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

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 профиля / специализации)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка и администрирование информационно-расчетной системы внесения показаний ИПУ»

Обучающийся Д. Е. Резников
(И.О. Фамилия) (личная подпись)

Руководитель к.п.н., доцент О.В. Оськина
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент С.А. Гудкова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2024

Аннотация

Название выпускной работы: «Разработка и администрирование информационно-расчетной системы внесения показаний ИПУ».

Эта дипломная работа посвящена созданию веб-приложения, цель которого – улучшить процесс учета и контроля показаний индивидуальных приборов учета в жилищно-коммунальной сфере. Работа включает введение, три основные главы, заключение и содержит 23 иллюстрации, 2 таблицы и список из 21 источников.

Основная задача данной дипломной работы заключается в анализе эффективности управления и прозрачности отчетности о потреблении коммунальных ресурсов с помощью технологических решений. В исследовании анализируется взаимодействие пользователей с управляющей компанией и автоматизация процесса сбора данных с коммунальных счетчиков.

Объект исследования – интеграция современных веб-технологий, таких как Spring Boot, Thymeleaf и PostgreSQL, в инфраструктуру жилищно-коммунального хозяйства. Предмет исследования – разработка системы, которая не только упрощает электронную передачу показаний счетчиков, но и оптимизирует административные процессы управления коммунальными ресурсами

Особое внимание уделяется архитектурному проектированию, анализу требований и выбору подходящих технологий разработки. Мы подробно описываем функциональность системы, включая удобные интерфейсы для подачи показаний счетчиков и просмотра истории потребления. Дипломная работа разделена на логически связанные части, охватывающие концептуализацию системы, дизайн, реализацию и функциональное тестирование.

Также мы исследуем, как приложение соответствует установленным требованиям через тщательное тестирование, демонстрируя его

эффективность и работоспособность в реальных условиях. В конечном итоге, внедрение этой системы предполагается улучшить прозрачность расчетов и оптимизировать процессы учета в жилищно-коммунальной сфере.

В заключение, данное исследование значительно вносит вклад в модернизацию систем управления жилищно-коммунальным хозяйством и может служить образцом для будущих разработок в этой области. Успешное внедрение этой системы может привести к ее широкому принятию в городах, стремящихся улучшить свои практики управления коммунальными услугами.

Работа также затрагивает перспективы дальнейшего развития и улучшения системы, учитывая возможные расширения функционала и адаптацию к изменяющимся требованиям пользователей и регулирующих органов. Систематическое внедрение новых технологий и методов позволит поддерживать высокую эффективность и надежность системы в долгосрочной перспективе.

Abstract

The title of the graduation work is «Development and Implementation of an Information and Calculation System for Transmitting Individual Metering Device Readings».

The senior paper consists of an introduction, three main chapters, a conclusion, tables, illustrations, and a list of references including foreign sources.

The key issue of the thesis is the design and development of a web application for the efficient recording and monitoring of individual meter readings in the housing and utility sector.

The graduation work is divided into several logically connected parts: analysis of the existing management systems for meter readings, justification of the need for an automated web application, design and development of the system using technologies such as Spring Boot, Thymeleaf, and PostgreSQL, solving the problem of manual data entry errors, improving transparency in calculations, and enhancing integration with existing utility management systems.

Finally, we present the successful development and implementation of the web application, which contributes to improving the transparency and efficiency of communal resource management, speeding up the process of data collection, and increasing customer satisfaction.

In conclusion, this work is relevant in addressing the efficiency and transparency of managing communal resource consumption. The results can be valuable for utility management companies and their clients, and the developed system can significantly enhance the overall quality of service in the housing and utility sector.

Оглавление

Введение

В условиях современной экономики информационные технологии играют ключевую роль в управлении бизнес-процессами различных отраслей, включая жилищно-коммунальное хозяйство. Разнообразие информационных систем и сайтов позволяет компаниям ускорять процессы обработки данных, повышать качество обслуживания и управлять потоками клиентов более эффективно. Именно поэтому создание и внедрение современных информационно-расчётных систем становится не просто актуальной задачей, но и необходимостью, направленной на оптимизацию работы коммунальных служб и улучшение взаимодействия с жителями.

Цель данной дипломной работы – разработка и администрирование информационно-расчетной системы (ИРС) для эффективного сбора, хранения и анализа данных о потреблении коммунальных ресурсов как для компаний-исполнителей, так и для клиентов – жильцов многоквартирных домов. Система должна обеспечить удобство и наглядность использования, что позволит пользователям не только контролировать свои расходы, но и способствовать повышению энергоэффективности и экономии ресурсов.

Основной задачей работы является создание веб-приложения, которое позволит минимизировать человеческий фактор при вводе данных и автоматизировать процесс передачи показаний счетчиков в жилищно-коммунальных компаниях. В процессе разработки будут использованы современные инструменты веб-разработки, включая базы данных, фреймворки для создания веб-приложений, а также алгоритмы для анализа данных и языки разметки.

Актуальность проекта подчеркивается необходимостью сокращения прямых контактов при передаче показаний, что становится особенно значимым в контексте современных требований к социальной дистанции и цифровизации общественных процессов. Внедрение предлагаемой системы позволит не только улучшить качество предоставляемых услуг и удобство их

использования для конечных пользователей, но и способствует устранению недопониманий между жильцами и управляющими компаниями, обеспечивая прозрачность и честность расчетов.

Таким образом, дипломная работа направлена на решение актуальных проблем современного жилищно-коммунального хозяйства с помощью применения современных информационных технологий, что делает ее важным вкладом в развитие данной отрасли. Коммунальные услуги благодаря данной системе станут более прозрачными, наглядными и надежными как для жителей, так и для управляющих компаний. В дальнейшем это позволит внедрять подобные решения в других областях управления городским хозяйством, способствуя их эффективной цифровизации. Системы подобного рода не только облегчают жизнь пользователям, но и содействуют более рациональному использованию ресурсов, что актуально в условиях глобальной экологической повестки.

Глава 1 Анализ предметной области и постановка задачи

1.1 Характеристика сферы ЖКХ и систем передачи показаний

Сфера жилищно-коммунальных хозяйства ЖКХ является необходимой частью инфраструктуры любого города. Эта сфера охватывает большой спектр услуг таких как водоснабжение, поставки электричества уборка улиц и другие услуги.

Вода распределяется по городу из Реки волги на водосборные станции и проходит несколько этапов отчистки. Частая вода поступает на водопроводную сеть города и там уже распределяется по жилым домам. Водоснабжение города связанно с унитарными и коммунальными службами, которые в свою очередь являются муниципальными и государственными предприятиями. Также к этим предприятиям также относиться водосборные станции и их зона ответственности включает в себя отчистку воды и распределению её по городу. Компании, которые осуществляют расчет и принимаю оплату за коммунальные услуги связанны с управлением водоснабжение через управление платежами, обслуживания домов как ЖКХ и сотрудничеством с водоснабжающими компаниями.

ООО «Квартплата 24» – это IT-компания онлайн-сервис, которая позволяет оплачивать квитанции за коммунальные платежи воды и света, передавать показания индивидуальных приборов учета (ИПУ) и хранить все данные в одном месте.

Рассмотрим структуру организации предприятия.

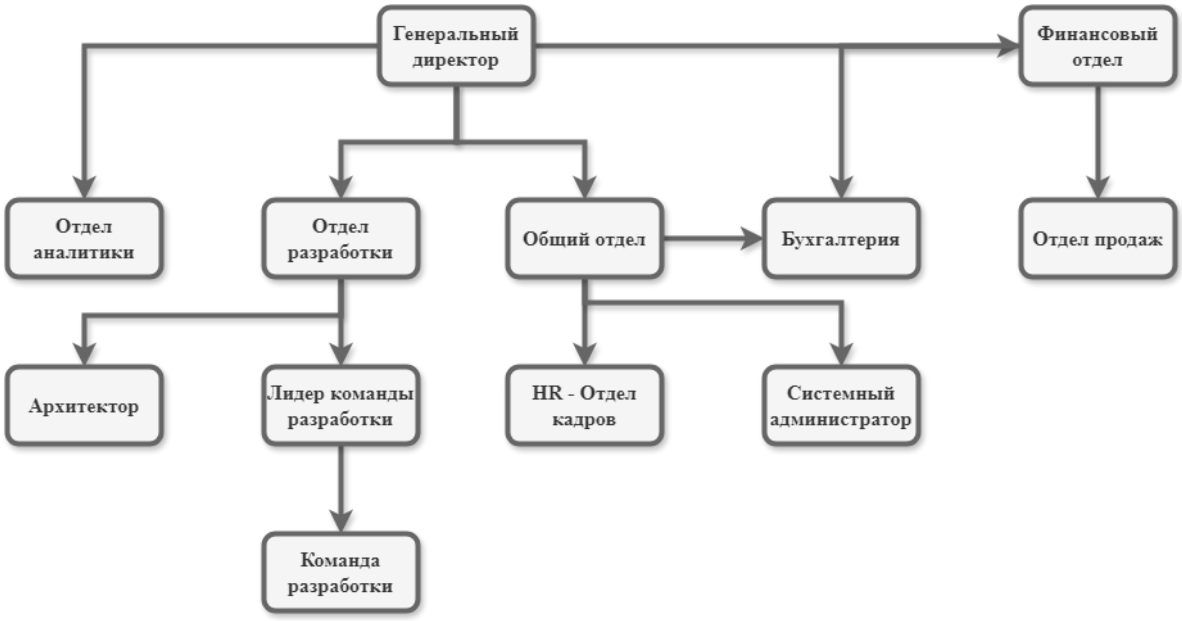


Рисунок 1 – Организационная структура ООО «Квартплата 24»

На рисунке 1 отображается общая структура компании, где в зоне общения генерального директора входят четыре основных отдела: общий отдел, отдел разработки, отдел аналитики и отдел продаж.

Директор отслеживает работу предприятия по созданным бизнес-процессам и принимает общие решения по развитию и изменениях в компании, управляет службами и отделами сотрудников.

Финансовый отдел обращается в общий для решения вопросов основных компании, ведет управление бухгалтерией и отделом продаж.

Бухгалтерия работает с финансовыми расчетами и всеми банковскими операциями. Отдел продаж отвечает за рекламу и продажу информационных услуг в которое выходит разработка и обслуживание программного обеспечения.

1.2 Разработка и анализ учета приборов Модель AS-IS «как есть»

Одной из основных задач будет правильно и корректное моделирование предметной области, где проводиться анализ схемы и

алгоритм процесса компании, наглядного отображения аспектов для последующей работы.

Варианты методологии моллирования представленные в анализе ниже включают в себя: BPMN, ARIS и IDEF0.

BPMN (Business Process Model and Notation) является инструментом для описания бизнес-процессов и понимания их всеми участниками проекта. На данный момент BPMN пользуется большой популярностью, и большинство вендоров систем BPM, которые предоставляют возможность работы с BPMN и создавать бизнес-процессы, интегрировать их в веб-приложения.

ARIS (Architecture of Integrated Information Systems) отличается простотой и легкостью восприятия и использования. Цвет элементов делают модель привлекательной визуально, что особенно важно для понимания и презентации результатов. ARIS позволяет легко визуализировать сложные и параллельные процессы, обеспечивая гибкость в моделировании.

IDEF0 (Integration Definition Function Modeling) разворачивает методологию, в которой модель разворачивается как слева направо, так и сверху вниз, что позволяет выделить основные и главные объекты и их взаимосвязи. IDEF0 подходит для моделирования процессов управления.

Таблица 1 – Сравнительный анализ методологий

Критерии сравнения	BPMN	ARIS	IDEF0
Уровень абстракции	Высокий	Средний	Низкий
Подход к проектированию	Процессный	Объектно-ориентированный	Функциональный
Удобство использования	Сложный	Средний	Простой
Гибкость моделирования	Строгая	Средняя	Неограниченная
Поддержка инструментов	Широкая	Ограниченная	Ограниченная

«Модель IDEF0 разворачивается одновременно слева направо и сверху вниз, по диагонали. Объекты, расположенные левее выше, доминируют над теми, которые находятся правее ниже. Доминирующие объекты могут

включать в себя зависимые: например, доставка заказа – это элемент, входящий в состав более масштабного процесса управления заказами» [7].

