Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись
«_____» _____ 2023 г.

БАКАЛАВРСКАЯ РАБОТА

Веб-приложение для управления умным домом

09.03.01 – «Информатика и вычислительная техника»

Руководитель _____ ст. преподаватель К. В. Пушкарев
подпись, дата

Выпускник _____ А.С. Васильев
подпись, дата

Нормоконтролер _____ К. В. Пушкарев
подпись, дата

Красноярск 2023

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись
«_____» _____ 20 ___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Красноярск 2022

Студенту Васильеву Александру Сергеевичу.

Группа: КИ19-06Б. Направление (специальность): 09.03.01 «Информатика и вычислительная техника».

Тема выпускной квалификационной работы: «Веб-приложение для управления умным домом».

Утверждена приказом по университету № 4765/С от 23.03.2023.

Руководитель ВКР: Пушкарев К. В., старший преподаватель кафедры ВТ ИКИТ СФУ.

Исходные данные для ВКР: разработать веб-приложение для управления умным домом в соответствии с указанными ниже требованиями.

Перечень разделов ВКР:

1. Анализ задания на выпускную квалификационную работу.
2. Проектирование и реализация системы.
3. Инструкции.

Перечень графического материала: презентация в формате PDF, видеодемонстрация работы разработанного приложения.

1 Задание

Разработать веб-приложение для управления умным домом.

2 Основные требования

Приложение должно быть кроссплатформенным для настольных и мобильных систем (ОС Windows, Linux, MacOS, Android, iOS). Также иметь графический пользовательский интерфейс, открытый исходный код и обладать следующими возможностями:

а) предоставлять графический интерфейс для управления умными устройствами;

б) поддержка многопользовательской работы;

в) назначение пользователям прав доступа.

Руководитель ВКР _____ К. В. Пушкарев
подпись

Задание принял к исполнению _____ А. С. Васильев
подпись

31 октября 2022 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Веб-приложение для управления умным домом» содержит 45 страниц текстового документа, 20 иллюстраций, 5 таблиц, 26 использованный источник, 1 приложение.

ВЕБ-ПРИЛОЖЕНИЕ, MQTT, JAVASCRIPT, REACT, NODE.JS, POSTGRESQL, УМНЫЙ ДОМ.

Цель работы: разработать веб-приложение для управления умным домом.

Для достижения цели в работе решаются следующие задачи:

- анализ задания на выпускную квалификационную работу;
- проектирование;
- реализация приложения.

В результате анализа задания на выпускную квалификационную работу были рассмотрены существующие решения, также был проведён обзор инструментов. Была составлена диаграмма прецедентов, построена модель базы данных. Также были составлены диаграммы классов.

В итоге была спроектировано и реализовано приложение для управления умным домом, которое обладает следующими основными возможностями:

- управление умными устройствами;
- поддержка многопользовательской работы;
- группировка и поиск по списку устройств;
- назначение пользователям прав доступа;
- сбор и отображение статистических данных по работе отдельного устройства.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Все больше людей становятся пользователями умного дома в различных его проявлениях. Сегодня на рынке мы можем найти большое количество модулей умного дома от разных производителей, работающих с помощью разных протоколов. Появляется потребность в платформе для управления умным домом, работающем на едином протоколе с устройствами от разных производителей.

Цель работы: разработать веб-приложение для управления умным домом.

Клиентская часть приложения должна быть кроссплатформенной для настольных систем (ОС Windows, Linux, MacOS). Серверная часть приложения должна работать на ОС Linux. Приложение должно иметь открытый исходный код и обладать следующими возможностями:

- предоставлять графический интерфейс для управления умными устройствами;
- поддержка многопользовательской работы;
- группировка и поиск по списку устройств;
- назначение пользователям прав доступа;
- сбор и отображение статистических данных по работе отдельного устройства.

Для достижения цели в работе решаются следующие задачи:

- анализ задания на выпускную квалификационную работу;
- проектирование;
- реализация приложения.

1 Анализ задания на выпускную квалификационную работу

Необходимо разработать веб-приложение для управления модулями умного дома с возможностью мониторинга показаний подключенных устройств. Для обеспечения конкурентоспособности приложения, выполнен анализ характеристик аналогичных приложений.

1.1 Анализ существующих аналогов

На рынке уже существует большое количество приложений для решения поставленной задачи. На сегодняшний день существует множество протоколов, специализирующихся на интернете вещей.

Прямыми аналогами разрабатываемого ПО являются Дом с Алисой, Xiaomi Mi Home, Умный дом Sber. Перечисленные сервисы имеют широкий функционал в виде управления голосом, подключения модулей разного назначения, создания сложных сценариев, системы мониторинга и т.д. В таблице 1 приведены основные сведения о существующих сервисах.

Таблица 1 – Основные сведения о существующих сервисах

Название	**Кроссплат-форменность**	**Возможность подключения устройств сторонних производителей**	**Возможность подключения DIY (англ. do it yourself)-устройств**	**Возможность работы без интернет-подключения**
Умный дом с Алисой	iOS, Android	+	-	-
Xiaomi MiHome	iOS, Android	-	-	При покупке отдельного хаба
Умный дом Sber	Android	+	+	-

1.1.1 Умный дом с Алисой

Сервис разработан компанией Яндекс [1]. Центром умного дома Яндекса является голосовой помощник Алиса. Отдать команду можно с помощью умной колонки, именуемой Яндекс Станция [2], или приложения, в котором интегрирован голосовой помощник. Яндекс Станция имеет несколько способов подключения: Wi-Fi IEEE 802.11b/g/n/ac (2,4 и 5 ГГц), Bluetooth 4.1/BLE, HDMI 1.4. Для бесперебойной работы устройства требуется интернет-подключение со скоростью не менее 2 Мбит/с. Данная система поддерживает как устройства от Яндекса, так и от сторонних производителей. В перечень возможностей также входит настройка сценариев для одновременной работы нескольких устройств, расписания для автономной работы устройств.

Главным недостатком является потребность умной системы в постоянном доступе к интернету. Но, эта проблема была частично решена, начиная со второй версии Яндекс Станции [3]. Яндекс Станция второго поколения поддерживает Bluetooth 5.0 и протокол передачи данных ZigBee, что позволяет подключать умные устройства, работающие по данному протоколу. Но для полноценной работы все еще необходимо бесперебойное интернет-подключение со скоростью от 2 Мбит/с. Помимо этого, для работы в локальном режиме модули умного дома должны поддерживать протокол ZigBee.

1.1.2 Xiaomi MiHome

Система от китайской компании Xiaomi является одной из самых популярных ввиду большого количества умных устройств разного назначения с широким ценовым диапазоном. Центром управления является приложение MiHome [4].

Каждый модуль умного дома продается отдельно, что делает систему умного дома легко расширяемой. Но некоторые устройства имеют разные протоколы передачи данных. Из-за чего при установке более масштабной системы можно столкнуться с проблемой настройки сценариев для совместной работы устройств с разными интерфейсами подключения. Например, датчик

Xiaomi Mi Window and Door Sensor 2 использует для связи Bluetooth, а умная лампа Xiaomi Mi Smart LED Bulb Essential White and Color использует для этих целей Wi-Fi. Пользователь может использовать эти устройства по отдельности, но настроить сценарий на автоматическое включение лампочки при открытии двери нет [5, разд. «Зачем и кому нужен шлюз?».]. Также для работы требуется подключение к интернету. Так как управление происходит путем обмена сообщениями с удалёнными серверами производителя, скорость реакции устройств ниже, чем у локальных систем умного дома.

Для решения этих проблем компанией Xiaomi был выпущен специальный блок управления Xiaomi Gateway [6] для объединения всех устройств в одну систему, который продается отдельно. Данное устройство поддерживает подключение через Wi-Fi IEEE 802.11b/g/n/ac (2,4 и 5 ГГц), Bluetooth 5.0.

Главным недостатком системы умного дома от Xiaomi является невозможность подключения устройств сторонних производителей. В приложении MiHome в каталоге поддерживаемых устройств для подключения пользователь просто не увидит устройства не из экосистемы Xiaomi [7]. Все устройства обмениваются данными по собственному протоколу, который называется miIO [8]. Данный протокол предназначен для обмена данными в пределах локальной сети между устройствами экосистемы Xiaomi и приложением MiHome и является проприетарным. И хотя ряд крупных производителей систем умного дома поддерживают устройства Xiaomi, для большей части производителей они все еще не доступны.

1.1.3 Умный дом Sber

Производителем данной системы является российская компания Сбербанк [9]. Умный дом от Сбербанка предлагает широкий спектр возможностей, включая управление техникой удаленно с помощью голосового ассистента или через интерфейс, возможность группового управления устройствами. Имеется поддержка как устройств от Сбербанка, так и от ряда сторонних производителей [10].

Отличительной особенностью является официальная поддержка DIY-устройств, благодаря сервису MQTT-to-Cloud. Компания Сбербанк предоставляет возможность пользователям подключать свои самодельные устройства через протокол MQTT, или Message Queue Telemetry Transport, к сервисам умного дома Sber. Разработчикам рекомендуется использовать официальную документацию [11].

Управление происходит либо через приложение Салют, либо через приставку SberBox. Подключается приставка через Wi-Fi IEEE 802.11b/g/n/ac (2,4 и 5 ГГц) и Bluetooth 5.0 [12].

В связи с проблемами с иностранными сервисами, на декабрь 2022 года устройства компании не продаются в розничных магазинах. Но производитель обещает, что вскоре они снова станут доступны [13].

1.2 Выбор инструментов

1.2.1 Выбор платформы

В качестве протокола передачи данных был выбран протокол MQTT [14]. Данный протокол специально создан для интернета вещей и имеет большую популярность в этой отрасли. Среди преимуществ можно выделить стабильную передачу данных, масштабируемость, передача данных группе вещей. Главным критерием при выборе является его совместимость с другими протоколами. Большинство популярных протоколов, таких как ZigBee или LoRa, окончательно преобразуются в протокол MQTT [15].

Особенностью работы выбранного протокола является наличие брокера — сервера, который выступает в качестве посредника при передаче данных между пользователем и модулями умного дома. В вопросе выбора платформы для работы приложения локальный сервер является оптимальным решением, так как предоставляет возможность работы умного дома без подключения к интернету, в отличие от использования удалённых сервисов.

В качестве локального сервера выступит одноплатный компьютер, как устройство с достаточной для обмена сообщениями между модулем и панелью управления производительностью и компактностью для удобного размещения такого сервера в любом жилом помещении. Для выбора подходящей платформы приведена таблица 2 с характеристиками популярных одноплатных компьютеров [16].

Таблица 2 – Характеристики одноплатных компьютеров

Название	**Процессор**	**Оперативная память**	**Сеть**	**Bluetooth**
Onion omega 2+	MT7688 с тактовой частотой 568 МГц	64 МБ	WiFi 802.11 2.4 ГГц b/g/n	Отсутствует
Raspberry Pi 4	ARM Cortex-A72 с тактовой частотой 1,5 ГГц	4 ГБ LPDDR4 SDRAM	WiFi 802.11 2.4ГГц/5 ГГц b/g/n/ac, до 1000 Мбит RJ45 Ethernet	Bluetooth 5.0
Orange Pi Zero	Quad core Cortex A7 с тактовой частотой 1.2 ГГц	512 МБ	WiFi 802.11 2.4ГГц b/g/n, до 100 Мбит RJ45 Ethernet	Отсутствует
NVIDIA Jetson Nano Developer Kit	ARM Cortex A57 с тактовой частотой 1.43 ГГц	4 ГБ LPDDR4	до 1000 Мбит RJ45 Ethernet	Отсутствует

Окончание таблицы 2

Название	Процессор	Оперативная память	Сеть	Bluetooth
ASUS Tinker Board S	Rockchip Quad-Core RK3288 с тактовой частотой 1.8 ГГц	2 ГБ LPDDR3	WiFi 2.4GHz 802.11b/g/n, до 1000 Мбит RJ45 Ethernet	Bluetooth 4.0

Для подключения большего количества устройств одноплатный компьютер должен обладать наибольшим количеством беспроводных интерфейсов подключения и при этом оптимальной производительностью. Среди перечисленных плат в таблице 2 только Raspberry Pi 4 и ASUS Tinker Board S обладают как WiFi, так и Bluetooth модулями. Плата от ASUS имеет более производительный процессор, но Raspberry работает с новейшей версией Bluetooth 5.0 и поддерживает работу WiFi в диапазоне 5 ГГц, что для задачи подключения большего количества умных устройств является более важными характеристиками.

