МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра информационных технологий управления

Курсовая работа

Сбор показаний индивидуальных приборов учёта в многоквартирных домах и выставление счетов за потреблённые услуги

09.03.02 Информационные системы и технологии

Преподаватель_____В.С. Тарасов, ст. преподаватель __.__.20__

Обучающийся_____А.А. Сарайкин, 3 курс, д/о

Обучающийся_____Я.В. Солодовникова, 3 курс, д/о

Воронеж 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В обязанности практически всех людей входит оплата коммунальных услуг. Для этого необходимо каждый месяц передавать показания счетчиков в компании, отвечающие за предоставление таких услуг, и производить оплату в соответствии с квитанциями.

Раньше был доступен только один способ оплаты: через отделение банка или почты. Данный способ можно считать достаточно затратным по времени и усилиям, так как необходимо добраться до отделения банка, после чего дождаться своей очереди. Также стоит учитывать, что график работы отделений может быть неудобным и ограничивать возможности клиента. Например, большинство банков и почтовых отделений не работают по воскресеньям.

С распространением смартфонов появился еще один способ оплаты: через мобильное приложение банка. Клиенту не надо тратить время на очереди, ходить куда-либо и подстраиваться под график работы. Несмотря на эти преимущества, есть ряд недостатков. Например, нельзя в одном месте просмотреть историю передаваемых показаний, для внесения оплаты по нескольким квитанциям необходимо потратить много усилий, при заполнении реквизитов можно допустить ошибку и оплатить другой счет.

Возможен еще один способ оплаты: через специализированное приложение. Его преимущество, что в одном месте можно передать показания всех счетчиков и в одном месте произвести оплату для всех видов коммунальных платежей. Также можно просматривать историю и статистику, собранную по всем видам потребляемых услуг.

Данный проект направлен на создание приложения, которое позволит сократить время и количество действий на передачу показаний ИПУ и их оплату.

1 Постановка задачи

1.1 Постановка задачи

Целью данного курсового проекта является создание приложения для сбора показаний индивидуальных приборов учёта в многоквартирных домах и выставление счетов за потреблённые услуги.

Помимо этого, будут также доступны функции:

— Расчёт статистики по потреблённым услугам;

— Составление прогноза на потребление коммунальных услуг в следующем периоде.

1.2 Обзор аналогов

1.2.1 Квартплата+

Мобильное приложение, направленное на оплату ЖКУ, домофона, связи и других услуг, а также передачу показаний ИПУ.

Данное приложение не имеет онбординг экрана, сразу после входа пользователь видит экран авторизации. С экрана авторизации можно попасть на экран «О приложении» (Рисунок 1) с помощью кнопки в правом верхнем углу.

После авторизации осуществляется переход на экран «Мои услуги» (Рисунок 1). Сразу после этого осуществляется поиск платежей по адресу пользователя. По нажатию кнопки «Оплатить», можно произвести оплату выбранных счетов. Приложение не сообщает никакой информации, которая должна помочь в навигации. Интуитивно не понятно, куда нужно нажать пользователю, чтобы передать показания счетчиков. Это можно узнать только в чате поддержки, если задать соответствующий вопрос (Рисунок 2).

На главном экране, в строке адреса находится кнопка «Счетчики». Если нажать на нее, осуществляется переход на экран со списком счетчиков. При нажатии на конкретный счетчик открывается экран, на котором можно посмотреть информацию о счетчике и передать показания.

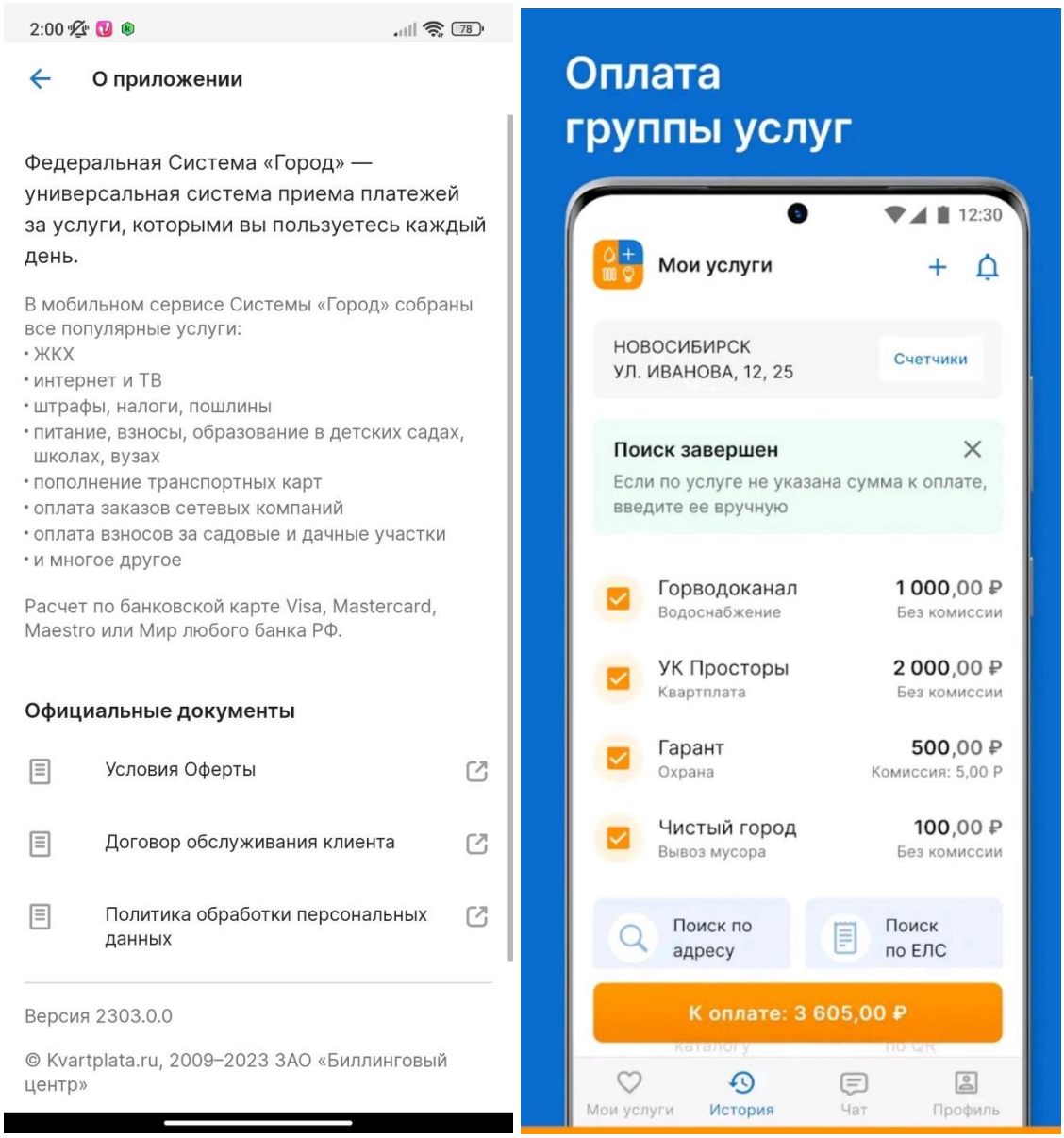


Рисунок 1 - Информационный и главный экраны

Для передачи показаний необходимо пройти через 3 экрана: Главный → Счетчики → Счетчик. От экрана «Счетчики» можно перейти только на экран одного конкретного счетчика. Соответственно, если пользователь передал показания счетчика, чтобы передать показания следующего, ему нужно вернуться на экран назад и затем снова перейти на экран необходимого счетчика.

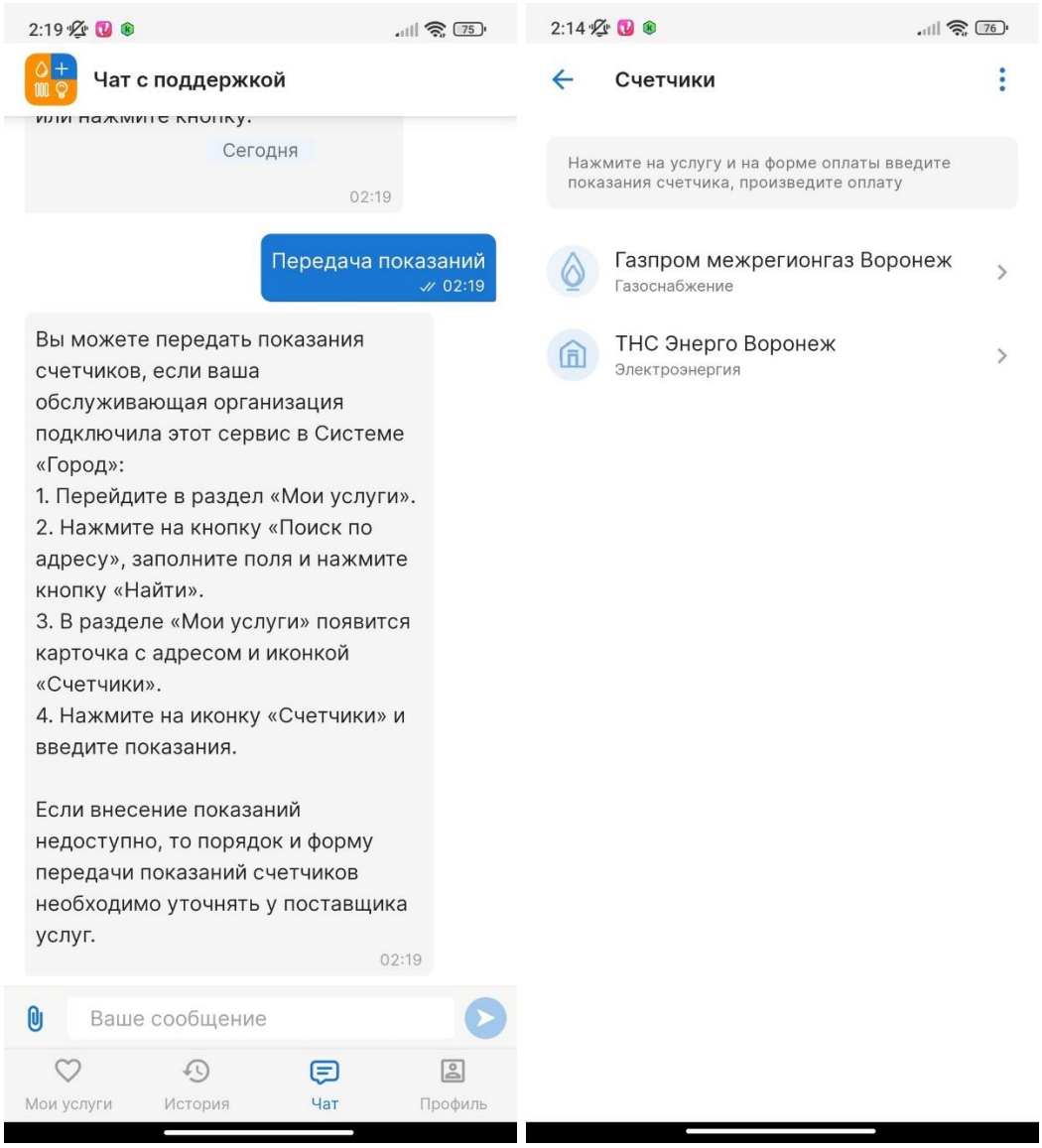


Рисунок 2 - Чат с поддержкой и экран со списком счетчиков

Преимущества приложения:

— Автоматический поиск счетов на оплату по адресу пользователя;

— Чат с поддержкой.

Недостатки приложения:

— Отсутствие информации о навигации в приложении даже при первом открытии;

— Для передачи показаний ИПУ необходимо совершить много действий.

В отличие от реализуемого проекта, Квартплата+ не обладает функциями просмотра статистики потребления коммунальных услуг или просмотра прогноза потребления на следующий период.

