ФГБОУ ВПО Уральский государственный горный университет

Инженерно-экономический факультет

Кафедра информ@тики

Курсовой проект

По дисциплине «Технологии программирования» На тему «Интеллектуальная система управления Умным Домом»

Выполнил:

Студент гр. АУБП-22-1

Новосёлов Д.А.

Проверила:

ст. преп. каф. Информатики

Волкова Е.А.

Екатеринбург, 2025г.

Содержание

[1. Постановка задачи 3](#_Toc199329917)

[1.1. Характеристика объекта и проблемной области 3](#_Toc199329918)

[1.2. Постановка целей и задач автоматизации/информатизации 4](#_Toc199329919)

[1.3. Краткое техническое задание 5](#_Toc199329920)

[2. Проектирование системы 7](#_Toc199329921)

[2.1. Моделирование системы 8](#_Toc199329922)

[2.2. Проектирование БД 11](#_Toc199329923)

[2.3. Проектирование интерфейса 13](#_Toc199329924)

[3. Разработка системы 16](#_Toc199329925)

[3.1. Выбор средств реализации 17](#_Toc199329926)

[3.2. Структура проекта 18](#_Toc199329927)

[3.3. Реализация 23](#_Toc199329928)

1. Постановка задачи

В современных домах и квартирах всё чаще используются различные электронные устройства и системы автоматизации. Для повышения удобства, безопасности и энергоэффективности требуется интеллектуальная система, обеспечивающая централизованное управление устройствами, автоматизацию сценариев и ведение истории событий.

* 1. Характеристика объекта и проблемной области

Объектом автоматизации является жилое помещение (дом или квартира), оснащённое различными электронными устройствами: датчиками температуры, влажности, движения, исполнительными механизмами (реле, выключатели, отопление и др.), а также средствами отображения информации и управления.

Проблемная область:

С развитием технологий в современных домах появляется всё больше устройств, требующих управления и контроля. Ручное управление каждым устройством по отдельности становится неудобным и неэффективным, особенно при большом количестве оборудования. Возникают следующие проблемы:

* отсутствие единого интерфейса для управления всеми устройствами;
* невозможность гибко настраивать автоматические сценарии (например, включение отопления при снижении температуры, автоматическое освещение при обнаружении движения и т.д.);
* отсутствие централизованного журнала событий и истории действий пользователей;
* сложности с разграничением доступа между пользователями (например, администратор и обычный пользователь);
* необходимость интеграции с современными технологиями и обеспечения безопасности данных.

На рынке представлены различные решения для автоматизации умного дома – как коммерческие (например, системы от крупных производителей), так и открытые платформы (например, Home Assistant, OpenHAB). Однако многие из них требуют сложной настройки, не всегда поддерживают нужные сценарии или не предоставляют удобного интерфейса для конечного пользователя.

Кроме того, часто отсутствует возможность гибкой доработки под индивидуальные потребности, а также полноценная поддержка разграничения прав доступа и ведения истории событий.

* 1. Постановка целей и задач автоматизации/информатизации

Основная цель автоматизации – создание интеллектуальной системы управления умным домом, обеспечивающей централизованное, удобное и безопасное управление всеми устройствами, автоматизацию сценариев и ведение истории событий для повышения комфорта, безопасности и энергоэффективности жилого помещения.

Для достижения этой цели необходимо решить следующие задачи:

1. Разработать структуру хранения данных, позволяющую учитывать пользователей, устройства, правила автоматизации, условия и действия, а также журналировать все события.
2. Реализовать механизм регистрации, аутентификации и авторизации пользователей с разграничением прав доступа (администратор, обычный пользователь).
3. Обеспечить удобный веб-интерфейс для управления устройствами, создания и редактирования правил автоматизации.
4. Реализовать возможность настройки автоматических сценариев (правил), основанных на показаниях датчиков и состоянии устройств.
5. Вести централизованный журнал событий и действий пользователей с возможностью просмотра истории.
6. Обеспечить безопасность хранения и передачи данных между пользователем и системой.
7. Реализовать административный интерфейс для управления пользователями, устройствами и правилами
   1. Краткое техническое задание

Наименование системы: Интеллектуальная система управления умным домом

Назначение системы: Автоматизация управления устройствами умного дома, настройка сценариев и ведение истории событий через веб-интерфейс.