Таким образом, в качестве платформы для работы веб-приложения была выбрана Raspberry Pi 4.

1.2.2 Выбор языка программирования

При выборе учитывались ориентированность языка на поставленную задачу и наличие уже готовых инструментов для ее решения. Исходя из поставленной задачи, был выбран язык JavaScript. Наличие фреймворков как для фронтенда (Vue.js, React), так и для бэкенда (Node.js, Nest.js, Express) делает возможным создать быстрый сервер и современный пользовательский интерфейс. Помимо этого, JavaScript является популярным языком программирования с большим сообществом, что делает доступным широкий спектр уже готовых инструментов и библиотек для решения разных задач [17].

1.2.3 Выбор библиотек для разработки

JavaScript [18] предлагает множество инструментов для создания веб-приложений, поэтому необходим сравнительный анализ для окончательного выбора средств для разработки.

Node.js [19] – данный фреймворк позволяет писать серверный код для веб-приложений и веб-страниц. Это основной инструмент, который позволяет превратить JavaScript из узкоспециализированного языка в язык общего назначения. Это делает Node.js обязательной частью полноценного веб-приложения.

Express [20] – популярный фреймворк, работающий внутри среды исполнения Node.js, позволяющий реализовывать ряд функций, необходимых для создания эффективного веб-приложения. По сравнению с другими фреймворками для бэкенд разработки (Nest.js, Next.js и др.) обладает высокой поддержкой сообщества и большим количеством обучающего материала [21]. Express необходим для быстрой разработки с использованием Node.js и де-факто является его стандартным инструментом.

Для фронтенд разработки существуют два самых популярных фреймворка – React [22] и Vue [23]. React обладает рядом преимуществ: поддержка большими компаниями и разработчиками, масштабируемость, большое количество уже готовых решений и библиотек. Vue не обладает такой поддержкой, так как был начат отдельным разработчиком и долгое время не был известен [24]. Хотя к 2021 году его популярность сильно возросла [25]. По сравнению с React, Vue имеет отличную документацию и простоту использования. В итоге, Vue больше подходит для написания небольших приложений со стандартным функционалом, а React для написание сложных решений с большим выбором сторонних библиотек. Исходя из этого, для фронтенд части приложения был выбран React.

1.3 Диаграмма прецедентов

На рисунке 1 представлена диаграмма прецедентов.

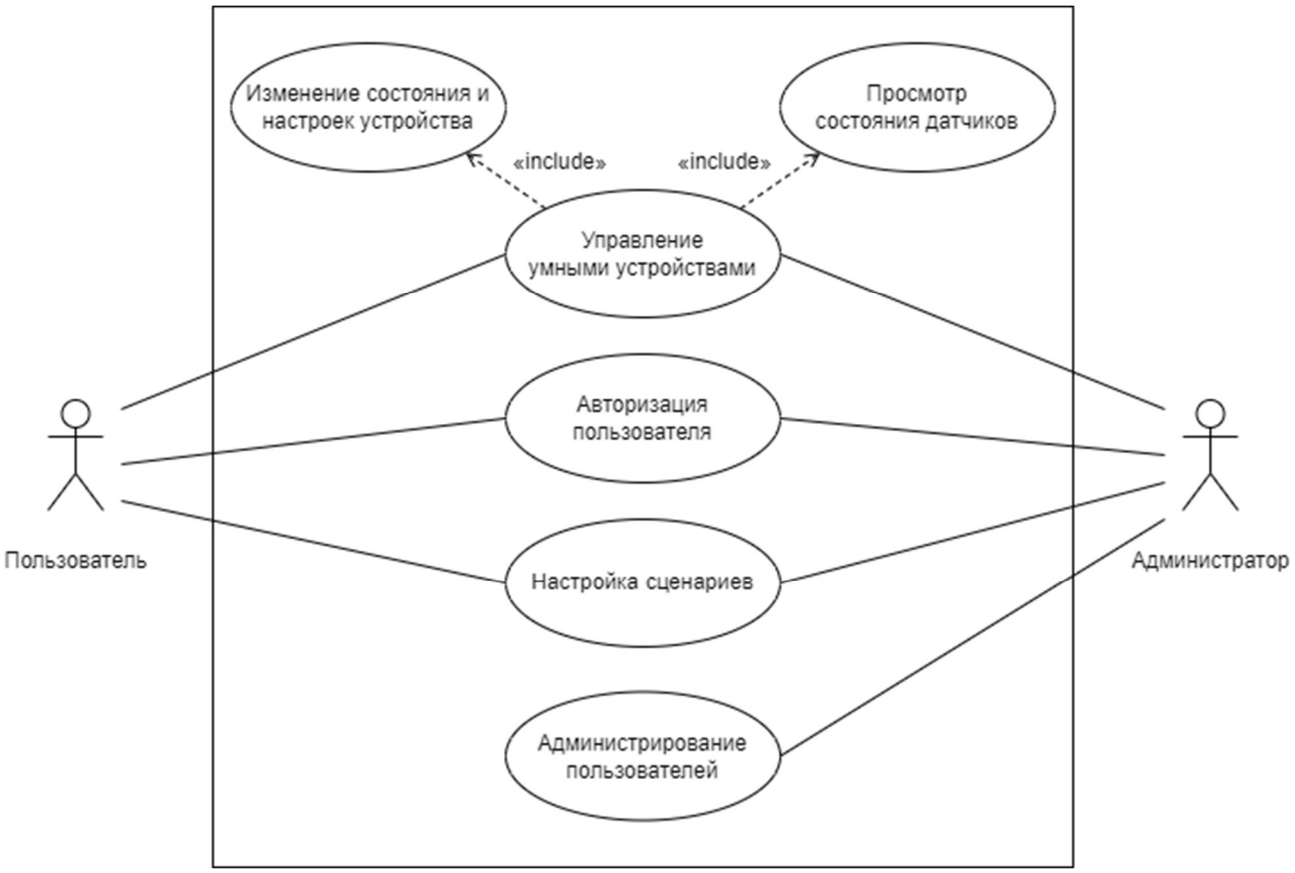


Рисунок 1 – Диаграмма прецедентов

Текстовое описание прецедентов приведено ниже.

Название: «Управление умными устройствами».

Предусловие: Пользователь прошел авторизацию и перешел на страницу «Мои устройства».

Основной сценарий:

А. Пользователь увидит данные всех доступных устройств: их состояния и собранную статистику.

Б. Пользователь выбирает нужное устройство и переключателем изменяет его состояние.

В. Пользователь открывает расширенные настройки выбранного устройства, изменяет их и нажимает кнопку «Сохранить».

Г. Программа говорит об успешной настройке устройства и возвращается на главный экран.

Условие ввода в действие альтернативных сценариев.

Условие 1. Пользователь не смог пройти авторизацию

А. Вывод сообщения об ошибке авторизации и возвращение пользователя на экран авторизации.

Условие 2. Программа не смогла подключиться к устройствам

А. Вывод сообщения о неудачной попытке подключения к устройствам.

Б. Программа автоматически выполняет попытки повторного подключения к устройствам до тех пор, пока лимит попыток повторного подключения не будет достигнут.

Условие 3. Пользователь нажимает кнопку «Назад» в окне с расширенными настройками устройства до нажатия кнопки «Сохранить».

А. Вывод сообщения о несохраненных данных с просьбой подтверждения выхода из настроек.

Название: «Аутентификация пользователя».

Предусловие: Пользователь переходит на страницу аутентификации.

Основной сценарий:

А. Пользователь вводит логин и пароль от своей учетной записи.

Б. Пользователь нажимает кнопку «Войти».

В. Пользователь автоматически переходит на страницу «Мои устройства».

Условие ввода в действие альтернативных сценариев.

Условие 1. Пользователь прошел авторизацию успешно, но не обладает правами администратора**.**

А. На странице «Мои устройства» в боковом меню пользователь не увидит пункта «Пользователи»**.**

Условие 2. Пользователь прошел авторизацию успешно и обладает правами администратора**.**

А. На странице «Мои устройства» в боковом меню пользователь увидит пункт «Пользователи» для открытия страницы с администрированием пользователей.

Условие 3. Пользователь не прошел авторизацию.

А. Пользователь остается на странице авторизации и видит ошибку, что введенные данные не верны.

Название: «Настройка сценариев».

Предусловие: Пользователь перешел на страницу сценариев.

Основной сценарий:

А. Пользователь переходит на страницу сценариев и видит список сценариев.

Б. Пользователь нажимает кнопку «Редактировать».

В. Пользователь переходит на страницу с настройками выбранного сценария. После внесения изменений нажимает кнопку «Сохранить».

Условие ввода в действие альтернативных сценариев.

Условие 1. Пользователю необходимо удалить сценарий.

А. Пользователь нажимает кнопку «Удалить» около выбранного сценария. Данный сценарий удаляется.

Условие 2. Пользователю необходимо создать сценарий.

А. Пользователь нажимает кнопку «Создать сценарий».

Б. В открывшемся окне пользователь выбирает устройство из списка доступных, выбирает, как должно измениться состояние устройства, и время этого изменения. После нажимает кнопку «Сохранить».

Название: «Администрирование пользователей».

Предусловие: Пользователь прошел авторизацию и перешел на страницу «Мои устройства».

Основной сценарий:

А. Пользователь переходит на страницу «Пользователи» и видит список зарегистрированных пользователей.

Б. Пользователь видит информацию о наличии прав администратора у каждого пользователя.

В. Пользователь нажимает кнопку «Удалить» около выбранного пользователя и программа удаляет пользователя из базы данных.

Г. Пользователь нажимает кнопку «Добавить пользователя» в верхней части страницы и видит модальное окно для добавления пользователя. После чего вводит необходимые данные и нажимает кнопку «Добавить».

1.4 Выводы по главе

В этой главе были освещены недостатки и преимущества аналогов крупных компаний. Не было найдено системы умного дома, которая бы удовлетворяла всем характеристикам, перечисленным в таблице 1.

В итоге, были сформулированы точные требования к разрабатываемому проекту, а именно:

- кроссплатформенность клиентского приложения, т. е. работа как на мобильных платформах, так и на настольных;

- возможность подключения устройств из разных экосистем с последующей настройкой сценария выполнения;

- возможность работы с DIY-устройствами по открытому протоколу MQTT.

В качестве инструмента разработки были выбраны следующие фреймворки: Node.js, Express – для создания серверной части, React – для создания пользовательского интерфейса.

2 Проектирование и реализация приложения

2.1 Алгоритм работы MQTT протокола

Главный принцип работы MQTT протокола в подписке и публикации сообщений на отдельные темы. Устройство подписывается на тему и получает сообщения, которые опубликованы на эту тему. Тем самым мы можем организовать в сети умного дома строгую иерархию, по которой сможет работать большое количество устройств. Алгоритм работы MQTT протокола состоит в следующем. Имеется сервер, который выступает в роли посредника между клиентом и конечным устройством, называемый брокером. Все устройства системы подключаются к брокеру, после чего он организовывает доставку сообщений до получателя. Пример обмена информацией в такой системе представлен на рисунке 2.

