Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

У Т В Е Р Ж Д А Ю

**Зав. кафедрой ВММБ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Ю. Столбов

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

***ЗАДАНИЕ***

***НА КУРСОВУЮ РАБОТУ***

***по дисциплине***

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Кольчерин Александр Алексеевич, Юнусов Василь Равилевич, Шаехов Андрей Евгеньевич ЦТУ-20-3б

(фамилия, имя, отчество студента; группа)

1. **Тема курсовой работы**

Модуль корпоративной информационной системы для администрирования системы подачи показаний счетчиков

2. **Срок сдачи студентами отчета:** « » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

3. **Содержание отчета:**

Проектирование базы данных. Реализация модуля корпоративной информационной системы для администрирования системы подачи показаний счетчиков.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание приняли к исполнению

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Министерство науки и высшего образования Российской Федерации

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**КУРСОВАЯ РАБОТА**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Модуль корпоративной информационной системы для администрирования системы подачи показаний счетчиков»**

Выполнили:

ЦТУ-20-3б

студенты гр. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (*подпись*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (*подпись*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (*подпись*)

Принял:

ст. преподаватель, Банников Р.Ю.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2024**

# Содержание

[Введение 4](#_Toc163408893)

[Задание 6](#_Toc163408894)

[Средства реализации 8](#_Toc163408904)

[Структура БД 10](#_Toc163408907)

[Программный продукт 16](#_Toc163408908)

[Вывод 21](#_Toc163408913)

[Список литературы 22](#_Toc163408914)

[Приложение 22](#_Toc163408915)

# Введение

С каждым годом количество новых многоквартирных домов увеличивается. За последние 10 лет жилой фонд в России увеличился на 20%, а количество квартир в домах за последний год на 5,7% [1]. Каждая квартира в этих домах оборудована приборами учета для измерения потребления электричества, воды и газа. Сбор данных с этих приборов учета является важной задачей для эффективного управления ресурсами и определения платежей за коммунальные услуги.

Однако, сбор данных с приборов учета может включать в себя несколько проблем. Вот некоторые из них:

* возможные технические проблемы с самими приборами учета, такие как сбои в работе, неправильное измерение или неполадки в передаче данных. Это может привести к неточности или неполным данным, что затрудняет правильный учет потребления ресурсов;
* отсутствие доступа к приборам учета: в некоторых случаях, доступ к приборам учета может быть ограничен или затруднен. Например, приборы учета могут быть установлены в отдельных квартирах или помещениях, к которым управляющая компания не имеет свободного доступа. Это может затруднить сбор данных и контроль за потреблением ресурсов;
* неправильная эксплуатация приборов учета: некорректная эксплуатация приборов учета может привести к искажению данных. Например, если жильцы не следят за состоянием и правильной работой приборов учета, это может привести к неправильному измерению потребления ресурсов;
* проблемы с обработкой данных: сбор данных приборов учета может быть только первым шагом. Далее, эти данные должны быть обработаны и проанализированы для получения полезной информации. Проблемы могут возникнуть при обработке и анализе данных, такие как ошибки в программном обеспечении или недостаточные навыки у персонала, занимающегося обработкой данных;
* проблемы совместного использования данных: в ТСЖ может возникнуть проблема совместного использования данных приборов учета. Например, если в одном доме установлены приборы учета от разных поставщиков, может быть сложно объединить данные для получения общей картины потребления ресурсов.

Многие из данных проблем уже решены. К примеру, проблема неправильной эксплуатацией решена с помощью создания базы счетчиков в ТСЖ и регулярными проверками для отслеживания их состояния. Проблема отсутствия доступа к приборам учета была решена вынесением их в общедомовой коридор.

Но во многих домах данные счетчики находятся в квартирах и людям приходится самим записывать данные приборов учета и отправлять их в ТСЖ, к примеру, записывая данные на бумажке и кладя в ящик, позже данные бумажки собирает ТСЖ и записывает данные. Однако, данный способ передачи данных имеет несколько весомых проблем, приведенных ниже.

При ручной записи данных на бумаге существует риск возникновения ошибок и неточностей. Жильцы могут неправильно записать показания счетчиков или допустить опечатки, что может привести к неправильному учету потребления ресурсов;

Передача данных через запись на бумагу может привести к задержкам и потере информации. Эти записи могут быть утеряны или задержаться в ящике, что затрудняет своевременный сбор и обработку данных;

Ручная запись данных на бумаге и их последующая передача в ТСЖ может быть неудобной и неэффективной процедурой. Жильцам приходится тратить время и усилия на запись и сбор данных, а управляющей компании - на их обработку и ввод в систему;

При передаче данных через запись на бумаге отсутствует прозрачность и контроль за процессом. Нет возможности проверить правильность записи данных или убедиться в их достоверности.

Можно предложить несколько вариантов решения данной проблемы:

* Автоматизированные системы сбора данных, к примеру, умные счетчики, которые автоматически будут отправлять данные в ТСЖ: этот метод сбора данных очень эффективен и удобен, поскольку умные счетчики автоматически отправляют данные в ТСЖ. Однако на сегодняшний день этот метод имеет недостаток в виде высокой стоимости установки и обслуживания таких систем. Кроме того, может потребоваться специализированное оборудование и программное обеспечение для обработки и анализа данных.
* Установка специальных приборов, которые подключаются к счетчику, считывают с него данные и отправляют в ТСЖ: данный метод сбора по сравнению с умными счетчиками имеет меньшую стоимость, но может подходить не ко всем счетчикам, а также требовать специального программного обеспечения.
* Разработка мобильного приложения для сбора показаний приборов учета: в современном мире, когда у большинства людей есть смартфоны и доступ в интернет, мобильное приложение может стать отличным решением для сбора показаний приборов учета. Пользователи могут вводить показания прямо в приложении, которое автоматически отправляет эти данные в ТСЖ. Это не только упрощает процесс сбора данных для жителей, но и уменьшает вероятность ошибок при передаче показаний. Кроме того, приложение может предоставлять дополнительные функции, такие как отслеживание истории показаний, уведомления о необходимости передать показания, информацию о тарифах и расчетах, и даже возможность онлайн-оплаты услуг.

Чтобы решить данные проблемы и предоставить людям удобное решение, решено разработать мобильное приложение, которое предоставит людям, имеющим смартфон и интернет быстро отправлять данные в ТСЖ.

Добавление возможности администрирования такой системы может принести множество преимуществ.

* Уменьшение ошибок - ручной ввод данных всегда связан с риском ошибок, которые могут привести к неправильным расчетам и, как следствие, к финансовым потерям. Автоматизация этого процесса позволяет минимизировать этот риск.
* Удобство использования - современные технологии позволяют создавать удобные и интуитивно понятные интерфейсы, которые могут использовать даже люди, не имеющие глубоких знаний в области IT.
* Улучшение контроля - администрирование системы подачи показаний счетчиков позволяет лучше контролировать процесс сбора данных, а также обеспечивает возможность быстро реагировать на возникающие проблемы.
* Улучшение качества обслуживания - быстрый и точный сбор данных позволяет улучшить качество обслуживания клиентов, что в свою очередь может привести к увеличению их лояльности.

В целом, добавление возможности администрирования системы подачи показаний счетчиков является важным шагом на пути к цифровизации и автоматизации бизнес-процессов, что в современных условиях является одним из ключевых факторов успеха любой организации.

# Задание

## Исходные данные

Данная система должна облегчить процесс подачи показаний в соответствующие организации путём предоставления пользователю возможности организации данного процесса при помощи фотографирования лицевой стороны счетчика и подтверждения правильности показаний в мобильном приложении. Ниже приведен список требований к данному проекту:

* Регистрация и авторизация пользователей;
* Реализация личного кабинета пользователя для предоставления пользователю возможности добавления/редактирования информации своей учетной записи;
* Реализация функционала меню администрирования для корректирования ошибочных данных;
* Предоставление пользователю возможности регистрации ТСЖ, в котором он будет являться управляющим. На данном этапе пользователь должен указать дома и квартиры, которые состоят в этом ТСЖ и далее, когда пользователи будут регистрироваться в системе и указывать квартиры, в которых они проживают, они должны автоматически быть оформлены, как участники данного ТСЖ;
* Обработка функционала пользователей в зависимости от их ролей. На данный момент предполагается использование пяти ролей:
  + Администратор. Имеет возможность корректировки ошибочных данных в системе;
  + Управляющий ТСЖ. Имеет возможность добавления и исключения пользователей в ТСЖ, управляющим которого он является;
  + Владелец квартиры. Имеет возможность регистрации счетчиков и подачи показаний в своей квартире. Также, данная роль позволяет назначать других пользователей арендаторами в своей квартире;
  + Арендатор. Подразумевает возможность подачи показаний в квартире, арендатором которой он является;
  + Пустая роль. Присваивается по умолчанию при регистрации, предполагается, что позже, при присоединении пользователя к ТСЖ, его роль будет изменена на роли владельца или арендатора квартиры.

• Предоставление пользователю возможности сканирования показаний посредством фотографирования лицевой стороны счетчика;

• Предоставление пользователю возможности отслеживания поданных показаний и приблизительная стоимость оплаты услуг за каждый период;

## Цели

* Создать удобную и интуитивно понятную платформу для администрирования системы подачи показаний счетчиков.
* Обеспечить возможность соблюдения целостности данных.

## Основной функционал

## Регистрация и авторизация

* Пользователи должны иметь возможность создать учетную запись.
* Система должна обеспечивать проверку подлинности пользователей и защиту от несанкционированного доступа.
* Система должна поддерживать разграничение функционала по ролям:
* Роль администратор – пользователь, имеющий возможность редактирования данных.
* Роль оператор службы поддержки – пользователь, который может просматривать отчеты.

## Генерация отчетов

Система должна генерировать 7 отчетов:

1. Вывод данных показателей приборов учета всех пользователей за месяц
2. Вывод данных показателей приборов учета определенного пользователя за месяц
3. Вывод информации о жильцах, их квартирах по ТСЖ
4. Ввод информации о жильцах определенной квартиры
5. Вывод всех ТСЖ и принадлежащих им ресурсоснабжающих компаний. Этот отчет будет экспортироваться в xls файл.
6. Вывод информации о квартирах, в которых зарегистрирован арендатор.
7. Вывод всех показаний дома определенного месяца.