Функциональные требования (F — Functionality):

Система должна обеспечивать регистрацию и аутентификацию пользователей с разграничением ролей (администратор, пользователь).

1. Пользователь должен иметь возможность просматривать список доступных ему устройств, их текущее состояние и историю изменений.
2. Пользователь должен иметь возможность добавлять, удалять и изменять устройства (при наличии соответствующих прав).
3. Пользователь должен иметь возможность создавать, редактировать и удалять правила автоматизации (сценарии), состоящие из условий и действий.
4. Система должна автоматически выполнять действия по правилам при наступлении заданных условий.
5. Система должна вести журнал всех действий пользователей и автоматических событий с возможностью фильтрации по дате, устройству, пользователю и типу действия.
6. Администратор должен иметь доступ к управлению пользователями, устройствами и правилами всех пользователей.

Требования к удобству использования (U — Usability):

1. Веб-интерфейс должен быть реализован на русском языке.
2. Все основные действия должны быть доступны не более чем в 3 клика от главной страницы.
3. Интерфейс должен быть адаптирован для экранов с шириной от 1024px (десктоп) и выше.
4. Цветовая схема интерфейса: фон — #111, основные элементы — #fff, акцентные цвета — #222, #f55.

Надёжность (R — Reliability):

1. Система должна обеспечивать корректную работу при одновременной работе не менее 10 пользователей.
2. Все критические операции (регистрация, изменение пароля, добавление/удаление устройств и правил) должны сопровождаться подтверждением и валидацией данных.
3. В случае сбоя соединения с базой данных пользователь должен получать информативное сообщение об ошибке.

Производительность (P — Performance):

1. Время отклика на основные действия пользователя (открытие страниц, выполнение операций) не должно превышать 2 секунд при нормальной нагрузке.
2. Система должна поддерживать хранение не менее 1000 записей в журнале действий без снижения производительности.

Поддержка (S — Supportability):

1. Система должна быть реализована с использованием Java 17+, Spring Boot 3.x, MySQL 8.x, что обеспечивает кроссплатформенность и возможность развертывания на большинстве современных серверов.
2. Исходный код и инструкции по развертыванию должны быть предоставлены в виде архива .zip.
3. Все миграции БД должны быть оформлены с помощью Flyway.
4. Проектирование системы

В рамках курсового проекта разработана многоуровневая интеллектуальная система управления умным домом. Архитектура системы включает три основных уровня:

* Уровень представления — веб-интерфейс для пользователей, реализованный с помощью Thymeleaf, HTML и CSS.
* Уровень бизнес-логики — серверная часть на Java (Spring Boot), обеспечивающая обработку пользовательских запросов, выполнение сценариев автоматизации и контроль доступа.
* Уровень данных — реляционная база данных MySQL для хранения информации о пользователях, устройствах, правилах, действиях и истории событий.

В системе реализованы основные сущности: пользователь, устройство, правило, условие, действие, журнал действий и связь пользователь-устройство.

Взаимодействие между компонентами построено по принципу MVC: пользователь взаимодействует с веб-интерфейсом, запросы обрабатываются контроллерами, бизнес-логика реализована в сервисах, а данные хранятся и извлекаются через репозитории.

Для обеспечения безопасности используется Spring Security с JWT-аутентификацией и разграничением прав доступа.

Миграции структуры базы данных выполняются с помощью Flyway, что обеспечивает удобство поддержки и масштабирования системы.

* 1. Моделирование системы

В системе «Интеллектуальная система управления умным домом» выделяются следующие основные участники: Пользователь(USER), Администратор(ADMIN).