1.2.2 РВК.Услуги

Мобильное приложение для клиентов ГК «Росводоканал». Оно позволяет оплачивать ЖКУ и передавать показания ИПУ.

Данное приложение имеет онбординг экран, после которого пользователь переходит на экран авторизации (Рисунок 3). Предлагается выбор: войти как клиент или как подрядчик.

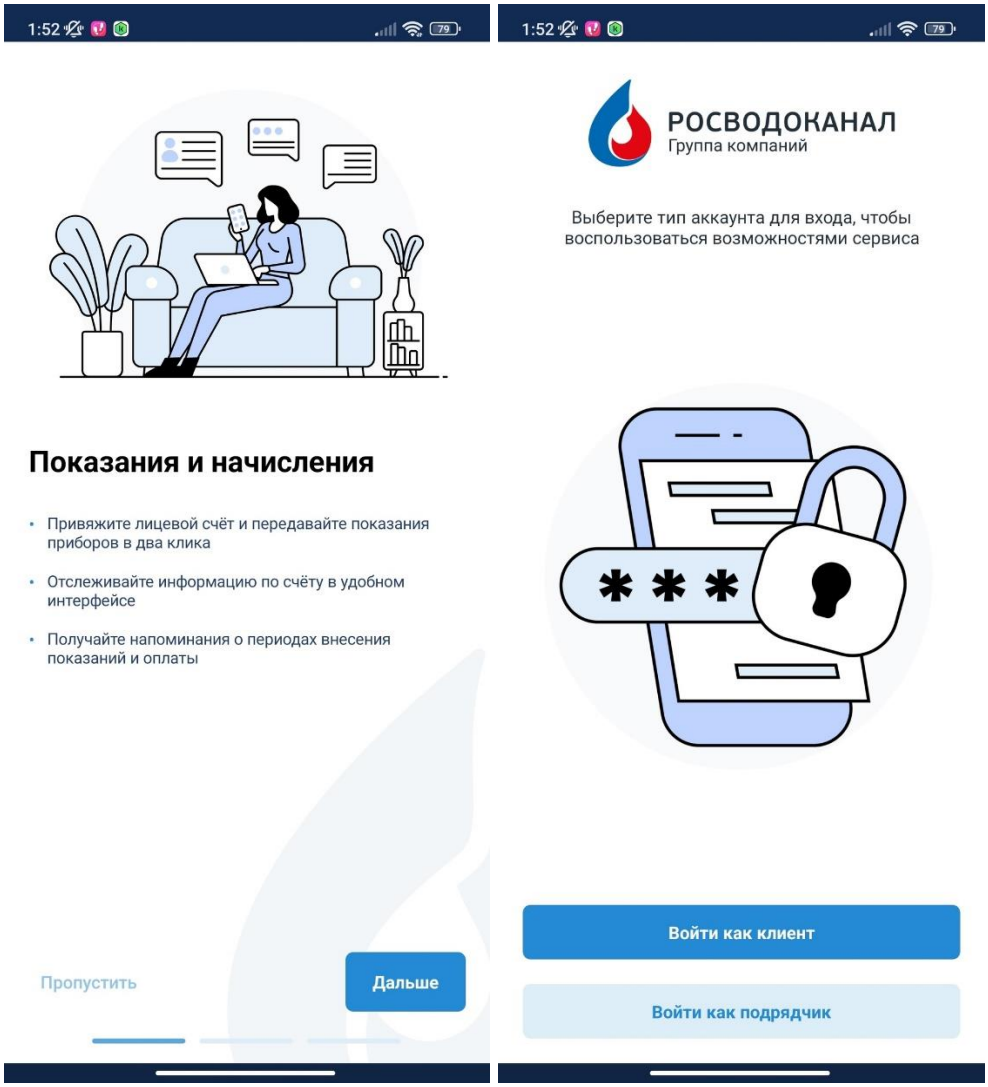


Рисунок 3 - Онбординг экран и экран авторизации

После авторизации осуществляется переход на экран «Счета» (Рисунок 4). На нем отображаются все лицевые счета, которые привязаны к

пользователю. При нажатии на кнопку «Управлять лицевым счетом» осуществляется переход на экран «Лицевой счет» (Рисунок 4).

Даже если у пользователя всего 1 лицевой счет, ему все равно необходимо на экране «Счета» выбирать конкретный лицевой счет каждый раз при запуске приложения. Таким образом, чтобы передать показания необходимо пройти 3 экрана: «Счетчики» → «Лицевой счет» → «Передать показания».

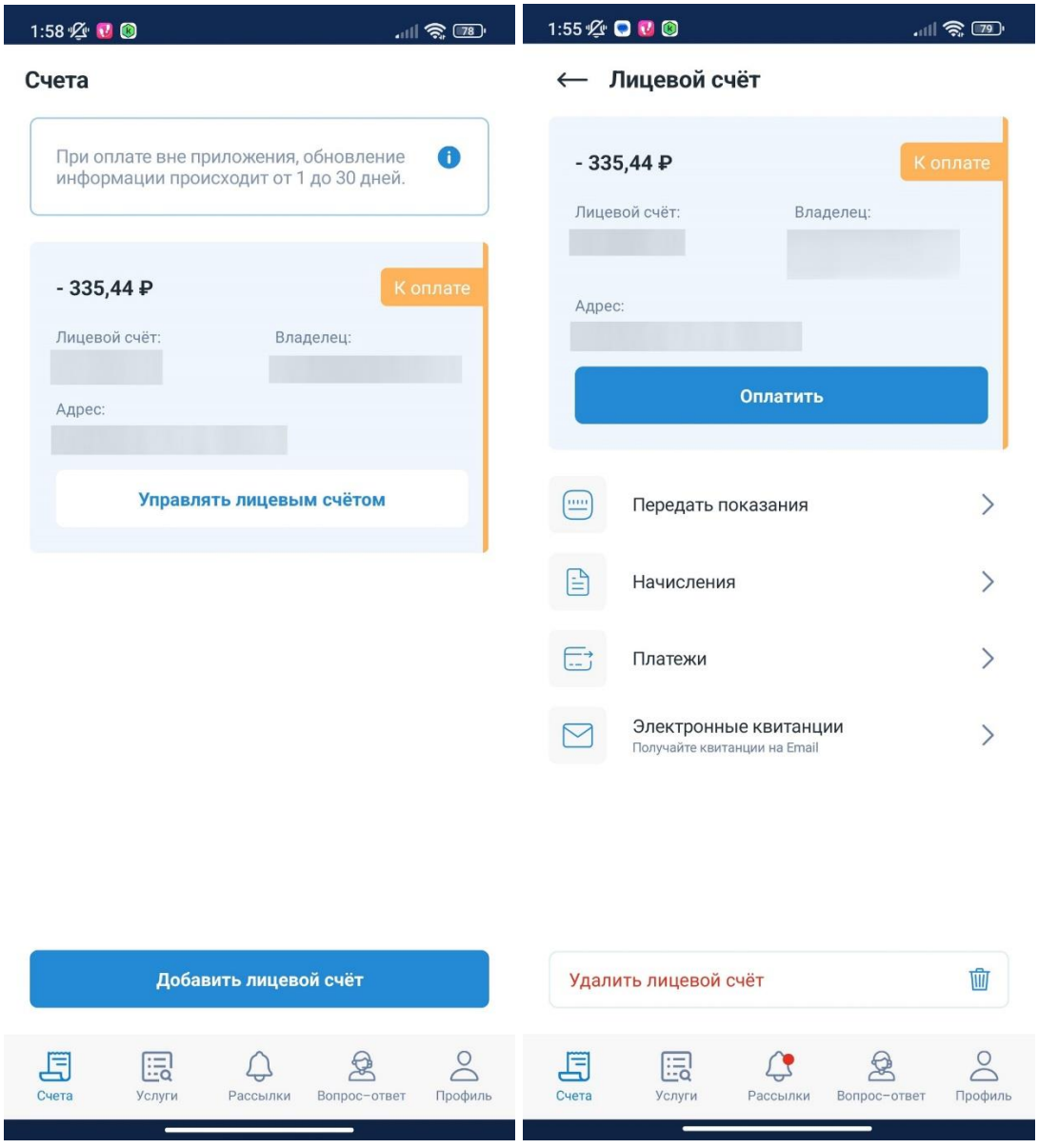


Рисунок 4 - Экраны «Счета» и «Лицевой счет»

С экрана «Лицевой счет» можно выполнить все доступные по нему операции: передача показаний, просмотр истории начислений, просмотр истории платежей, управление электронными квитанциями.

На экране «Услуги» можно посмотреть список доступных услуг и оформить необходимые. На экране «Рассылки» можно посмотреть новости, рассылки и уведомления.

Преимущества приложения:

— Возможность оформления заявок на определенные услуги;

— Возможность получения электронных квитанций на Email;

— Чат с поддержкой.

Недостатки приложения:

— Для передачи показаний ИПУ необходимо совершить много действий.

В отличие от реализуемого проекта, РВК.Услуги не обладает функциями просмотра статистики потребления коммунальных услуг или просмотра прогноза потребления на следующий период.

Подводя итог, можно выделить следующие недостатки, присущие многим мобильным приложениям для оплаты ЖКУ и передачи показаний ИПУ: для передачи показаний нужно сделать много действий. В реализуемом приложении будет необходимо совершить меньшее количество шагов для передачи показаний.

2 Анализ предметной области

2.1 Глоссарий

База данных (БД) — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. [1]

Индивидуальные приборы учета (ИПУ) — это приборы, которые учитывают личное потребление коммунальных ресурсов клиентом.

Жилищно-коммунальные услуги (ЖКУ) — услуги по поддержанию и восстановлению надлежащего технического и санитарно-гигиенического состояния зданий, сооружений, оборудования, коммуникаций и объектов коммунального назначения. [2]

Личный кабинет — это персональная страница пользователя, доступная после авторизации. [3]

Лицевой счет физического лица — тип учетной записи в реестрах налоговой, пенсионного фонда, банков, поставщиков коммунальных услуг и услуг связи. [4]

Тариф — установленная стоимость за единицу ресурса. [5]

Онбординг — приветственный экран, который пользователи видят после установки приложения или выполнения целевого действия на сайте. [6]

HTTP (Hypertext Transfer Protocol) запрос — способ взаимодействия между клиентом и сервером в сети Интернет. [7]

2.2 Сценарии пользователей

— Пользователь: Лидия Короткова, 58 лет.

Описание: владелица двухкомнатной квартиры в жилом комплексе, обладает общими навыками использования телефона.

Пользовательская история: Лидии необходимо передать показания ИПУ и произвести оплату счетов, за предоставляемые ЖКУ, с целью своевременного исполнения обязанностей потребителя ЖКУ в соответствии с договором.

— Пользователь: Иван Моисеев, 22 года.

Описание: владелец однокомнатной квартиры в жилом комплексе, обладает продвинутыми навыками в использовании мобильного телефона.

Пользовательская история: Ивану необходимо узнать прогноз потребления ЖКУ на следующий период, чтобы заранее планировать свой бюджет и траты на месяц.

— Пользователь: Мария Трунова, 30 лет.

Описание: владелица двухкомнатной квартиры, обладает продвинутыми навыками использования мобильного телефона.

Пользовательская история: Марии необходимо узнать свою статистику потребления ЖКУ, чтобы оценить объемы потребления и внести при необходимости коррективы.

2.3 Сценарии воронок конверсии

В приложении можно выделить следующие сценарии воронок конверсии:

— Передача показаний: Начало сессии → Авторизация → Передача показаний;

— Оплата: Начало сессии → Авторизация → Оплата;

— Выход из аккаунта: Начало сессии → Авторизация → Выход из аккаунта.

Эти сценарии представляют собой ключевые цели пользователей, значит, сосредоточение на оптимизации этих сценариев может привести к повышению конверсии и удовлетворенности пользователей.

2.4 Диаграммы, описывающие работу системы

2.4.1 Диаграмма прецедентов (Use-case diagram)

На рисунке 5 приведена use-case диаграмма системы.