## Обеспечение устойчивости системы

* Система должна реагировать на исключительные ситуации выводом сообщения об ошибке.
* Система должна корректно валидировать входные данные от пользователя.

Кроме того, администратор и оператор службы поддержки могут корректировать данные в системе, что предоставляет дополнительный фактор устойчивости системы.

## Требования к дизайну

UI/UX часть приложения должна удовлетворять следующим требованиям:

* интуитивно понятный интерфейс: приложение должно быть легким в использовании и понятным для пользователей всех уровней навыков. Это включает в себя простую навигацию, понятные иконки и интуитивно понятные элементы управления;
* четкая структура и организация: приложение должно иметь четкую структуру и организацию, чтобы пользователи могли легко найти нужные функции и информацию;
* единый дизайн: все элементы интерфейса должны быть оформлены в едином стиле и иметь согласованный дизайн. Это поможет создать единое визуальное впечатление и обеспечить легкость восприятия информации;
* ясные инструкции и подсказки: приложение должно предоставлять ясные инструкции и подсказки, чтобы помочь пользователям понять, как использовать различные функции и выполнять задачи;
* визуальная привлекательность: приложение должно иметь привлекательный дизайн и графику, чтобы привлечь внимание пользователей и создать положительное впечатление;
* удобство использования: приложение должно быть удобным в использовании и предоставлять пользователю возможность выполнять задачи быстро и эффективно;

Защита данных

Пользовательские данные будут защищены от несанкционированного доступа и передачи. Система должна поддерживать парольную аутентификацию. Требуется, чтобы пароли хранились в хешированном виде.

## Сроки выполнения работы

На стадии «Технический (и рабочий) проект» должны быть выполнены перечисленные ниже этапы работ:

* Проектирование хранилища. Форма отчётности – отчёт. Срок выполнения – не позднее 13.02.2024.
* Проектирование интерфейса. Форма отчётности – отчёт. Срок выполнения – не позднее 20.02.2024.
* Разработка диаграммы классов. Форма отчётности – отчёт. Срок выполнения – не позднее 27.02.2024.
* Программная реализация. Форма отчётности – отчёт и демонстрация заявленного

функционала ИС. Срок выполнения – не позднее 19.03.2024.

* Тестирование. Форма отчётности – отчёт. Срок выполнения – не позднее 30.03.2024.

# Средства реализации

Выбор стека технологий является важным аспектом для разработки. Данный проект состоит из трех основных частей: frontend, backend и СУБД для хранения данных.

## Средства реализации frontend части приложения

В данном проекте использовались 3 основных инструмента React, Typescript и antd –это мощные инструменты для реализации frontend части любого приложения.

1. React: Это библиотека JavaScript для создания пользовательских интерфейсов. React делает возможным создание сложных UI из маленьких, изолированных частей кода, называемых "компонентами".
2. TypeScript: Это надмножество JavaScript, добавляющее статическую типизацию в язык. TypeScript позволяет писать более безопасный и читаемый код, идеально подходящий для больших проектов.
3. antd (Ant Design): Это набор высококачественных компонентов UI для React. Ant Design позволяет быстро и легко создавать красивые и отзывчивые пользовательские интерфейсы.

Совместное использование этих трех инструментов может значительно упростить и ускорить процесс разработки frontend. React позволяет создавать переиспользуемые компоненты UI, TypeScript обеспечивает строгую типизацию для предотвращения ошибок на этапе компиляции, а Ant Design предоставляет широкий выбор готовых к использованию компонентов, что уменьшает время на разработку собственных компонентов.

## Средства реализации backend части приложения

Backend часть приложения состоит из двух основных частей – API и СУБД, с которой API будет взаимодействовать и хранить все необходимые данные.

В качестве основного инструмента реализации API части приложения был выбран надёжный и проверенный временем фреймворк для языка программирования Java – Spring Boot. Данный выбор был обусловлен тем, что данный фреймворк является одним из самых популярных для построения подобного рода систем. Это значит, что при возникновении каких-либо затруднений при построении системы, можно будет легкой найти их решение. Его использование в разы упрощает построение API части любого приложения [2]. В него входят удобные средства для решения таких рутинных задач, как:

* взаимодействие с базой данных;
* создание компонентов для реализации бизнес-логики;
* работа с HTTP запросами (прием, обработка и отправка);
* различные типы тестирования компонентов приложения;
* внедрение зависимостей.

Разработчику не нужно задумываться о том, где, как и когда создавать сущности необходимых компонентов.

Кроме того, важным критерием выбора данного фреймворка было то, что у команды разработки данного проекта уже был опыт построения систем с его использованием.

Выбор версии фреймворка был тесно связан с выбором версии языка программирования Java. В качестве версии языка программирования была выбрана 8 версия Java, так как эта версия является стабильной версией и рассчитана на долгосрочную поддержку [3]. В связи с этим, была выбрана 1.5.2 версия для Spring фреймворка.

Кроме API части приложения, нужно определиться еще и со средствами реализации хранения данных. Современные информационные системы не могут функционировать без использования баз данных, поскольку они позволяют систематизировать огромные объемы данных, обеспечивают быстрый доступ к информации, гарантируют целостность и безопасность этих данных [4]. Есть несколько видов баз данных, в том числе реляционные, объектно-реляционные, базы данных в оперативной памяти и NoSQL [5].

Реляционные базы данных — это один из самых популярных видов баз данных [6]. Популярность данного вида баз данных является вполне обусловленной, они проверены временем и являются основным видом баз данных для разработки подобного типа систем. В них данные организованы в виде таблиц, а отношения между таблицами определяются с помощью ключей. Для работы с данными используется язык запросов SQL.

Базы данных необходимы для эффективного управления данными в разных ситуациях. Например, в бизнес-приложениях они используются для хранения данных клиентов, продуктов, заказов и другой бизнес-информации. Веб-сайты и веб-приложения используют их для хранения данных пользователей. В мобильных приложениях они используются для сохранения данных пользователей, покупок, прогресса в играх и так далее.

Таким образом, базы данных играют важную роль во многих современных приложениях и системах, помогая хранить, организовывать и анализировать большие объемы данных для обеспечения эффективности и безопасности. Выбор типа используемой базы данных зависит от особенностей конкретного проекта.

Как уже было описано ранее, реляционный тип базы данных отлично подходит для реализации подобных систем, следовательно, для наших целей была выбрана база данных именно этого типа, а именно PostgreSQL. Конкретно данная СУБД была не случайно. Её выделяет то, что, как и большинство инструментов, которые были выбраны для реализации данного проекта, она является одной из самых популярных и современных систем управления базами данных. Более того, ее отличает широкий функционал и высокая надёжность. Несомненным плюсом данной СУБД является и то, что она распространяется бесплатно и имеет открытый исходный код. Также, PostgreSQL легко масштабируется, данная СУБД может без особых проблем горизонтально масштабироваться [7].

# Структура БД

**Проектирование БД**

В базе данных были определены основные сущности и их атрибуты, разработана модель БД, которая представлена на рис. 3 и рис. 4. Основной сущностью данной системы является пользователь. Сущность содержит основную информацию о пользователе, логин, пароль, ФИО и номер телефона. Каждый пользователь связан с квартирой, посредством промежуточной таблицы, которая содержит роль жильца данной квартиры. Сущность ТСЖ так же привязана к пользователю. К данной сущности привязан адрес, к которому уже привязана квартира. К каждой квартире привязан счетчик, который содержит информацию о данном счетчике, для его идентификации. История показаний счетчика является отдельной сущностью. Она содержит в себе показания приборов учета, которые занес пользователь и приблизительную стоимость услуг. Так же к счетчику привязана сущность «ресурс» (газ, вода, электричество) и «цена» на данный ресурс. К ресурсу привязан провайдер, который предоставляет услуги.

**Логическая модель**

Логическая модель данных корпоративной информационной системы представлена на рисунке 1.

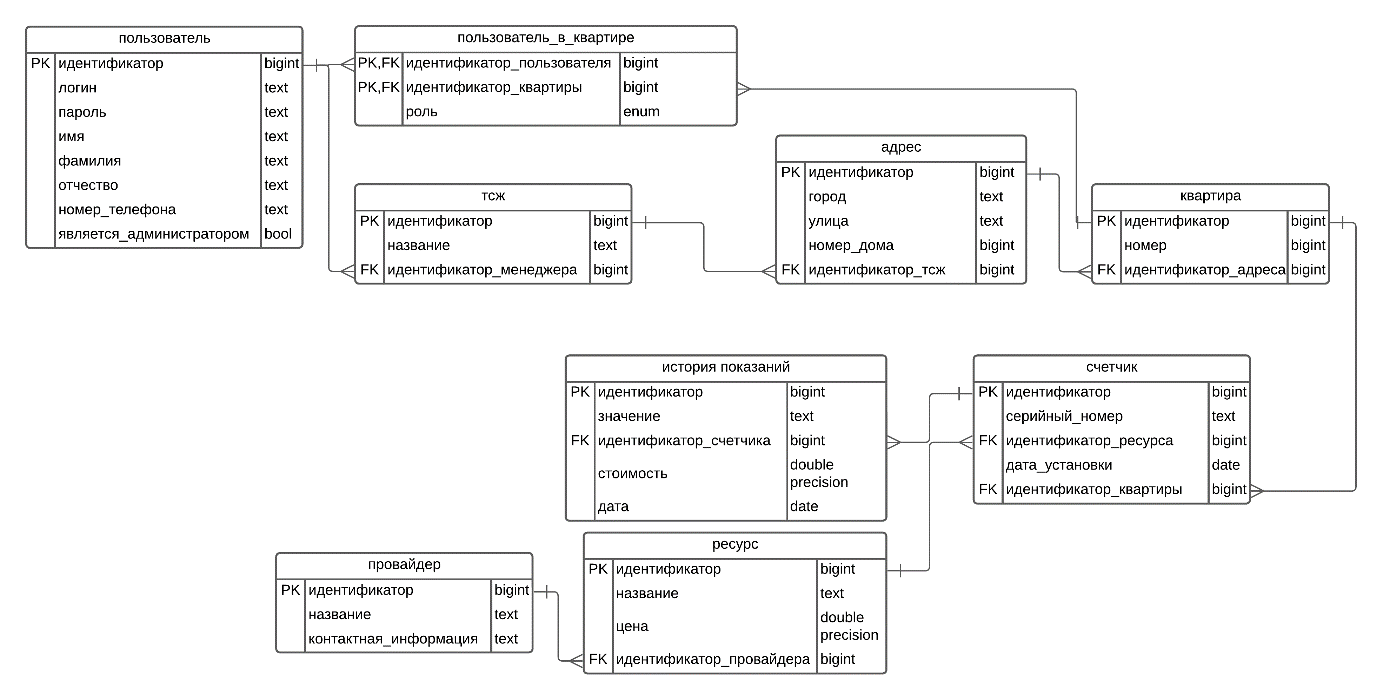


Рисунок 1. Логическая модель системы.