Основные варианты использования:

* Регистрация и вход в систему;
* Просмотр списка устройств и их состояния;
* Добавление/удаление устройств (Администратор);
* Управление устройствами (изменение состояния);
* Создание, редактирование и удаление правил автоматизации;
* Просмотр и фильтрация журнала действий;
* Управление пользователями (Администратор)

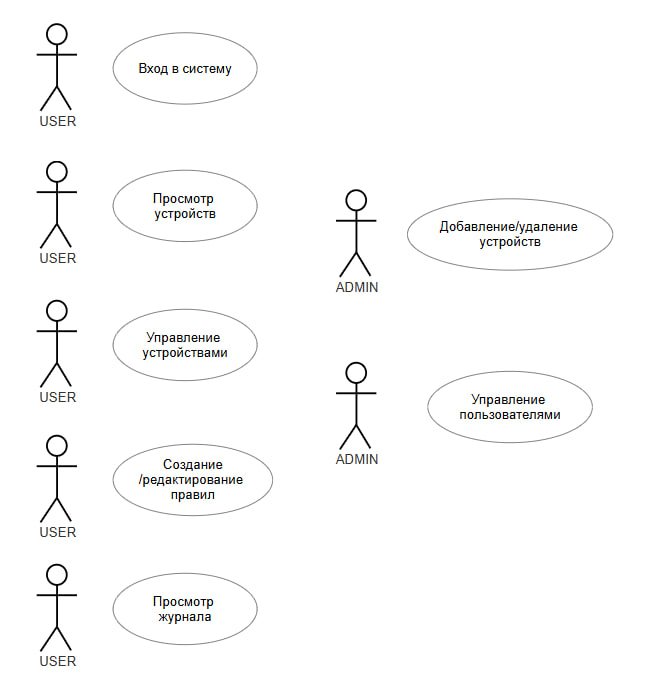


Рисунок 1 Диаграмма Use Case вариантов использования системы управления умным домом

Контекстная диаграмма отражает взаимодействие внешних участников с системой:

Пользователь/Администратор → (отправляет запросы через браузер) → Веб-интерфейс (Thymeleaf)

Веб-интерфейс (Thymeleaf) ↔ (HTTP-запросы/ответы) ↔ Spring Boot Backend

Spring Boot Backend ↔ (SQL-запросы/ответы) ↔ MySQL Databas

Изображение выглядит как текст, Шрифт, Самоклеющийся листок, Красочность

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 Контекстная диаграмма взаимодействия компонентов системы

На диаграмме классов системы управления умным домом показаны основные сущности проекта и связи между ними. Один пользователь может иметь много устройств и записей в журнале. Одно устройство может быть связано с несколькими пользователями, условиями, действиями и журналами. Одно правило содержит несколько условий и действий. Журнал действий связывает пользователя, устройство и правило, фиксируя все события.

Изображение выглядит как текст, диаграмма, План, схематичный

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 UML-диаграмма классов системы управления умным домом

На диаграмме потоков данных показано, как пользователь взаимодействует с системой при добавлении устройства или при срабатывании автоматического правила. Все действия проходят через веб-интерфейс и сервер приложения, данные сохраняются в базе данных, а события фиксируются в журнале действий. Модуль автоматизации анализирует условия и инициирует действия при необходимости.Изображение выглядит как текст, диаграмма, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 DFD-схема потоков данных системы управления умным домом

* 1. Проектирование БД

В качестве основы для проектирования базы данных использована реляционная модель, реализованная в MySQL. На ER-диаграмме (в нотации Crow’s Foot) отражены основные сущности системы: пользователи, устройства, правила, условия, действия, журнал действий, а также связи между ними.