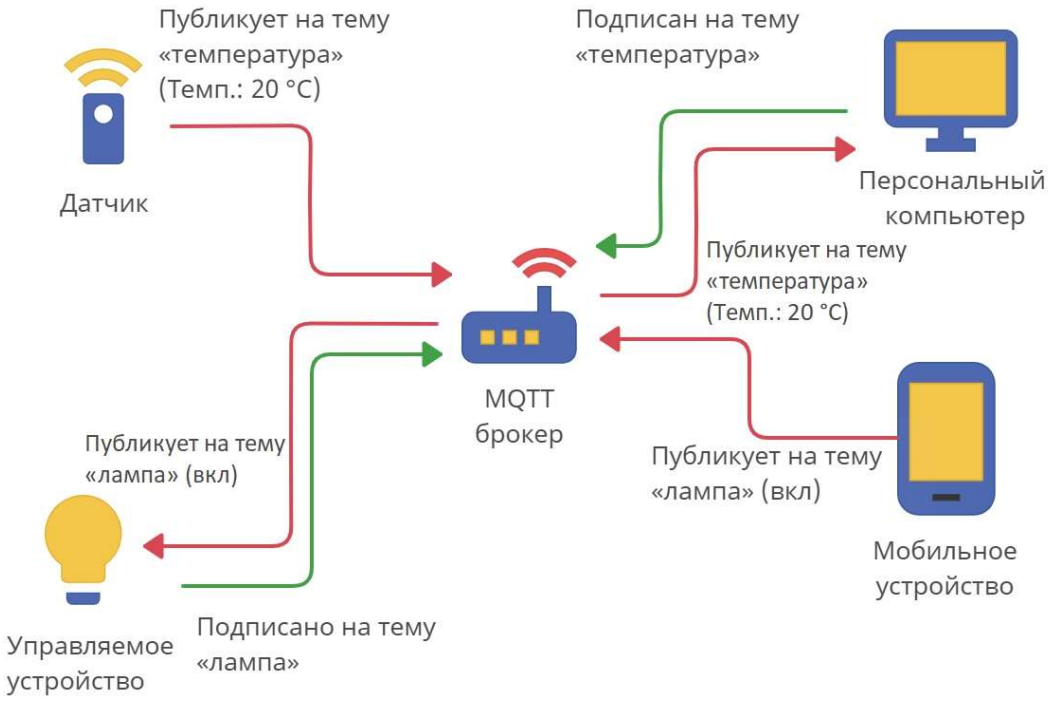


Рисунок 2 – Схема обмена информацией

После подключения устройство, или подписчик, отправляет брокеру темы, на которые оно хочет получать сообщения. Как только любое другое устройство,

или публикатор, отправит сообщение на эту тему, оно будет доставлено подписчику. После чего последний перейдет к выполнению полученных инструкций.

2.2 Архитектура веб-приложения

На рисунке 3 представлена схема архитектуры веб-приложения. Веб-приложение работает на удаленном сервере. Через клиента пользователь попадает в приложение, где уже может взаимодействовать с данными, хранящимися в удаленной базе PostgreSQL на сервере, и с умными устройствами через MQTT-брокера. К MQTT-брокеру подключены все умные устройства.

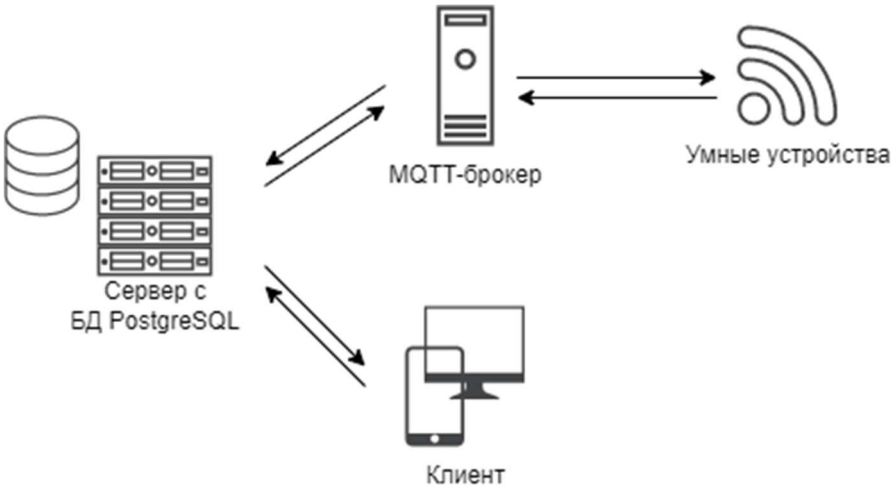


Рисунок 3 – Архитектура веб-приложения

На сервере находится реализация серверного компонента приложения и клиентского.

Серверный компонент имеет доступ к данным базы данных и принимает запросы на работу с ними от клиента. Также здесь хранится конфиденциальная информация – токены доступа пользователей.

Клиентский компонент выполняет отрисовку пользовательского интерфейса и отслеживает взаимодействие пользователя с ним. Также клиент передает полученные из БД данные нужному элементу интерфейса и управляет его жизненным циклом (период между созданием и удалением элемента).

2.3 Модель базы данных

После анализа задания было выделено 6 основных сущности. Центральной из них является сущность card (карточка устройства). Пользователи взаимодействуют с устройствами при помощи карточек. Каждая карточка привязана к отдельному пользователю (сущность user) и доступна только для него. Каждая карточка привязана к отдельному устройству (сущность device) для управления параметрами этого устройства и отображения актуальной информации о нем. Каждой карточке отведена отдельная комната (сущность room), в которой находится физическое устройство. У пользователя может быть множество карточек или ни одной. У устройства и комнаты также может быть множество карточек или ни одной.

Каждому сценарию (сущность scenario) соответствует устройство, которое будет выполнять его. У устройства может быть несколько сценариев или ни одного. Каждый пользователь имеет брокер (сущность broker) для связи с устройствами. К одному брокеру может быть подключено ни одного или несколько пользователей.

На рисунке 4 представлена логическая модель базы данных.

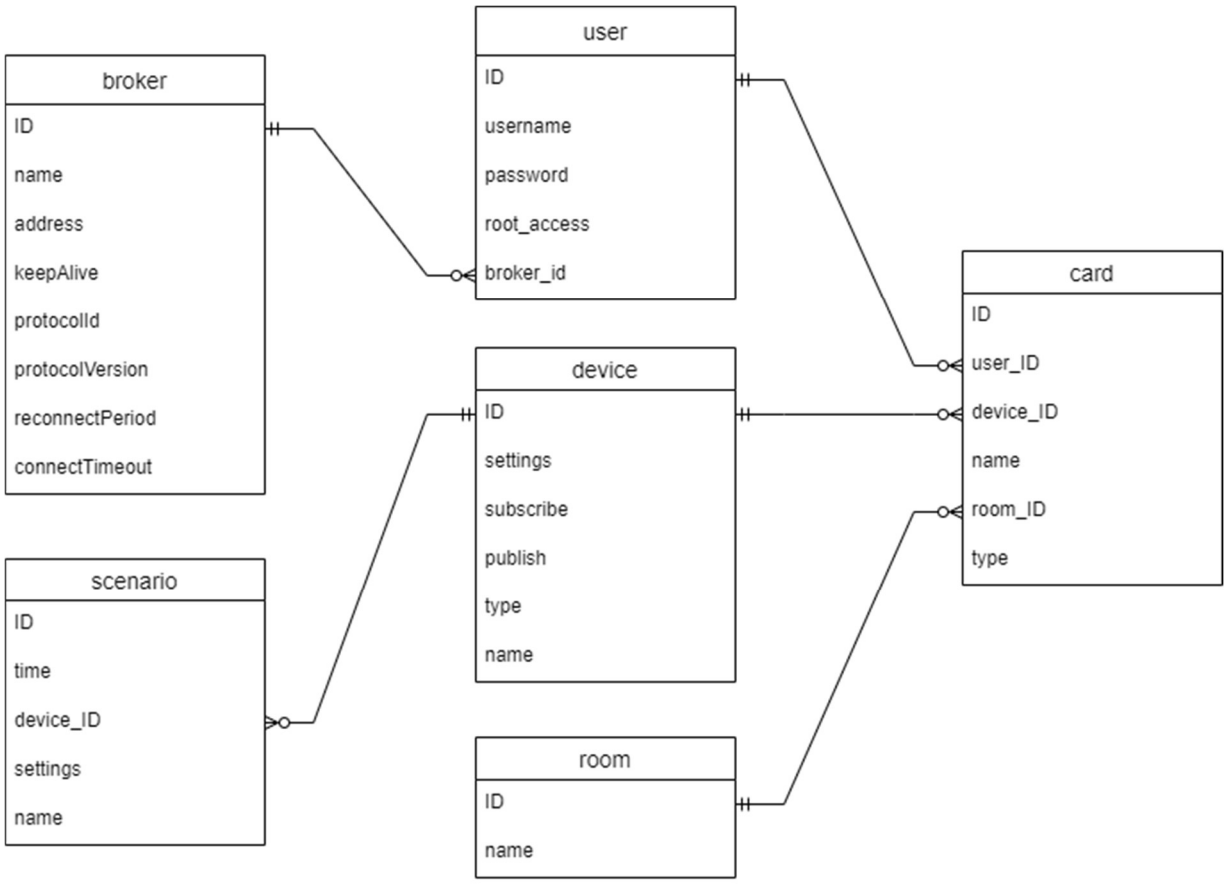


Рисунок 4 – Логическая модель базы данных

В таблице 3 представлено описание сущностей.

Для хранения пользователей отведена таблица user. Каждый пользователь должен иметь имя пользователя и пароль для входа в свой аккаунт. Также имеется поле root_access, которое говорит о наличии или отсутствии прав администратора у пользователя. В поле password для защиты хранится сформированный хэш пароля. В качестве хэш-функции используется BCrypt.

Для хранения всех подключенных устройств отведена таблица device, в которой хранятся настройки работы устройства, тема для подписки и тема для публикации, тип устройства и его наименование.

Для отображения устройства на экране пользователя используется сущность card. Здесь хранится информация о номере устройства, которое должно отображаться на карточке, информация о пользователе, создавшем эту карточку, название, информация о номере комнаты, в которой находится устройство, и тип карточки.

Для хранения информации для подключения к брокеру используется сущность broker. Здесь хранится информация о имени брокера для удобства пользователя, его адрес в сети, времени передачи данных при разрыве соединения, наименование протокола, версии протокола, интервал повторного подключения, время ожидания перед получением ответа.

Для хранения сценариев отведена таблица Scenario. Здесь хранится номер сценария, его наименование, время выполнения сценария, номер исполняющего устройства и настройки с состоянием, в которое должно будет перейти устройство.

Таблица 3 - Описание сущностей

Сущность	**Поля и их назначение**	**Тип данных поля**
card	ID – идентификатор	integer
	user_ID – идентификатор связанного пользователя	integer
	device_ID – идентификатор связанного устройства	integer
	name – наименование	text
	room_ID – идентификатор связанной комнаты	integer
	type – тип отображения карточки	text
user	ID – идентификатор	integer
	username – логин пользователя для входа в учетную запись	text
	password – хэш пароля	text
	root_access – наличие прав администратора	boolean

Продолжение таблицы 3

Сущность	Поля и их назначение	Тип данных поля
device	ID – идентификатор	integer
	settings – настройки устройства	text
	subscribe – тема подписи на сообщения	text
	publish – тема публикаций сообщений	text
	type – тип устройства по назначению	text
	name – наименование	text
room	ID – идентификатор	integer
	name – наименование	text
broker	ID – идентификатор	integer
	name – наименование	text
	address – адрес брокера в сети	text
	keepAlive – время передачи данных при разрыве соединения	integer
	protocolId – наименование протокола	text
	protocolVersion – версии протокола	text
	reconnectPeriod – интервал повторного подключения	integer
	connectTimeout – время ожидания перед получением ответа	integer
scenario	ID – идентификатор	integer
	time – время для выполнения сценария	text
	device_ID – идентификатор связанного устройства	integer
	settings – настройки устройства	text
	name – наименование	text

2.4 Описание классов

Для работы с базой данных (БД) используется Sequelize ORM (Object-Relational Mapping). Для работы сервера используется Node.js и фреймворк Express.js. Серверная часть сведена к описанию структуры БД, созданию «контроллеров» для запросов к каждой из сущностей БД и настройке «маршрутов», обращаясь к которым клиентская часть будет взаимодействовать с нужным «контроллером».

На рисунке 5 представлена диаграмма классов для работы с базой данных.