**Физическая модель**

Физическая модель данных корпоративной информационной системы представлена на рисунке 2.

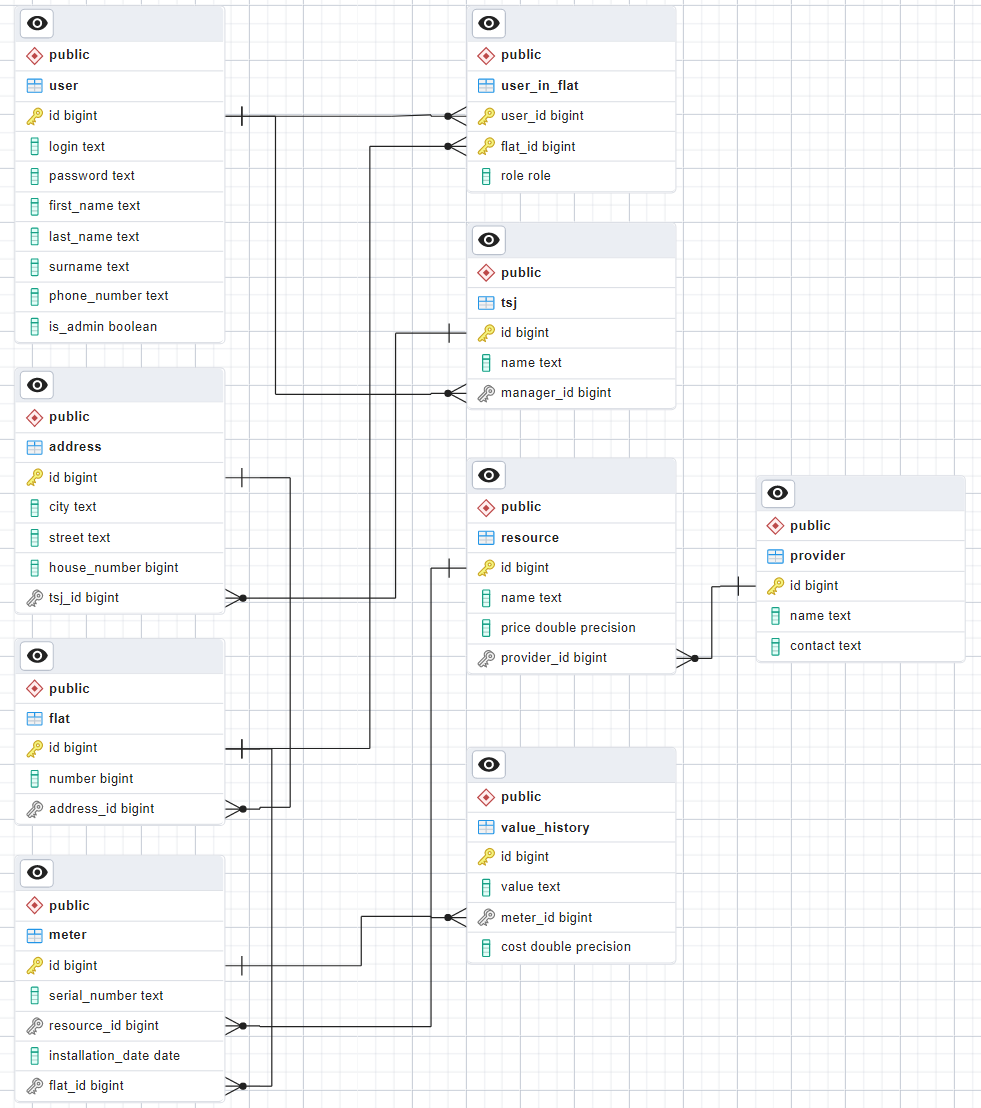


Рисунок 2. Физическая модель системы.

**Обоснование выбранных типов данных:**

Все текстовые поля базы данных хранятся в типе text, который неограничен по длине. Такое решение было принято в связи с тем, чтобы в дальнейшем не возникло никаких проблем, связанных с физическими ограничениями базы данных, хотя такое решение является не самым оптимальным.

Поля таблицы «Пользователь» как правило хранятся в типах text. Это связано с тем, чтобы не ограничивать пользователя в длине его данных. Поле «является\_администратором» может содержать всего два значения, поэтому было выбран тип данных bool.

Поле «серийный номер» таблицы «Счетчик» и поле «значение» таблицы «История показаний» были выбраны типа text для того, чтобы система была максимально универсальной и могла хранить серийные номера и значения счетчиков любых типов и форматов.

Поле «контактная\_информация» таблицы «Провайдер» хранится в типе text, так как эта контактная информация может содержать в себе хоть номер телефона, хоть адрес, хоть какую-либо другую информацию, полезную для пользователя.

Таблицы «Адрес» и «Квартира» разбиты на две разные по причине того, чтобы не хранить всю информацию об адресе для каждой квартиры. Кроме того, так более удобно добавлять новые квартиры, в уже зарегистрированный дом, когда дом уже зарегистрирован.

**Таблица user:**

Содержит в себе информацию о пользователе.

Поля таблицы:

* Id – идентификатор, первичный ключ
* Login – логин пользователя, указываемый при регистрации
* Password – пароль пользователя
* First\_name – имя пользователя
* Last\_name – фамилия пользователя
* Surname – отчество пользователя
* Phone\_number – телефонный номер пользователя
* Is\_admin – определеяет, является ли пользователем администратором

**Таблица address:**

Содержит в себе информацию об адресе, к которому привязывается ТСЖ.

Поля таблицы:

* Id – идентификатор, первичный ключ
* City – название города
* Street – название улицы
* House\_number – номер дома
* Tsj\_id – идентификатор из таблицы tsj, нужен для привязки определенного ТСЖ к конкретному адресу, внешний ключ

**Таблица tsj:**

Содержит в себе информацию о ТСЖ.

Поля таблицы:

* Id – идентификатор, первичный ключ
* Name – название ТСЖ
* Manager\_id – идентификатор пользователя из таблицы user, внешний ключ

**Таблица flat:**

Содержит в себе информацию о квартире.

Поля таблицы:

* Id – идентификатор, первичный ключ
* Number – номер квартиры
* address\_id – идентификатор адреса из таблицы address, необходим для привязки квартиры к определенному адресу, внешний ключ

**Таблица user\_in\_flat:**

Содержит в себе информацию об соответствующих идентификаторах пользователей и квартир, т.е. по данной таблице можно узнать к какому пользователю принадлежит та или иная квартира. Также из данной таблицы можно узнать о роли пользователя в системе.

Поля таблицы:

* user\_id – идентификатор пользователя из таблицы user, первичный ключ
* flat\_id – идентификатор квартиры из таблицы flat, первичный ключ
* role – роль пользователя, определяющая его права. Существующие роли по возрастанию значимости их прав: [“BLANK”, “RENTER”, “OWNER”]

**Таблица provider:**

Содержит в себе информацию о поставщике ресурсов.

Поля таблицы:

* Id – идентификатор, первичный ключ
* Name – название ресурсоснабжающей компании
* Contact – контактная информация ресурсоснабжающей компании

**Таблица resource:**

Содержит в себе информацию о поставляемых ресурсах ресурсоснабжающей компанией.

Поля таблицы:

* Id – идентификатор, первичный ключ
* Name – название ресурса
* Price – цена за единицу ресурса
* Provider\_id - идентификатор ресурсоснабжающей компании из таблицы provider, которая поставляет данный ресурс, внешний ключ

**Таблица meter:**

Содержит в себе информацию о счетчике.

Поля таблицы:

* Id – идентификатор, первичный ключ
* Serial\_number – серийный номер счетчика
* Installation\_date – дата установки счетчика
* Resource\_id – идентификатор ресурса из таблицы resource, с которым связан данный счетчик, внешний ключ
* Flat\_id – идентификатор квартиры из таблицы flat, к которой привязан данный счетчик, внешний ключ

**Таблица value\_history:**

Содержит в себе информацию о полученных показаниях счетчика.

Поля таблицы:

* Id – идентификатор, первичный ключ
* Value – значение, которое было определено приложением
* Cost – стоимость услуги относительно предыдущего показания счетчика
* Date – дата получения показания
* Meter\_id - идентификатор счетчика из таблицы meter, внешний ключ

# Программный продукт

## Авторизация

Работа с приложением начинается с авторизации.