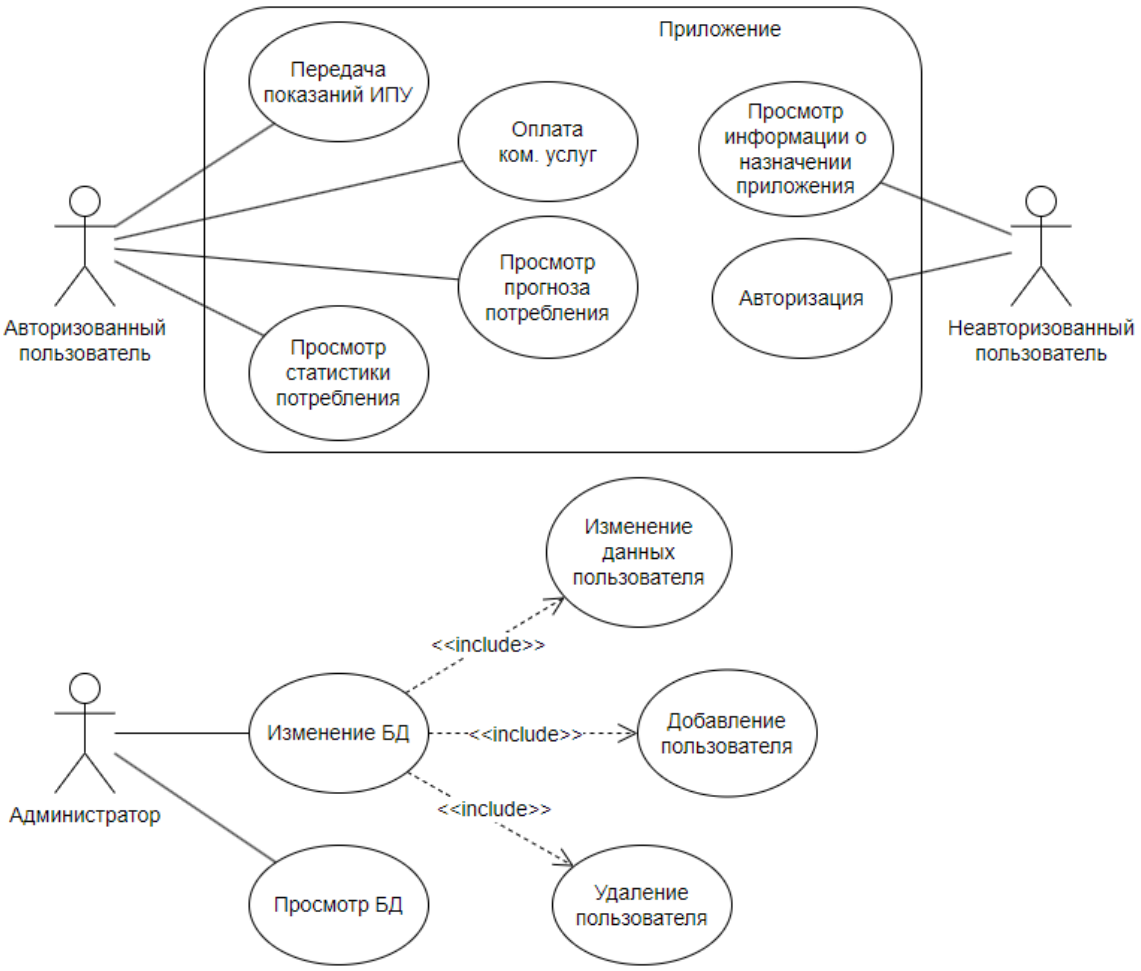


Рисунок 5 - Use-case diagram

Неавторизованный пользователь — пользователь, не прошедший авторизацию или не зарегистрированный в системе. Данный пользователь может:

— Просматривать информацию о назначении приложения;

— Авторизоваться.

Авторизованный пользователь — пользователь, прошедший авторизацию в системе. Помимо функций, доступных неавторизированному пользователю, данный пользователь может:

— Передавать показания ИПУ;

— Получить счет на оплату коммунальных услуг;

— Просматривать статистику потребления за определенный период времени;

— Просматривать прогноз потребления услуг на следующий период.

Администратор — сотрудник компании, который может:

— Производить регистрацию клиентов в системе;

— Просматривать БД;

— Вносить изменения в БД (например, удалять или изменять данные клиента).

2.4.2 Диаграмма классов (Class diagram)

Диаграмма классов (class diagram) является одним из основных видов диаграмм UML, используемых для моделирования статической структуры системы, показывая классы, их атрибуты, методы и взаимосвязи между ними.

Основная цель диаграммы классов состоит в визуализации статической структуры системы и описании классов, их атрибутов и методов. Она помогает понять, как объекты системы организованы, какие свойства у них есть и как они взаимодействуют друг с другом.

На рисунках 6-8 приведены диаграммы классов приложения.

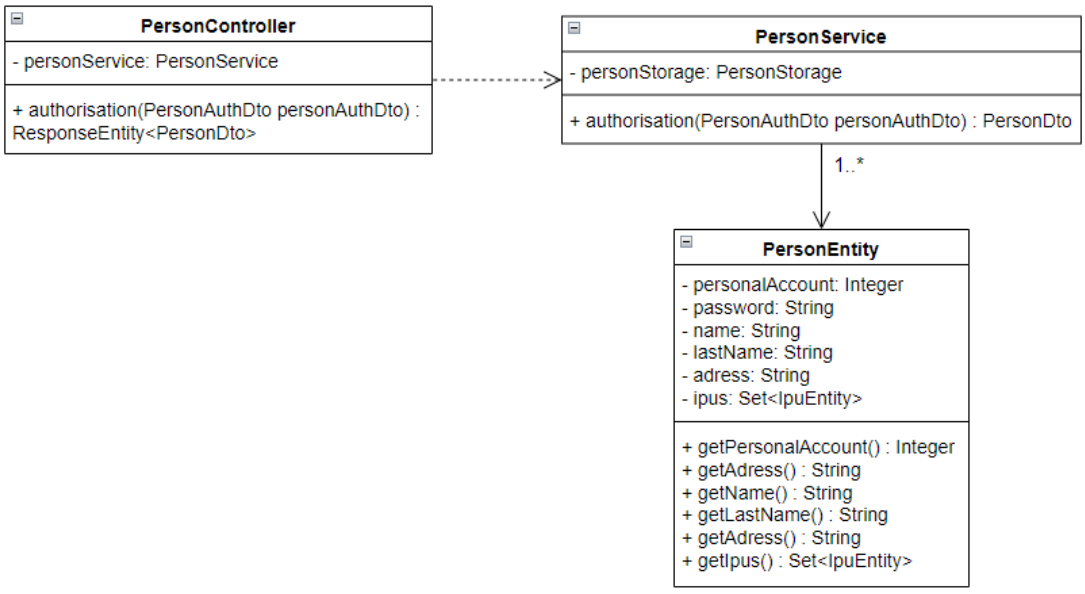


Рисунок 6 - Диаграмма классов, связанных с пользователями

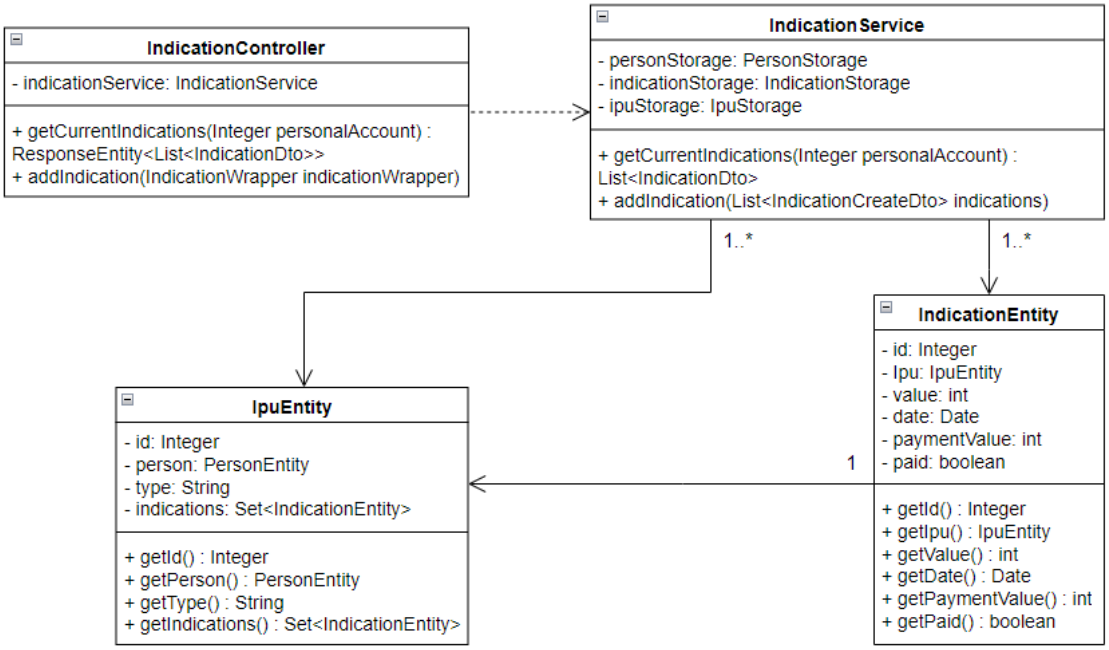


Рисунок 7 - Диаграмма классов, связанных с показаниями

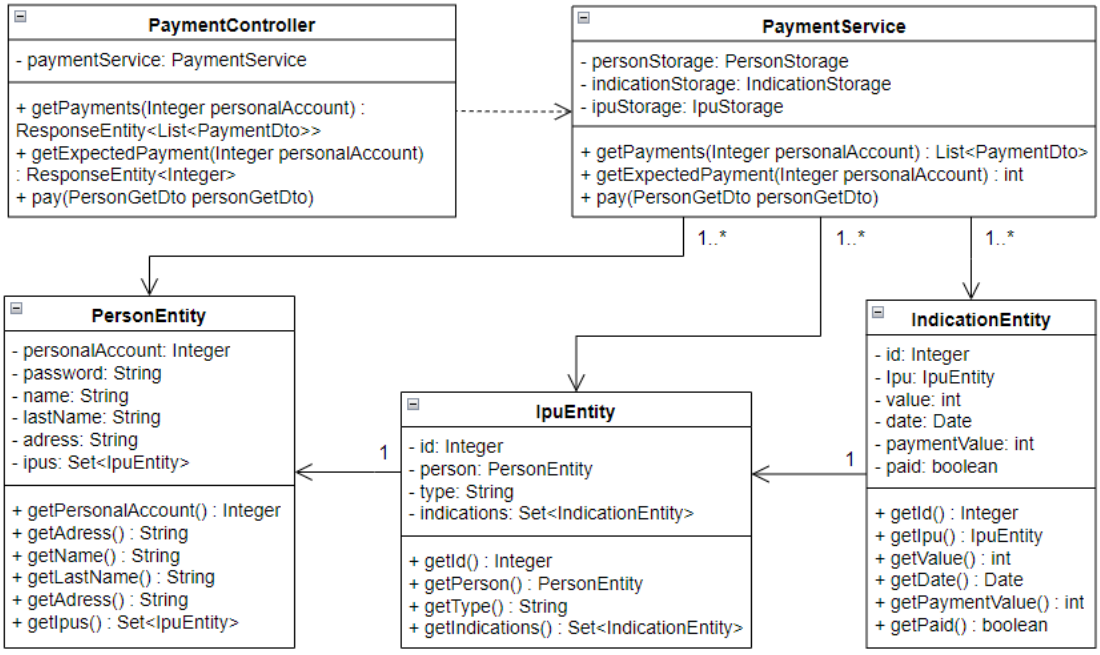


Рисунок 8 - Диаграмма классов, связанных с платежами

2.4.3 Диаграмма активностей (Activity diagram)

