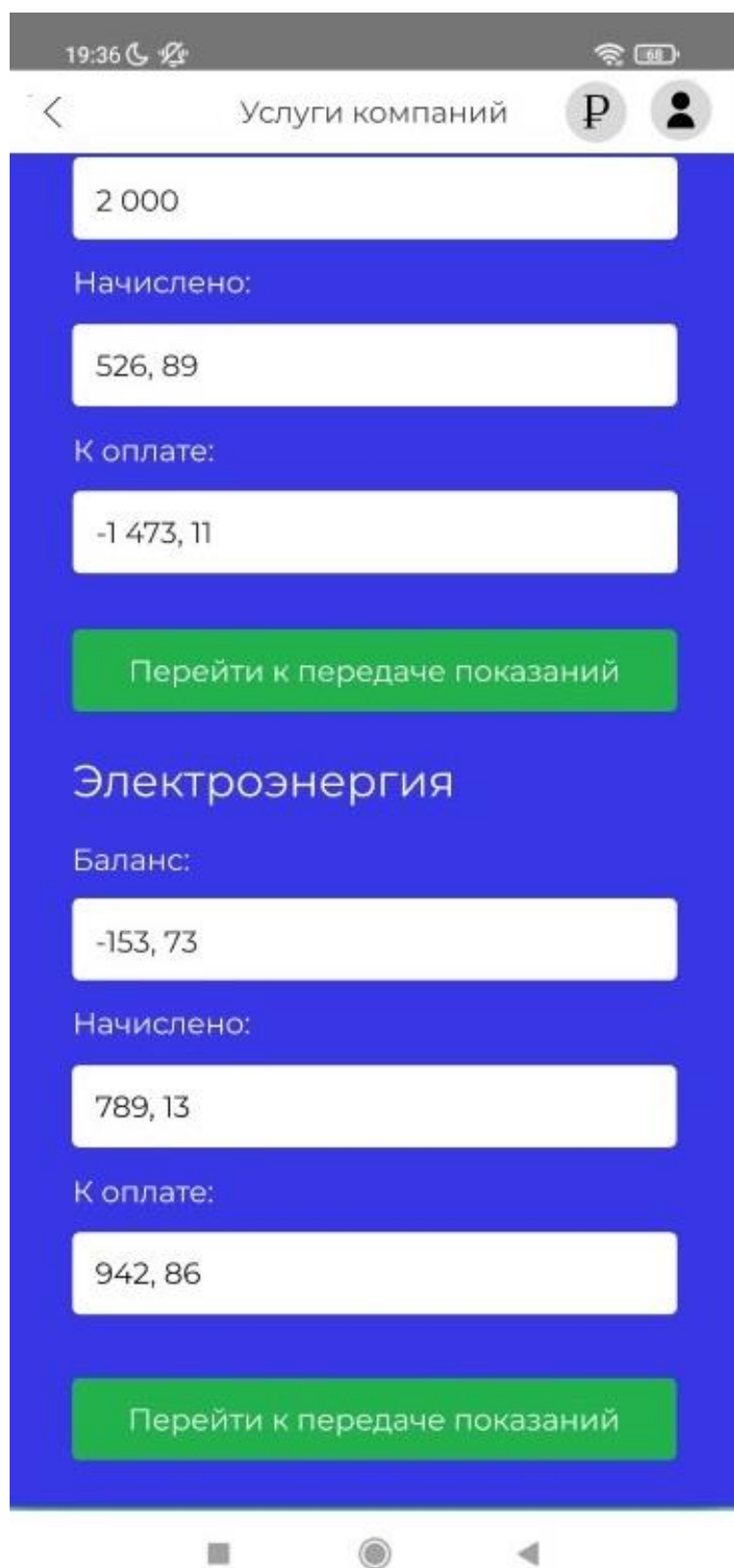