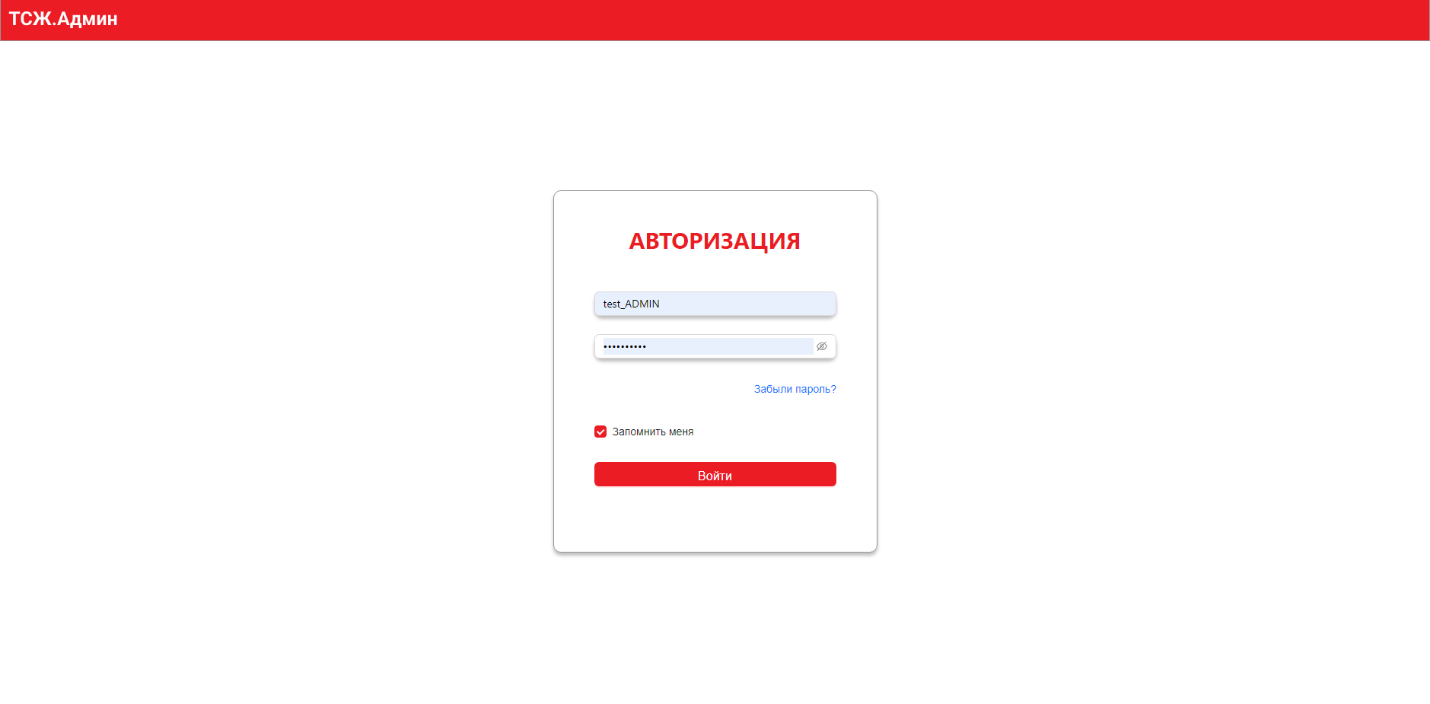


Рисунок 3. Страница авторизации.

После ввода данных необходимо нажать кнопку «Войти», пользователь авторизуется в системе, после чего откроется пустое окно с отчетами. Это продемонстрировано на рисунке 4.

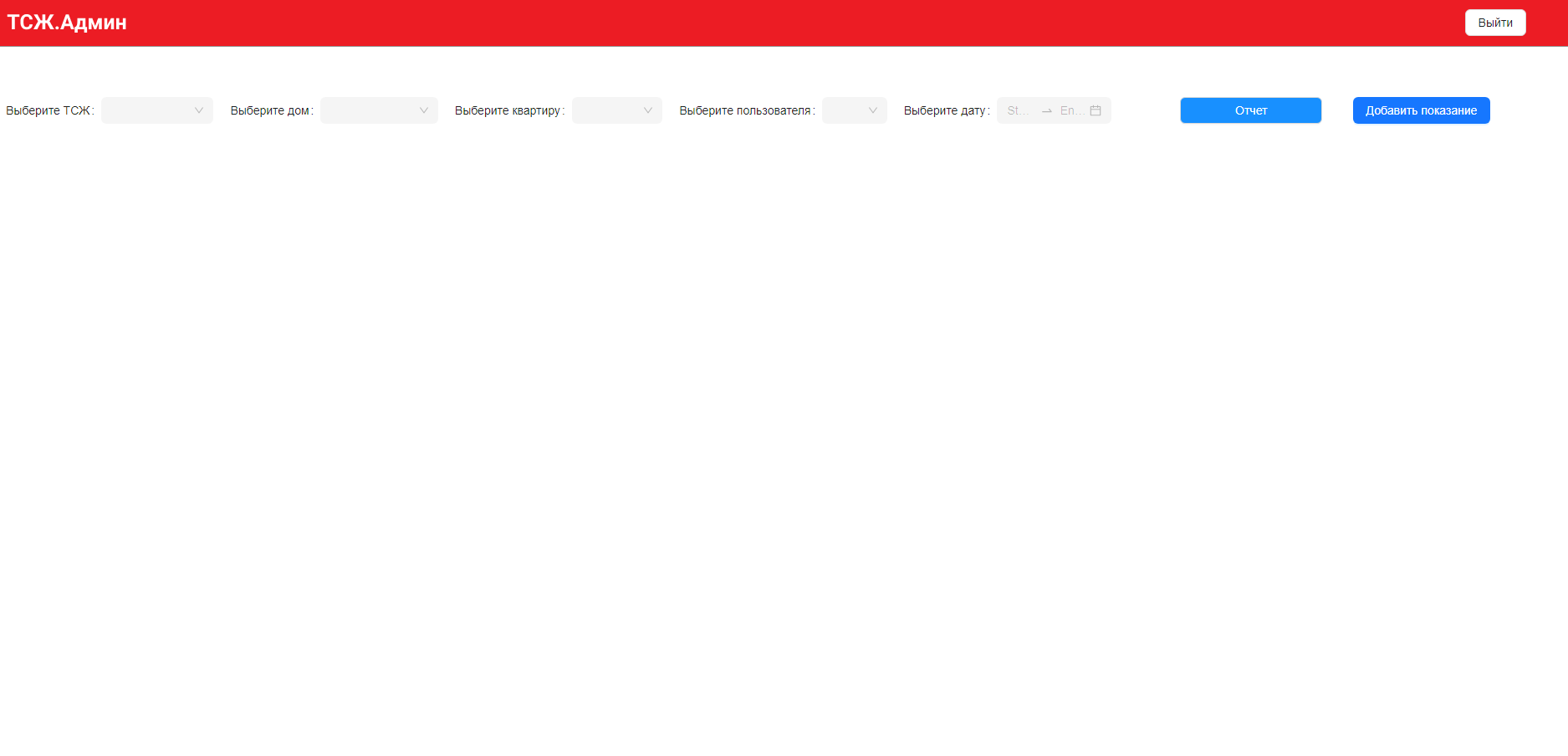


Рисунок 4. Пустое окно с отчетами.

## Генерация отчетов

Далее пользователь выбирает данные для формирования отчета и выбирает его форму при помощи нажатия на кнопку «Отчет».

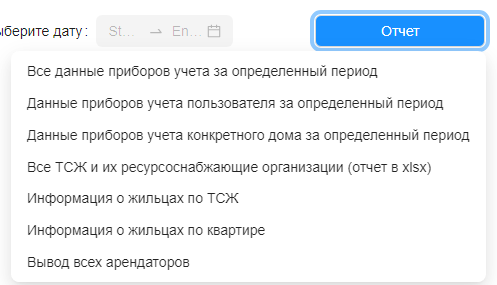


Рисунок 5. Разновидности форм отчета.

На рисунке 6 показан результат отчета по выводу всех показаний счетчиков. При помощи кнопки «Изменить» можно редактировать данные о значениях в показаниях. Кнопка «Удалить» удаляет выбранное показание счетчика.

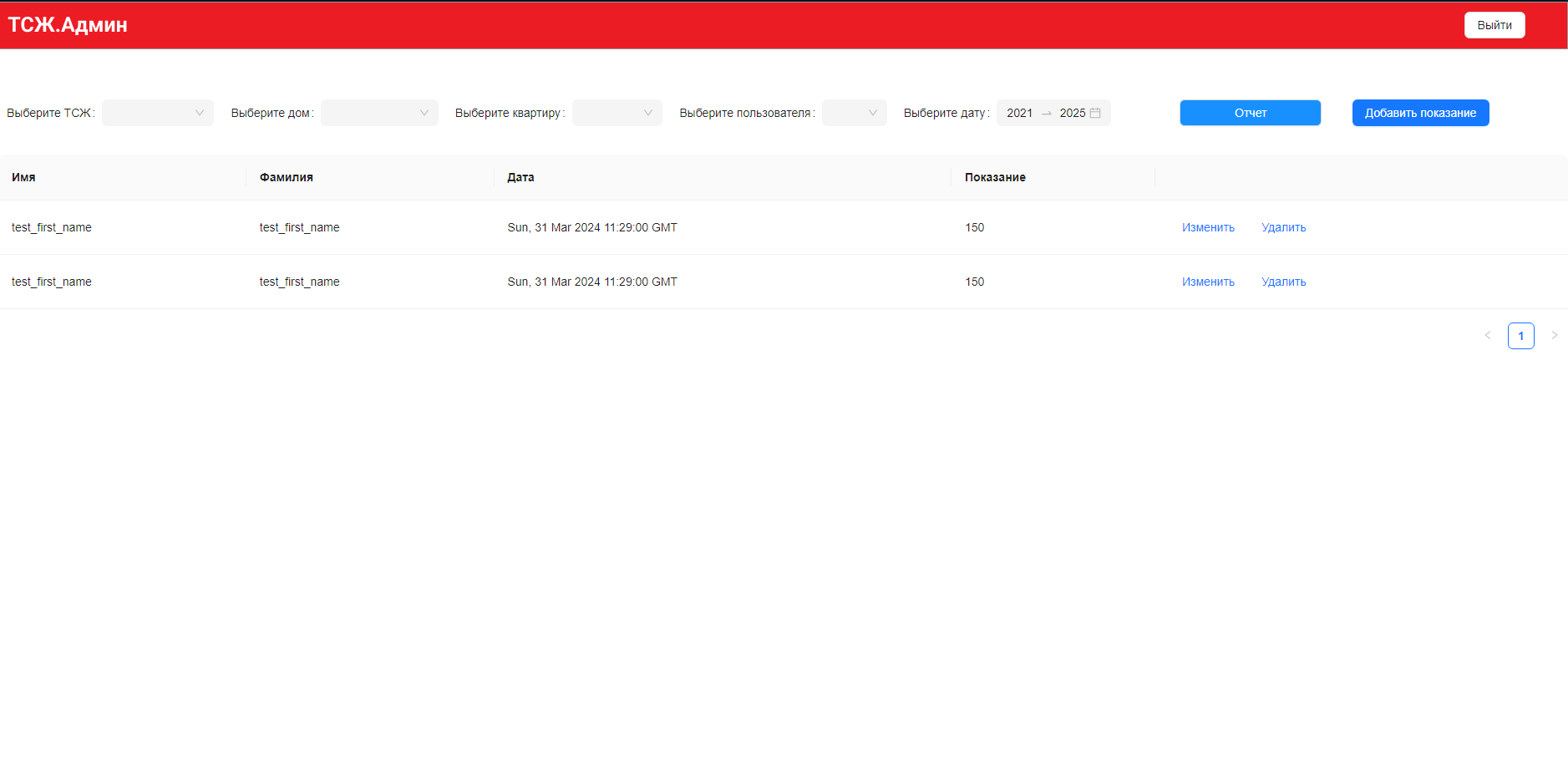


Рисунок 6. Отчет по выводу всех показаний счетчиков.