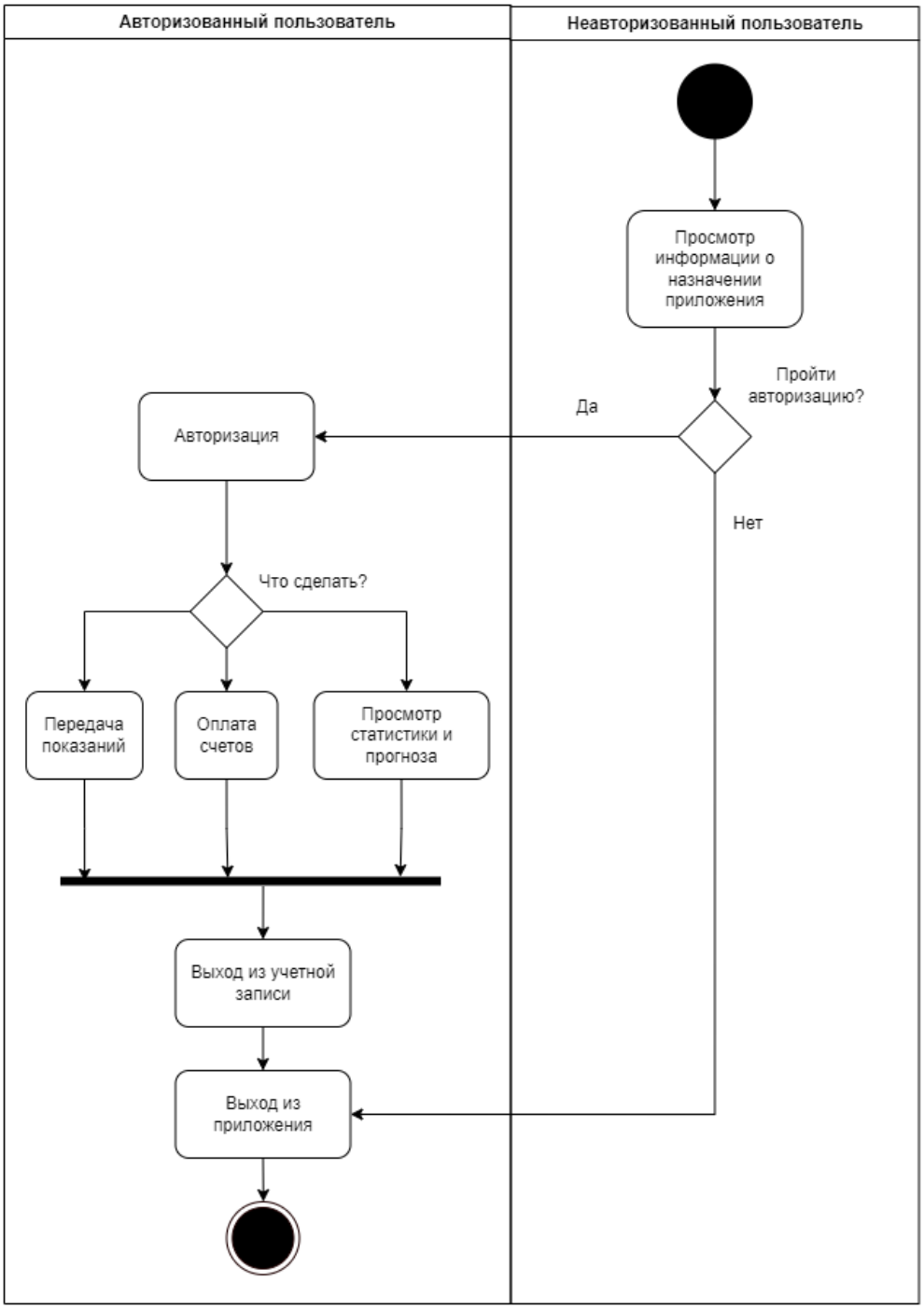


Рисунок 9 - Activity diagram

На рисунке 9 приведена диаграмма активностей для разрабатываемой системы. При входе в систему неавторизованный пользователь видит онбординг экраны с информацией о приложении и его возможностях. После просмотра всех экранов осуществляется переход на экран авторизации. Пользователь может пройти авторизацию или выйти из приложения. После

авторизации пользователь может: передать показания, оплатить счета, посмотреть статистику или прогноз.

2.4.4 Диаграмма последовательности (Sequence diagram)

Диаграмма последовательности (sequence diagram) является одним из видов диаграмм UML, используемых для визуализации взаимодействия между объектами или компонентами системы в определенной последовательности.

Основная цель диаграммы последовательности заключается в визуализации динамического поведения системы или процесса. Она помогает понять, как объекты взаимодействуют друг с другом, какие операции вызываются и как они влияют на состояние системы.

На рисунке 10 приведена диаграмма последовательности системы.

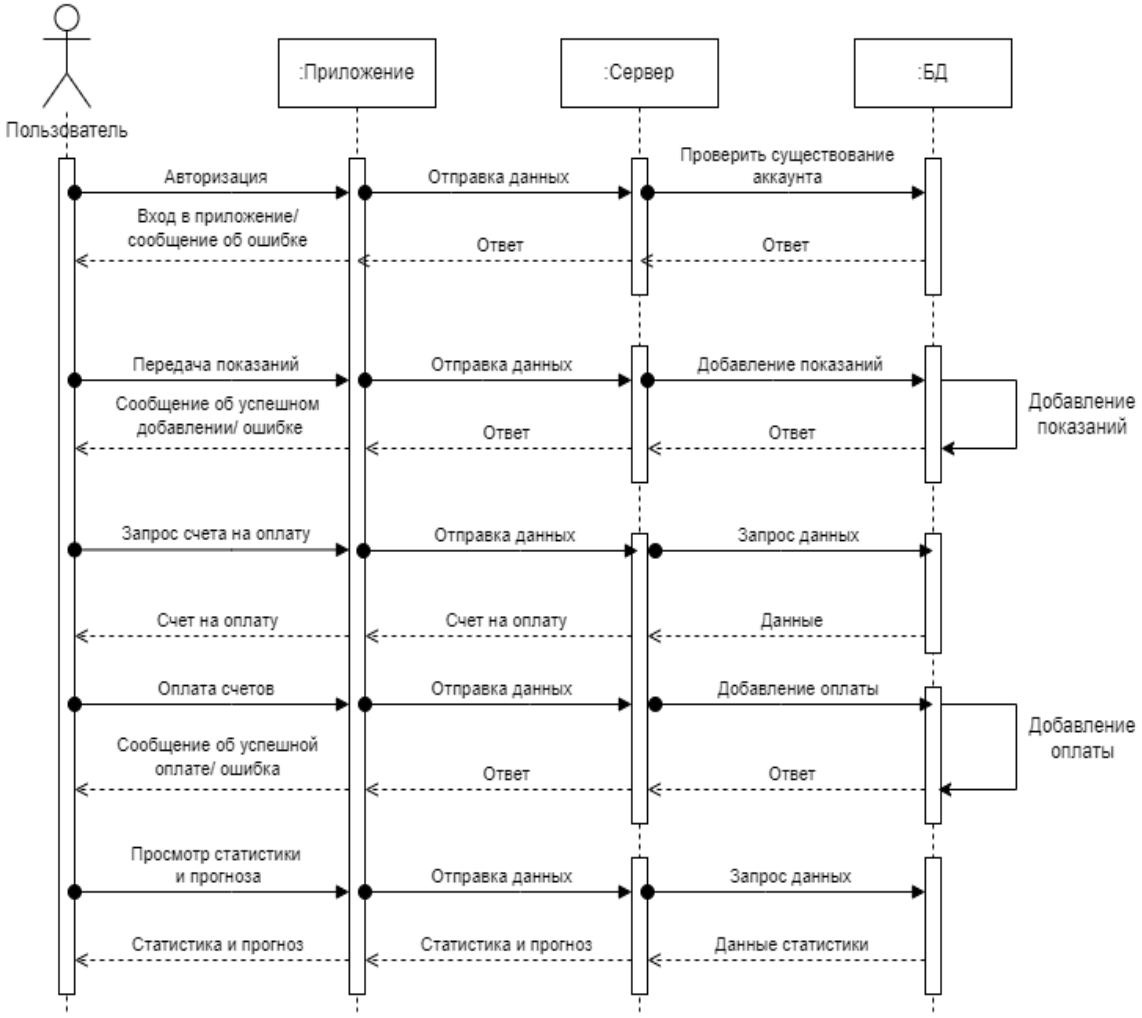


Рисунок 10 - Sequence diagram

2.4.5 Диаграмма развёртывания (Deployment diagram)

Диаграмма развёртывания (deployment diagram) является одним из видов диаграмм UML, используемых для моделирования физической конфигурации и развёртывания компонентов системы на аппаратном и программном обеспечении.

Основная цель диаграммы развёртывания состоит в визуализации физического размещения компонентов системы и их взаимодействия на различных устройствах. Она помогает понять, как компоненты развертываются на аппаратных устройствах, какие связи и зависимости между ними существуют, а также какие ресурсы (процессор, память, хранилище и т.д.) требуются для работы системы.

На рисунке 11 приведена диаграмма развертывания системы.

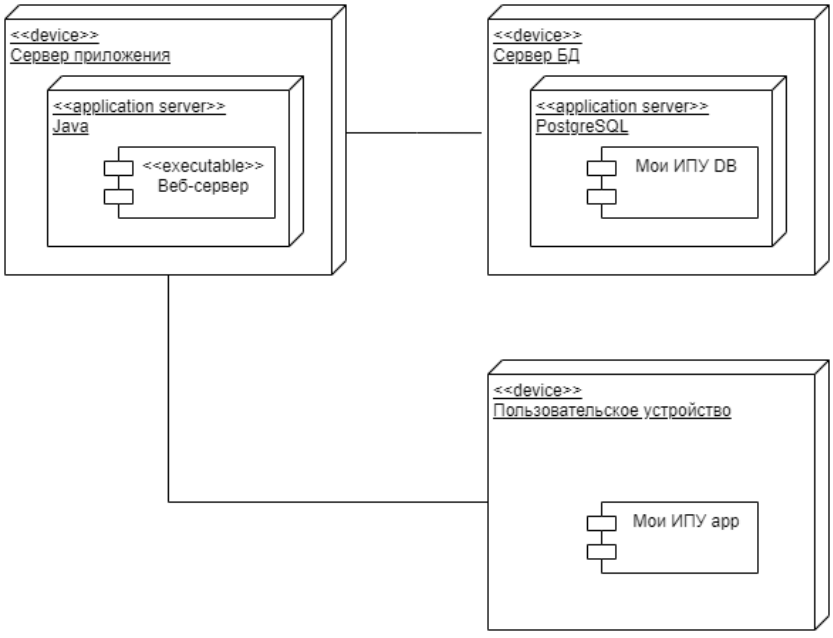


Рисунок 11 - Deployment diagram

2.4.6 Диаграмма сотрудничества (Collaboration diagram)

Диаграмма сотрудничества (collaboration diagram), также известная как диаграмма коммуникации, является одним из видов диаграмм UML, используемых для визуализации взаимодействия между объектами или компонентами системы.

Основная цель диаграммы сотрудничества заключается в визуализации взаимодействия и коммуникации между объектами или компонентами системы. Она помогает понять, как объекты сотрудничают друг с другом, какие сообщения они обмениваются и каким образом взаимодействие влияет на поведение системы.

На рисунках 12-14 приведены диаграммы сотрудничества системы.

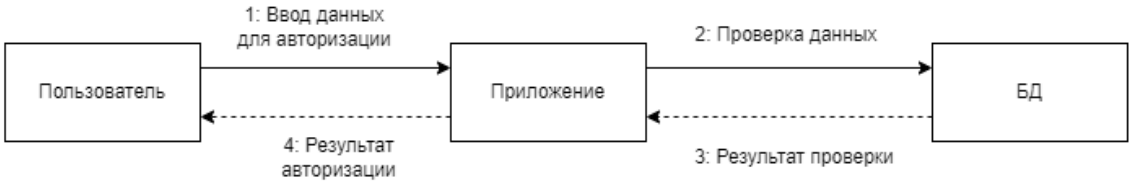


Рисунок 12 - Авторизация

Рисунок 13 - Передача показаний

Рисунок 14 - Оплата счета

2.4.7 Диаграмма объектов (Object diagram)

Диаграмма объектов (object diagram) является одним из видов диаграмм UML, используемых для визуализации статической структуры системы, отображая объекты и их отношения в конкретном моменте времени.