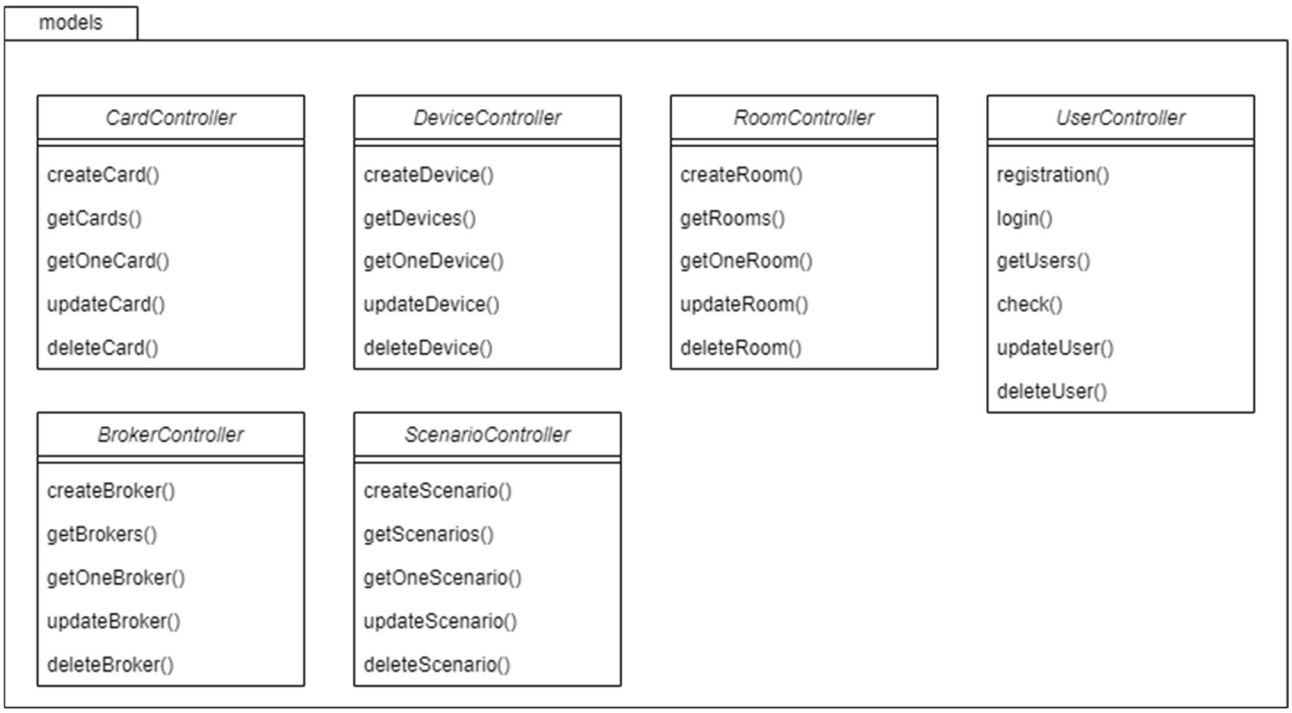


Рисунок 5 – Диаграмма классов сервера

В модуле models создаются атрибуты с сущностью каждой таблицы: User, Device, Card, Room, Broker, Scenario. Здесь описаны поля каждой из сущностей: название, тип данных и прочее.

В классах CardController, DeviceController, RoomController, UserController, ScenarioController, BrokerController реализованы методы для взаимодействия с БД. Методы каждого контроллера реализуют полный CRUD (Create, Read, Update, Delete) цикл, т.е. четыре основные операции базы данных.

Методы createCard(), createDevice(), createRoom(), createBroker(), createScenario() создают запрос с добавлением нового элемента

соответствующей таблицы. Метод принимает параметром объект со значениями всех полей нового элемента таблицы.

Методы getCards(), getUsers(), getDevices(), getRooms(), getScenarios(), getBrokers() создают запрос на получение всех элементов соответствующей таблицы. Возвращается массив со всеми объектами таблицы.

Методы getOneCard(), getOneDevice(), getOneRoom(), getOneScenario(), getOneBroker()создают запрос на получение одного элемента таблицы. Метод берет идентификатор из маршрута запроса и возвращают объект соответствующего класса с данным идентификатором.

Методы updateCard(), updateUser(), updateDevice(), updateRoom(), updateScenario(), updateBroker() создают запрос на обновление одного элемента таблицы. Метод берет идентификатор из маршрута запроса и находит объект соответствующего класса с данным идентификатором. Метод принимает параметром объект с новыми значениями всех полей элемента таблицы.

Методы deleteCard(), deleteUser(), deleteDevice(), deleteRoom(), deleteScenario(), deleteBroker() создают запрос на удаление одного элемента таблицы. Метод берет идентификатор из маршрута запроса и находит объект соответствующего класса с данным идентификатором, после чего этот объект удаляется из БД.

Так как операции с таблицей User подразумевает работу с JWT (JSON Web Tokens) для проверки авторизованности пользователя и BCrypt для защищенного хранения паролей, CRUD методы для таблицы User немного отличаются от всех остальных. Метод registration() необходим для создания нового пользователя. Метод принимает параметром объект с данными нового пользователя. После этого хэш-функция принимает пароль и формирует ключ. Производится запрос на создание нового элемента таблицы user с именем пользователя, хэш-ключом и наличием прав администратора.

В методе login() хэш-функция формирует хэш-ключ введенного пароля и сравнивает со значением в БД. Метод принимает параметром объект с введенными значениями при авторизации.

Метод check() нужен для проверки, авторизован ли сейчас кто-то. Если нет, то интерфейс пользователя не будет доступен, пока пользователь не авторизуется.

2.5 Графический интерфейс

В приложении А представлена диаграмма классов для реализации графического интерфейса.

Был реализован класс App – главный класс программы, который связывает графический интерфейс с сервером. В данном классе создается атрибут loading, который хранит логическое значение о наличие данных об авторизованном пользователе. Пока данные о проверке не пришли с сервера по этому значению запускается анимация загрузки страницы. Для установки данных об авторизованном пользователе в локальное хранилище присутствуют метод setUser(). И метод setIsAuth() устанавливает логическое значение авторизованности пользователя.

В зависимости от результата проверки авторизованности пользователь переадресуется на главную страницу или страницу авторизации. Для этого класс App связан с классами Home и Login соответственно.

Класс Login служит для отображения страницы авторизации. Атрибуты username и password нужны для хранения введенных имени пользователя и пароля. Метод signIn() из введенных данных передает запрос к серверу для авторизации.

Класс Home служит для отображения главной страницы приложения. Атрибуты cardVisible, roomVisible, deviceVisible хранят логические значения, которые указывают отображать или нет модальные окна для добавления карточки, комнаты или устройства. Атрибут searchedCards хранит массив карточек для отображения на странице, отфильтрованный согласно поиску карточек по названию на главной странице. Атрибут searchQuery содержит значение этой поисковой строки.

Классы Sidebar и Navbar нужны для отображения дополнительных элементов интерфейса главной страницы – бокового меню и верхнего заголовка страницы.

Классы CreateCard, CreateDevice, CreateRoom, CreateUser, CreateScenario отображают модальные окна для добавления соответствующего компонента. Атрибуты этих классов созданы для хранения введенных данных. Методы addCard(), addDevice(), addRoom(), signUp(), addScenario() – для их отправки в базу данных. Методы setCreateValue() и handleChange() в классах CreateCard и CreateScenario, метод handleClick() в классе CreateUser() нужны для отслеживания изменения состояния в полях для ввода.

На главной странице отображается графическое представление класса WidgetList. Класс WidgetList содержит атрибут token, в котором хранится токен пользователя из локального хранилища, созданный при авторизации. С помощью него проверяется принадлежность карточки авторизованному пользователю. Тем самым у каждого пользователя отображается только созданные им карточки. WidgetList содержит в себе вызов класса Widget.

Класс Widget нужен для графического отображения отдельной карточки. Для хранения данных об устройстве, к которому привязана карточка присутствует атрибут device. В атрибуте data хранятся данные, которые отображаются на карточке. Метод handleChange() предназначен для отслеживания действий пользователя на данной карточке и их последующей их обработки.

Класс Home связан с классами InfoCard, UsersPage, DevicePage, ScenarioPage и SettingsPage, которые реализуют отображение страницы с настройками карточки, со списком пользователей, со списком устройств, со списком сценариев и настройками брокера соответственно.

Класс InfoCard содержит атрибут deleteVisible, который хранит логическое значение, которое указывает отображать или нет модальное окно для удаления карточки. Атрибуты card, device, room необходимы для хранения данных о карточке, прикрепленного к этой карточке устройства и о комнате, в которой

находится устройство. Метод handleChange() отслеживает изменения введенных данных. Метод saveChanges() сохраняет измененные настройки карточки.

Класс UsersPage содержит атрибут addVisible, который хранит логическое значение, которое указывает отображать или нет модальное окно для добавления нового пользователя.

Методы deleteUser() и deleteDevice() в классах UsersPage и DevicePage удаляют выбранного пользователя или устройство.

Класс ScenarioPage содержит атрибуты addVisible, deleteVisible и методы setAddVisible(), setDeleteVisible() для добавления и удаления сценариев.

Класс SettingsPage содержит атрибут token для получения ID пользователя и атрибут fieldValue для хранения введенных значений. Методы setFieldValue(), handleChange() нужны для отслеживания введенных значений, а метод saveChanges() для сохранения изменений и отправки их в БД.

3 Инструкции

3.1 Инструкция пользователя

В начале работы пользователь открывает приложение. Если пользователь впервые заходит в него или из аккаунта пользователя выходили, то откроется страница аутентификации (рисунок 6).

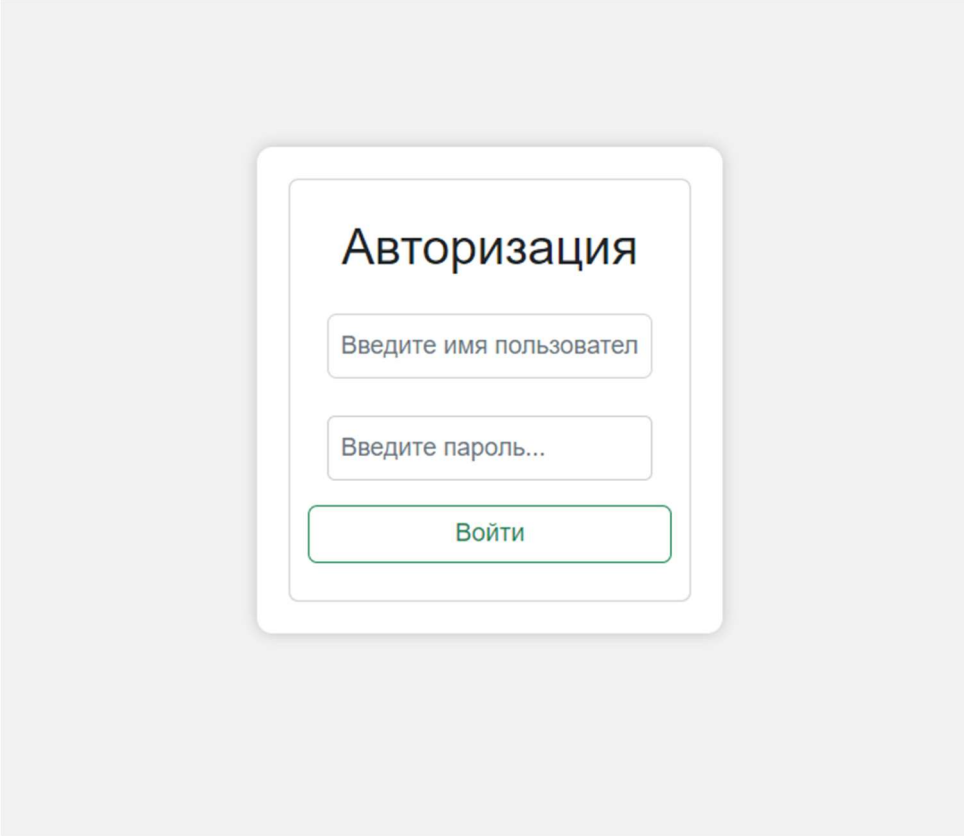


Рисунок 6 – Страница аутентификации

После чего пользователь вводит свои логин и пароль и нажимает кнопку «Войти». На главном экране пользователь увидит панель с карточками, где он может либо посмотреть собранные показания с датчиков, либо изменить базовые настройки устройства, например, выключить или включить устройство (рисунок 7). Если пользователю понадобятся расширенные настройки, то их можно изменить, нажав на кнопку «Настройки» на нужной карточке.