При выборе вида отчета «Все ТСЖ и их ресурсоснабжающие организации (отчет в xlsx)» открывается окно выбора места сохранения отчета в формате xlsx. Это продемонстрировано на рисунке 7.

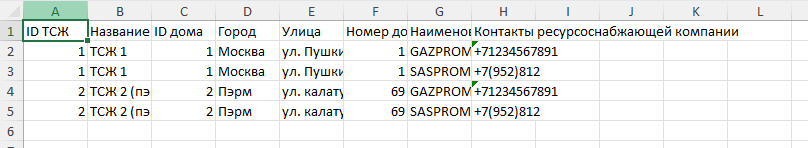


Рисунок 7. Отчет с выводом всех ТСЖ и их ресурсоснабжающих организаций в Excel.

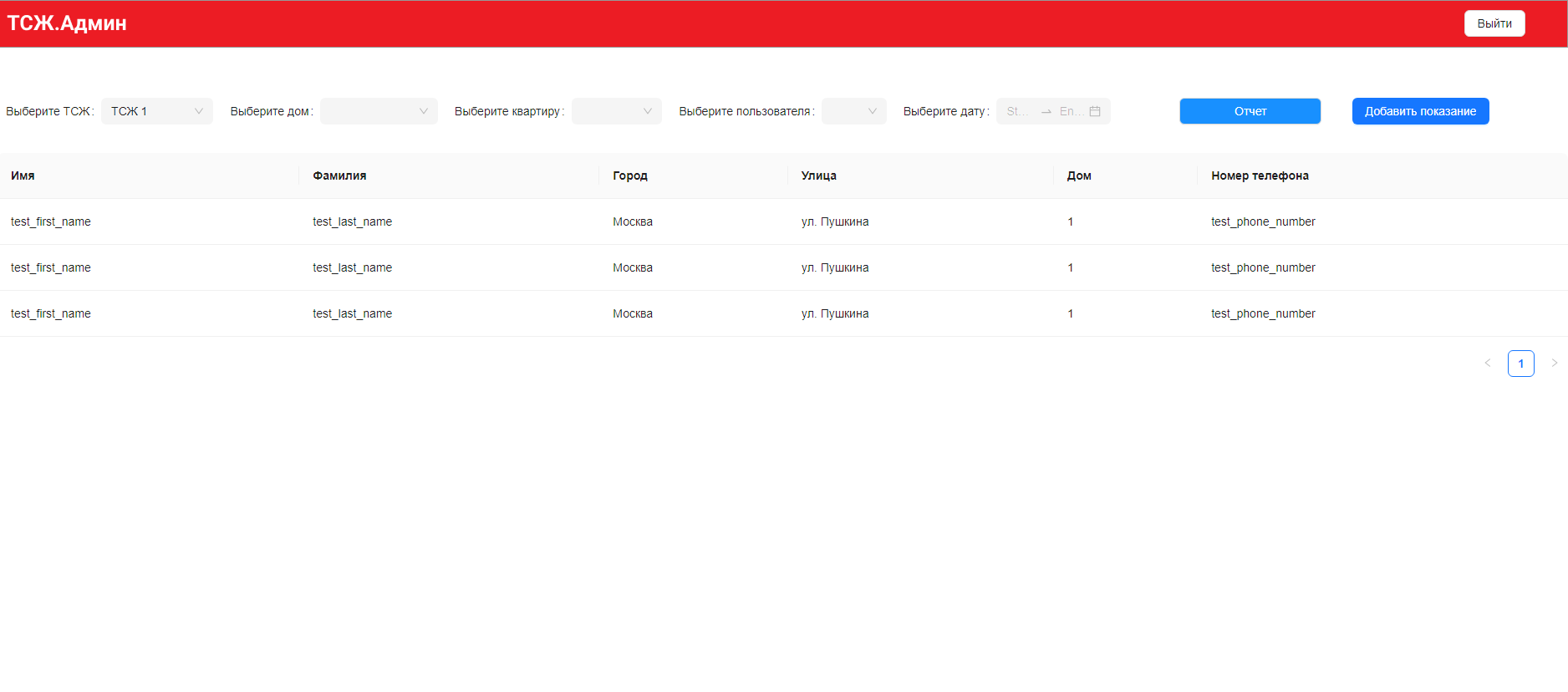


Рисунок 8. Отчет с выводом информации о жильцах по выбранному ТСЖ.

## Добавление, удаление и изменение данных

На рисунке 9 показаны изначальные показания счетчиков.

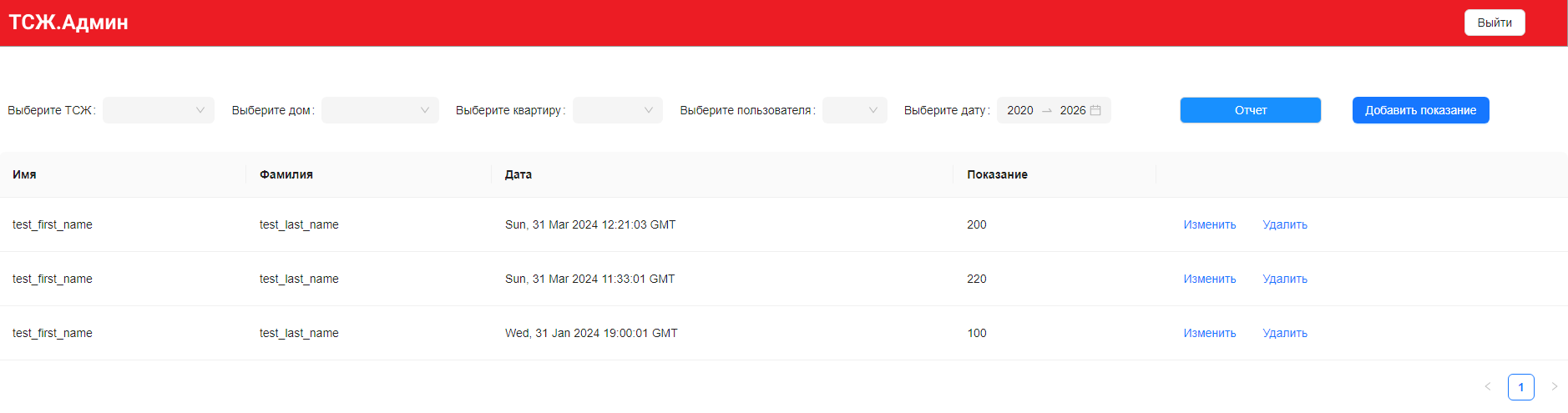


Рисунок 10. Вывод всех значений счетчиков

После нажатия на кнопку «Добавить показание» появится форма для заполнения данных о новом показании, как показано на рисунке 11.

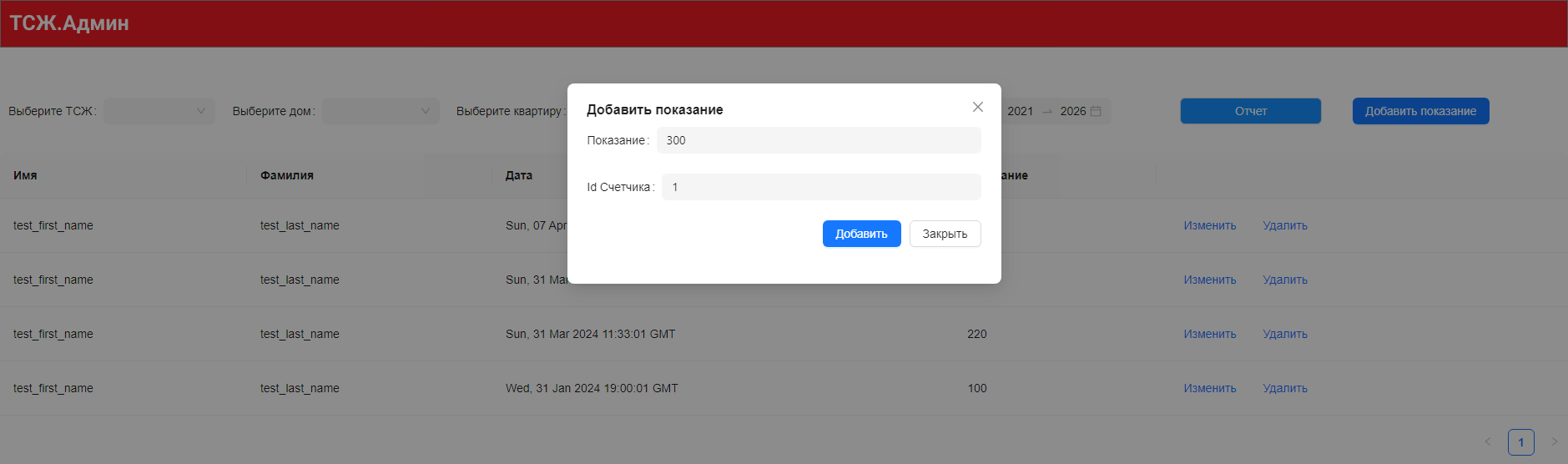
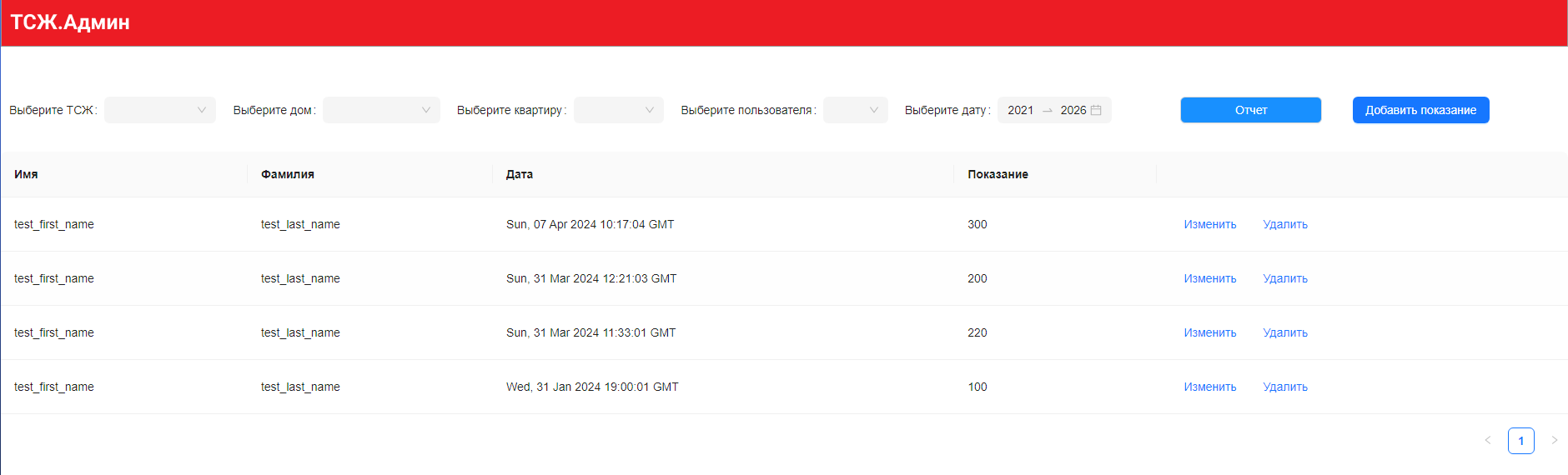