Изображение выглядит как текст, диаграмма, План, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 ER-диаграмма

Основные таблицы:

users – пользователи системы;

устройства – устройства умного дома;

правила – правила автоматизации;

условия – условия срабатывания;

правил действия – действия, выполняемые по правилам;

журнал\_действий м журнал событий и действий;

user\_devices – связь между пользователями и устройствами;

данные\_устройств – история значений устройств

Описание всех основных таблиц и полей в системе:

Пользователи (users): Таблица хранит информацию о пользователях системы. Каждый пользователь имеет уникальный идентификатор, имя, хэш пароля, роль (администратор или обычный пользователь) и электронную почту.

Устройства (устройства): В этой таблице содержатся сведения об устройствах, подключённых к системе умного дома. Для каждого устройства указывается его название, тип (например, сенсор или реле), расположение, текущее состояние, дата добавления и идентификатор владельца (пользователя).

Правила (правила): В этой таблице описываются автоматические сценарии (правила), которые могут быть активны или неактивны. Для каждого правила хранится его название, статус активности и дата создания.

Условия (условия): Таблица условий определяет, при каких обстоятельствах срабатывают правила. Каждое условие связано с определённым правилом и устройством, содержит оператор сравнения (например, больше(">"), меньше("<"), равно("="), значение для сравнения и логику объединения с другими условиями (AND/OR).

Действия (действия): В этой таблице описываются действия, которые выполняются при срабатывании правил. Для каждого действия указывается, к какому правилу и устройству оно относится, тип действия (например, включить, выключить, установить значение) и, при необходимости, само значение.

* 1. Проектирование интерфейса

В данном разделе представлены скетчи основных экранов (представлений) приложения для системы умного дома. Интерфейс разрабатывался с учётом удобства пользователя, интуитивной навигации и быстрого доступа к основным функциям.

1. Экран авторизации

Пользователь вводит логин и пароль для входа в систему. На экране предусмотрены ввод логина, пароля, кнопка "Войти", а также, если пользователь ещё не зарегистрирован, то есть ссылка "Зарегистрироваться"

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 Экран авторизации в систему Умный Дом

1. Главная страница (Панель управления)

После успешного входа пользователь попадает на главную страницу, где отображается список всех его устройств с указанием их текущего состояния, типа и расположения. На экране есть кнопка "Управление устройствами" для подключения новых устройств, кнопка "Перейти к правилам" для просмотра правил про каждое устройство, которое пользователь добавил себе в Умный Дом, а также кнопка "Выйти" для выхода из системы.

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7 Главная страница пользователя

1. Страница правил

На этой странице пользователь видит список всех созданных правил устройств, но только те устройства, которые он добавил в "свои". Для каждого правила отображается его название, статус (активно/неактивно) и дата создания. Это было создано для того, чтобы пользователь понимал, как работает устройство и нужно ли оно ему.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 Страница правил устройств пользователя

1. Журнал действий

Пользователь может просмотреть историю всех выполненных действий и событий в системе. На экране отображается таблица с датой и временем события, названием устройства, описанием действия и результатом выполнения. Для удобства предусмотрены фильтры по дате, устройству и результату выполнения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9 Журнал действий

1. Разработка системы

В данном разделе подробно описывается процесс создания программного обеспечения для системы Умного Дома, реализованного в рамках данного проекта. Приводится обоснование выбора технологий, структура проекта, а также основные этапы реализации ключевых компонентов приложения. Система реализована как веб-приложение на языке программирования Java с использованием фреймворка Spring Boot. В качестве системы управления базами данных используется MySQL, причём все таблицы и столбцы в базе данных имеют русскоязычные наименования(за исключением user\_devices), что потребовало особого внимания при настройке ORM и аннотаций сущностей. Для взаимодействия с базой данных применяется Spring Data JPA. Веб-интерфейс реализован с использованием шаблонов (Thymeleaf) и стандартных средств Spring MVC. Для обеспечения безопасности реализована система аутентификации и авторизации пользователей с разграничением ролей (Администратор, Пользователь).