Основная цель диаграммы объектов заключается в визуализации статической структуры системы и отношений между объектами. Она помогает увидеть, какие объекты существуют в системе, какие значения у них есть для атрибутов, и как они связаны друг с другом.

На рисунке 15 приведена диаграмма объектов системы.

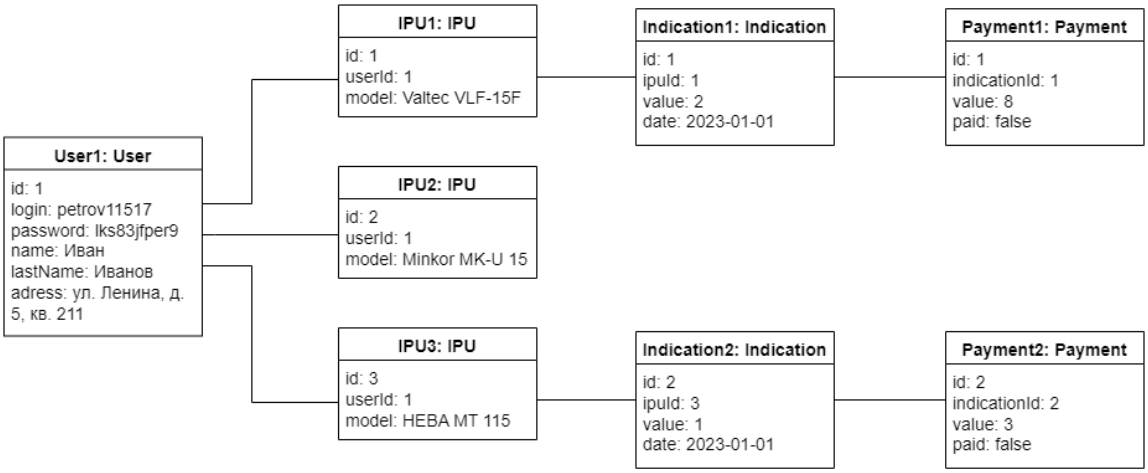


Рисунок 15 - Object diagram

2.4.8 Диаграмма состояний (Statechart diagram)

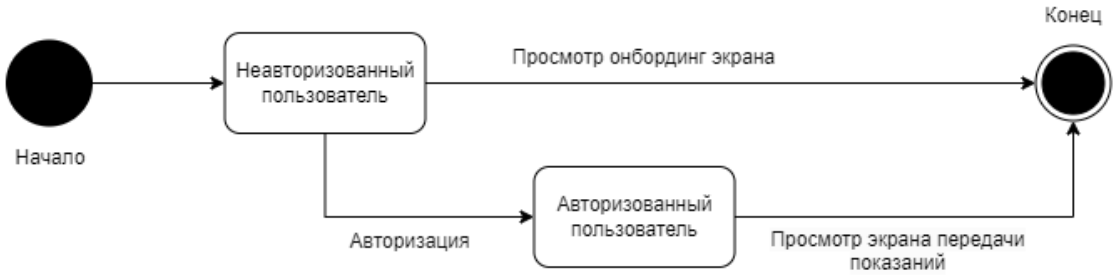


Рисунок 16 - Диаграмма состояний пользователя

На рисунке 16 отображены возможные состояния пользователя. При входе в приложение он неавторизованный, после прохождения авторизации — авторизованный.

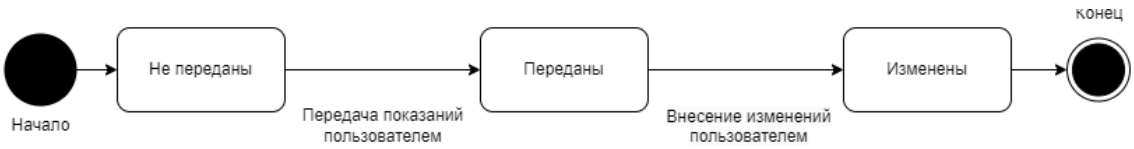


Рисунок 17 - Диаграмма состояний показаний

На рисунке 17 отображены возможные состояния показаний. Изначально они не переданы, после передачи пользователем — переданы, после повторной передачи — изменены.

Рисунок 18 - Диаграмма состояний платежа

На рисунке 18 отображены возможные состояния платежа. Пока не переданы показания он не сформирована, после передачи показаний — сформирована, после оплаты пользователем — оплачены.

2.4.9 IDEF0 диаграмма

Основной целью IDEF0 диаграммы является описание и анализ процессов и функций системы. Эта диаграмма позволяет визуализировать потоки информации и управления в системе, а также идентифицировать возможные улучшения и оптимизации процессов.

Для понимания процесса передачи показаний и оплаты коммунальных услуг через мобильный банк составим IDEF0 диаграмму (рисунок 19-20).

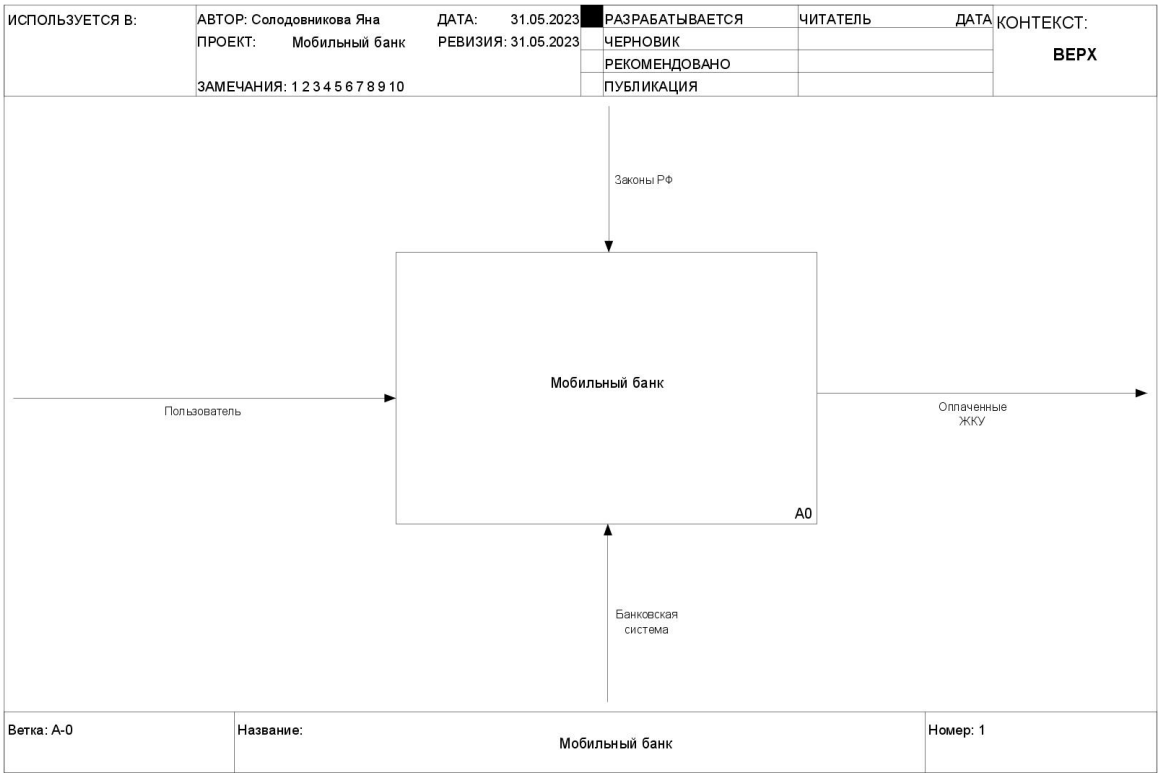


Рисунок 19 - Мобильный банк (0 уровень)

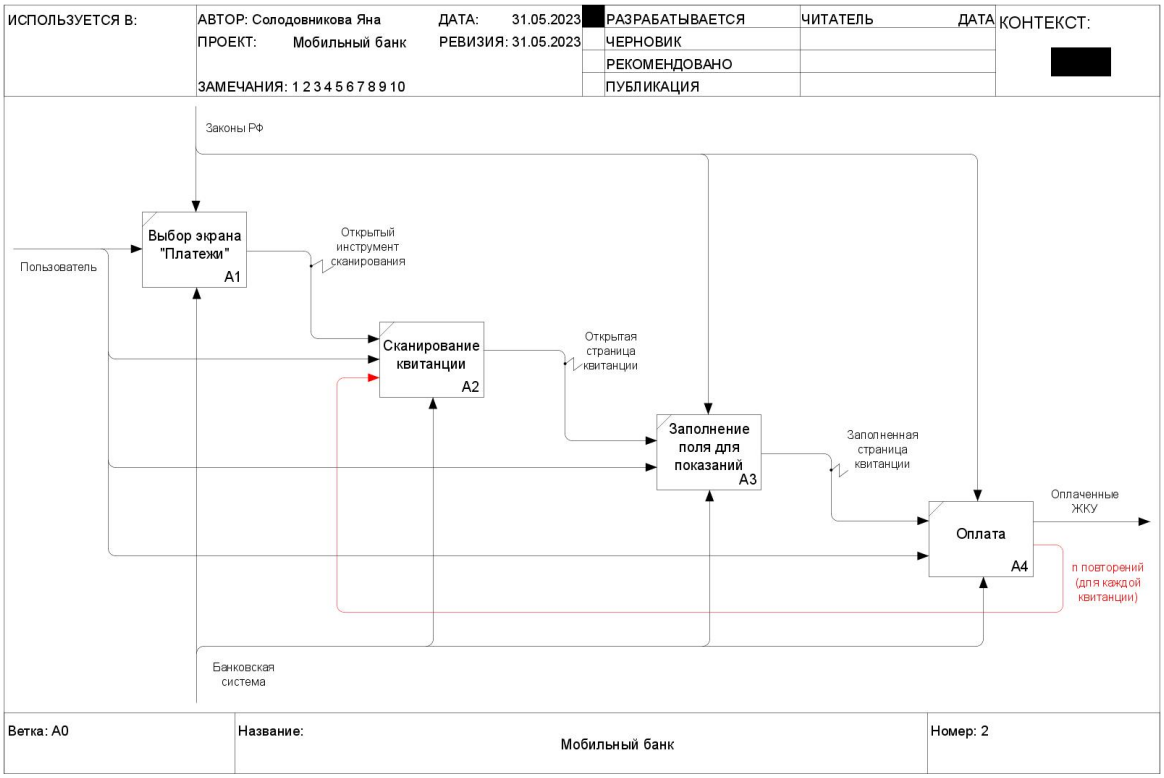


Рисунок 20 - Мобильный банк (1 уровень)

Проведя анализ, можно понять, что для передачи показаний и оплаты счетов таким способом, необходимо выполнить $3n + 1$ действий, где n — количество квитанций. Данный процесс можно оптимизировать. В лучшем случае получится избавиться от линейно растущего количества действий.

На рисунках 21-22 приведены IDEF0 диаграммы системы «Мои ИПУ». Пользователю достаточно выбрать экран «Передать показания» и заполнить поля с показаниями. Если рассматривать это как 1 действие, то в приложении достаточно выполнить всего 1 действие для передачи показаний. Для оплаты необходимо осуществить переход на экран «Оплата» и нажать кнопку «Оплатить». Если рассматривать это как 1 действие, то в приложении для оплаты счетов достаточно выполнить всего 1 действие.

Таким образом, в «Мои ИПУ» достаточно будет выполнить 2 действия независимо от количества потребляемых коммунальных услуг. Данный результат можно считать оптимальным, поскольку получилось избавиться от линейного роста количества действий.