Рисунок 11. Окно добавления значений в показания счетчика.

После добавления показания, в список добавится новое показание. Это продемонстрировано на рисунке 12.

Рисунок 12. Вывод обновленных значений счетчиков

Для изменения конкретного показания, можно нажать кнопку «Изменить». После этого откроется окно для ввода данных:

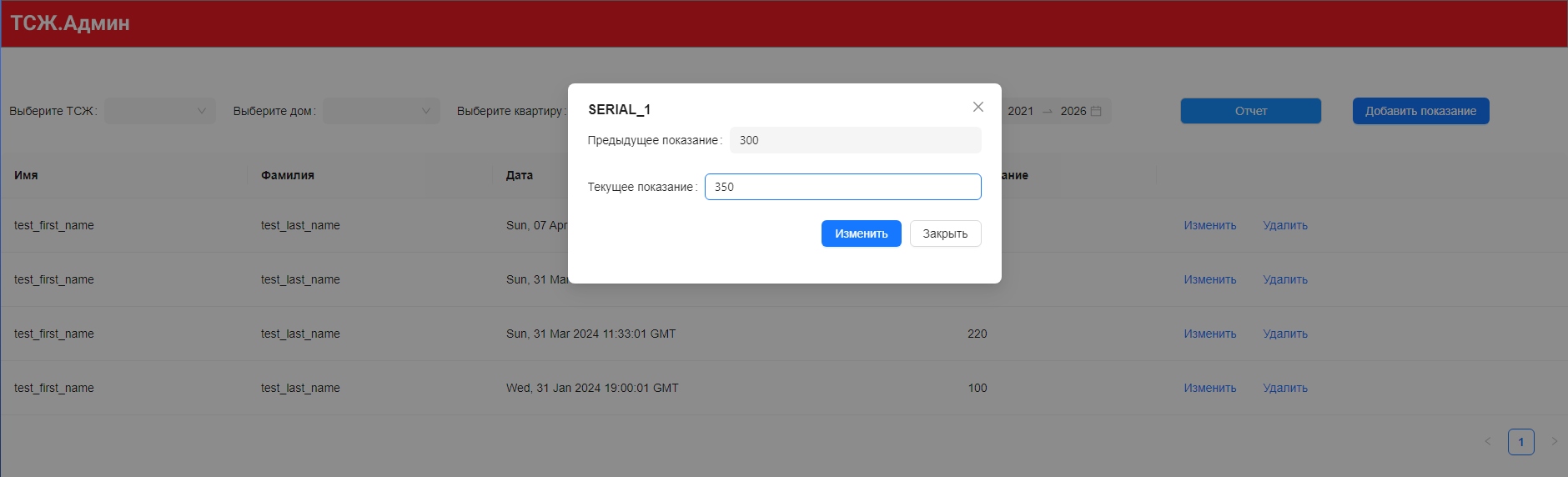


Рисунок 13. Окно для изменения показания счетчика.

После изменения показания, список со всеми значениями будет выглядеть так:

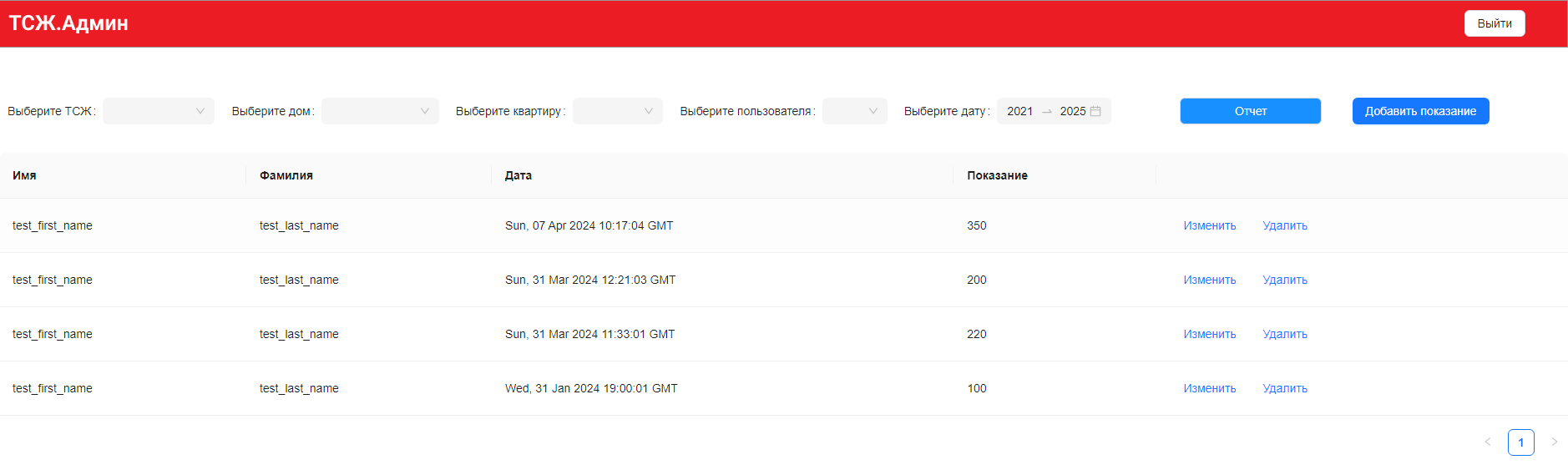


Рисунок 14. Вывод обновленных значений счетчиков

Для удаления показания необходимо нажать кнопку «Удалить» рядом с этим показанием. Список всех показаний, после удаления первого показания будет выглядеть так:

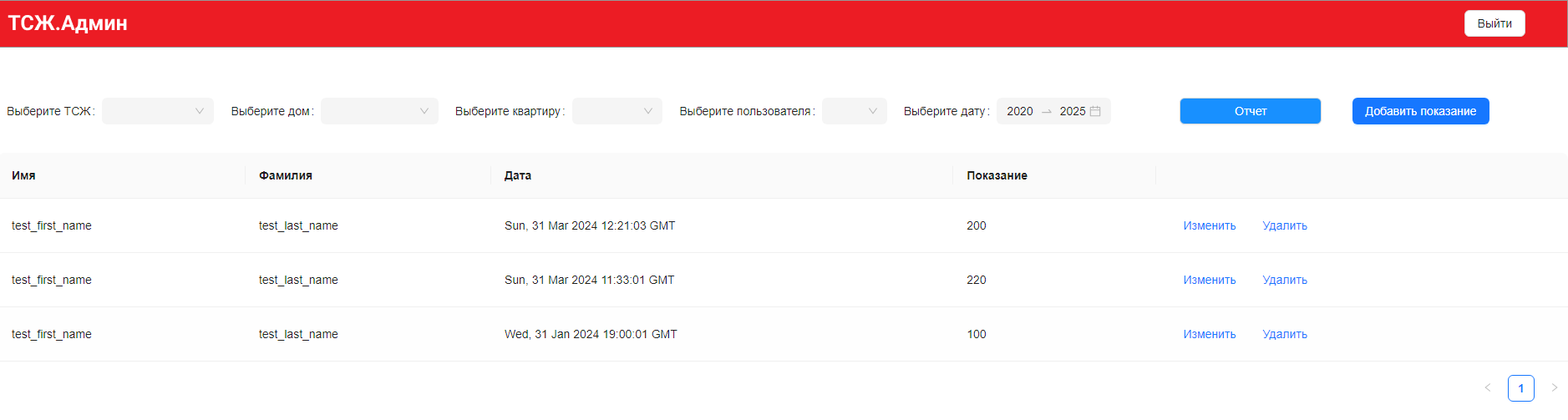


Рисунок 15. Вывод обновленных значений счетчиков.

## Обработка исключительных ситуаций

Система предусматривает обработку исключительных ситуаций, к примеру на этапах авторизации и генерации отчетов.

На рисунке 16 продемонстрирована реакция ИС на отсутствие введенных данных.