Благодаря анализу, проведенному в таблице 1, была подобранна IDEF0 оптимальная модель, которая будет в основе моделирования процесса передачи показаний приборов учета предприятия.

IDEF0 – «это методология графического моделирования процессов, используемая для реализации систем и разработки программного обеспечения. Эти методы используются в функциональном моделировании данных, симуляции, объектно-ориентированном анализе и приобретении знаний.» [20].

В процессе работы организации веб-приложение и компания решают задачи:

- консультация колл-центров и общение с поддержкой сайта в веб-приложении;
- собирают показания на сайте и в очных панках передачи показаний ЖКХ и почты;
- хранят и рассчитывают их клиентов на сумму потребления с учетом тарифов;

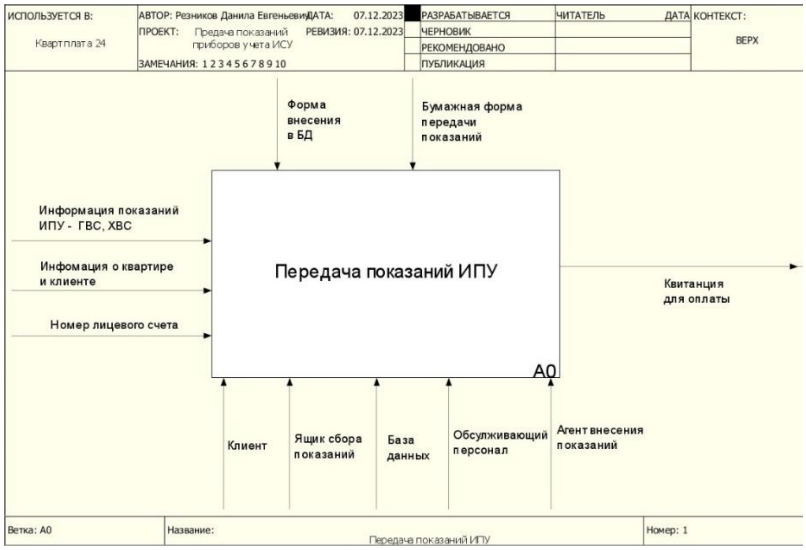


Рисунок 2 – Модель «Как есть» процесса передачи показаний.

«Отдельно следует заметить, что в рамках проводимых преобразований одно из важнейших мест должно отводиться работе с людьми. Без воли поддержки высшего руководства и без желания что-то изменить, разработанные модели бизнес-процессов могут так и остаться лишь описаниями с красивыми картинками, по которым никто не работает» [1].

В данный процесс на рисунке 2 входят сами показания за горячую и холодную воду с приборов учета ГВС и ХВС, информация о квартире потребителя и сведенье о нем прикрепленные к лицевому счету.

Декомпозиция блока А1 позволит увидеть процесс в развернутом виде и развитие на подпроцессы на рисунке 3.

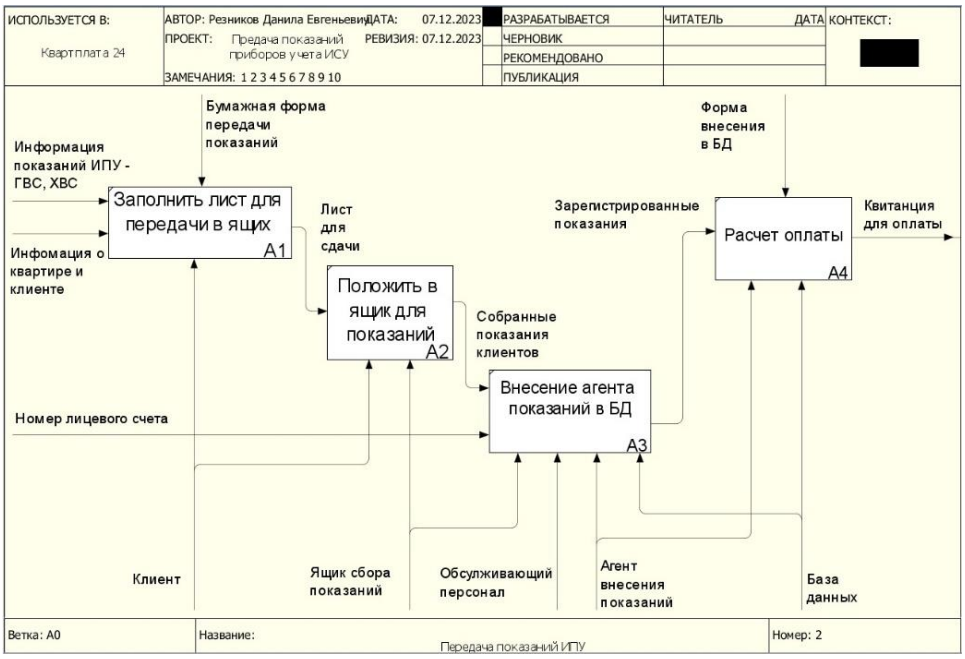


Рисунок 3 – Декомпозиция процесса передачи показаний клиента «как есть»

Процесс А1 «Заполнить лист для показаний» представляет, что клиент водоснабжающих услуг приходит на почту и берет лист со специальной формой для заполнения показаний, где прописывается данные ФИО, номер квартиры и лицевой счет.

Процесс А2 «Положить лист в ящик для показаний» подразумевает, передачу в ящик сбора показаний, которые находить в пунктах сбора, ЖКХ и почтах, где последствии они передаются в отдел внесения показной компаний коммунальных платежей.

Процесс А3 «Внесение агента показаний в базу данных» представляет собой процесс, где данные переносятся специальными анкетами внесения показаний в базу данных для массового расчета через информационную систему коммунальных платежей.

Процесс А4 «Расчет оплаты» происходит массовый расчет, после распечатанная квитанция направляется клиенту для оплаты.

На основе диаграммы изображенной на рисунке 3 можно сделать вывод, что главной задачей клиента в данном процессе передать свои показания в необходимом формате, чтобы они были доставлены до агента внесения и учета для занесения их в базу данных, для хранения и последующего расчета.

1.3 Разработка требований к информационной системе на веб - приложении или постановка задачи ИРС по ИПУ

В стандартной модели AS-IS для управления показаниями счетчиков в многоквартирных домах можно выделить ряд значительных недостатков, каждый из которых способствует неэффективности текущей системы сбора и управления данными о коммунальных услугах. Рассмотрим эти недостатки более подробно, оценивая их влияние как на компании-поставщики услуг, так и на конечных пользователей – жителей.

Негибкость в обновлении показаний: одним из основных недостатков модели AS-IS является жесткость в обновлении или изменении показаний после их ввода. В традиционной системе, как только показание счетчика подано, внесение исправлений или обновлений не является простым процессом. Это часто требует ручного вмешательства со стороны

сотрудников коммунальной службы, что может привести к задержкам и увеличению рабочей нагрузки. Эта неэластичность особенно проблематична в случаях, когда начальные показания были зарегистрированы некорректно или когда необходимо обновить показания из-за последующих перекалибровок оборудования для учета.

Невозможность отслеживания исторических данных: модель AS-IS обычно не поддерживает простой доступ к историческим данным о потреблении для каждого пользователя. Это ограничение мешает жителям контролировать свои модели потребления с течением времени, что может быть полезно для управления затратами на энергию и выявления необычных колебаний в потреблении коммунальных услуг. Без доступа к историческим данным жители также оказываются в невыгодном положении при оспаривании счетов или при попытках понять изменения в своих расходах.

Разбросанные точки подачи показаний: жители многоквартирных домов, таких как крупные жилые комплексы, часто сталкиваются с неудобством подачи показаний в нескольких различных местах. Это особенно затруднительно для управляющих имуществом или лиц, владеющих или управляющих несколькими единицами в разных местах. Каждое имущество может иметь свой набор протоколов и точек подачи, что усложняет процесс эффективного управления коммунальными счетами.

Проблемы с подачей показаний на расстоянии: для жителей, которые отсутствуют в своем основном месте жительства, возможно, из-за путешествий или если они живут в другом городе. Модель AS-IS не предлагает практического решения для удаленной подачи показаний. Это отсутствие возможности удаленного доступа вынуждает пользователей либо ехать назад к месту установки счетчика, либо устраивать кого-то другого для подачи показания от их имени, что является неудобным и не всегда

На стандартной модели AS-IS можно выделить около 7-10 существенных недостатков:

– изменение показаний невозможно или будет являться не быстрым

процессом;

- невозможность отследить историю своих показаний;
- наличии нескольких квартир сдача показаний будет находиться в разных пунктах;
- трата времени на очное посещение, при нахождении в другом городе невозможность передачи показаний;
- высокая затрата человеческих ресурсов работы на внесения показаний в информационно-расчетную систему через веб-приложения с бумажных форм;
- физической передачи показаний, возможность потери данных.

1.4 Сравнительный анализ существующих аналогов систем для передачи показаний

Проанализируем и рассмотрим аналоги доступные пользователям коммунальных услуг. Это необходимо для разработки примерного плана моделирования новой системы на основе существующих вариантов с учетом обновления и внесения своих улучшений и учета ошибок других проектов.

«Квартплата 24» – веб-приложение, сервис с название информационно-расчетной системы, которая позволяет оплачивать квитанции за квартиру, передавать показания за свет и воду, отслеживать задолженности и хранить все данные о пользователях.

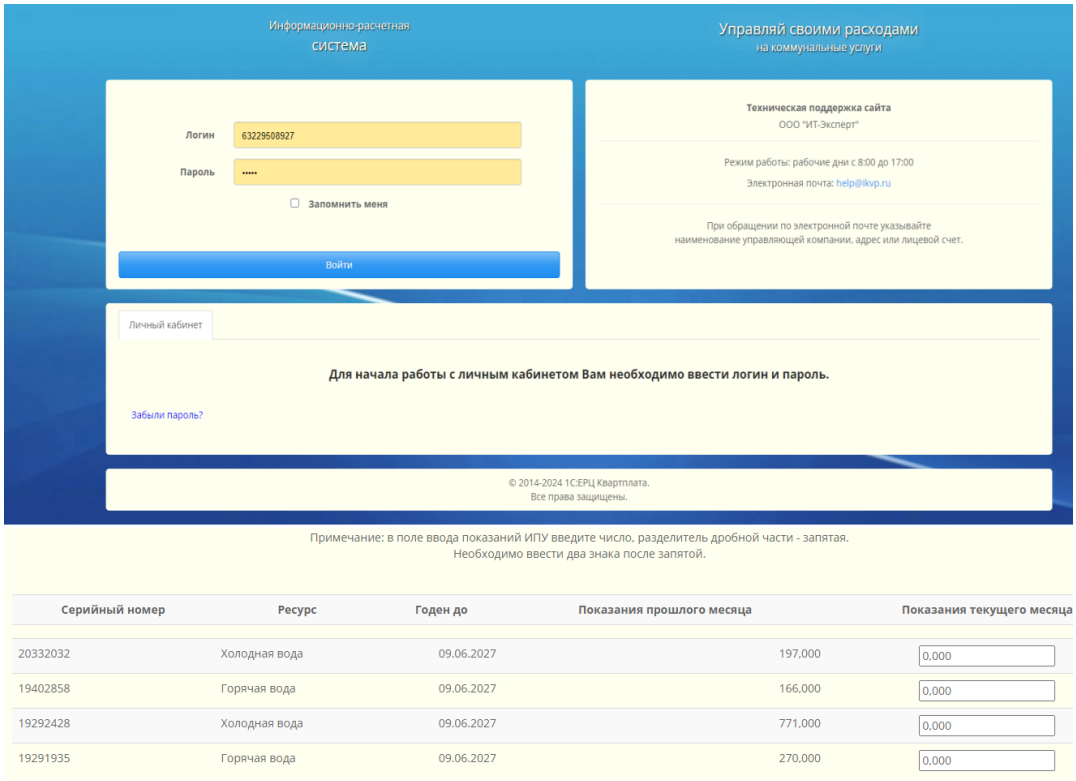


Рисунок 4 – Сайт ИРС «Квартплата 24»

На рисунке 4 сервис располагается по адресу https://ikvp.ru/, и представляет собой сервис для передачи показаний за воду и свет. Ряд преимуществ, которые он может предоставить:

– удобная авторизация на сайте,
– небольшое количество кнопок с URL адресами,
– возможность отследить историю показаний.