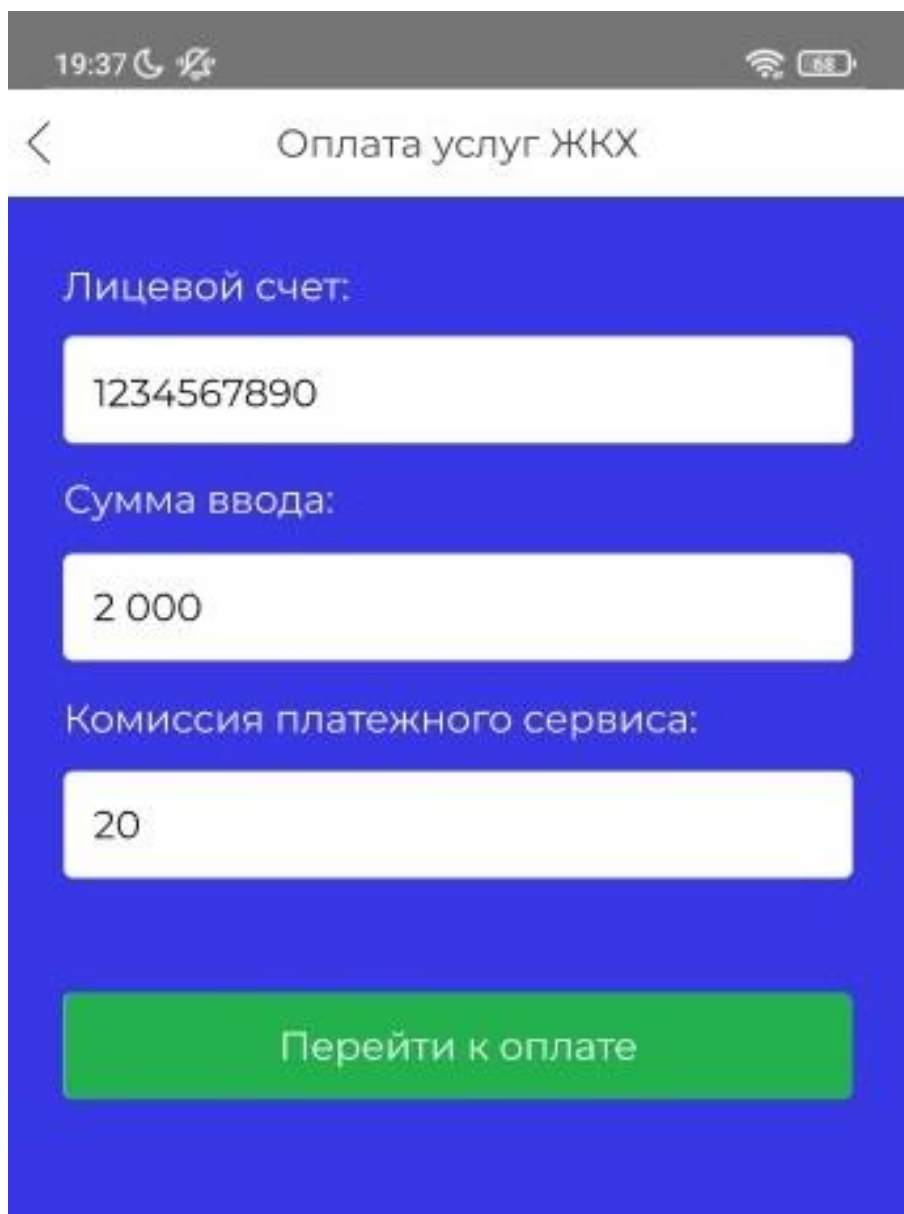