Проект структурирован по принципу разделения ответственности: выделены слои контроллеров, сервисов, репозиториев и сущностей. Для хранения и обработки данных используются сущности, соответствующие структуре базы данных. Взаимодействие с пользователем осуществляется через веб-интерфейс, реализованный с помощью MVC-подхода.

В результате реализации был получен программный продукт, позволяющий пользователям управлять устройствами Умного Дома, создавать автоматические сценарии, просматривать историю событий и взаимодействовать с системой через удобный веб-интерфейс. Система поддерживает регистрацию и аутентификацию пользователей, разграничение прав доступа, а также хранение и обработку данных в базе данных MySQL с использованием русскоязычных наименований таблиц и столбцов.

* 1. Выбор средств реализации

Для реализации системы Умного Дома я выбрал язык программирования Java. Этот язык мне хорошо знаком, он широко используется для создания надёжных серверных приложений и обладает богатой экосистемой библиотек. Кроме того, Java обеспечивает кроссплатформенность, что позволяет запускать приложение на различных операционных системах. В качестве основного фреймворка я использовал Spring Boot. Он значительно упрощает разработку веб-приложений, позволяет быстро запускать проект и легко подключать необходимые модули. Мне особенно понравилось, что Spring Boot поддерживает автоматическую конфигурацию и предоставляет удобные средства для построения REST API и MVC-приложений. Для работы с базой данных я выбрал Spring Data JPA. Этот модуль позволяет удобно описывать сущности и репозитории, а также избавляет от необходимости писать большую часть SQL-запросов вручную. В моём проекте структура базы данных использует русскоязычные имена таблиц и столбцов, поэтому возможность явно указывать соответствие между полями классов и колонками таблиц оказалась очень полезной. В качестве СУБД я решил использовать MySQL. Это одна из самых популярных и надёжных реляционных баз данных, с которой у меня уже был опыт работы. MySQL хорошо интегрируется с Java-приложениями и поддерживается Spring Boot без дополнительных сложностей. Кроме того, я выбрал русскоязычные наименования таблиц и полей, чтобы структура базы данных была максимально понятна для русскоязычных пользователей. Для обеспечения безопасности я реализовал аутентификацию и авторизацию с помощью Spring Security. Это стандартное решение для Java-приложений, которое позволяет гибко настраивать права доступа и защищать пользовательские данные. В системе предусмотрено разграничение ролей (администратор и обычный пользователь), что важно для управления доступом к различным функциям. Веб-интерфейс я реализовал на основе Spring MVC с использованием шаблонов (Thymeleaf). Такой подход позволяет создавать динамические страницы и обеспечивает удобное взаимодействие пользователя с системой. Для управления зависимостями и сборки проекта я использовал Maven. Это удобный инструмент, который позволяет легко подключать необходимые библиотеки и управлять процессом сборки. В процессе разработки и тестирования я пользовался средой IntelliJ IDEA, а также встроенными средствами Spring Boot для запуска и мониторинга приложения. Таким образом, я выбрал именно эти технологии, потому что они хорошо сочетаются между собой, позволяют быстро и удобно разрабатывать надёжные веб-приложения, а также обеспечивают все необходимые возможности для реализации системы умного дома.

* 1. Структура проекта

Проект построен по стандартной структуре Spring Boot-приложения. В каталоге src/main/java/com/smarthome размещён основной исходный код, который разделён на отдельные пакеты. Ниже предствлена структура моего проекта Умного Дома.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 10 Структура проекта(1 часть)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11 Структура проекта(2 часть)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 Структура проекта(3 часть)

В моём проекте основное внимание уделялось сущностям, потому что они напрямую отражают структуру базы данных и бизнес-логику Умного Дома: User, Device(Устройство), Rule(Правило), Condition (Условие), Action (Действие), DeviceData (Данные устройства), ActionLog (Журнал действий), UserDevice.