Существенные недоставки, которые нуждаться в доработки:

– непредусмотренная возможности хранить несколько квартир на одном лицевом счете;
– плохой дизайн, что делает вид очень сложным для восприятия;
– нет виджета с технической поддержкой.

ЭНЕРГОСБЫТ t+ – сайт веб-приложение принадлежит объединенной энергосбытовой компании АО «ЭнергосбыТ Плюс» группы «Т Плюс» с филиальной сетью из 14 региональных филиалов на территории Российской

Федерации. Здесь можно передать показания, не регистрируясь и не авторизовываясь в системе, через виджет с указание лицевого счета.

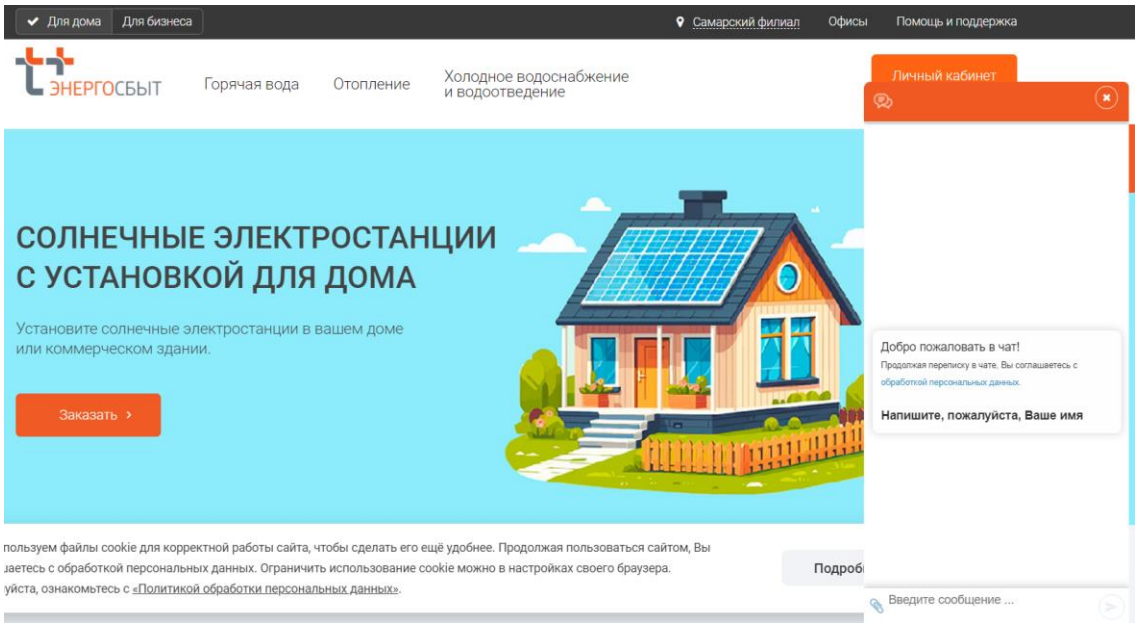


Рисунок 5 – Сайт придачи показаний горячей воды «ЭНЕРГОСБЫТ t+»

На рисунке 5 сайт, придачи показаний электроэнергии, расположение по адресу https://samara.esplus.ru и достоинства, которыми может предоставить данный сервис состоит в том, что там есть:

– возможность задавать показания не авторизируешь на сайте;
– удобная авторизация;
– эргономичный и красивый интерфейс;
– узкоспециализированный и собирает показания за горячую воду, где также видеться расчет тепловой энергии, которая была затрачена на разогрев воды.

Недостатки:

– отделенный сервис, который может повторно дублировать показания на других сервисах по сдаче показаний за ИПУ воды;
– нет возможности хранения нескольких квартир на одном аккаунте.

Это сайт компании ПАО «Самараэнерго», которая, предлагает брать в

аренду оборудование для проведения различных видов измерений. Так же показания за электричество можно сдавать у на данном сервисе.

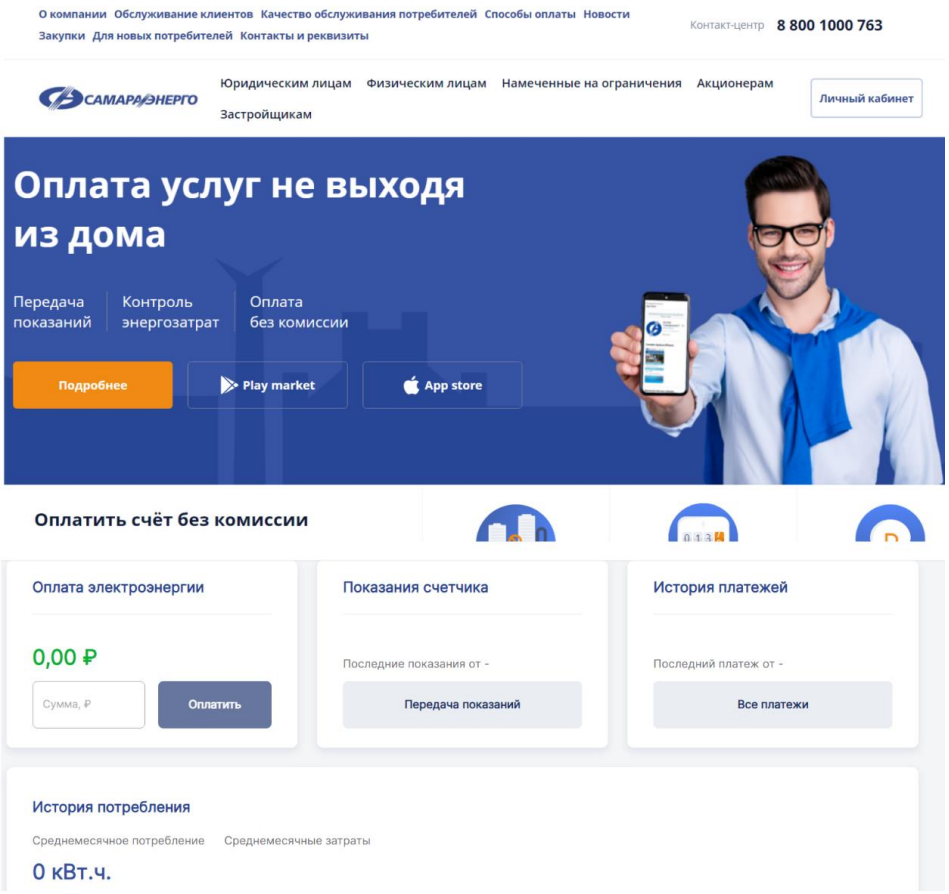


Рисунок 6 – Система передачи показаний «Самара Энерго»

Интерфейс сайта «Самара Энерго» изображен на рисунке 6 и расположен по адресу https://samaraenergo.ru/.

Преимущества:

– простая и быстра авторизация;
– очень мало функций и кнопок, что облегчает понимание работы с сервисом.

Недостатки:

– непредусмотренная возможности хранить несколько объектов недвижимости;

– большое количество сбоев в работе сайта и временное отключение;

Таблица 2 – Сравнения аналогов

Требования	Ikvp.ru	T+ Энергосбыт	Самара Энерго
Хранения несколько квартир на одном аккаунте	Один лицевой счет на аккаунте	Один лицевой счет на аккаунте	Один лицевой счет на аккаунте
Удобный эргономичный интерфейс	Средний	Сложный	Легкий
Отслеживание истории показаний	История в виде таблицы	Нет истории	История в виде диаграммы
Получение расчетной суммы для оплаты	Оплата на сайте	Нет оплаты	Оплата на сайте
Итого	2/4	0/4	3/4

Сравнив три существующих варианта из таблицы 2, можно сделать общий вывод, что причисленные веб-приложения имеют несущественные недоставки, на которые, важно, обратить внимание, для удобства клиента.

1.5 Разработка модели бизнес-процесса «как должно быть»

Разработка информационно-расчётной системы значительно упрощает и оптимизирует процесс обработки данных и общение с клиентами. Благодаря интеграции веб-приложения, процедура регистрации пользователей и передачи показаний становится более доступной и удобной для всех сторон. Это позволит снижать время на обработку данных и уменьшает вероятность ошибок, так как информация теперь автоматически обрабатывается системой.

Рисунок 7 – Контекстная модель «Как должно быть»

На рисунке 7 наглядно отображен процесс, где клиенты получают возможность регистрации на сайте, это дает им возможность самостоятельно вносить показания приборов учета в любое время и из любого места. Все данные, сразу же попадают в единую базу, что облегчает дальнейший расчет оплаты и формирование квитанций. Также клиенты имеют прямой доступ к истории своих показаний и квитанций, что делает систему более прозрачной и понятной для пользователей.

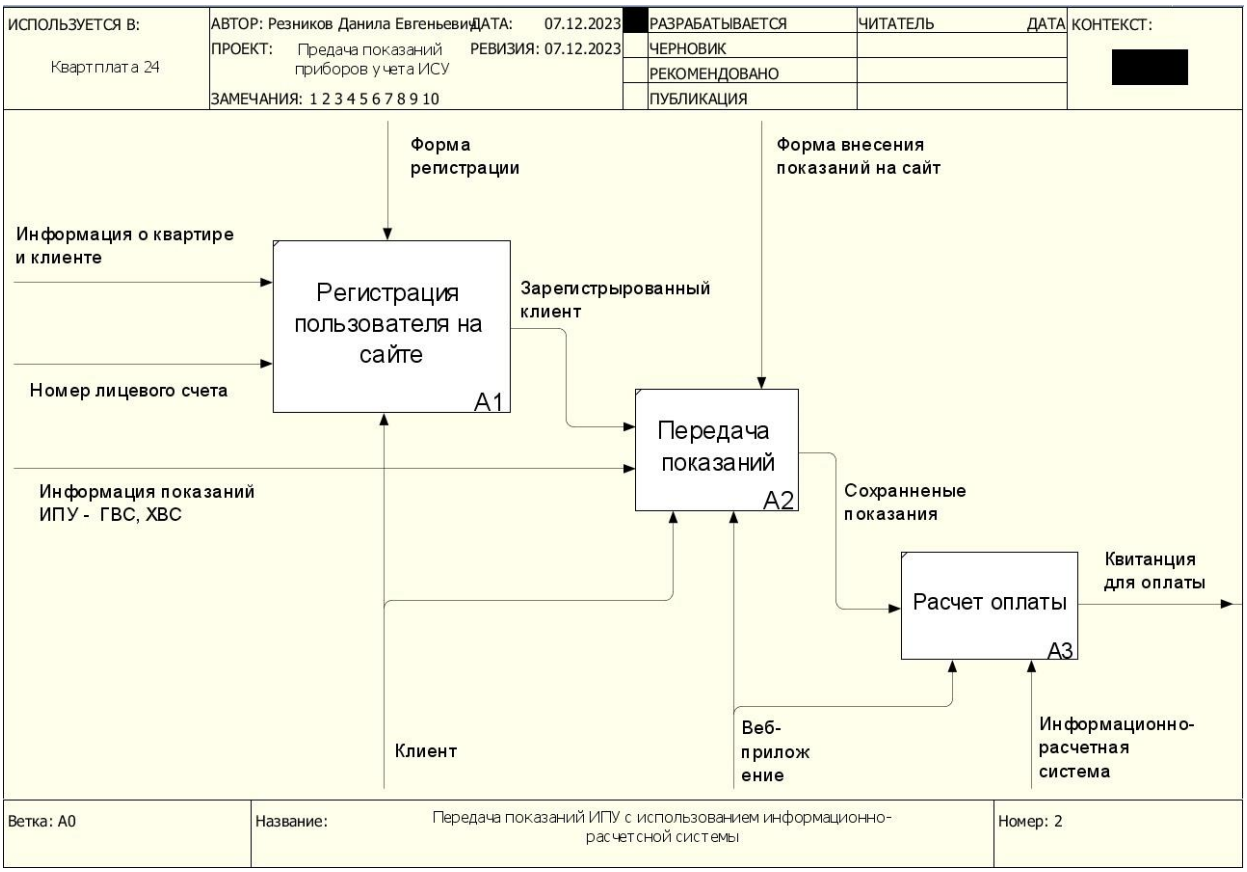


Рисунок 8 – Декомпозиция контекстной модели «Как должно быть»

С точки зрения управления, это позволяет компании быстро реагировать на изменения в показаниях, а также упрощает процесс создания счетов и общение с клиентами. Внедрение этой системы обеспечивает мгновенное обновление данных, что минимизирует вероятность ошибок в блоке A2 на рисунке 8. Кроме того, введение веб-приложения повышает удобство использования сервиса, что может привести к повышению удовлетворенности клиентов, так как они смогут легко получать доступ к необходимой информации и управлять своими счетами онлайн. Это особенно важно в условиях высокой конкуренции, где качество и скорость обслуживания клиентов играют ключевую роль.