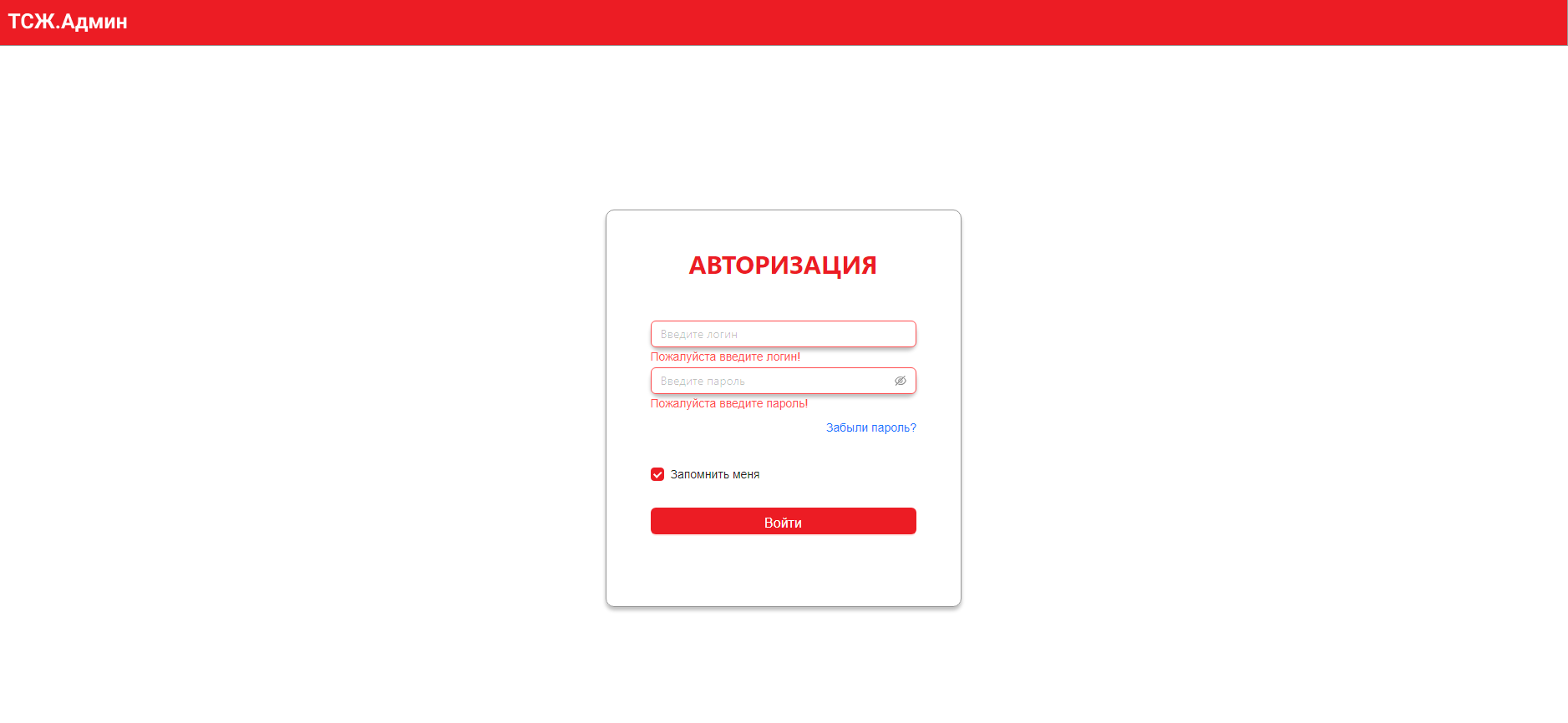


Рисунок 16. Реакция ИС на отсутствие введенных данных для авторизации.

На рисунке 17 показана неудавшаяся авторизация из-за некорректных данных.

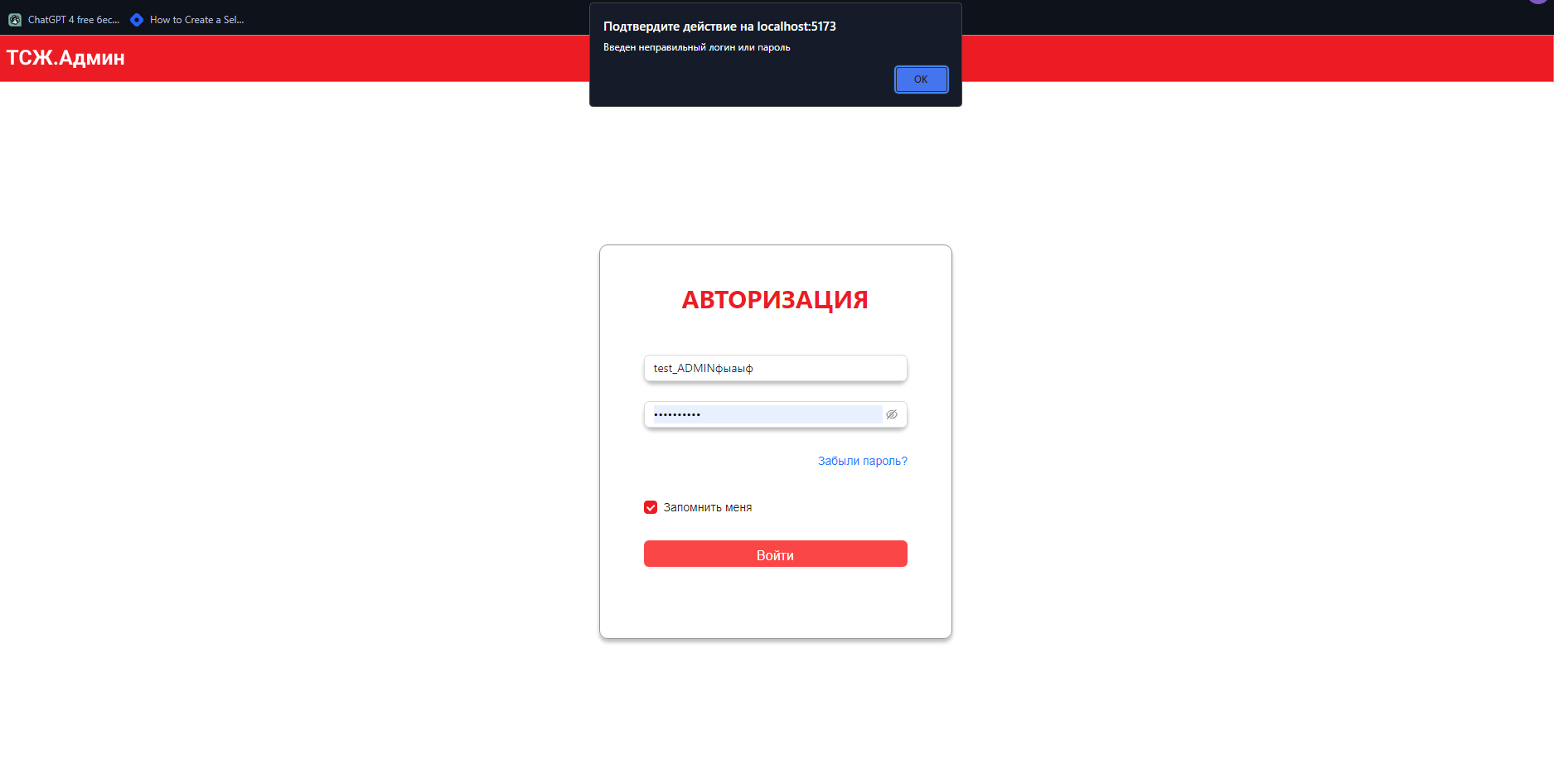


Рисунок 17. Реакция ИС при неудавшейся авторизации.

На рисунке 18 показана реакция ИС на отсутствие введенных данных при попытке генерации отчета.

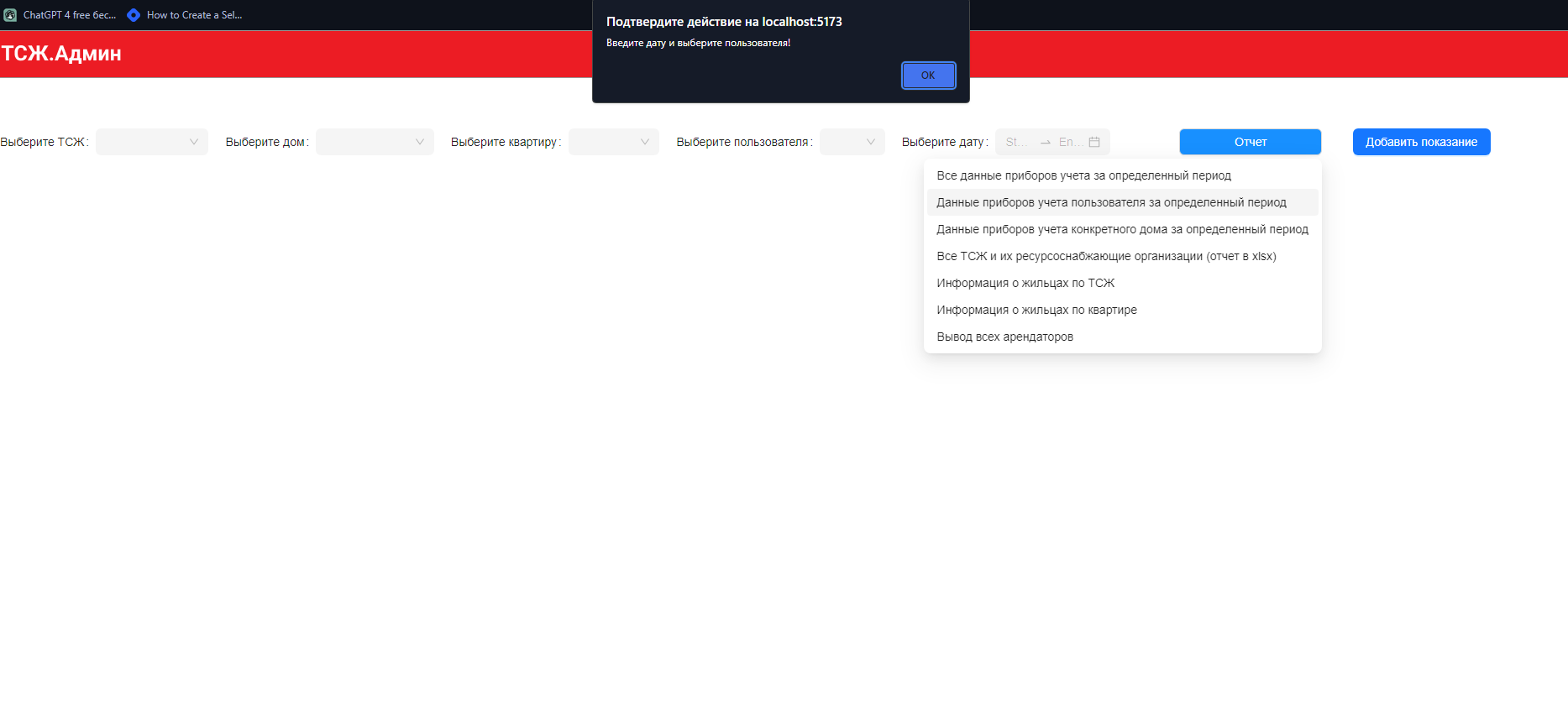


Рисунок 18. Реакция ИС на отсутствие введенных данных.