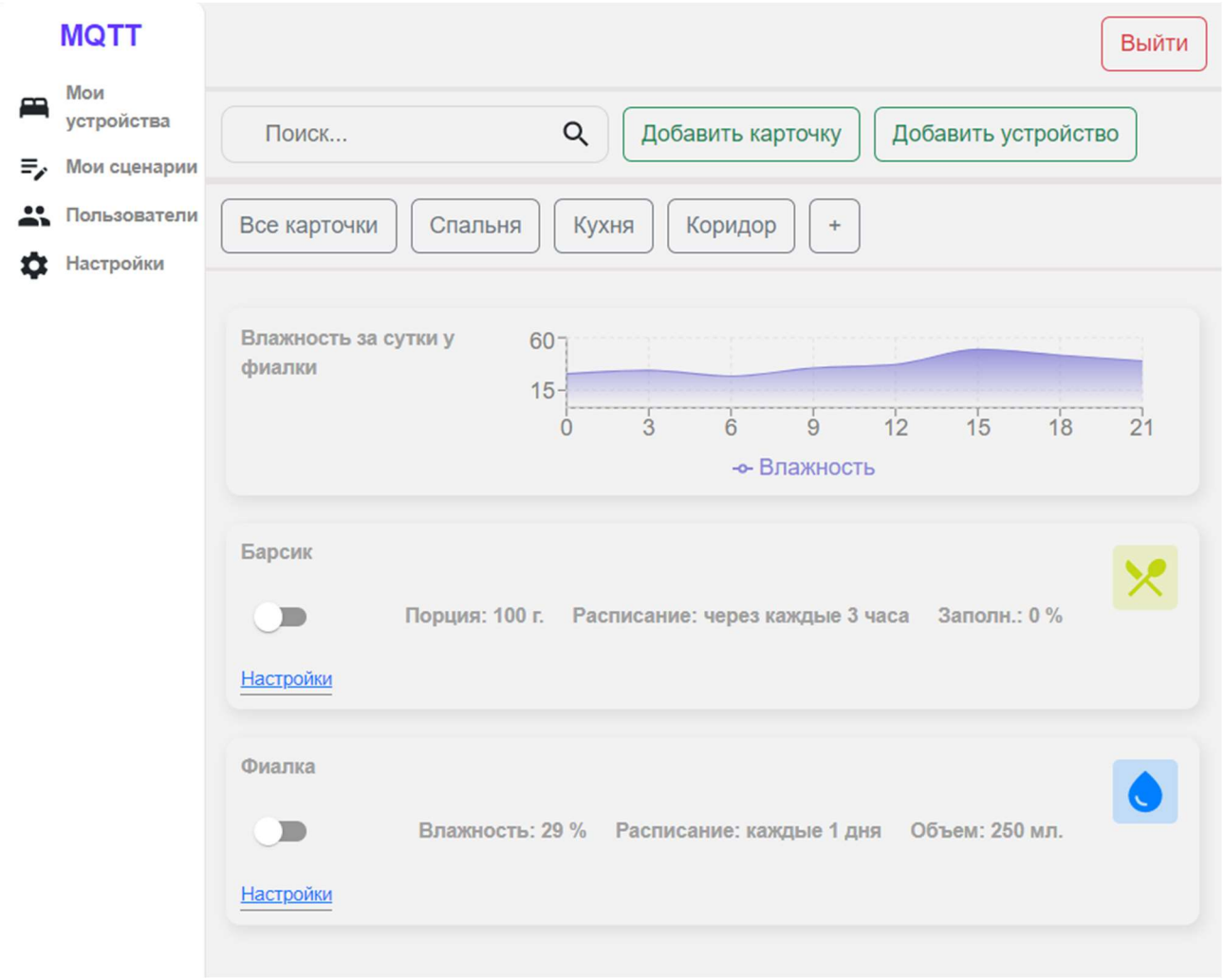


Рисунок 7 – Интерфейс пользователя

На странице с настройками можно увидеть подробную более информацию о карточке: название комнаты с устройством, название устройства. Также изменить все параметры для работы устройства (рисунок 8).

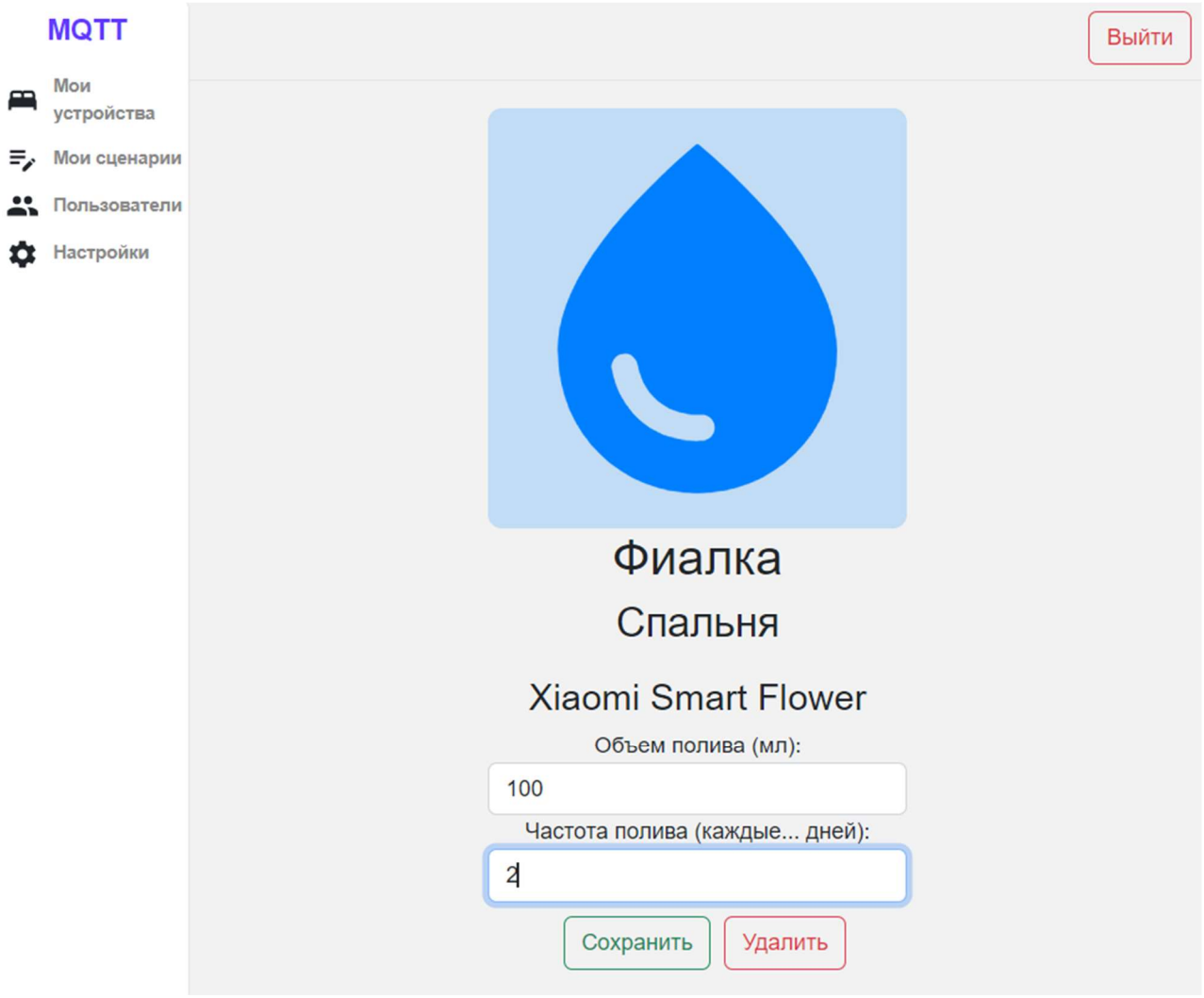


Рисунок 8 – Настройки карточки

На главной странице пользователь также может увидеть строку для поиска нужной карточки по названию, фильтр для отображения по комнатам и кнопки для добавления новых карточек, устройств и комнат (рисунок 9).

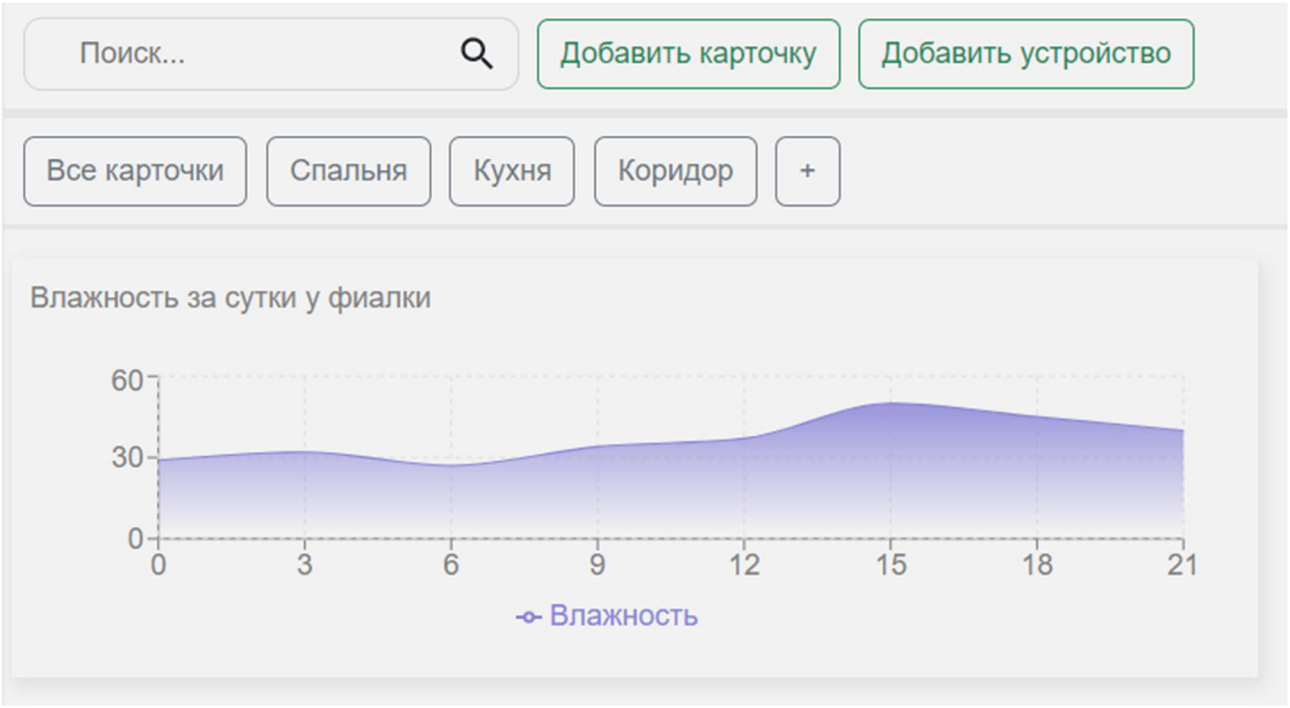


Рисунок 9 – Верхняя часть главной страницы

При создании новой карточки пользователь может задать название карточки, выбрать устройство и комнату, в которой находится устройство. В зависимости от выбранного устройства будут отображаться доступные параметры для первоначальной настройки (рисунок 10).

Для добавления нового устройства нужно нажать кнопку «Добавить устройство» на главной странице (рисунок 9). После нажатия откроется окно, в которое пользователь вводит название и тип устройства (рисунок 11).

Для добавления новой комнаты пользователю необходимо нажать кнопку «+» рядом с вкладками комнат (рисунок 9). После чего откроется окно, в котором пользователь может задать название новой комнаты (рисунок 12).