В целом, разработка информационно-расчетной системы и веб-приложения в бизнес-процесс компании создает более стройный,

автоматизированный и эффективный рабочий процесс, позволяющий сэкономить время и ресурсы, а также предоставляет более высокое качество услуг для пользователей. Это способствует не только улучшению внутренних процессов компании, но и повышению ее конкурентоспособности на рынке, так как позволяет предоставлять клиентам более надежный и удобный сервис.

Выводы по первой главе

Первая глава отображает представление функционального моделирования предметной области. Проведен анализ технологии концептуального моделирования. При этом была разработана и проанализирована модель бизнес-процесса «как есть». Как следует, проанализированы аналоги существующих систем передачи показаний ИПУ, выявлены их достоинства и недостатки. На основе существующих процессов, составлен план и набросок будущего моделирования логической и физической модели данных. Эти действия позволили выявить основные направления для дальнейшего совершенствования системы и определили ключевые моменты, требующие внимания в ходе реализации проекта.

Глава 2 Логическое моделирование веб-приложения для работы с передачей показаний коммунальных услуг

2.1 Выбор технологии логического моделирования веб-приложения

«Логическое моделирование основано на анализе возможных исходов вследствие изменения отдельных элементов в состоянии управляемой системы, при функционировании которой сложилась конкретная ситуация.

Такое моделирование представляет собой неотъемлемую составную часть традиционной системы управления в области принятия решений только на основе мысленного обобщения полученной технической или технико-экономической информации. Общение с памятью ЭВМ при данном методе проводится редко, использование этого метода не позволяет, как правило, решать задачи по текущему отысканию и согласованию технического и экономического оптимумов, поскольку традиционными методами сделать это невозможно» [6].

В контексте веб-приложения логическое моделирование подразумевает собой создание модели приложения и его информационной базы на основе концептуальной модели «как должно быть».

Это несомненно является неотъемлемым этапом для принятий решений на основе стандартных систем управления мысленного обобщения полученной технико-экономической информации.

Для этого применяют диаграммы вариантов использования, вариантов деятельности, классов, последовательности и UML Использование CASE-инструментов позволит повысить эффективность работы, сократить время и уменьшить количество ошибок в проекте.

StarUML - один из программных инструментов моделирования с открытым исходным кодом, который поддерживает стандартизированный UML для моделирование систем и программного обеспечения.

После выбора технологии логического моделирования, приступаем к

разработке логической модели веб-приложения для работы с ИРС по внесению показаний через веб-приложение.

2.2 Разработка логической модели веб-приложения

Логическая модель информационно-расчетной системы индивидуальных приборов учета (ИРС ИПУ) показывает взаимодействия между функциональными возможностями ИРС и пользователями системы. Модель разрабатывается с учетом требований, предъявляемых к системе со стороны ее пользователей, и предполагает определение основных сценариев использования ИРС.

«Диаграмма вариантов использования в UML — диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему» [5].

Реализация функциональных требований в модели:

Функциональные требования реализованы в виде сценариев использования (Use Cases), которые включают:

Вход в систему: Аутентификация клиентов и администраторов для доступа к функционалу ИРС.

Передача показаний: Процесс ввода данных о потреблении ресурсов пользователями.

Просмотр истории потребления: Возможность анализа исторических данных о потреблении для оптимизации использования ресурсов.

Расчет по тарифу: Автоматизированный расчет суммы к оплате на основе введенных показаний и действующих тарифов.

Создание квитанции: Формирование документов для оплаты

потребленных ресурсов.

Коррекция данных: Исправление ошибочно введенных данных в показаниях.

Чат с администратором: Общение клиентов с администраторами для решения возникающих вопросов.

Подтверждение лицевого счета: Процедура верификации данных клиента для обеспечения надежности системы.

«Диаграмма деятельности – UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнен и подчинённых элементов – вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого» [4].

Регистрация в ИС: Включение новых пользователей в систему.

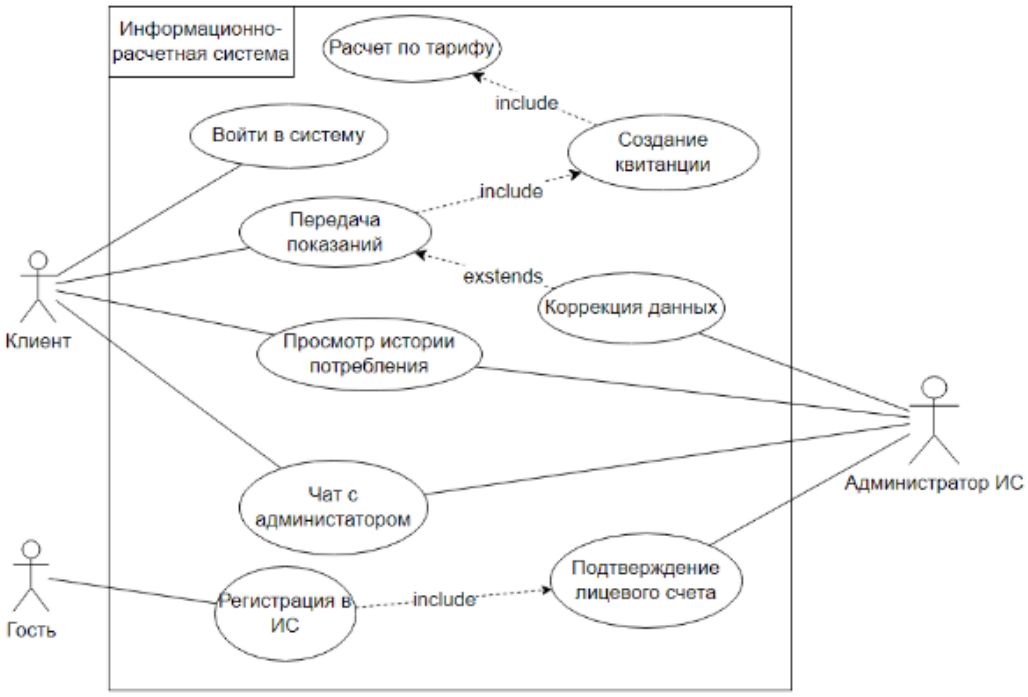


Рисунок 9 - UML Case диаграмма.

На диаграмме рисунка 9 вариантов использования представлены роли администратора ИС, клиента и гостя, которые будут распределять некоторое количество запросов по правам доступа.

– клиент – который набирает URL веб-приложения и может использовать функционал передачи показываний просмотра истории своего потребления, обраться в поддержку на сайте;
– администратор ИС – пользователь (работник) компании «Квартплата 24» может регистрировать клиента, добавить ему место проживания с его счетчиками, отредактировать результаты ошибочных показаний или передать показания вместо клиента, если у него нет возможности седеть это через веб-приложение;
– гость – незарегистрированный пользователь или пользователь иных услуг, которые не внес в систему как клиент или дожидаться регистрации, а пока не может пользоваться функционалом сайта.

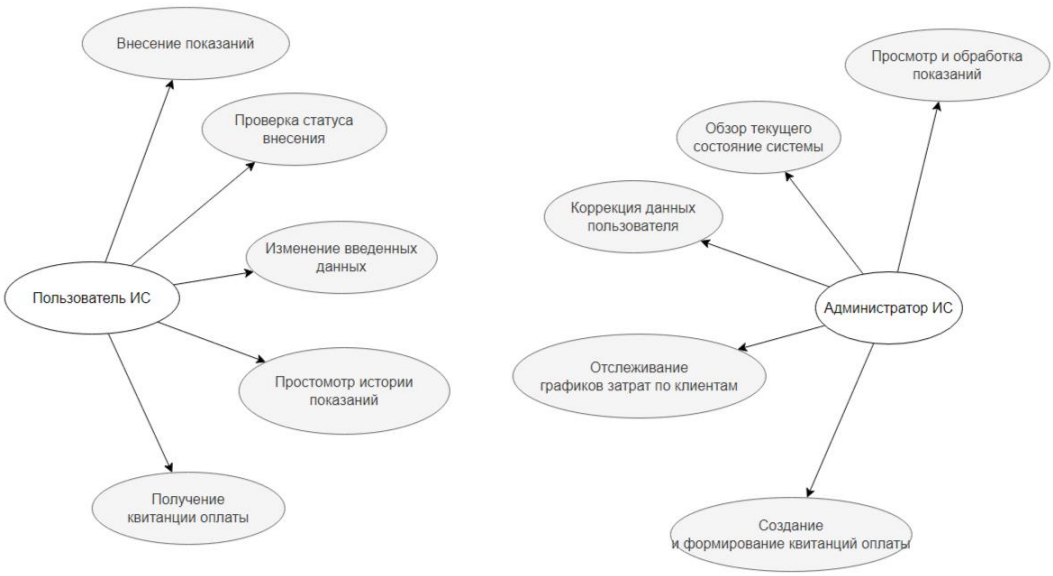


Рисунок 10 - Идентификация опорных точек зрения

Внутри информационной системы действия пользователя взаимосвязаны и упорядочены в алгоритмах, разбиваться на некоторые действия и зависят от ролей, которыми обладает система.

Из рисунка 10 видно, что гость, который посещает систему должен зарегистрироваться и дождаться подтверждения лицевого счета у администратора системе.

2.3 Требования к аппаратно-программному обеспечению веб-приложения

Для обеспечения бесперебойной работы веб-приложения Информационно-расчетной системы индивидуальных приборов учета (ИРС ИПУ) необходимо соблюдение определенных требований к аппаратно-программному обеспечению. Система должна максимально эффективно использовать имеющиеся ресурсы для достижения высокого уровня производительности и надежности.

Компоненты технического комплекса:

Сервер СУБД: является основой для хранения, обработки и защиты данных.

Персональный компьютер (ПК) пользователей: используется для доступа к функционалу ИРС через веб-интерфейс.

Мобильное устройство пользователя: обеспечивает доступ к системе в мобильном режиме.

Минимальные требования к серверу СУБД:

Процессор: С частотой не менее 2.4 ГГц.

Оперативная память (ОЗУ): не менее 4 Гб.

Свободное место на жёстком диске: не менее 25 Гб.

Интернет-соединение: для обеспечения доступа к серверу.

Операционная система: Windows или любая другая с поддержкой требуемого стека технологий.

Минимальные требования к ПК пользователя:

Процессор: С частотой от 2 ГГц.

Оперативная память (ОЗУ): не менее 2 Гб.

Интернет-соединение: для доступа к веб-приложению.

Операционная система: Любая популярная ОС (Windows, Linux, MacOS и др.).

Веб-браузер: Любой современный браузер (Google Chrome, Mozilla Firefox, Яндекс Браузер и др.).

Выводы по второй главе

Во второй главе был выполнен детализированный анализ логического моделирования веб-приложения для работы с передачей показаний коммунальных услуг. Рассмотрены и выбраны технологии, подходящие для разработки логической модели, включая диаграммы вариантов использования и классов, что обеспечило четкое понимание функциональных требований к системе.

Была разработана логическая модель веб-приложения ИРС ИПУ, отражающая взаимодействия между функциональными возможностями системы и её пользователями. Использование диаграмм вариантов использования позволило определить основные сценарии работы пользователей и администраторов, а также структурировать процессы подачи показаний, просмотра истории потребления и взаимодействия с системой.

Таким образом, вторая глава представила комплексный подход к логическому моделированию системы, обеспечив основу для её дальнейшей реализации и интеграции в существующую инфраструктуру ЖКХ.

Глава 3 Реализация информационно-расчетной системы на веб-приложении

3.1 Выбор инструментов разработки веб-приложения

В разделе рассматриваются основные средства и платформы, которые были выбраны для создания веб-приложения, способного эффективно обрабатывать пользовательские данные и интегрироваться с информационно-расчетной системой.