# Вывод

В ходе данной работы была разработана КИС для администрирования системы подачи показаний приборов учета. Все цели и требования были выполнены. К плюсам разработанной КИС можно отнести то, что пользователю не требуется установка дополнительного ПО, так-как доступ к приложению осуществляется через браузер. С точки зрения развертывания веб-приложения на сервере, система хороша тем, что для её работы не нужно ничего, кроме кроссплатформенной Java 8 и СУБД PostgreSQL с nginx, которые адаптированы под большинство популярных ОС.

# Список литературы

1. За десять лет жилой фонд России увеличился на 20% / [Электронный ресурс] //РБК: [сайт]. URL: <https://www.rbc.ru/industries/news/654276e59a79474d8ec43c17> (дата обращения: 06.01.2024).
2. Гаврилов, С. В., Трошина, М. В. UX/UI-Дизайн интерфейса при разработке мобильных приложений / С. В. Гаврилов, М. В. Трошина [Текст] // Наука. Технология. Производство – 2023. – Уфа: Уфимский государственный нефтяной технический университет, 2023. – С. 136-137.
3. Маношин, Д. А. Использование возможностей Java в современной веб-разработке / Д. А. Маношин // Colloquium-Journal. – 2019. – № 10-2(34). – С. 122-124.
4. А. Ильясов, Г. Чарыева База данных и требования к базе данных // Ceteris Paribus. 2022. №12.
5. Сравнительные характеристики SQL и NOSQL СУБД, влияющие на разработку приложений баз данных / Е. М. Куприянчик, Т. В. Зудилова, И. В. Ананченко [и др.] // Современные наукоемкие технологии. – 2022. – № 7. – С. 74-78.
6. Оптимизация SQL-запросов СУБД / М. Ю. Перухин, Д. С. Сафиуллин, Г. Н. Ямалеева, Р. Н. Гайнуллин // Наука и образование: проблемы, идеи, инновации. – 2018. – № 7(10). – С. 27-30.
7. Левшин, Д. В. Алгоритмы интеграции субд PostgreSQL с семантическим веб / Д. В. Левшин, А. С. Марков // Программирование. – 2009. – Т. 35, № 3. – С. 22-34.

# Приложение

|  |
| --- |
| @Override public boolean hasPermission(Authentication authentication, Object resource, Object minRole) {  SecurityUser user;  if (authentication.getPrincipal() instanceof User)  user = (SecurityUser) authentication.getPrincipal();  else  throw new UserNotFoundException("Ошибка получения пользователя");  if (user.getUser().isAdmin())  return true;   RequestAttributes requestAttributes = RequestContextHolder.getRequestAttributes();   if (resource.equals("tsj")) {  Long tsjId = null;  try {  tsjId = Long.parseLong(((ServletRequestAttributes)requestAttributes).getRequest().getParameter(TSJ\_ID));  } catch (Exception e) {  throw new IllegalArgumentException(e.getMessage());  }   return user.getUser().getId().equals(tsjService.getManager(tsjId).getId());  }  else if (resource.equals("flat")) {  Long flatId = null;  try {  flatId = Long.parseLong(((ServletRequestAttributes)requestAttributes).getRequest().getParameter(FLAT\_ID));  } catch (Exception e) {  throw new IllegalArgumentException(e.getMessage());  }   return userInFlatService.checkRole(user.getUser().getId(), flatId, Role.valueOf(minRole.toString()));  }  throw new EntityIllegalArgumentException("Данный объект не поддерживается");  } |
| Исходный код, отвечающий за разделенный доступ к информационной базе. |

|  |
| --- |
| public interface TsjRepository extends JpaRepository<Tsj, Long> {  Tsj findById(Long id);  void deleteById(Long id);  @Query(value = "select distinct\n" +  "tsj.id as tsjId, tsj.name as tsjName, a.id as addressId, a.city as city, a.street as street,\n" +  "a.house\_number as housenumber, p.name as providername, p.contact as providercontact\n" +  "from tsj\n" +  "left join address a on tsj.id = a.tsj\_id\n" +  "left join flat f on a.id = f.address\_id\n" +  "left join meter m on m.flat\_id = f.id\n" +  "left join resource r on r.id = m.resource\_id\n" +  "inner join provider p on p.id = r.provider\_id\n" +  "order by tsj.id asc", nativeQuery = true)  List<Object[]> getReport();  } |
| Исходный код, отвечающий за добавление, изменение и удаление данных о сущности ТСЖ в БД. |

|  |
| --- |
| @Service public class TsjReporter {   private final TsjService tsjService;   public TsjReporter(TsjService tsjService) {  this.tsjService = tsjService;  }    public void getTsjReport(HttpServletResponse response) {  XSSFWorkbook workbook = new XSSFWorkbook();  XSSFSheet sheet = workbook.createSheet();   createHeaderRow(sheet, 0);   List<TsjAndProviderDto> reportData = tsjService.getTsjAndProviderDtos();  for (int i = 0; i < reportData.size(); i++)  addFilledRow(sheet, i + 1, reportData.get(i));   try {  response.addHeader("Content-Disposition", "attachment; filename=report.xlsx");  workbook.write(response.getOutputStream());  }  catch (Exception e) {  throw new EntityIllegalArgumentException("Не удалось произвести эскпорт отчета!");  }  }   private void createHeaderRow(XSSFSheet sheet, int index) {  XSSFRow row = sheet.createRow(index);   row.createCell(0).setCellValue("ID ТСЖ");  row.createCell(1).setCellValue("Название ТСЖ");  row.createCell(2).setCellValue("ID дома");  row.createCell(3).setCellValue("Город");  row.createCell(4).setCellValue("Улица");  row.createCell(5).setCellValue("Номер дома");  row.createCell(6).setCellValue("Наименование ресурсоснабжающей компании");  row.createCell(7).setCellValue("Контакты ресурсоснабжающей компании");  }   private void addFilledRow(XSSFSheet sheet, int index, TsjAndProviderDto dto) {  XSSFRow row = sheet.createRow(index);   row.createCell(0).setCellValue(dto.getTsjId());  row.createCell(1).setCellValue(dto.getTsjName());  row.createCell(2).setCellValue(dto.getAddressId());  row.createCell(3).setCellValue(dto.getCity());  row.createCell(4).setCellValue(dto.getStreet());  row.createCell(5).setCellValue(dto.getHouseNumber());  row.createCell(6).setCellValue(dto.getProviderName());  row.createCell(7).setCellValue(dto.getProviderContact());  } } |
| Исходный код, отвечающий за формирование отчетов в xlsx формате. |

|  |
| --- |
| @ControllerAdvice  public class ExceptionController {  @ExceptionHandler(BaseException.class)  @ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)  @ResponseBody  public ErrorResponseEntity handleBaseException(BaseException e) {  return createErrorResponseEntity(e, HttpStatus.INTERNAL\_SERVER\_ERROR);  }  @ExceptionHandler(EntityAlreadyExistsException.class)  @ResponseStatus(HttpStatus.CONFLICT)  @ResponseBody  public ErrorResponseEntity handleEntityAlreadyExistsException(EntityAlreadyExistsException e) {  return createErrorResponseEntity(e, HttpStatus.CONFLICT);  }  @ExceptionHandler(EntityConflictException.class)  @ResponseStatus(HttpStatus.CONFLICT)  @ResponseBody  public ErrorResponseEntity handleEntityConflictException(EntityConflictException e) {  return createErrorResponseEntity(e, HttpStatus.CONFLICT);  }  @ExceptionHandler(EntityHasDetailException.class)  @ResponseStatus(HttpStatus.CONFLICT)  @ResponseBody  public ErrorResponseEntity handleEntityHasDetailException(EntityHasDetailException e) {  return createErrorResponseEntity(e, HttpStatus.CONFLICT);  }  @ExceptionHandler(EntityIllegalArgumentException.class)  @ResponseStatus(HttpStatus.BAD\_REQUEST)  @ResponseBody  public ErrorResponseEntity handleEntityIllegalArgumentException(EntityIllegalArgumentException e) {  return createErrorResponseEntity(e, HttpStatus.BAD\_REQUEST);  }  @ExceptionHandler(EntityNotFoundException.class)  @ResponseStatus(HttpStatus.NOT\_FOUND)  @ResponseBody  public ErrorResponseEntity handleEntityNotFoundException(EntityNotFoundException e) {  return createErrorResponseEntity(e, HttpStatus.NOT\_FOUND);  }  @ExceptionHandler(UserNotFoundException.class)  @ResponseStatus(HttpStatus.UNAUTHORIZED)  @ResponseBody  public ErrorResponseEntity handleUserNotFoundException(UserNotFoundException e) {  return createErrorResponseEntity(e, HttpStatus.UNAUTHORIZED);  }  @ExceptionHandler(EntityValidationException.class)  @ResponseStatus(HttpStatus.BAD\_REQUEST)  @ResponseBody  public ErrorResponseEntity handleEntityValidationException(EntityValidationException e) {  return createErrorResponseEntity(e, HttpStatus.BAD\_REQUEST);  }  private static ErrorResponseEntity createErrorResponseEntity (BaseException e, HttpStatus httpStatus) {  return new ErrorResponseEntity(e.getMessage(), httpStatus.getReasonPhrase(), httpStatus.value());  }  } |
| Исходный код, отвечающий за обработку исключительных ситуаций. |