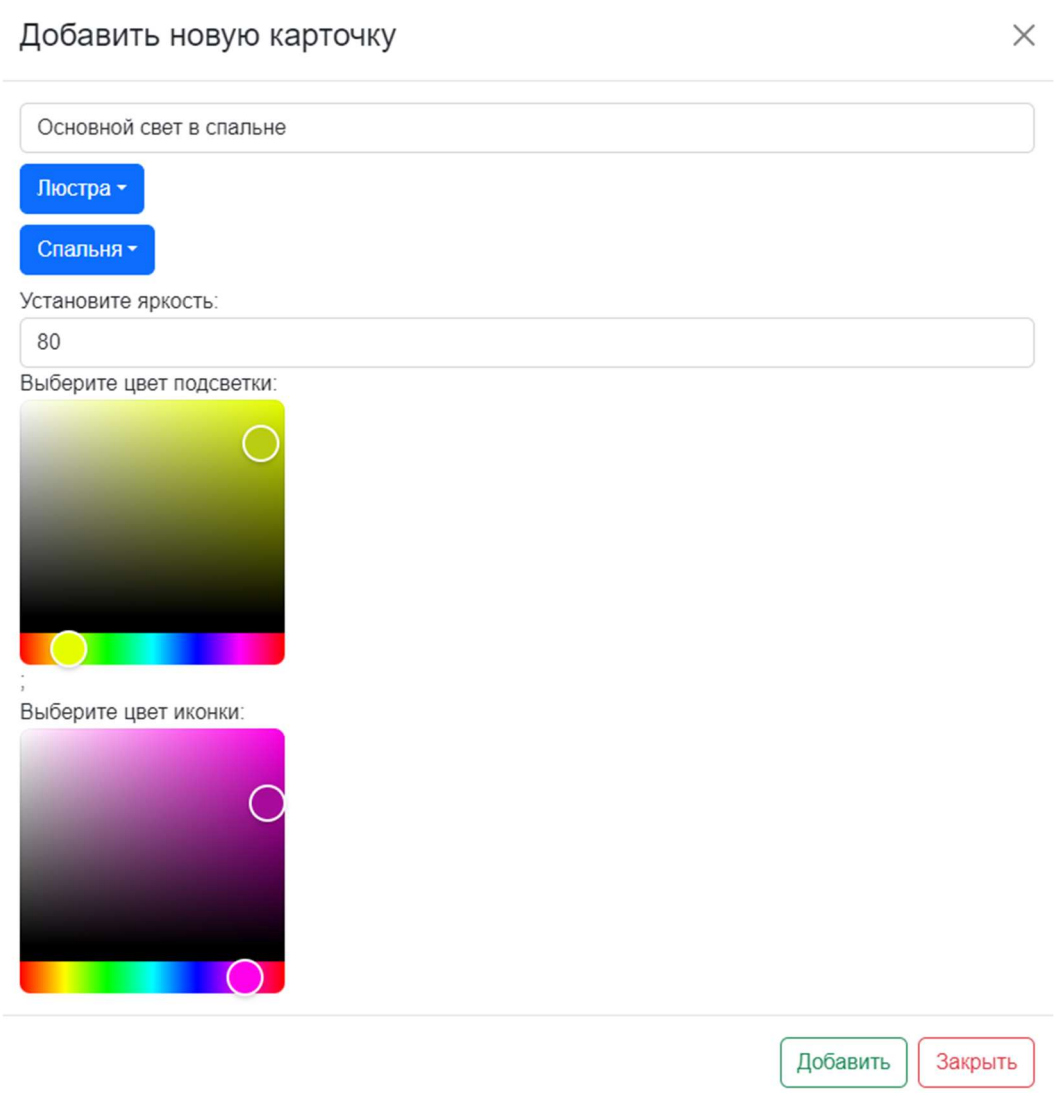


Рисунок 10 – Создание новой карточки

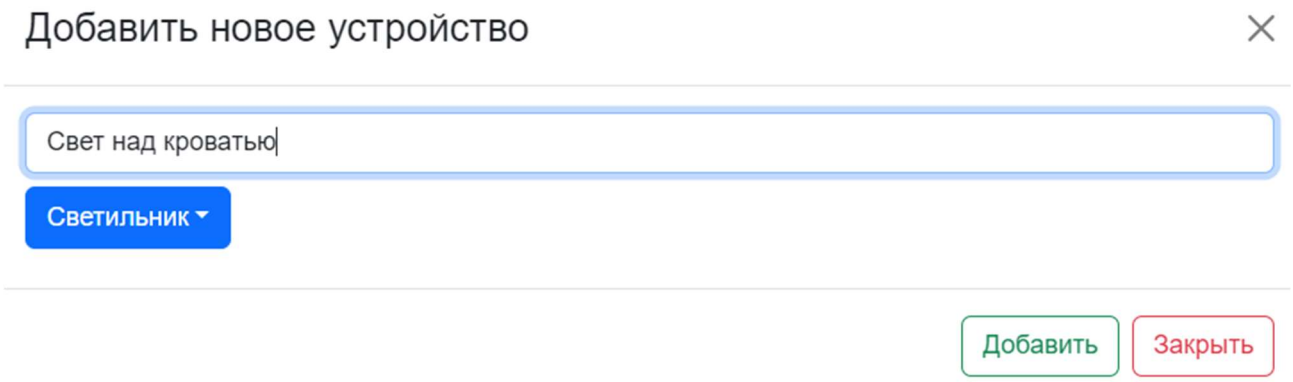


Рисунок 11 – Окно добавления нового устройства

Рисунок 12 – Окно добавления новой комнаты

Пользователь в боковом меню видит пункт «Мои сценарии», нажав на который пользователь переместится на страницу со списком сценариев (рисунок 13). Около каждого сценария есть кнопки для его редактирования и удаления.

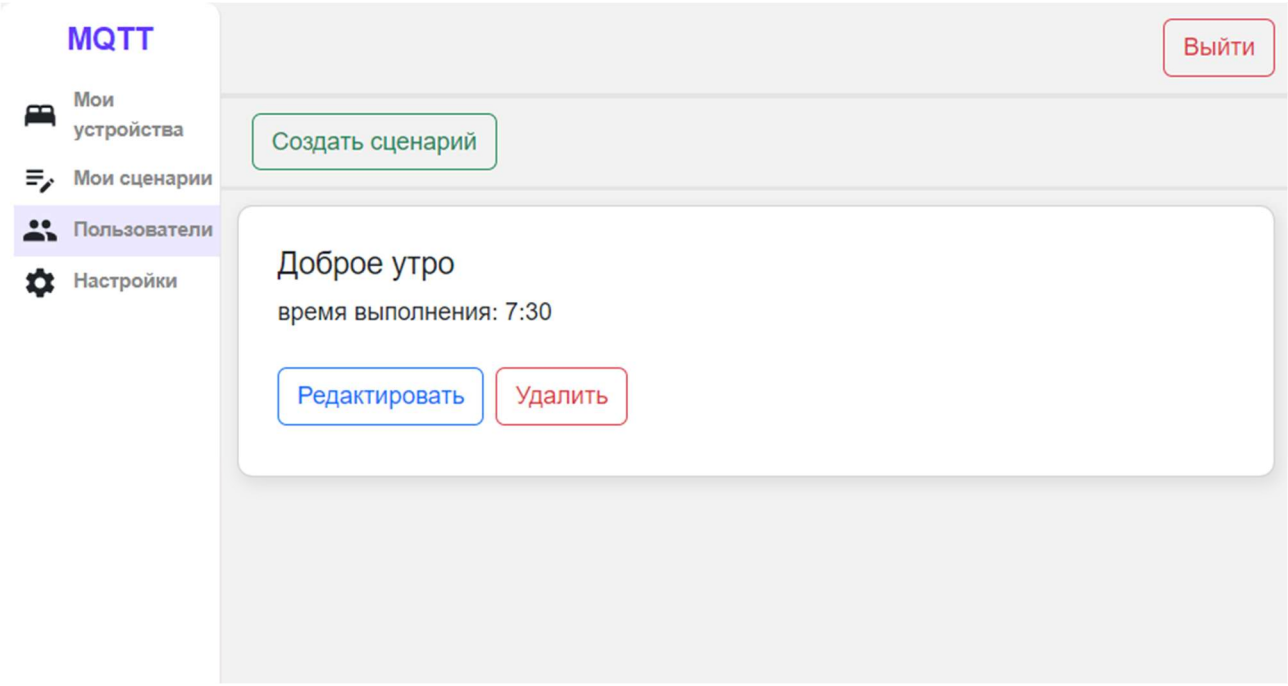


Рисунок 13 – Страница сценариев

После нажатия кнопки «Создать сценарий» открывается окно для создания нового сценария (рисунок 14). Здесь пользователь вводит название сценария, выбирает устройство, которое должно изменить свои параметры, время выполнения сценария и новые параметры этого устройства.

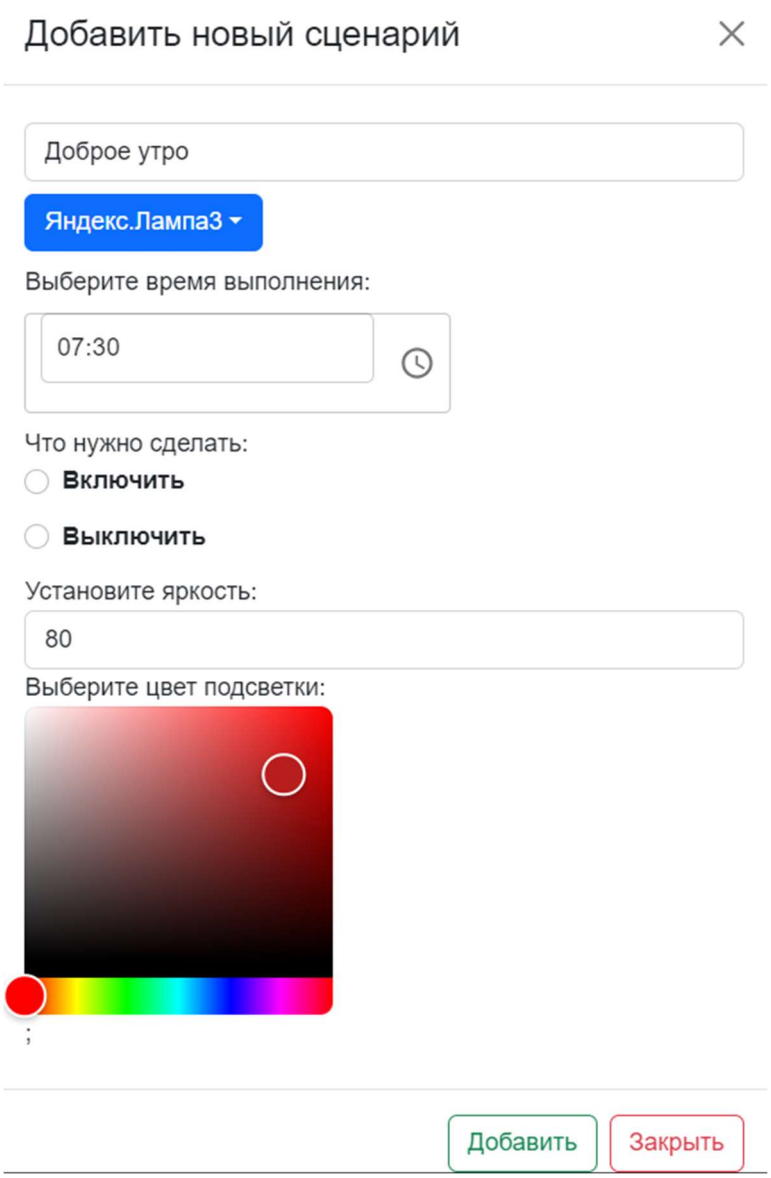


Рисунок 14 – Окно создания сценария

Если пользователь нажмет кнопку «Редактировать» около сценария, откроется страница редактирования (рисунок 15). Здесь пользователь может изменить существующие параметры сценария.