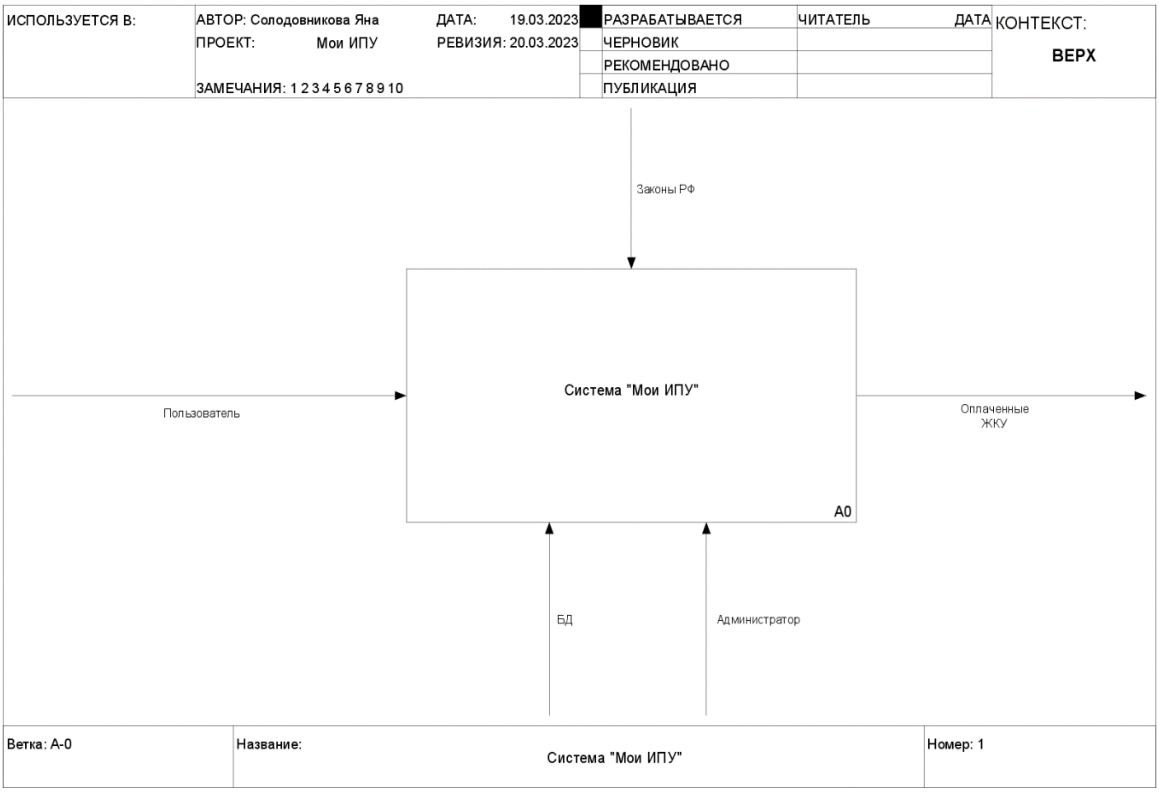


Рисунок 21 - Система «Мои ИПУ» (0 уровень)

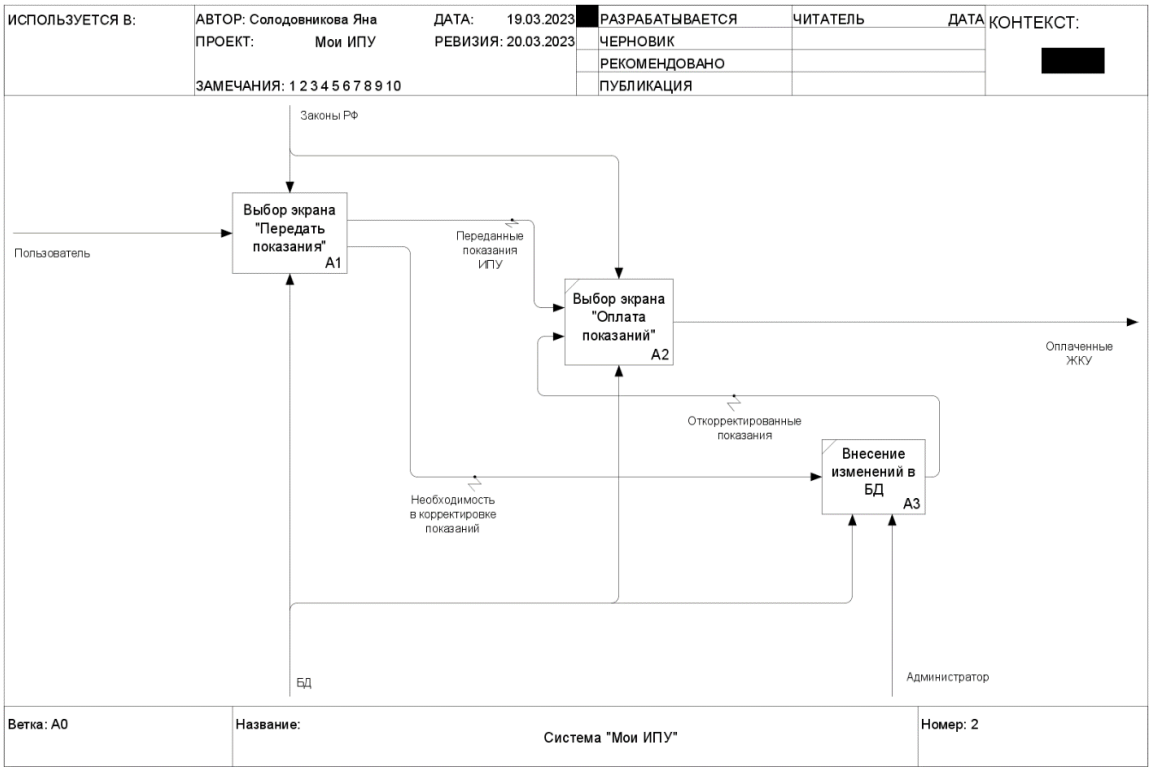
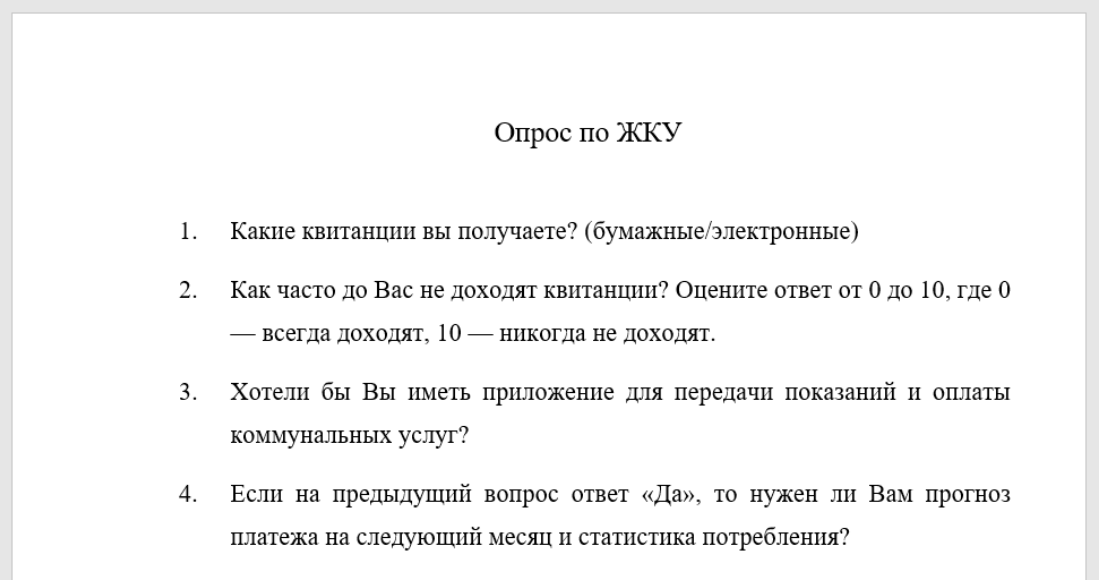


Рисунок 22 - Система «Мои ИПУ» (1 уровень)

2.5 Опрос

Среди 24 жильцов многоквартирного дома был проведен опрос, содержание которого представлено на рисунке 23.



Опрос по ЖКУ

1. Какие квитанции вы получаете? (бумажные/электронные)
2. Как часто до Вас не доходят квитанции? Оцените ответ от 0 до 10, где 0 — всегда доходят, 10 — никогда не доходят.
3. Хотели бы Вы иметь приложение для передачи показаний и оплаты коммунальных услуг?
4. Если на предыдущий вопрос ответ «Да», то нужен ли Вам прогноз платежа на следующий месяц и статистика потребления?

Рисунок 23 - Бланк опроса

Полученные результаты приведены в таблице 1.

Таблица 1 - Результаты опроса

Вопрос	Результат
Какие квитанции вы получаете? (Бумажные/электронные)	80% — бумажные, 20% — электронные
Как часто до Вас не доходят квитанции? От 0 до 10, где 0 — всегда доходят, 10 — никогда не доходят.	В среднем 1
Хотели бы Вы иметь приложение для передачи показаний и оплаты коммунальных услуг?	90% — «Да», 10% — «Нет»
Нужен ли Вам прогноз платежа на следующий месяц и статистика потребления?	95% — «Да», 5% — «Нет»

3 Реализация

3.1 Средства реализации

В качестве языка программирования для разработки серверной части был выбран Java — широко используемый объектно-ориентированный язык программирования, предлагающий ряд преимуществ, среди которых:

— Переносимость;

— Большая библиотека;

— Объектно-ориентированный подход;

— Безопасность.

В качестве фреймворка для разработки серверной части приложения был выбран Spring — популярный фреймворк для разработки приложений на языке Java, предлагающий ряд преимуществ, среди которых:

— Внедрение зависимостей (DI);

— Модульность.

В качестве СУБД была выбрана PostgreSQL — мощная и расширяемая система управления базами данных, предлагающий ряд преимуществ, среди которых:

— Надежность;

— Поддержка SQL-стандартов.

В качестве языка программирования для разработки клиентской части был выбран Kotlin — современный статически типизированный язык программирования, разработанный компанией JetBrains. Он предлагает ряд преимуществ, среди которых:

— Поддержка Android;

— Поддержка нулевых значений (null safety);

— Поддержка кроссплатформенных приложений.

3.2 Разработка Frontend

Для разработки клиентской части приложения была выбрана архитектура MVI (Model-View-Intent). Такая архитектура хорошо подходит для мобильного приложения, потому что:

— Почти полностью покрывается тестами;

— Хорошо масштабируется;

— Позволяет хранить состояние экранов, что упрощает отладку и нахождение ошибок;

— Поток данных имеет единственное направление, что упрощает понимание работы приложения.

Разработка клиентской части приложения проходила в соответствии со следующими этапами:

— Инициализация;

— Вёрстка экранов;

— Реализация запросов к серверу.

3.2.1 Инициализация

На этом этапе необходимо было провести настройку проекта. В неё входили:

— Подключение необходимы зависимостей;

— Добавление стилей приложения;

— Настройка темы приложения.

3.2.2 Вёрстка экранов

На этом этапе необходимо было сверстать экраны и добавить основную логику отображения всех элементов без привязки к серверу.

Каждый экран представляет из себя отдельный модуль в проекте. Такие модули именуются по следующему правилу: «f-screen-name».

Для каждого экрана используется FragmentView, который отвечает за отображение данных. Для сохранения состояния экранов используется

ScreenState, где хранится вся информация, которая будет отображена на экране. Для обработки и изменения состояния используется ScreenReducer. Для отправки команд о том, что нужно показать сообщение об ошибке или успешно проведённой операции, используется CommandHolder.

3.2.3 Реализация запросов к серверу

На этом этапе необходимо было реализовать логику запросов к серверу.

Для запросов к серверу создаётся отдельный модуль. Такие модули именуются по следующему правилу: «i-request-name».