Java – «это объектно-ориентированный язык, основанный на классах, который разработан для переносимости, что означает, что Java код может работать на различных аппаратных средствах и операционных системах. Java широко используется для разработки приложений корпоративного уровня, мобильных приложений, видеоигр и других типов программного обеспечения. Он известен своей философией "напиши один раз, запусти где угодно", поскольку код Java может быть скомпилирован для запуска на любой платформе, поддерживающей виртуальную машину Java (JVM).» [11] Этот язык программирования идеально подходит для написания веб-приложения.

Spring Framework – «фреймворк с открытым исходным кодом, написанный на Java. Его можно использовать для разработки на всех этих языках. Spring предоставляет огромный набор инструментов и библиотек, которые упрощают и ускоряют процесс разработки, позволяя сосредоточиться на бизнес-логике приложения.» [2]

Для начала разработки необходимо выбрать инструменты, которые позволят реализовать поставленные задачи с оптимальным балансом производительности, удобства использования и поддержки. Среди множества доступных вариантов был сделан выбор в пользу IntelliJ IDEA и Visual Studio Code в связке с Spring Framework и Lombok для создания серверной части приложения.

«Microsoft Visio – программный продукт, часть пакета Microsoft Office и предназначенный для построения диаграмм и работы с данными» [12].

IntelliJ IDEA выбрана как основная интегрированная среда разработки (IDE) из-за её высокого уровня интеграции с Spring Framework, обширных возможностей по управлению зависимостями и встроенных инструментов для рефакторинга и отладки кода. Эта среда разработки обеспечивает удобную работу с кодом, поддерживает плагины для работы с базами данных и предоставляет обширные аналитические функции, которые помогут в оптимизации и устранении потенциальных проблем.

«Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, UWP а также веб-сайты, вебприложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework и Silverlight» [17].

Spring Framework был выбран как основа для серверной части приложения, поскольку он предоставляет удобные абстракции для создания веб-приложений и REST API, интегрируется с системами управления базами данных и поддерживает модель программирования, основанную на инверсии управления (IoC), что упрощает управление жизненным циклом компонентов приложения.

Lombok используется для уменьшения бойлерплейт-кода, автоматической генерации геттеров, сеттеров, методов equals и hashCode, а также конструкторов, что делает код более читаемым и уменьшает вероятность возникновения ошибок в результате ручного копирования и вставки шаблонного кода.

Для возвращения представлений в методах контроллера используются

шаблоны Thymeleaf, позволяющие создавать динамические HTML-страницы на стороне сервера. Thymeleaf обладает возможностью интеграции с Spring MVC и поддерживает естественные шаблоны, что позволяет веб-дизайнерам и разработчикам работать с визуально нормальными HTML-страницами, которые затем могут быть легко интегрированы с серверной логикой.

Итак, выбор инструментов разработки для данного веб-приложения направлен на создание гибкой, масштабируемой и легко поддерживаемой платформы, что является ключевым фактором для обеспечения долгосрочного успеха проекта.

3.2 Разработка физической модели данных веб приложения

Разработка физической модели данных для веб-приложения — это процесс создания структуры базы данных, которая будет использоваться для хранения и управления информацией, необходимой для работы веб-приложения. Физическая модель данных не только определяет, как данные будут храниться в базе данных, но и как они будут взаимодействовать друг с другом, что напрямую влияет на производительность и масштабируемость приложения.

«Система баз данных – это, по сути, не что иное, как компьютеризированная система хранения однотипных записей» [3].

«PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird. PostgreSQL может похвастаться поддержкой uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, 36 битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов» [10].

«MySQL – свободная реляционная система управления базами данных.

Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря

такому заказу почти в самых ранних версиях появился механизм репликации» [18].

«Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка» [16].

«Oracle Database – это объектно-реляционная система управления базами данных (СУБД) от компании Oracle. Она используется для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации».

«Firebird (FirebirdSQL) – свободная кроссплатформенная реляционная система управления базами данных, работающая на macOS, Linux, Microsoft Windows и некоторых Unix-платформах» [14].

В результате анализа существующих баз данных и СУБД была выбрана наиболее подходящая система PostgreSQL, к которому существует дополнительная технология использования Java кода для генерации SQL запросов через среду выполнения с использованием Spring Framework.

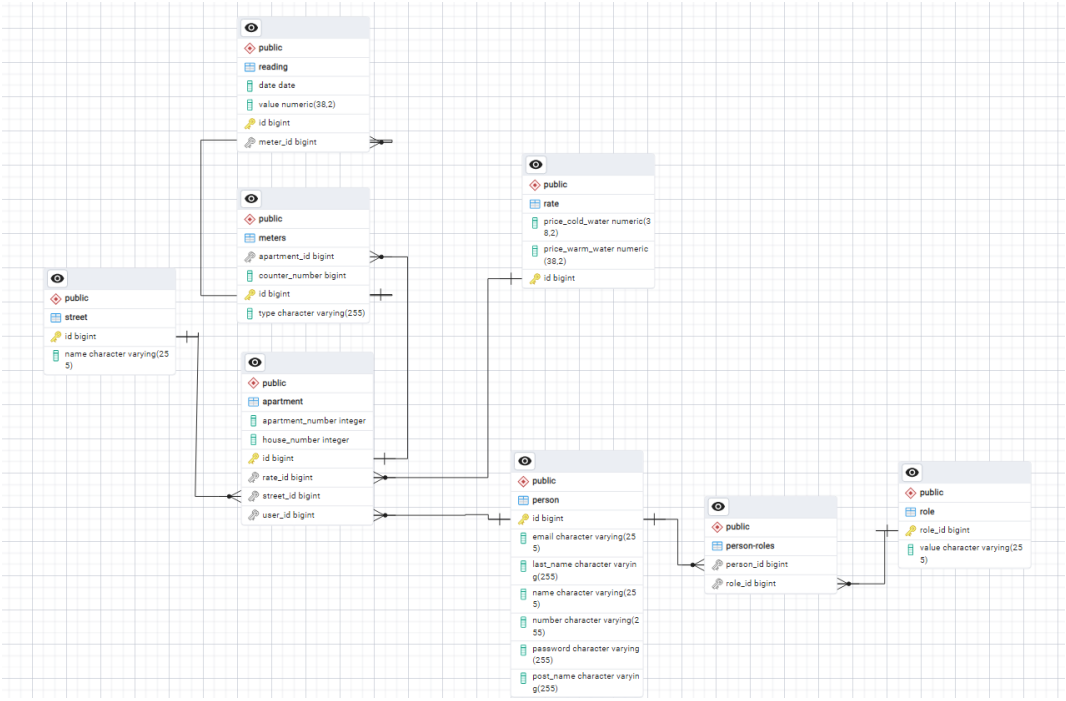


Рисунок 11 – Физическая модель данных, ERD диаграмма PostgreSQL

На рисунке 11 представлена схема взаимосвязей между различными таблицами базы данных. Каждая связь означает отношение между таблицами, «атрибуты – это характеристики сущности, отношения «многие ко многим» или отношения «один к одному». [13]

Внешние ключи (street_id, user_id, rate_id, apartment_id, meter_id, person_id, role_id) соединяют данные между собой, обеспечивая целостность и логическую структуру данных в базе.

3.3 Разработка веб-приложения для передачи показаний ЖКУ

В этой главе описывается процесс разработки веб-приложения для аналитики данных счетчиков учета ресурсов. Веб-приложение использует Spring Boot, что позволяет создать стандартизированное и легко масштабируемое приложение.

Конфигурация Maven (pom.xml):

Spring Boot Starter Parent: определяет базовую конфигурацию и

устанавливает версию Spring Boot, что гарантирует совместимость и правильное управление зависимостями.

Java Person: указывает на версию Java, используемую в проекте. Здесь это Java 21, что обеспечивает доступ к последним возможностям языка.

Dependencies: Секция зависимостей определяет необходимые библиотеки и фреймворки.

Spring Boot Starter Freemarker: предоставляет интеграцию с шаблонизатором Freemarker для генерации представлений.

Spring Boot Starter Security: добавляет безопасность на основе Spring Security, защищая веб-приложение от неавторизованных доступов.

Thymeleaf Extras Springsecurity6: подключает функции Thymeleaf для работы с Spring Security, позволяя контролировать элементы на страницах в зависимости от ролей и прав пользователей.

Spring Boot Starter Thymeleaf: включает поддержку шаблонизатора Thymeleaf для создания динамических HTML-страниц.

Spring Boot Starter Web: интегрирует основные компоненты для создания веб-приложения, такие как Spring MVC.

Spring Boot Starter Data JPA: обеспечивает интеграцию с JPA для работы с базами данных, упрощая работу с персистентностью данных.

PostgreSQL: Драйвер для подключения к базе данных PostgreSQL, который используется в качестве системы управления базой данных.

Lombok: упрощает разработку за счет генерации стандартного кода, такого как getters/setters, конструкторы и т.д.

Spring Boot Starter Test и Spring Security Test: предоставляют инструменты для тестирования веб-приложения и компонентов Spring Security.

На рисунке 12 изображена модель таблицы apartment также с помощью аннотаций в этом классе прописываться геттеры и сеттеры.

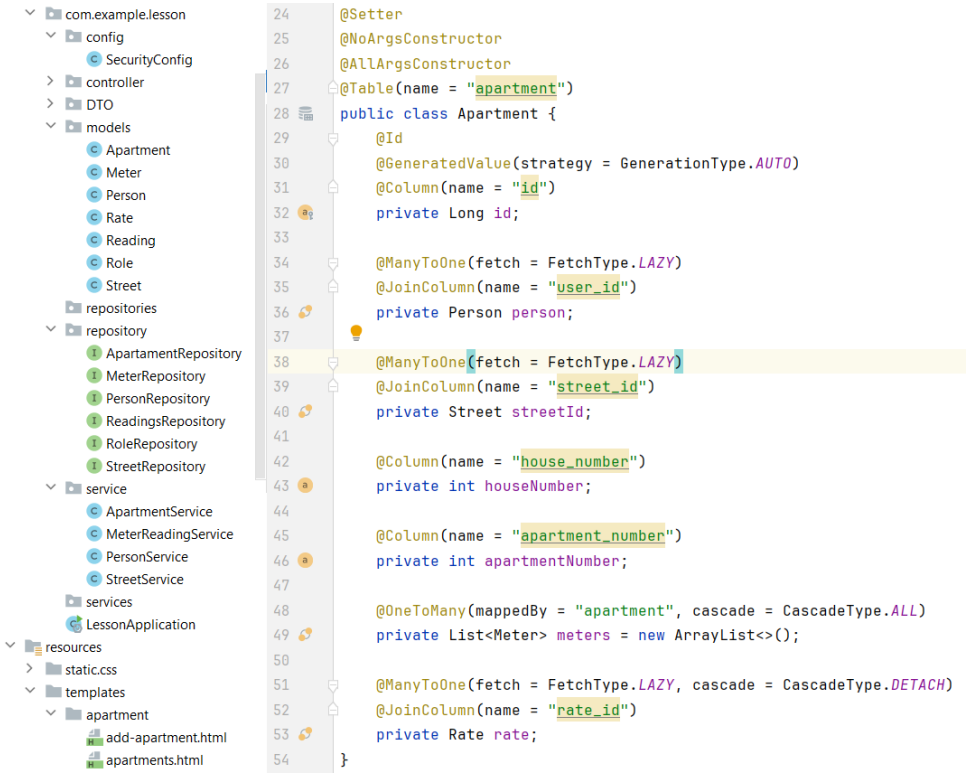


Рисунок – 12 Код файла Apartment

На рисунке 12 изображён контроллер в контексте веб-приложения, основанного на модели MVC (Model-View-Controller), является критически важным компонентом, который действует как посредник между моделью (данными), представлением (интерфейс пользователя) и запросами, отправляемыми клиентом. В роли координатора, контроллер обрабатывает входящие запросы, делегирует задачи моделям для получения/изменения данных и определяет, какое представление должно быть отображено пользователю.