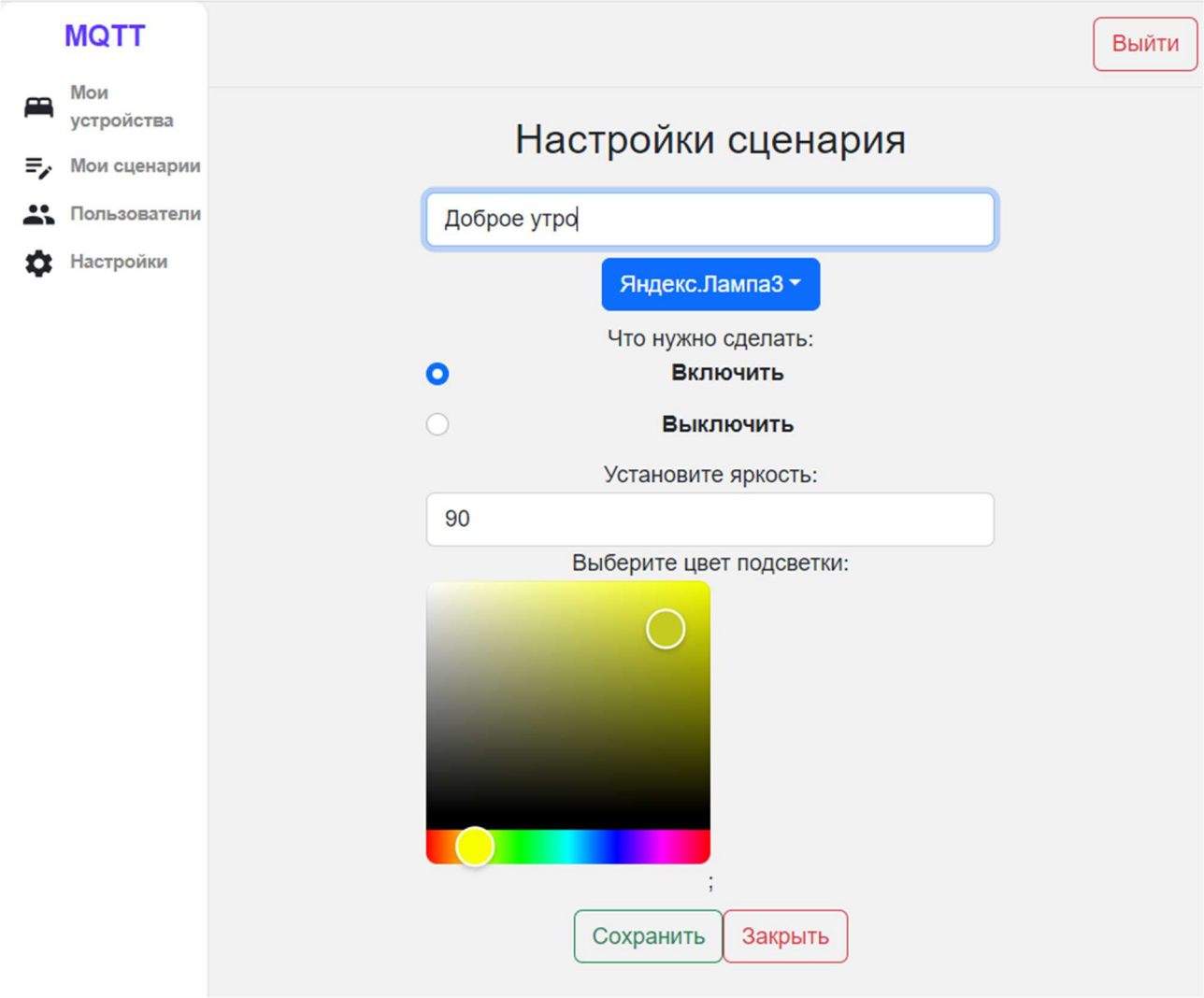


Рисунок 15 – Страница редактирования сценария

3.2 Инструкция администратора

Чтобы развернуть приложение, на сервере должен быть установлено следующее:

- Git;

- Node.js;

- NPM (Node Package Manager).

Клонируем репозиторий проекта с GitHub [26] в корневой каталог сервера с помощью команды git clone.

После этого командой npm install в каталоге server установим все необходимые модули. Теперь можно запустить сервер с помощью команды npm run start.

Для клиента командой npm install в каталоге client установим все необходимые модули. Теперь необходимо провести сборку приложения – вводим команду npm build. Можно запустить приложение командой npm start.

К приложению можно подключиться, введя в поисковую строку ip сервера и «:8080».

Если пользователь обладает правами администратора, то в боковом меню ему будут доступен пункт «Пользователи», нажав на него, администратору откроется страница со списком всех пользователей (рисунок 16). Здесь администратор может увидеть наличие прав администратора у других пользователей и удалить пользователей при необходимости. Вверху страницы отображается кнопка «Добавить пользователя», которое открывает соответствующее окно (рисунок 17).

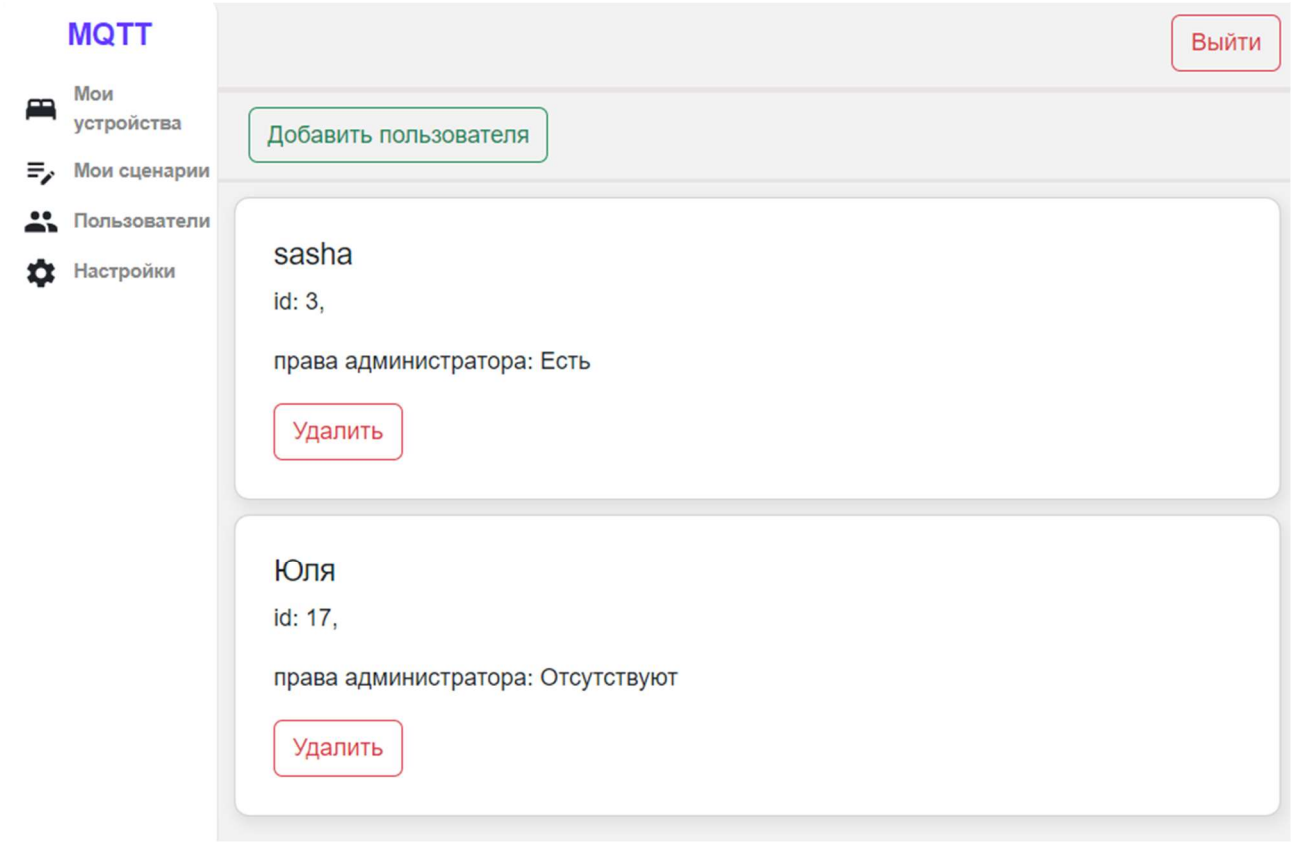


Рисунок 16 – Страница пользователей

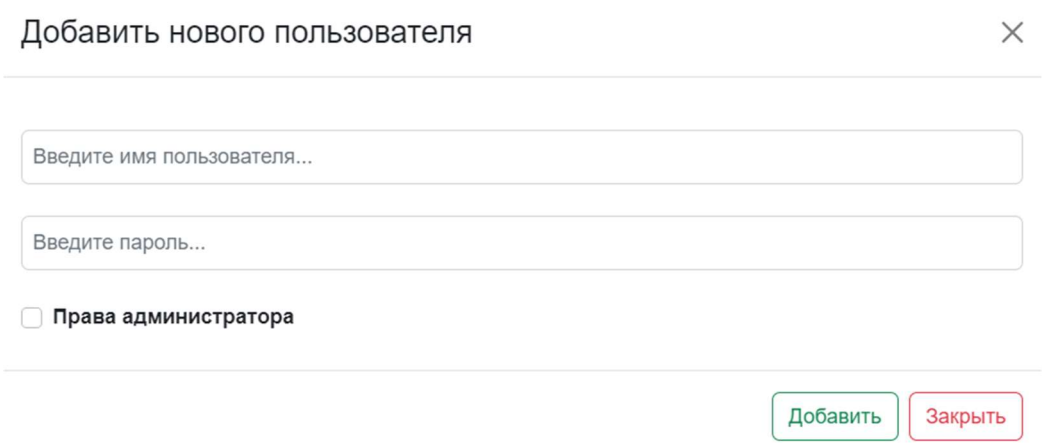


Рисунок 17 – Добавление нового пользователя

Также администратору доступен в боковом меню пункт «Настройки», нажав на который администратор переместится на страницу с настройками брокера. (рисунок 18)

На странице находятся следующие поля:

- имя брокера;
- адрес брокера для подключения;
- время передачи данных при разрыве соединения;
- наименование протокола;
- версия протокола;
- интервал повторного подключения;
- время ожидания перед получением ответа.

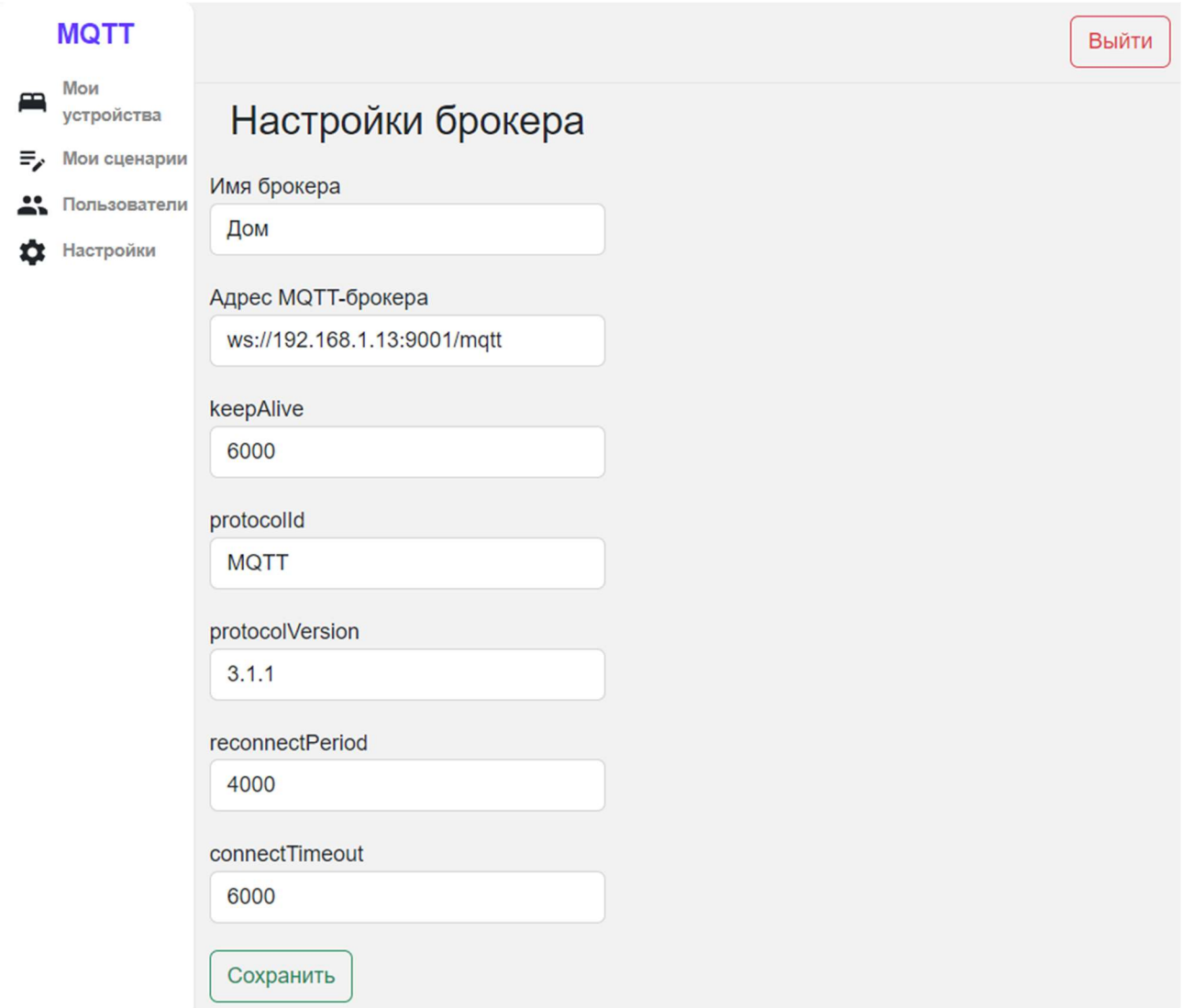


Рисунок 18 – страница с настройками брокера

3.3 Инструкция разработчика

Для работы с исходным кодом необходимо:

- IDE Webstorm 2022;
- фреймворк ReactJS версия не ниже 18.2.0;
- фреймворк ExpressJS версия не ниже 4.18.2.