Для выполнения запросов к северу используется библиотека Retrofit2 и классы RequestApi, которые она создаёт. После выполнения запроса, результат попадает в RequestIntercator, где происходит преобразование полученных данных в используемые приложением сущности. Связь запросов в сеть с приложение осуществляется с помощью Middleware. Внутри него происходит выполнение запросов, результат которых отправляется в Reducer для изменения состояния.

3.3 Разработка Backend

Для разработки серверной части приложения была выбрана архитектура MVC (Model-View-Controller). Такая архитектура хорошо подходит для REST-сервера, потому что:

— Разделяет приложение на три основных компонента;

— Каждый компонент может быть разработан и изменен независимо;

— Позволяет легко поддерживать разные форматы ответа для REST-сервера.

Разработка серверной части приложения проходила в соответствии со следующими этапами:

— Инициализация сервера без привязки к БД;

— Проектирование БД;

— Создание БД;

— Связь сервера с БД;

— Реализация REST запросов.

3.3.1 Инициализация сервера без привязки к БД

На первом этапе необходимо было создать и настроить связь между классами: Controller, Service, Entity, DTO, Storage. В качестве entity выступают Indication, Ipu и Person.

Сервер разработан в соответствии с MVC архитектурой. За компонент Model отвечают Service и Entity классы. За компонент Controller отвечают Controller классы, которые обрабатывают входящие HTTP-запросы. Компонент View представлен исходящими от сервера JSON ответами, которые потом обрабатываются на клиентской части приложения.

Для обработки HTTP-запросов, связанных с пользователем необходим PersonController. В нем есть POST метод authorization, который используется для авторизации пользователя. Он принимает объект класса PersonAuthDto, содержащий лицевой счет и пароль, и возвращает объект класса ResponseEntity<PersonDto>.

Для реализации логики запросов необходим PersonService. В нем реализован метод authorization, который осуществляет процесс авторизации пользователя. Он принимает объект класса PersonAuthDto в качестве параметра и возвращает объект класса PersonDto, содержащий полную информацию о пользователе.

Для хранения и обработки информации в PersonService используется PersonStorage. На данном этапе PersonStorage является Map, где ключом является personalAccount, а значением — объект класса PersonEntity.

Для обработки HTTP-запросов, связанных с показаниями необходим IndicationController. В нем есть POST метод addIndications, который используется для передачи показаний. Он принимает IndicationWrapper в качестве параметра и ничего не возвращает. Данный оберточный класс

обрабатывает входящий List<IndicationCraeteDto>. Также есть GET метод getCurrentIndications, который используется для получения текущих показаний. Он принимает personalAccount в качестве параметра и возвращает объект класса ResponseEntity< List<IndicationDto>>.

Для реализации логики запросов необходим IndicationService. В нем реализован метод addIndications, который осуществляет процесс добавления новых показаний. Он принимает List<IndicationCraeteDto> в качестве параметра и ничего не возвращает. Также в нем реализован метод getCurrentIndications, который осуществляет процесс получения текущих показаний. Он принимает personalAccount в качестве параметра и возвращает List<IndicationDto>.

Для хранения и обработки информации о показаниях в IndicationService используется IndicationStorage. На данном этапе IndicationStorage является Map, где ключом является id, а значением — объект класса IndicationEntity. Для хранения информации об ИПУ IndicationService используется IpuStorage. На данном этапе IpuStorage является Map, где ключом является id, а значением — объект класса IpuEntity.

Для обработки HTTP-запросов, связанных с платежами необходим PaymentController. В нем есть POST метод pay, который используется для оплаты счетов. Он принимает объект класса PersonGetDto в качестве параметра и ничего не возвращает. GET метод getPayments, который используется для получения текущих счетов на оплату. Он принимает personalAccount в качестве параметра и возвращает объект класса ResponseEntity<List<PaymentDto>>. Также есть GET метод getExpectedPayment, который используется для получения ожидаемого размера платежа в следующем месяце. Он принимает personalAccount в качестве параметра и возвращает объект класса ResponseEntity<Integer>.

Для реализации логики запросов необходим PaymentService. В нем реализован метод pay, который осуществляет процесс оплаты счетов. Он принимает объект класса PersonGetDto в качестве параметра и ничего не

возвращает. Реализован метод getPayments, который осуществляет процесс получения текущих счетов на оплату. Он принимает personalAccount в качестве параметра и возвращает List< PaymentDto>. Также реализован метод getExpectedPayment, который осуществляет процесс получения ожидаемого размера платежа на следующий расчетный период. Он принимает personalAccount в качестве параметра и возвращает int значение.

Для хранения и обработки информации о платежах в PaymentService используется IndicationStorage. На данном этапе IndicationStorage является Map, где ключом является id, а значением — объект класса IndicationEntity. Для хранения информации об ИПУ — IpuStorage, для хранения информации о пользователях — PersonStorage.

Таким образом был проинициализирован сервер, созданы классы и настроена связь между ними.

3.3.2 Проектирование БД

На этапе проектирования БД были созданы ER-диаграмма и физическая модель БД.

ER-диаграмма (сущность-связь) — это графическое представление структуры и взаимосвязей данных в базе данных. Она моделирует сущности (объекты) в системе, их атрибуты (характеристики) и связи между ними.

На рисунке 24 приведена ER-диаграмма, на которой есть сущности Person, IPU, Indication, Payment, а также настроена связь между ними. У пользователя может быть несколько ИПУ или не быть их совсем, у ИПУ должен быть один пользователь. И ИПУ может быть несколько показаний или не быть их вообще, у показания должен быть один ИПУ. У каждого показания должен быть один платеж, у каждого платежа должно быть одно показание.

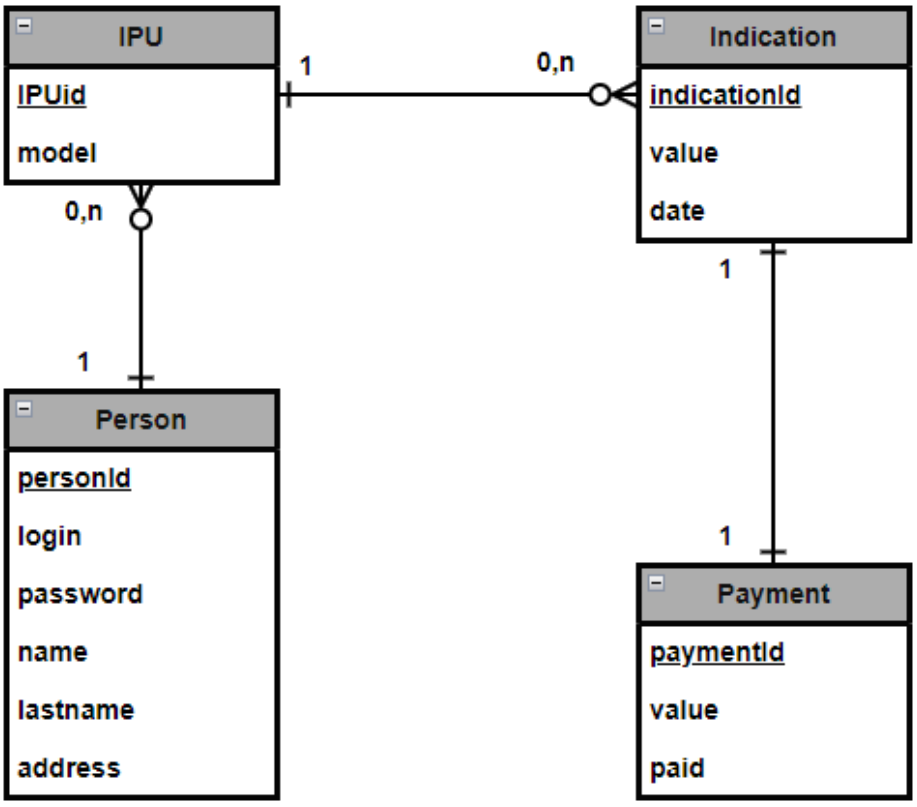


Рисунок 24 - ER диаграмма

По ER-диаграмме составлена физическая модель БД (Рисунок 25).

Физическая модель базы данных представляет собой конкретное описание структуры базы данных, которое определяет, как данные будут храниться на физическом уровне. Она определяет способ организации таблиц, индексов, ограничений, связей и других аспектов базы данных, с учетом особенностей конкретной системы управления базами данных (СУБД).

Сущность Indication и Payment были объединены в 1 таблицу Indication with Payment. paymentId стало альтернативным ключом, а indicationId — первичным. Сущности Person и IPU остались без изменений.

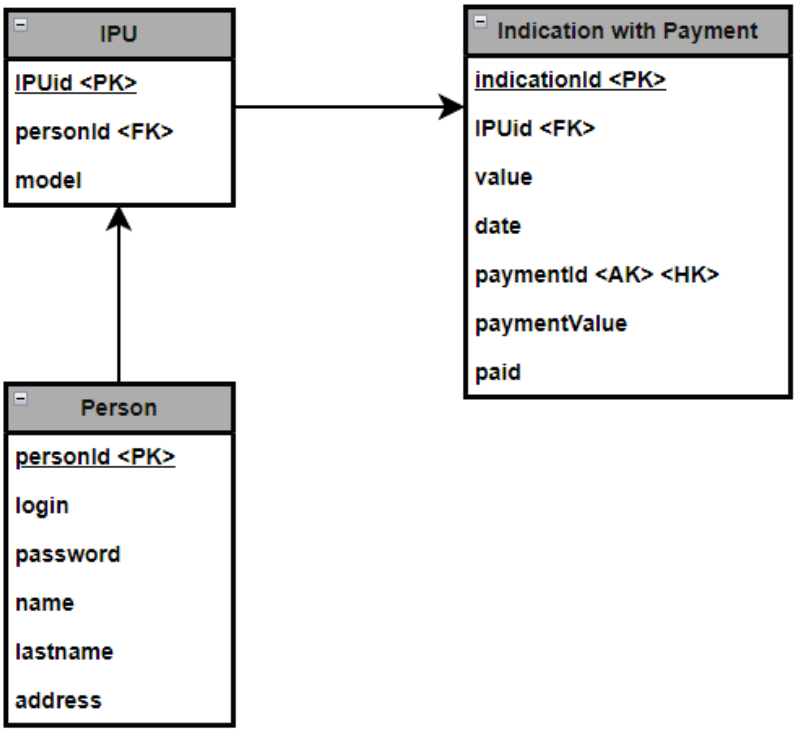


Рисунок 25 - Физическая модель БД

3.3.3 Создание БД

На данном этапе необходимо создать локальную БД. В ней не нужно создавать таблицы, поскольку это будет реализовано самим сервером на следующем этапе.

Во фреймворке Spring существует механизм ORM (Object-Relational Mapping), который используется для создания таблиц БД на основе сущностей (Entity). Он включает в себя компоненты Hibernate/JPA.

Hibernate/JPA — это фреймворк и API для работы с базами данных в Java. Они позволяют упростить взаимодействие с базой данных, оперируя объектами и связями между ними, а не таблицами и столбцами. Hibernate является реализацией JPA и предоставляет удобные инструменты для маппинга объектов на таблицы, выполнения запросов и управления транзакциями.

3.3.4 Связь сервера и БД

На данном этапе для создания связи между сервером и БД необходимо было задать параметры подключения к БД в файле application.properties.

Также необходимо было заменить все Storage классы на интерфейсы, наследующие JpaRepository. В них реализованы все необходимые методы для взаимодействия с БД.

Для создания таблиц по Entity, в них необходимо было добавить аннотации @Entity, @Table и @Column. В соответствии с физической моделью БД между Entity существуют различные связи, для реализации которых использовались аннотации @ManyToOne и @OneToMany.

3.3.5 Реализация REST запросов

На первом этапе был проинициализирован сервер, на данном этапе необходимо реализовать логику REST запросов.

Метод authorization в PersonService находит по логину и паролю данные пользователя в таблице Person и возвращает их. В случае, если пользователя в таблице найти не удалось, возвращается null значение.