«REST API не сильно отличаются от веб-сайтов. Оба они отвечают на HTTP-запросы. Но ключевое отличие заключается в том, что вместо того, чтобы отвечать на запросы с помощью HTML, как это делают веб-сайты, REST API обычно отвечают с помощью формата, ориентированного на данные, такого как JSON или XML. [19]

На рисунке 13 изображён класс MeterReadingController, который представляет собой контроллер в веб-приложении, использующем Spring Framework. Этот контроллер обрабатывает запросы, связанные с показаниями счетчиков учета.

```
@Controller
@RequiredArgsConstructor
@RequestMapping("/meter-readings")
public class MeterReadingController {

    private final MeterReadingService meterReadingService;
    private final StreetService streetService;
    private final PersonService personService;
    private final ApartmentService apartmentService;

    Danila Reznikov
    @PostMapping("/add")
    public String submitMeterReadings(@RequestParam Map<String,String> allParams) {
        meterReadingService.saveReading(allParams);
        return "redirect:/meter-readings";
    }
    Danila Reznikov
    @GetMapping()
    public String showMeterReadingsForm(Model model) {
        Map<Apartment, List<Meter>> meters = meterReadingService.findPersonMeters();
        List<Reading> readings = meterReadingService.findPersonReadings();
        Person user = personService.getUser();
        List<Apartment> apartments = apartmentService.getApartmentsByPerson();
        Map<Apartment, Street> apartmentsWithStreets = streetService.getAllStreetByApartments(apartments);

        model.addAttribute( attributeName: "apartmentsWithStreets", apartmentsWithStreets);
        model.addAttribute( attributeName: "user",user);
        model.addAttribute( attributeName: "user",user);
        model.addAttribute( attributeName: "meters", meters);
        model.addAttribute( attributeName: "readings", readings);
        return "site/meter_readings";
```

Рисунок – 13 Код файла MeterRaedingController

Обработка запросов на добавление данных: Метод submitMeterReadings() с аннотацией @PostMapping обрабатывает POST-запросы по определенному URI (в данном случае "/add"). Этот метод принимает параметры из запроса, передаёт их в сервис MeterReadingService для сохранения и перенаправляет пользователя на страницу с показаниями счетчиков (redirect:/meter-readings).

Отображение данных пользователю: Метод showMeterReadingsForm () с аннотацией @GetMapping обрабатывает GET-запросы, отвечающие за

отображение

На изображении 14 представлен фрагмент кода класса MeterReadingService, который является сервисом в рамках архитектуры Spring Framework. Этот сервис отвечает за логику извлечения информации о показаниях приборов учета, связанных с конкретным пользователем.

Рисунок 14 – Код файла MeterReadingService

Для доступа к базе-данных создаться интерфейс репозитории с использование Spring Data JPA, который дает возможность генерировать SQL запросы к данным. На изображении 15 представлен интерфейс StreetRepository, который расширяет JpaRepository из Spring Data JPA. Этот репозиторий определяет методы для работы с объектами Street в базе данных:

«JPA означает Java Persistence API. Это платформа объектно-реляционного сопоставления (ORM), которая позволяет нам сопоставлять объекты Java с таблицами в реляционной базе данных. Другими словами, JPA предоставляет способ сохранения объектов Java в базе данных с помощью

набора аннотаций, определяющих сопоставление между классами Java и таблицами базы данных» [15].

```
6   import org.springframework.data.jpa.repository.JpaRepository;
7
8   import java.util.List;
9   import java.util.Optional;
10
        Danila Reznikov
11  public interface StreetRepository extends JpaRepository<Street, Long> {
12
            Danila Reznikov
13      public Street findAllById(Apartment apartment);
14
            Danila Reznikov
15      public Street findAllById(Long id);
16
            Danila Reznikov
17      public Street findByApartments(Apartment apartment);
18
19  }
```

Рисунок 15 – Код репозитория StreetRepository

На представленном рисунке 15 демонстрируется фрагмент кода HTML-шаблона, который использует Thymeleaf – современный серверный Java-шаблонизатор, обеспечивающий связь данных на сервере с HTML-шаблонами в браузере. Thymeleaf позволяет вам использовать HTML-файлы в качестве фронтенд-шаблонов, поддерживающих динамическое добавление содержимого на стороне сервера.

В данном шаблоне Thymeleaf используется для отображения формы и таблицы:

Форма для отправки показаний счетчика:

Использует Thymeleaf синтаксис th:action="@{/meter-readings/add}" для определения URL, на который форма будет отправлена методом POST.

th: each="meter, meterStat : ${meters}" пробегает по коллекции meters,

переданной из контроллера, и для каждого элемента в коллекции генерирует соответствующие поля ввода.

th:text и th:value используются для вывода данных счетчиков, таких как номер счетчика и текущие показания.

Таблица истории показаний:

th: each="reading : ${readings}" перебирает список показаний, переданный из контроллера, для отображения в таблице истории.

```
<section class="meter-reading-container">
    <div th:each="entry : ${meters}">
        <h2 class="section-nav-header" th:text="'ИН квартиры'+${entry.key.getId()}"></h2>
        <!-- Ваш action должен указывать на URL, который обрабатывается вашим методом POST в контроллере -->
        <form th:action="@{/meter-readings/add}" method="post">
            <div th:each="meter, meterStat : ${entry.value}">
                <label th:text="${'Номер счетчика ' + meter.getCounterNumber()}"></label>
                <input type="text" th:id="'reading-' + ${meter.getId()}" th:name="${meter.getCounterNumber()}" placeholder="Показания" required><br>
            </div>
            <button type="submit">Отправить</button>
        </form>
    </div>
    <p>Прием показаний разрешен с 18 до 23 числа текущего месяца.</p>
</section>
<section class="meter-history-container">
    <h2 class="section-nav-header">История внесения показаний ИПУ</h2>
    <table>
        <thead>
        <tr>
            <th>Номер счетчика</th>
            <th>Месяц</th>
            <th>Показания</th>
        </tr>
        </thead>
        <tbody>
        <tr th:each="reading : ${readings}">
            <td th:text="${reading.meter.counterNumber}">Номер счетчика</td>
            <td th:text="${#temporals.format(reading.date, 'MMMM')}">Месяц</td>
            <td th:text="${reading.value}">Показания</td>
        </tr>
```

Рисунок 16 – Форма для отправки показаний ИПУ за воду

Внутри таблицы для каждого reading отображаются: номер счетчика, дата показаний (форматируется с помощью #temporals) и значение показаний.

Thymeleaf предоставляет естественные шаблоны на рисунке 16, позволяя разработчикам использовать обычный HTML для разработки веб-страниц, который также может быть динамически преобразован на сервере. Объекты, такие как meters и readings, помещаются в модель (Model) в контроллере и передаются в шаблон, где Thymeleaf обращается к этим объектам.

3.4 Разработка двух точек зрения по ролям и администрирование ADMIN и USER для распределения деятельности и доступа.

На изображении 17 показан фрагмент Java-кода, который является частью конфигурации безопасности для Spring-приложения. В этом классе SecurityConfig определены настройки для обеспечения безопасности веб-приложения.

Рисунок 17 - Класс SecurityConfig для авторизации пользователей

Класс SecurityConfig:

Аннотирован с @EnableWebSecurity, что активирует поддержку безопасности веб-сервера и Spring Security.

@EnableGlobalMethodSecurity (prePostEnabled = true) позволяет использовать аннотации для проверки безопасности на уровне методов, например @PreAuthorize.

Bean SecurityFilterChain:

Определен как @Bean, что означает, что метод filterChain создает компонент Spring, управляющий безопасностью.

В методе конфигурации HttpSecurity настроены различные аспекты безопасности, включая:

Отключение защиты от межсайтовой подделки запросов (CSRF).

Настройку формы входа, указывая путь к странице входа и перенаправление после успешного входа.

Настройку процедуры выхода из системы и указание, что все пользователи могут его использовать.

Правило, что все запросы должны быть аутентифицированы.

Включение HTTP Basic аутентификации.

Bean UserDetailsService:

Предоставляет сервис UserDetailsService с пользовательским реализацией.

Внутри лямбда-выражения реализован метод loadUserByUsername, который извлекает данные пользователя (Person) из репозитория по электронной почте.

Если пользователь не найден, выбрасывается исключение UsernameNotFoundException.

Возвращает объект UserDetails, содержащий информацию о пользователе, включая имя пользователя, пароль и назначенные роли.

Этот код описывает стандартные шаги для настройки аутентификации и авторизации в Spring Security, позволяя контролировать доступ к различным частям приложения на основе учетных данных пользователя.

На данном рисунке 18 видно конфигурацию Spring Security для авторизации в приложении. Класс SecurityConfig настроен для управления безопасностью веб-приложения с использованием Spring Framework.

```java
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig {

    David Rezcov +1
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
                .csrf() CsrfConfigurer<HttpSecurity>
                .disable() HttpSecurity
                .formLogin(form -> form
                        .loginPage("/login")
                        .defaultSuccessUrl( defaultSuccessUrl: "/person/main", alwaysUse: true)
                        .permitAll())
                .logout() LogoutConfigurer<HttpSecurity>
                .permitAll()
                .and() HttpSecurity
                .authorizeHttpRequests((authz) -> authz.anyRequest().authenticated())
                .httpBasic();
        return http.build();
    }

    Danila Reznikov
    @Bean
    public UserDetailsService userDetailsService(PersonRepository personRepository) {
        return email -> {
            Person person = personRepository.findByEmail(email)
                    .orElseThrow(() -> new UsernameNotFoundException("Пользователь не найден: " + email));
            return new org.springframework.security.core.userdetails.User(person.getEmail(), person.getPassword(), person.getRoles());
        };
```

Рисунок 18 – Конфигурация Spring Security для авторизации

Аннотация @EnableWebSecurity активирует веб-безопасность для проекта.

Аннотация @EnableGlobalMethodSecurity с параметром prePostEnabled = true позволяет использовать аннотации безопасности перед выполнением и после выполнения методов (например, @PreAuthorize).

В методе filterChain настраивается цепочка фильтров безопасности:

.csrf().disable() отключает проверку токена безопасности CSRF для упрощения разработки.

.formLogin() настраивает вход пользователей через форму на рисунке 18, указывая путь к странице входа (/login) и целевую страницу после успешного входа (/person/main).

.logout() настраивает процесс выхода из системы, разрешая его всем пользователям.

«formLogin – настраивает страницу входа в аккаунт;

logout – настраивает страницу выхода из аккаунта.

В качестве примера настройки авторизации в демонстрационном приложении, структура и функциональность которого представлены в следующем разделе статьи, можно рассматривать класс TeacherSecurityConfig, который содержит три метода, помеченных аннотацией @Bean» [8].

.authorizeHttpRequests() устанавливает правила авторизации, требуя аутентификацию для всех запросов.

```
96
97      <section id="login-container">
98          <h2 class="header-login">Логин</h2>
99          <form action="/login" method="post">
100             <label for="email">Email</label>
101             <input type="email" id="email" name="username" required>
102
103             <label for="password">Пароль</label>
104             <input type="password" id="password" name="password" required>
105
106             <button type="submit">Войти</button>
107         </form>
108     </section>
```

Рисунок 19 – HTML форма для POST авторизации

На рисунке 19 представлена разметка формы регистрации с методом post, для отправки данных на сервер и обработки запроса.

Внутри сервиса используется репозиторий PersonRepository для поиска пользователя по электронной почте.

В случае отсутствия пользователя в базе данных генерируется исключение UsernameNotFoundException.

Возвращается объект UserDetails, который содержит информацию о пользователе, включая его email, пароль и роли.

3.5 Тестирование информационно расчётной системы

«Тестирование чёрного ящика или поведенческое тестирование — стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта. Иначе говоря, тестированием чёрного ящика занимаются тестировщики, не имеющие доступ к исходному коду приложения. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора. Стратегия поведенческого теста исходит из технических требований и их спецификаций» [9].

Достоинства метода «черный ящик»:

- тестирование методом «черного ящика» позволяет найти ошибки, которые невозможно обнаружить методом «белого ящика»;
- «Черный ящик» позволяет быстро выявить ошибки в функциональных спецификациях (в них описаны не только входные значения, но и то, что мы должны в итоге получить);
- тестировщику не нужна дополнительная квалификация;
- тестирование проходит «с позиции пользователя»;
- составлять тест-кейсы можно сразу после подготовки спецификации.

Недостатки метода:

- основным недостатком метода «черного ящика» являетсявозможность пропуска границ и переходов, которые не очевидны из
- спецификации, но есть в реализации кода;можно протестировать только небольшое количество возможныхвводных (входящих) значений» [17].