Изображение выглядит как текст, диаграмма, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 UML-диаграмма классов сущностей(Entity)

В проекте отдельное внимание уделялось контроллерам, так как именно они обеспечивают взаимодействие пользователя с системой через веб-интерфейс. Каждый контроллер отвечает за обработку запросов, связанных с определённой областью функциональности: UserController – за управление пользователями, DeviceController – за работу с устройствами, RuleController – за создание и управление правилами автоматизации. Контроллеры взаимодействуют с сервисным слоем, который реализует бизнес-логику, и через сервисы получают доступ к данным, хранящимся в базе. Такой подход позволяет чётко разделить ответственность между слоями приложения и обеспечивает удобство поддержки и расширения системы.

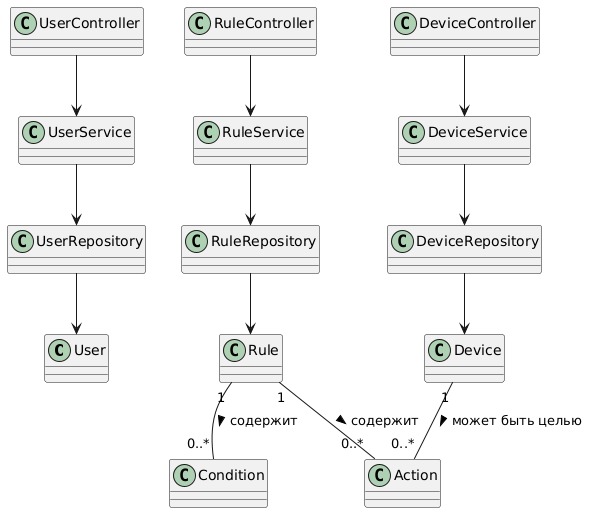


Рисунок 14 UML-диаграмма основных контроллеров проекта

В проект также были добавлены репозитории так как они обеспечивают доступ к данным, хранящимся в базе. Каждый репозиторий отвечает за работу с определённой сущностью: UserRepository – за пользователей, DeviceRepository – за устройства, RuleRepository – за правила автоматизации, DeviceDataRepository – за историю показаний устройств, ActionLogRepository – за журнал действий, UserDeviceRepository – за связи между пользователями и устройствами. Репозитории реализованы на основе Spring Data JPA, что позволяет удобно и эффективно выполнять операции поиска, сохранения и удаления данных без необходимости писать SQL-запросы вручную. Таким образом, работать с базой данных более удобнее и можно быстрее разрабатыватьновые функции и значительно упрощает поддежку.

Изображение выглядит как диаграмма, текст, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 UML-диаграмма репозиториев проекта

Краткое описание структуры решения:

В процессе разработки системы умного дома я придерживался принципа разделения ответственности, что позволило сделать проект более структурированным и удобным для поддержки. Вся логика приложения разделена на несколько основных слоёв. Контроллеры отвечают за обработку входящих HTTP-запросов и взаимодействие с пользователем через веб-интерфейс. В сервисах сосредоточена основная бизнес-логика: здесь реализуются все операции с устройствами, правилами, условиями и действиями. Для работы с базой данных я использовал сущности, которые полностью соответствуют структуре таблиц в MySQL, а доступ к данным реализован через репозитории на основе Spring Data JPA. Это позволило минимизировать количество ручного SQL-кода и упростить работу с данными.

В проекте также выделены ресурсы, где хранятся шаблоны представлений для веб-интерфейса, статические файлы (такие как стили и скрипты), а также конфигурационные файлы приложения. Такой подход позволил сделать архитектуру решения модульной и расширяемой: при необходимости можно легко добавить новые функции или изменить существующие, не затрагивая остальные части системы. Все основные сущности и их связи напрямую отражают структуру базы данных, что значительно облегчает интеграцию и дальнейшее развитие проекта.

* 1. Реализация

В данном разделе приведены примеры реалазиции ключевых функции моего проекта Умный Дом, описание структуры базы данных, а также примеры интерфейса приложения.