Метод getCurrentIndications в IndicationService по лицевому счету находит какие ИПУ принадлежат пользователю, перебирает каждый ИПУ и для него из таблицы Indication получает последние переданные показания. Все показания добавляются в список, который в конце и возвращает метод.

Метод addIndications в IndicationService перебирает все переданные показания, для каждого из них находит ИПУ, к которому оно относится, и рассчитывает размер платежа за потребленные услуги в соответствии с тарифом. После этого формирует запись и добавляет ее в таблицу Indication.

Метод getPayments в PaymentService по лицевому счету находит какие ИПУ принадлежат пользователю, перебирает каждый ИПУ и для него из таблицы Indication получает сумму всех неоплаченных счетов. Полученные платежи добавляются в список, который в конце и возвращает метод.

Метод getExpectedPayment в PaymentService по лицевому счету находит какие ИПУ принадлежат пользователю, перебирает каждый ИПУ и для него из таблицы Indication получает размер последних 5 платежей. Они суммируются и в конце полученное число делится на количество учтенных

платежей, то есть данный метод рассчитывает средней значение последних 5 платежей.

Метод pay в PaymentService по лицевому счету находит какие ИПУ принадлежат пользователю, перебирает каждый ИПУ и для него в таблице Indication устанавливает значение «оплачено» для всех неоплаченных показаний, относящихся к нему.

3.4 Тестирование

Для тестирования разработанного приложения были выбраны:

— Дымовое тестирование;

— GUI (Graphic User Interface) тестирование;

— Юзабилити-тестирование.

Дымовое тестирование использовалось на всех этапах разработки для проверки работоспособности системы после внесения изменений. Результаты тестирования после окончания работ приведены в таблице 2.

Таблица 2 - Результаты дымового тестирования

Сценарий	Результат
Авторизация	Пройден
Передача показаний	Пройден
Оплата показаний	Пройден
Получение прогноза оплаты	Пройден
Просмотр статистики	Пройден

GUI тестирование — процесс тестирования графического пользовательского интерфейса продукта. Оно проверяет общий внешний вид, например, читаемость текста, удобство расположения элементов. Также оно проверяет работоспособность графических элементов, например, кнопок или чек-листов.

Она проводилось на всех этапах разработки для своевременного выявления неисправностей и их устранения. Результаты тестирования после окончания работ приведены в таблице 3.

Таблица 3 - Результаты GUI тестирования

Шаги теста	Ожидаемый результат	Статус
Нажатие далее на онбординг экране	Переход к следующему онбординг экрану	Пройден
Нажатие пропустить на онборднг экране	Переход к экрану авторизации	Пройден
Нажатие кнопки «Войти» на экране авторизации	Переход на экран для передачи показаний	Пройден
Нажатие кнопки «Показания» на навигационном меню	Переход на экран для передачи показаний	Пройден
Нажатие кнопки «Передать показаний»	Передача показаний	Пройден
Нажатие кнопки «Оплата» на навигационном меню	Переход на экран для оплаты	Пройден
Нажатие кнопки «Оплатить»	Оплата потреблённых услуг	Пройден
Нажатие кнопки «Профиль» на навигационном меню	Переход на экран «Личный кабинет»	Пройден
Нажатие кнопки «Выйти» на экране профиля	Выход из аккаунта	Пройден

Для юзабилити-тестирования было отобрано 3 человека, которые получили доступ к законченному приложению «Мои ИПУ». Результаты тестирования после окончания работ приведены в таблице 4.

Таблица 4 - Результаты юзабилити-тестирования

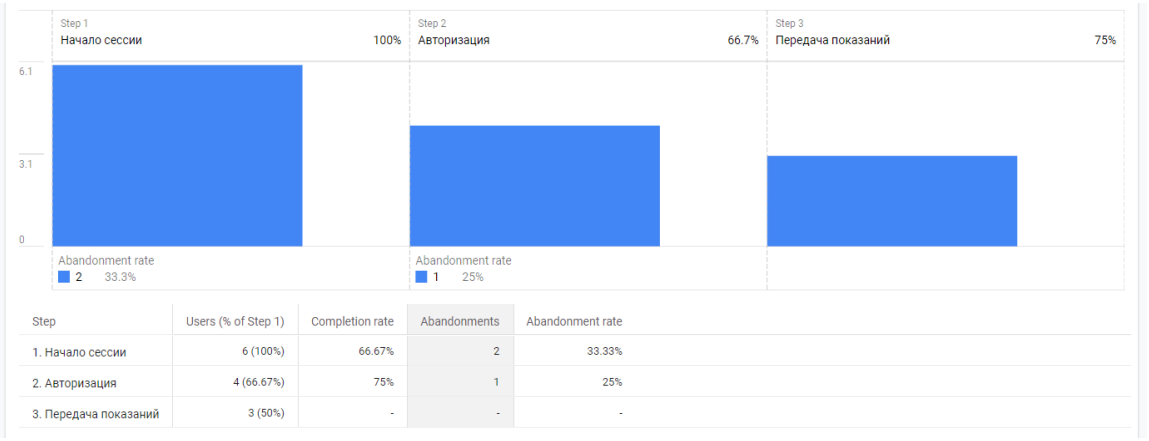
Сценарий	Пользователь 1	Пользователь 2	Пользователь 3
Авторизация	Пройден	Пройден	Пройден
Передача показаний	Пройден	Пройден	Пройден
Оплата показаний	Пройден	Пройден	Пройден

Продолжение таблицы 4

Получение прогноза оплаты	Пройден	Пройден	Пройден
Просмотр статистики	Пройден	Пройден	Пройден

3.5 Аналитика

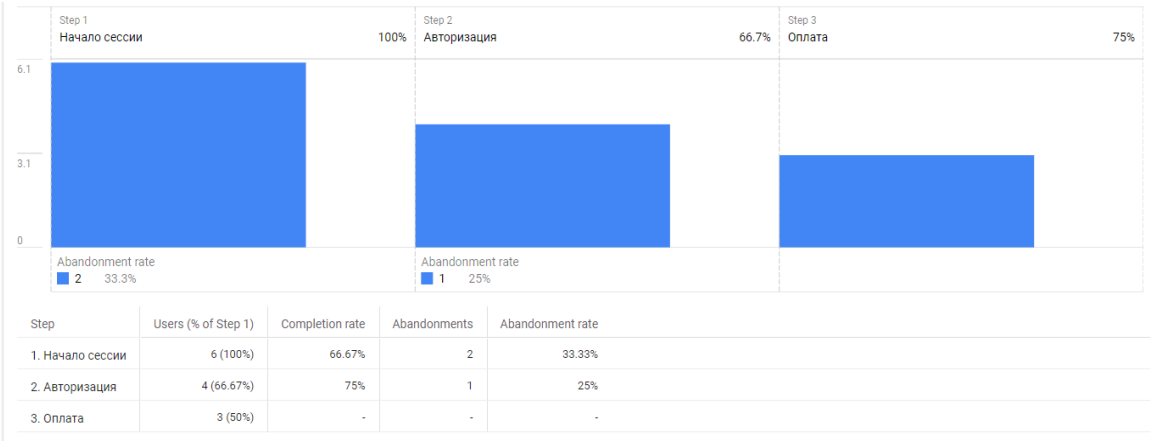
Для анализа приложения был использован сервис Google Analytics. Рассмотрим воронки конверсии для основных сценариев.



Step	Users (% of Step 1)	Completion rate	Abandonments	Abandonment rate
1. Начало сессии	6 (100%)	66.67%	2	33.33%
2. Авторизация	4 (66.67%)	75%	1	25%
3. Передача показаний	3 (50%)	-	-	-

Рисунок 26 - Воронки конверсии для передачи показаний

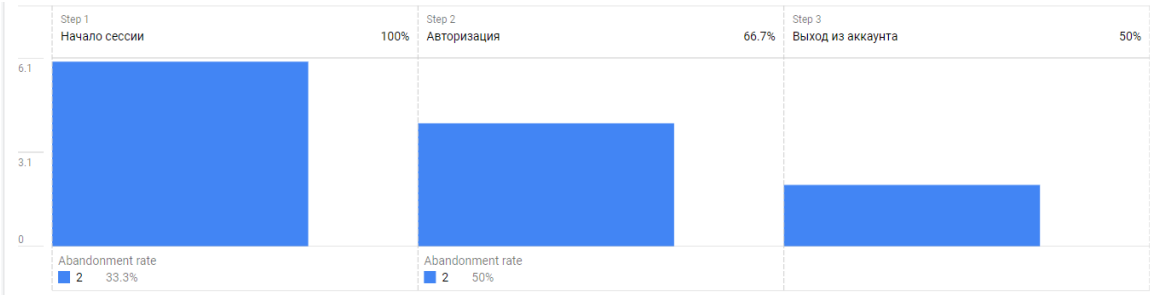
На рисунке 26 изображен сценарий передачи показаний. Как видно из данных коэффициент завершения для начала сессии равен 66,67% и для авторизации — 75%. Таким образом до передачи показаний дошло 50% пользователей, что можно считать хорошим результатом.



Step	Users (% of Step 1)	Completion rate	Abandonments	Abandonment rate
1. Начало сессии	6 (100%)	66.67%	2	33.33%
2. Авторизация	4 (66.67%)	75%	1	25%
3. Оплата	3 (50%)	-	-	-

Рисунок 27 - Воронки конверсии для оплаты

На рисунке 27 изображен сценарий оплаты счетов. Как видно из данных коэффициент завершения для начала сессии равен 66,67% и для авторизации — 75%. Таким образом до оплаты дошло 50% пользователей, что можно считать хорошим результатом.



Step	Users (% of Step 1)	Completion rate	Abandonments	Abandonment rate
1. Начало сессии	6 (100%)	66.67%	2	33.33%
2. Авторизация	4 (66.67%)	50%	2	50%
3. Выход из аккаунта	2 (33.33%)	-	-	-

Рисунок 28 - Воронки конверсии для выхода из аккаунта

На рисунке 28 изображен сценарий выхода из аккаунта. Как видно из данных коэффициент завершения для начала сессии равен 66,67% и для авторизации — 75%. Таким образом до выхода из аккаунта дошло 50% пользователей, что можно считать хорошим результатом.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было создано приложение для сбора показаний индивидуальных приборов учёта в многоквартирных домах и выставления счетов за потреблённые услуги.

Также были реализованы:

— Расчёт статистики по потреблённым услугам;

— Составление прогноза на потребление коммунальных услуг в следующем периоде.

«Мои ИПУ» прошло ряд тестирований, по результатам которых был оно удовлетворяет выставленным требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Что такое база данных? [Электронный ресурс]. – Режим доступа: https://www.oracle.com/cis/database/what-is-database. – Заглавие с экрана. – (Дата обращения: 10.03.2023).

2. ГОСТ Р 51929–2002. Услуги жилищно-коммунальные. Термины и определения [Текст]. – Введ. 2003-01-01. – М. : Изд-во стандартов, 2005. – 3 с.

3. Зачем нужен личный кабинет [Электронный ресурс]. – Режим доступа: https://inostudio.com/blog/articles-managment/zachem-nuzhen-lichnyy-kabinet. – Заглавие с экрана. – (Дата обращения: 11.03.2023).

4. Что такое единый лицевой счет [Электронный ресурс]. – Режим доступа: https://www.raiffeisen.ru/wiki/chto-takoe-edinyj-licevoj-schet. – Заглавие с экрана. – (Дата обращения: 12.03.2023).

5. Как формируются тарифы ЖКХ [Электронный ресурс]. – Режим доступа: https://vgkh.ru/articles/kak-formiruyutsya-tarify-zhkkh. – Заглавие с экрана. – (Дата обращения: 12.03.2023).

6. Основы дизайна онбординга [Электронный ресурс]. – Режим доступа: https://idbi.ru/blogs/blog/osnovy-dizayna-onbordinga. – Заглавие с экрана. – (Дата обращения: 21.05.2023).

7. HTTP request methods [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods. – Заглавие с экрана. – (Дата обращения: 16.05.2023).