При аутентификации пользователя Spring Security использует

концепцию «authorities», которая включает в себя роли, представленные как простые строки, например, «ROLE_ADMIN» или «ROLE_USER». Эти роли, как правило, хранятся в таблице ролей в базе данных.

Когда пользователь проходит процесс аутентификации на рисунке 20, Spring Security загружает его роли и использует их для определения его полномочий в системе. Роли позволяют определить, к каким действиям и правам у пользователя есть доступ. Например, администраторы могут управлять аккаунтами пользователей, в то время как обычные пользователи могут только просматривать определённые страницы или использовать определённые функции приложения.

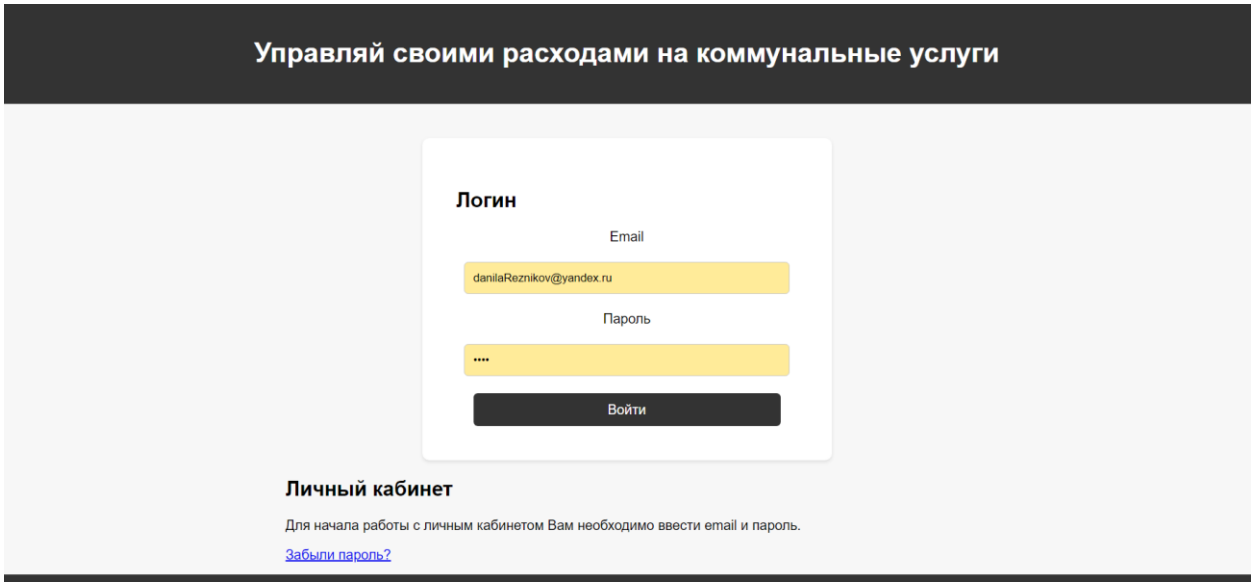


Рисунок 20 – Авторизация на веб-приложении ИРС ИПУ

В случае таблицы многие ко многим, каждый пользователь может иметь несколько ролей, а каждая роль может быть назначена множеству пользователей. Это позволяет создавать гибкие иерархии доступа и легко адаптировать систему к изменениям требований безопасности.

В Spring Security конфигурация доступа обычно происходит через определение URL-паттернов и соответствующих им ролей, необходимых для

доступа. Это может быть настроено в классе конфигурации безопасности через методы HttpSecurity.

В рамках дипломной работы была реализована главная страница веб-приложения Информационно-расчетной системы для индивидуальных приборов учета (ИРС ИПУ). Дизайн главной страницы на рисунке 21 продуман таким образом, чтобы обеспечить максимальное удобство и интуитивное взаимодействие пользователя с системой с первых секунд использования. На главной странице, непосредственно после входа в систему, пользователя встречает персонализированный интерфейс, где отображаются все необходимые элементы управления и навигации.

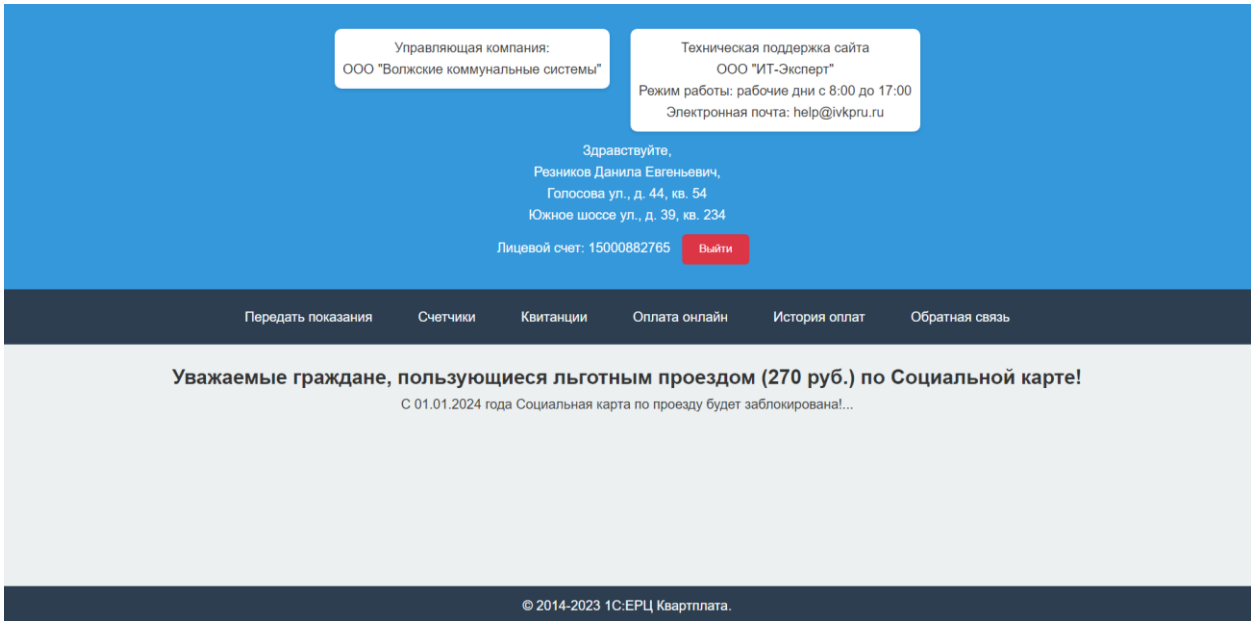


Рисунок 21 – Главная страница пользователя USER_ROLE

Центральное место на странице занимает пользовательское меню, включающее в себя ключевые функции веб-приложения, такие как передача показаний приборов учета, просмотр истории платежей и доступ к коммуникационным инструментам для обратной связи с управляющей компанией. Панель быстрого доступа позволяет с легкостью переходить к основным разделам системы, таким как счетчики, квитанции, оплата онлайн,

история оплат и обратная связь.

Дополнительно, в шапке страницы расположена информация об управляющей компании и технической поддержке, включая контактные данные, что делает коммуникацию с сервисными службами простой и доступной. Внизу страницы присутствует важное информационное сообщение для пользователей, использование которого подчеркивает заботу компании о своих клиентах.

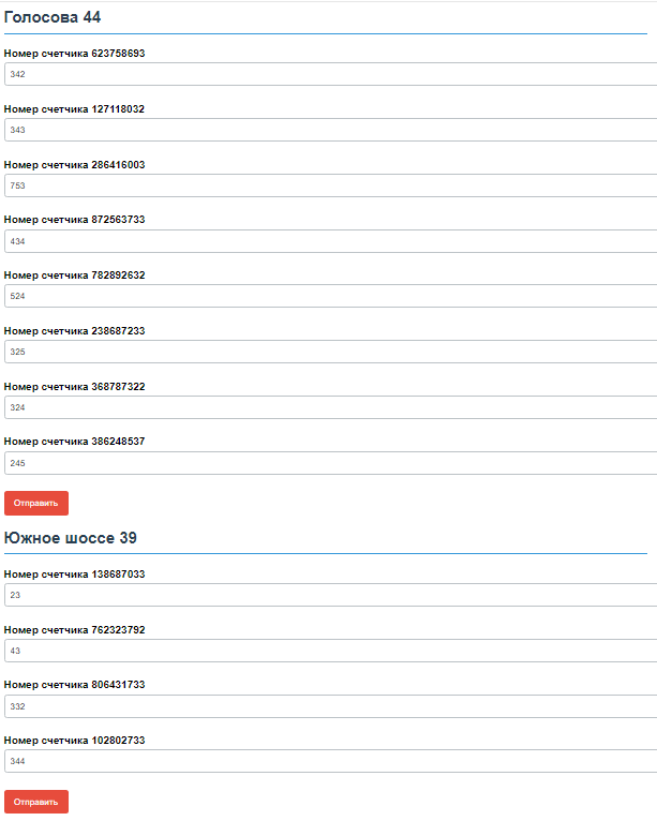


Рисунок 22 – Форма отправики показаний счетчиков

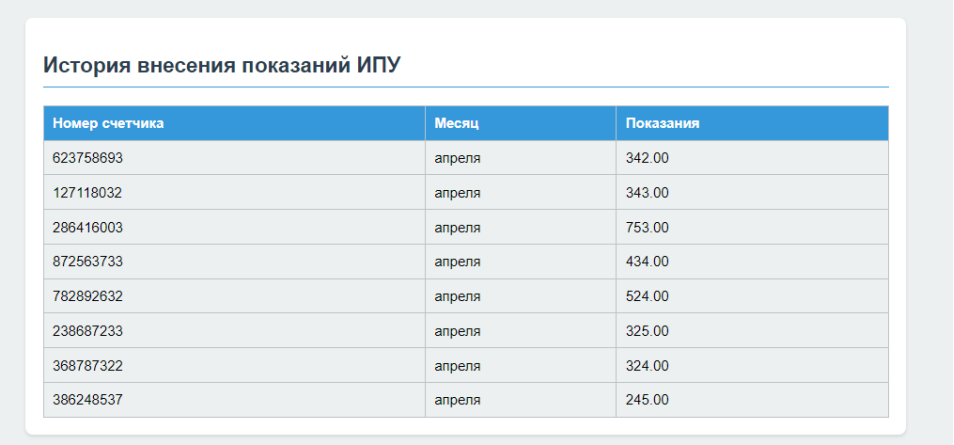
На рисунке 22 представлен скриншот интерфейса пользователя для ввода показаний счетчиков в системе учета коммунальных услуг. Экран разделен на две части, представляющие различные функциональные зоны.

В верхней части экрана находится заголовок с примером улицы, с домом, в котором расположены счетчики. Под этим заголовком располагается серия полей, предназначенных для ввода показаний различных

счетчиков. Каждый блок содержит метку «Номер счетчика» с уникальным номером, следующим за ним, и поле для ввода, где пользователи могут внести текущие показания.

Под серией полей расположена кнопка «Отправить», которая позволяет пользователю отправить введенные показания в систему для обработки.

В средней части экрана находится аналогичная структура с заголовком, что указывает на другой адрес, связанный с аккаунтом пользователя. Так же, как и в первой части, представлены поля для ввода показаний счетчиков, аналогичные по функционалу предыдущим.



История внесения показаний ИПУ

Номер счетчика	Месяц	Показания
623758693	апреля	342.00
127118032	апреля	343.00
286416003	апреля	753.00
872563733	апреля	434.00
782892632	апреля	524.00
238687233	апреля	325.00
368787322	апреля	324.00
386248537	апреля	245.00

Рисунок 23 – История показаний счетчиков

На рисунке 23 представлена таблица «История внесения показаний ИПУ» (индивидуальных приборов учета). В таблице три колонки: «Номер счетчика», «Месяц» и «Показания». Отображены записи с уникальными номерами счетчиков и соответствующими показаниями, сделанными в месяце апреле, выраженные в числовом формате. Этот раздел позволяет пользователям отслеживать предыдущие вводы данных по своим счетчикам за разные периоды времени.

Выводы по третьей главе

Третья глава описывает процесс реализации информационно-расчетной системы на веб-приложении для передачи показаний коммунальных услуг.

Была осуществлена выборка инструментов разработки, включая IntelliJ IDEA и Visual Studio Code в связке с Spring Framework и Lombok для серверной части, что обеспечило высокую производительность и удобство использования.