Рисунок 11 – Услуги компаний

Также сверху у нас есть две кнопки: «Оплата услуг ЖКХ» и «Профиль».

В «Оплате услуг ЖКХ» нужно ввести: лицевой счет и сумму. Также приложение высчитывает комиссию. Кнопка «Перейти к оплате» направит нас на приложение мобильного банка, где мы можем оплатить услуги ЖКХ (рисунок 12).



19:37

Оплата услуг ЖКХ

Лицевой счет:

1234567890

Сумма ввода:

2 000

Комиссия платежного сервиса:

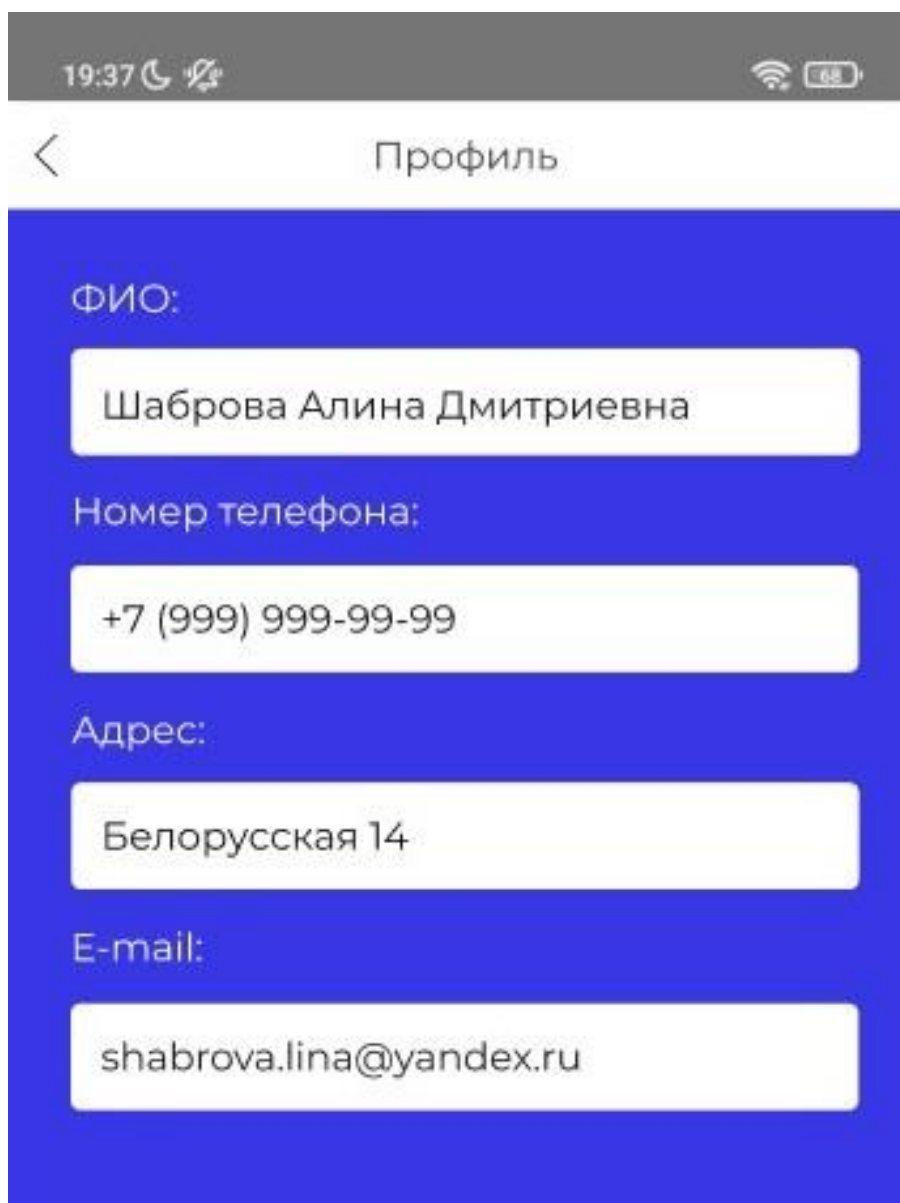
20




Перейти к оплате

Рисунок 12 – Оплата услуг ЖКХ

Также из главного меню мы можем перейти в «Профиль» по второй кнопке, которая находится справа сверху. Нажав на иконку «Профиль» мы

попадаем в новое окно, где лежат наши личные данные, а именно: ФИО, адрес, телефон и e-mail (рисунок 13).



19:37   

< Профиль

ФИО:
Шаброва Алина Дмитриевна

Номер телефона:
+7 (999) 999-99-99

Адрес:
Белорусская 14

E-mail:
shabrova.lina@yandex.ru

Рисунок 13 – Профиль пользователя

На главной странице в окне «ГВС» по кнопке «Перейти к передаче показаний», мы попадаем на страницу «Показания приборов учёта», здесь необходимо указать лицевой счёт и приложение выдаст информацию о счетчиках, а также здесь мы можем ввести новые показания (рисунок 14).



Рисунок 14 – Показания приборов учёта

Таким образом, подтверждена работоспособность приложения по выбранным пунктам.

Выводы по главе 3

В данной главе был проведён анализ средств разработки мобильных приложений, в результате которого был выбран наиболее подходящий фреймворк React Native.

С использованием указанного инструментария были реализованы функции мобильного приложения: передача новых показаний, оплата услуг ЖКХ.

С помощью JavaScript-кода было реализовано гибридное мобильное приложение.

Для разработки использовался фреймворк React Native, который в свою очередь использует Node.js - среду выполнения JavaScript.

Так же было протестировано мобильное приложение методом функционального тестирования.

Подтверждена работоспособность приложения по выбранным пунктам.

Таким образом, гибридное приложение успешно прошло функциональное тестирование.

Заключение

Бакалаврская работа посвящена разработке гибридного приложения для контроля показаний бытовых счетчиков.