1. Аутентификация пользователя (фрагмент AuthController)

@PostMapping("/login")  
public ResponseEntity<?> login(@Valid @RequestBody LoginRequest loginRequest) {  
 Authentication authentication = authenticationManager.authenticate(  
 new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword())  
 );  
  
 SecurityContextHolder.*getContext*().setAuthentication(authentication);  
 UserDetails userDetails = (UserDetails) authentication.getPrincipal();  
 String jwt = jwtTokenUtil.generateToken(userDetails);  
  
 User user = userRepository.findByUsername(loginRequest.getUsername())  
 .orElseThrow(() -> new RuntimeException("User not found"));  
  
 return ResponseEntity.*ok*(new AuthResponse(jwt, user.getUsername(), user.getRole().name()));  
}

Данный метод обрабатывает запрос на вход пользователя: выполняет аутентификацию, генерирует JWT-токен и возвращает его вместе с ролью пользователя.

1. Получение устройств пользователя (фрагмент DeviceController)

@GetMapping("/api/devices/user/{userId}")

public ResponseEntity<List<Device>> getDevicesByUserId(@PathVariable Integer userId) {

    return ResponseEntity.ok(deviceRepository.findByПользователь\_UserId(userId));

}

Метод возвращает список устройств, принадлежащих конкретному пользователю.

1. Получение истории данных устройства (фрагмент DeviceDataController)

@GetMapping("/device/{deviceId}")

public ResponseEntity<List<DeviceData>> getDeviceDataByDeviceId(@PathVariable Integer deviceId) {

    return ResponseEntity.ok(deviceDataRepository.findByУстройствоIdOrderByДатаВремяDesc(deviceId));

}

Метод возвращает историю показаний выбранного устройства, отсортированную по дате.

1. Пример работы с репозиторием (фрагмент UserRepository)

public interface UserRepository extends JpaRepository<User, Integer> {

    Optional<User> findByUsername(String *username*);

    Optional<User> findByEmail(String *email*);

    boolean existsByUsername(String *username*);

    boolean existsByEmail(String *email*);

}

Репозиторий предоставляет методы для поиска и проверки существования пользователей по логину и email.

Структура базы данных

В своей базе данных я реализовал все основные сущности, необходимые для работы системы Умного Дома, и продумал логические связи между ними. Пользователи хранятся в таблице users, а их устройства – в таблице устройства, где для каждого устройства указывается владелец. Историю показаний устройств я сохраняю в таблице данные\_устройств, каждая запись которой связана с конкретным устройством. Автоматизация реализована через таблицу правила: для каждого правила я могу задать условия (таблица условия) и действия (таблица действия), при этом каждое условие связано с определённым устройством, а каждое действие – с устройством-целью. Все события и выполненные действия фиксируются в журнале действий, где для каждой записи указывается, по какому правилу и с каким устройством было выполнено действие, а также, если применимо, какой пользователь его инициировал. Для поддержки совместного использования устройств я реализовал таблицу user\_devices, которая связывает пользователей и устройства по принципу «многие ко многим». Такая структура обеспечивает целостность данных, позволяет гибко управлять устройствами, автоматизировать сценарии и отслеживать всю историю событий в системе.

Ниже приведён листинг создание БД – smart\_home\_db:

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 16 Листинг БД(1 часть)

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 17 Листинг БД(2 часть)

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 18 Листинг БД(3 часть)

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 19 Листинг БД(4 часть)

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 20 Листинг БД(5 часть)

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 21 Скриншот экрана авторизации

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 22 Скриншот главной страницы(панель управления)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 23 Скриншот списка устройств

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 24 Скриншот списка правил

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 25 Скриншот журнал действий

В результате реализации был создан программный продукт управления Умным Домом, позволяющий пользователям управлять устройствами, создавать автоматические сценарии, просматривать историю событий и взаимодействовать с системой через удобный веб-интерфейс. Все основные функции реализованы с использованием современных технологий Java, Spring Boot и MySQL.