Была разработана физическая модель данных, использующая PostgreSQL, что обеспечило надежное хранение и управление данными. Диаграмма взаимосвязей таблиц (ERD) помогла структурировать данные и обеспечить их целостность.

В процессе разработки веб-приложения были реализованы ключевые компоненты, такие как контроллеры, сервисы и репозитории, обеспечивающие взаимодействие пользователей с системой. Использование шаблонов Thymeleaf позволило создать динамические HTML-страницы, обеспечивая удобный и интуитивно понятный интерфейс для пользователей.

Конфигурация безопасности была выполнена с использованием Spring Security, что обеспечило защиту данных и контроль доступа, и администрирование на основе ролей пользователей (ADMIN и USER). Реализованы процедуры аутентификации и авторизации, обеспечивающие безопасность и конфиденциальность пользовательских данных.

Тестирование системы подтвердило её работоспособность и соответствие функциональным требованиям. Внедрение разработанного веб-приложения позволит автоматизировать процесс передачи показаний, улучшить управление коммунальными ресурсами и повысить прозрачность расчетов.

Таким образом, третья глава демонстрирует успешную реализацию информационно-расчетной системы, готовой к интеграции в инфраструктуру ЖКХ, что значительно повысит эффективность и удобство использования для конечных пользователей.

Заключение

В ходе выполнения дипломной работы была разработана информационно-расчетная система для передачи показаний приборов учета. Разработанная система позволяет автоматизировать процесс сбора, хранения и анализа данных счетчиков, что повышает эффективность управления ресурсами и обеспечивает прозрачность расчетов для конечных пользователей.

Основной задачей работы была реализация веб-приложения, интегрированного с базой данных, с использованием современных технологий, таких как Spring Framework, Spring Security и Thymeleaf. Данный подход позволил создать безопасное и легко масштабируемое решение, отвечающее современным требованиям к подобного рода системам.

Результаты тестирования подтвердили работоспособность системы, ее надежность и удобство использования как для администраторов, так и для конечных пользователей. Внедрение системы в эксплуатацию позволит обеспечить оперативный доступ к актуальной информации о потреблении ресурсов, оптимизировать процесс учета и расчета платежей.

Возможные направления дальнейших исследований включают интеграцию с мобильными устройствами, использование искусственного интеллекта для предиктивного анализа потребления ресурсов и разработку индивидуальных решений для оптимизации расходов на коммунальные услуги.

Реализация данной дипломной работы показала, что применение современных информационных технологий в учете коммунальных ресурсов открывает широкие перспективы для повышения эффективности управления жилищным фондом и внесения вклада в экологическую безопасность и энергосбережение.

Список используемой литературы

1. Анализ и оптимизация бизнес процессов [Электронный ресурс] URL https://www.bazt.ru/services/gov/business-process-gos (Дата обращения 12.04.2024)
2. А что такое Spring и как он устроен [Электронный ресурс]. URL: https://ru.hexlet.io/blog/posts/spring-framework (дата обращения: 26.04.2024)
3. Дейт К. Дж. Введение в системы баз данных – Вильяме, 2005. – 1316 с
4. Диаграмма классов [Электронный ресурс] URL https://ru.wikipedia.org/wiki/Диаграмма_классов (Дата обращение 12.05.2024)
5. Диаграмма последовательности [Электронный ресурс] URL https://ru.wikipedia.org/wiki/Диаграмма_последовательности (дата обращения 12.04.2024)
6. Логическое моделирование [Электронный ресурс] URL https://www.ngpedia.ru/id159031p1.html (дата обращения 07.04.2024)
7. Нотации бизнес-процессов IDEF0. EPC. BPMN [Электронный ресурс] URL https://www.comindware.com/ru/blog-нотации-бизнес-процессовidef0-epc bpmn/ (дата обращения 29.04.2024).
8. Сакович В. В., Кожомбердиева Г. И., Бураков Д. П. Использование фреймворков семейства Spring Projects для разработки веб-приложений на платформе Java // Интеллектуальные технологии на транспорте. 2023. №2 (34). URL: https://cyberleninka.ru/article/n/ispolzovanie-freymvorkov-semeystva-spring-projects-dlya-razrabotki-veb-prilozheniy-na-platforme-java (дата обращения: 29.04.2024).
9. Тестирование по стратегии чёрного ящика [Электронный ресурс] URL https://ru.wikipedia.org/wiki/Тестирование_по_стратегии_чёрного_ящика (дата обращения 12.05.2024)
10. Хабр [Электронный ресурс] URL https://habr.com/ru/post/282764/ (дата обращения 12.03.2024)

11. Что такое Java? Определение, значение и особенности [Электронный ресурс]. URL: https://appmaster.io/ru/blog/chto-takoe-java-opredelenie-znachenie-osobennosti (дата обращения: 28.05.2024)

12. Diagrams.net. [Электронный ресурс] / Режим доступа: URL: https://en.wikipedia.org/wiki/Diagrams.net (дата обращения 10.05.2024)

13. Entity-Relationship Diagram Symbols and Notation [Электронный ресурс]. URL: https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning (дата обращения: 31.05.2024)

14. Firebird [Электронный ресурс] URL https://ru.wikipedia.org/wiki/Firebird (дата обращения 12.05.2024)

15. Java Persistence API (JPA) For Database Access [Электронный ресурс]. URL: https://www.turing.com/kb/jpa-for-database-access (дата обращения: 31.05.2024)

16. Microsoft SQL Server [Электронный ресурс] URL https://ru.wikipedia.org/wiki/Microsoft_SQL_Server (дата обращения 12.05.2024)

17. Microsoft Visual Studio [Электронный ресурс] URL https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio (дата обращения 12.05.2024)

18. MySQL [Электронный ресурс] URL https://ru.wikipedia.org/wiki/MySQL (дата обращения 12.05.2024)

19. Spring in Action 6th Edition by Craig Walls, c. 164

20. What is IDEF? [Электронный ресурс]. URL: https://www.edrawsoft.com/what-is-idef.html (дата обращения: 31.05.2024)

Приложение А

Код цепочки безопасности Spring Security

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig {
   @Bean
   public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
      http
            .csrf()
            .disable()
            .formLogin(form -> form
                  .loginPage("/login")
                  .defaultSuccessUrl("/person/main", true)
                  .permitAll())
            .logout()
            .permitAll()
            .and()
            .authorizeHttpRequests((authz) -> authz.anyRequest().authenticated())
            .httpBasic();
      return http.build();
   }
   @Bean
   public UserDetailsService userDetailsService(PersonRepository
personRepository) {
      return email -> {
         Person person = personRepository.findByEmail(email)
```

```
                .orElseThrow(() -> new
UsernameNotFoundException("Пользователь не найден: " + email));
            return new
org.springframework.security.core.userdetails.User(person.getEmail(),
person.getPassword(), person.getRoles());
        };
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

Приложение Б

Код сущности «клиент», «роль» и «квартира».

```
package com.example.lesson.models;


@Getter
@Setter
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "person")
public class Person {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;
    @Column(name = "name")
    private String name;
    @Column(name = "last_name")
    private String lastName;
    @Column(name = "post_name")
    private String postName;
    @Column(name = "number")
    private String number;
    @Column(name = "email")
    private String email;
    @Column(name = "password")
    private String password;
    @OneToMany(mappedBy = "person", cascade = CascadeType.ALL)
    private List<Apartment> apartments = new ArrayList<>();
```

```
    @ManyToMany(fetch = FetchType.EAGER, cascade =
CascadeType.DETACH)
    @JoinTable(name = "\"person-roles\"",
            joinColumns = @JoinColumn(name = "person_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private List<Role> roles;
}

@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "rate")
public class Rate {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;
    @OneToMany(mappedBy = "rate", cascade = CascadeType.DETACH)
    private List<Apartment> apartment = new ArrayList<>()
    @Column(name = "price_cold_water")
    private BigDecimal priceCold;
    @Column(name = "price_warm_water")
    private BigDecimal priceWarm;
}

@Entity
@Getter
```

```
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "apartment")
public class Apartment {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    private Person person;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "street_id")
    private Street streetId;
    @Column(name = "house_number")
    private int houseNumber;

    @Column(name = "apartment_number")
    private int apartmentNumber;
    @OneToMany(mappedBy = "apartment", cascade = CascadeType.ALL)
    private List<Meter> meters = new ArrayList<>();
    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.DETACH)
    @JoinColumn(name = "rate_id")
    private Rate rate;
    @Column(name = "acount_number")
    private Long acountNumder;
}
```

Приложение В

Конфигурация

```
spring.application.name=demo
spring.datasource.url=jdbc:postgresql://localhost:5432/kvartplata24
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Приложение Г

Код архитектуры получения показаний

```
@Controller
@RequiredArgsConstructor
@RequestMapping("/meter-readings")
public class MeterReadingController {

    private final MeterReadingService meterReadingService;
    private final StreetService streetService;
    private final PersonService personService;
    private final ApartmentService apartmentService;

    @PostMapping("/add")
    public String submitMeterReadings(@RequestParam Map<String,String> allParams) {
        meterReadingService.saveReading(allParams);
        return "redirect:/meter-readings";
    }
    @GetMapping()
    public String showMeterReadingsForm(Model model) {
        Map<Apartment, List<Meter>> meters = meterReadingService.findPersonMeters();
        List<Reading> readings = meterReadingService.findPersonReadings();
        Person user = personService.getUser();
        List<Apartment> apartments = apartmentService.getApartmentsByPerson();
        Map<Apartment, Street> apartmentsWithStreets = streetService.getAllStreetByApartments(apartments);

        model.addAttribute("apartmentsWithStreets", apartmentsWithStreets);
```

```
        model.addAttribute("user",user);
        model.addAttribute("user",user);
        model.addAttribute("meters", meters);
        model.addAttribute("readings", readings);
        return "site/meter_readings";
    }
}

@Service
@Slf4j
@RequiredArgsConstructor
public class MeterReadingService {

    private final MeterRepository meterRepository;
    private final ApartamentRepository apartamentRepository;
    private final PersonRepository personRepository;
    private final ReadingsRepository readingsRepository;
    public List<Reading> findPersonReadings(){
        List<Reading> readings = new ArrayList<>();
        List<Meter> allMeters = new ArrayList<>();
        String email =
SecurityContextHolder.getContext().getAuthentication().getName();
        Person person =
personRepository.findByEmail(email).orElseThrow(EntityNotFoundException::ne
w);
        List<Apartment> apartments =
apartamentRepository.findAllByPerson(person);
        for (Apartment apartment : apartments) {
            List<Meter> allByApartment =
```

```
meterRepository.findAllByApartment(apartment);
            for (Meter meter : allByApartment){
                List<Reading>                readingsOneApartment                =
readingsRepository.findAllByMeter(meter);
                for (Reading reading : readingsOneApartment){
                    readings.add(reading);
                    System.out.println(reading);
                }
            }
        }
        return readings;
    }
    public Map<Apartment, List<Meter>> findPersonMeters() {
        Map<Apartment, List<Meter>> allMeters = new HashMap<>();
        String                                email                                =
SecurityContextHolder.getContext().getAuthentication().getName();
        Person                                person                                =
personRepository.findByEmail(email).orElseThrow(EntityNotFoundException::ne
w);
        List<Apartment>                        apartments                        =
apartamentRepository.findAllByPerson(person);
        for (Apartment apartment : apartments) {
            List<Meter>                        allByApartment                =
meterRepository.findAllByApartment(apartment);
            allMeters.put(apartment, allByApartment);
        }
        return allMeters;
```

```
    }
    public List<Reading> saveReading(Map<String,String> allReadings){
        List<Reading> readings = new ArrayList<>();
        Iterator<Map.Entry<String, String>> itr = allReadings.entrySet().iterator();
        while(itr.hasNext()) {
            Map.Entry<String, String> entry =  itr.next();
            String counterNumber = entry.getKey();
            String value = entry.getValue();
            Meter meter =
meterRepository.findAllByCounterNumber(Long.parseLong(counterNumber));
            Reading reading = new Reading();
            reading.setDate(LocalDate.now());
            reading.setMeter(meter);
            reading.setValue(new BigDecimal(value));
            readings.add(reading);
            readingsRepository.save(reading);
        }
        return readings;
    }
}

public interface ReadingsRepository extends JpaRepository<Reading,Long> {
    public List<Reading> findAllByMeter(Meter meter);
}
```