В ходе выполнения ВКР был проведён анализ предметной области, выявлены проблемы, присущие исследуемой области, выдвинуто предложение по решению выявленных проблем путём разработки мобильного приложения.

Для проектируемого приложения были сформированы требования по методологии FURPS+, указан приоритет реализации каждого требования. Для увеличения охвата аудитории пользователей мобильного приложения было решено разработать мобильное приложение, работающее на операционных системах Android и iOS.

Также был проведён анализ аналогов, показавший отсутствие на рынке готового приложения, соответствующего предъявляемым требованиям. В ходе анализа были выявлены ключевые функции приложения, отличающие его от аналогов.

Дальше была разработана трёхзвенная архитектура «клиент-сервер», выбран вариант реализации поддержки нескольких операционных систем в форме гибридного приложения, описаны сценарии использования приложения. Также были разработаны функциональная диаграмма использования, описывающая взаимодействие пользователя с приложением и логическая схема взаимодействия компонентов.

А также был проведён анализ средств разработки мобильных приложений, в результате которого был выбран наиболее подходящий фреймворк React Native.

С использованием указанного инструментария были реализованы функции мобильного приложения: передача новых показаний, оплата услуг ЖКХ.

С помощью JavaScript-кода было реализовано наше мобильное приложение. Для этого мы воспользовались React Native, который в свою очередь использует Node.js, среду выполнения JavaScript.

Также было протестировано мобильное приложение методом функционального тестирования. Проверили работоспособность нашего приложения по выбранным пунктам, приложение успешно прошло тестирование.

Задачи, определённые для достижения цели работы, были выполнены в полном объёме.

Цель бакалаврской работы была достигнута – разработано гибридное приложение для контроля показаний бытовых счётчиков.

В ходе работы над ВКР были углублены знания и умения проектирования приложений, получены практические навыки работы с React Native и JavaScript.

Мобильное приложение, полученное в результате ВКР, может быть доработано и использовано в качестве альтернативного метода контроля показаний счётчиков и их передачи в компании 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. 573с.
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 стр.
4. Гагарина Л. Г., Киселев Д. В., Федотова Е. Л. Разработка и эксплуатация автоматизированных информационных систем М.: ИД «ФОРУМ»; ИНФРА-М, 2012. 384с.
5. Голицина О.Л., Максимов Н.В., Попов И.И. Базы данных: Учебное пособие. – М.: Форум: ИНФРА-М, 2013. 352 с.
6. Грофф Д. SQL: Полное руководство / Д. Грофф, П. Вайнберг. / - К.: BHV, 2001. 816с.
7. Дейт Д. Введение в системы баз данных. М.: Издательский дом "Вильямс", 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. Ихтиар В.Ф. Сравнение кроссплатформенных фреймворков // Вестник магистратуры. 2018. №1-3 (76). URL: <https://cyberleninka.ru/article/n/sravnenie-kross-platformennyh-freymvorkov> (дата обращения: 03.04.2022).
11. Калиневич Н., Гильванов Р. Г. Разработка кроссплатформенных приложений на языке Dart при помощи фреймворка Flutter // Интеллектуальные технологии на транспорте. 2021. №4 (28). URL:

<https://cyberleninka.ru/article/n/razrabotka-kross-platformennyh-prilozheniy-na-yazyke-dart-pri-pomoschi-freymvorka-flutter> (дата обращения: 19.03.2022).

12. Колесов Ю. Б. Моделирование систем. Объектно-ориентированный подход: учебное пособие / Ю. Б. Колесов, Ю. Б. Сениченков. СПб.: БХВ-Петербург, 2012. 192 с.

13. Конноли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Конноли, К. Бегг, А. Стратчан. М.: Вильямс, 2000. 1093 с.

14. Корнеев И. К. Информационные технологии: учебное пособие. М.: Проспект, 2007. 224 с. 9.

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

16. Новожилова А. Азбука клиента. Мобильные приложения: нативные, html5, гибридные // CMSMagazine. URL: <http://www.cmsmagazine.ru/> (дата обращения 10.12.2021)

17. Русанова И.В. Анализ платформ для разработки гибридного мобильного приложения для систем iOS и Android // Актуальные проблемы авиации и космонавтики. 2017. №13 [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/analiz-platform-dlya-razrabotki-gibridnogo-mobilnogo-prilozheniya-dlya-sistem-ios-i-android> (дата обращения: 09.02.2022).

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

19. 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.

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

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.