Далее инструкция приводится для разработки на ОС Windows, однако соответствующие действия могут быть выполнены и на других поддерживаемых ОС. После установки необходимого ПО, скачиваем репозиторий с GitHub [26].

Открыв WebStrom, в терминале редактора переходим сначала в папку с файлами серверного компонента приложения – server, затем папку с файлами клиентского компонента – client, и в каждой выполняем команду «npm install» (рисунок 19). Она установит все необходимые модули и библиотеки для работы.

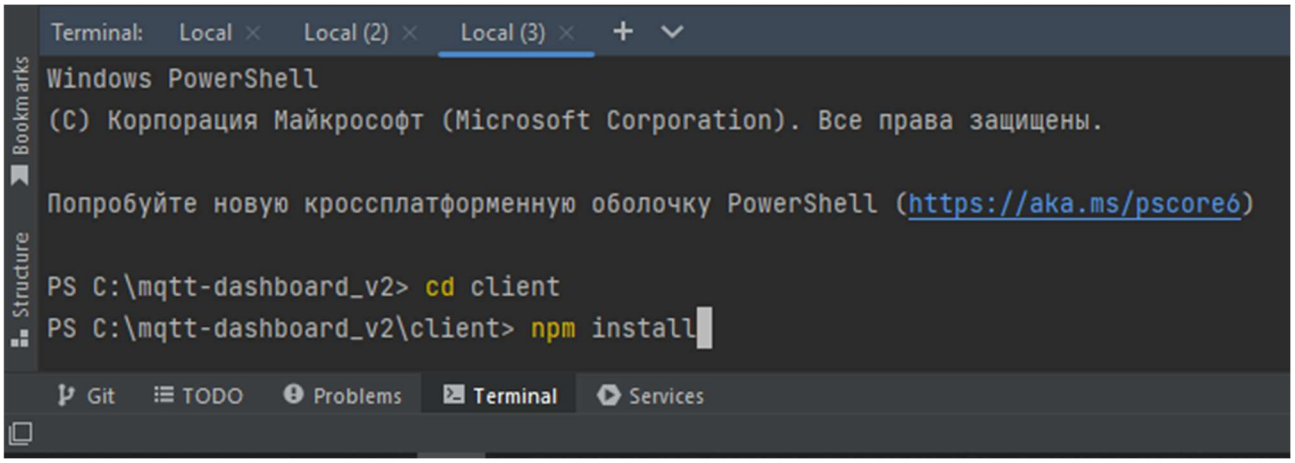


Рисунок 19 – Установка необходимых модулей

Теперь в одном окне терминала переходим в папку server, а во втором в папку client. В папке server выполняем команду «npm run dev», а в папке client «npm start» (рисунок 20). Это запустит проект в режиме разработчика, благодаря чему после каждого сохранения файла проект будет заново компилироваться и выдавать информацию о появившихся ошибках. Для добавления необходимых компонентов переходим в соответствующую папку проекта. Например, для добавления новой страницы для клиента переходим в client\src\pages, для

добавления нового «контроллера» для взаимодействия с БД на сервере в server\controller.

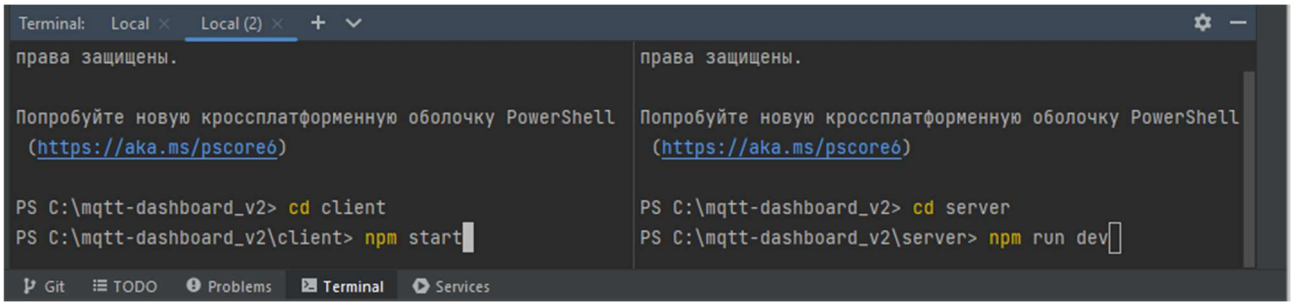


Рисунок 20 – Запуск проекта в режиме разработчика

Описание назначения файлов в проекте для клиентской части и серверной можно увидеть в таблицах 4 и 5 соответственно.

Таблица 4 – Список файлов клиентской части проекта и их назначение

Список файлов	**Назначение файлов**
App.js	Аутентификация пользователя, передача данных пользователя в локальное хранилище
Home.jsx	Получение данных с сервера о карточках, устройствах и комнатах. Отображение всех компонентов главной страницы
Sidebar.jsx	Отображение бокового меню и описание логики перенаправления на выбранную страницу
Navbar.jsx	Отображение верхнего поля навигации
Login.jsx	Отображение страницы авторизации и описание логики ее работы
InfoCard.jsx	Отображение страницы с информацией об отдельной карточке
SettingsPage.jsx	Отображение страницы настроек
InfoScenario.jsx	Отображение страницы с настройками сценария
Widget.jsx	Отображение карточки

Окончание таблицы 4

Список файлов	Назначение файлов
AppRouter.js	Описание логики отображения страниц при переадресации
WidgetList.jsx	Отображение элементов Widget.jsx на главной странице
CreateCard.jsx	Модальное окно для добавление новой карточки
ConfirmDeleteCard.jsx	Модальное окно для подтверждения удаление устройства
CreateDevice.jsx	Модальное окно для добавление нового устройства
DeviceStore.js CardStore.js RoomStore.js UserStore.js BrokerStore.js ScenarioStore.js	Описание локальных свойств и методов для сущностей device, card, room, user, broker и scenario
cardAPI.js roomAPI.js userAPI.js deviceAPI.js brokerAPI.js scenarioAPI.js	Описание функций для передачи запросов к серверу для сущностей базы данных card, room, user, device, broker и scenario соответственно

Таблица 5 – Список файлов серверной части проекта и их назначение

Список файлов	Назначение файлов
cardController.js deviceController.js userController.js room.controller.js broker.controller.js scenario.controller.js	Реализация запросов к базе данных для каждой сущности: card, device, user, room, broker, scenario соответственно

Окончание таблицы 5

Список файлов	**Назначение файлов**
card.routes.js device.routes.js user.routes.js room.routes.js broker.routes.js scenario.routes.js index.js	Настройка переадресации запроса по заданному URL к нужному контроллеру. index.js – файл для объединения переадресаций всех сущностей в один модуль
models.js	Описание сущностей базы данных и взаимосвязей между ними
authMiddleware.js	Обработка ошибок авторизации при обмене данных с сервером
errorHandingMiddleware.js	Обработка непредвиденных ошибок при обмене данных с сервером
checkRoleMiddleware.js	Аутентификация пользователя
ApiError.js	Описание полученных сообщений об ошибках при обмене данных с сервером
db.js	Описание параметров для подключение к базе данных
index.js	Инициализация фреймворка сервера и подключение к базе данных

ЗАКЛЮЧЕНИЕ

В результате работы было спроектировано и реализовано веб-приложение для управления умным домом.

Был разработан графический интерфейс программы, структура приложения. В приложении реализована возможность подключения к умным устройствам через протокол MQTT и обмена сообщения с ними, с помощью чего есть возможность управлять устройствами и получать нужные данные от них. Имеется возможность настроить автоматическое изменения состояния устройств в заданное время. Предусмотрена многопользовательская работа с устройствами.

Клиентское приложение работает на ОС Windows, Linux, MacOS. Серверное – на ОС Linux.

Последняя версия приложения размещена в git-репозитории [26].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Умный дом с Алисой [Электронный ресурс]. – Режим доступа: https://yandex.ru/alice/smart-home (дата обращения: 11.12.2022).
2. Характеристики Яндекс Станции первого поколения – Yandex [Электронный ресурс]. – Режим доступа: https://yandex.ru/support/station/meet/characteristics-gen1.html (дата обращения: 11.12.2022).
3. Характеристики Яндекс Станции 2 [Электронный ресурс]. – Режим доступа: https://yandex.ru/support/station/meet/characteristics-gen2.html (дата обращения: 11.12.2022).
4. Xiaomi Mi Home [Электронный ресурс]. – Режим доступа: https://xiaomi-smarthome.ru/xiaomi-mi-home/ (дата обращения: 11.12.2022).
5. Хаб для устройств умного дома xiaomi mi smart home hub xiaomi mi smart home hub [Электронный ресурс]. – Режим доступа: https://domoticzfaq.ru/khab-dlya-ustroystv-umnogo-doma-xiaomi-mi-smart-home-hub-xiaomi-mi-smart-home-hub/ (дата обращения: 11.12.2022).
6. Xiaomi Hub Gateway 4 (ZSWG01CM) [Электронный ресурс]. – Режим доступа: https://xiaomi-smarthome.ru/xiaomi-hub-gateway-4-zswg01cm/#1 (дата обращения: 11.12.2022).
7. Добавление Wi-Fi устройств в MiHome [Электронный ресурс]. – Режим доступа: https://rumihome.ru/materialy/dobavlenie-wi-fi-ustrojstv-v-mihome (дата обращения: 11.12.2022).
8. Модуль Xiaomi miIO – Протокол miIO [Электронный ресурс]. – Режим доступа: https://kb.mjdm.ru/xiaomimiio-protocol/ (дата обращения: 11.12.2002).
9. Умный дом Sber [Электронный ресурс]. – Режим доступа: https://sberdevices.ru/help/smarthome/ (дата обращения: 11.12.2022).

10. Что умеет умный дом Sber? [Электронный ресурс]. – Режим доступа: https://sberdevices.ru/help/smarthome/smarthome-about/use (дата обращения: 11.12.2022).

11. MQTT-to-Cloud для DIY [Электронный ресурс]. – Режим доступа: https://developers.sber.ru/docs/ru/smarthome/mqtt-diy/mqtt-to-diy (дата обращения: 11.12.2022).

12. Характеристики – SberBox [Электронный ресурс]. – Режим доступа: https://sberdevices.ru/help/sberbox/sberbox-meet/sberbox-specifications (дата обращения: 11.12.2022).

13. «Сбер» строит собственную платформу умного дома [Электронный ресурс]. – Режим доступа: https://www.cnews.ru/news/top/2022-05-06_sber_stroit_sobstvennuyu (дата обращения: 11.12.2022).

14. MQTT Specifications [Электронный ресурс]. – Режим доступа: https://mqtt.org/mqtt-specification/ (дата обращения: 19.12.2022).

15. MQTT, CoAP, ZigBee или LwM2M? Какой протокол выбрать? [Электронный ресурс]. – Режим доступа: https://wm-it.pro/mqtt-coap-zigbee-ili-lwm2m-kakoy-protokol-iot-vybrat (дата обращения: 19.12.2022).

16. Рейтинг лучших одноплатных микрокомпьютеров на 2022 год [Электронный ресурс]. – Режим доступа: https://vyborok.com/rejting-luchshih-odnoplatnyh-mikrokompyuterov/#_5000 (дата обращения: 19.12.2022).

17. The top programming languages [Электронный ресурс]. – Режим доступа: https://octoverse.github.com/2022/top-programming-languages (дата обращения: 19.12.2022).

18. JavaScript [Электронный ресурс]. – Режим доступа: https://www.javascript.com/ (дата обращения: 19.12.2022).

19. Node.js [Электронный ресурс]. – Режим доступа: https://nodejs.org/en/ (дата обращения: 19.12.2022).

20. Express – Fast, unopinionated, minimalist web framework for Node.js [Электронный ресурс]. – Режим доступа: https://expressjs.com/ (дата обращения: 19.12.2022).

21. Выберите лучший JavaScript-фреймворк для разработки на стороне сервера [Электронный ресурс]. – Режим доступа: https://www.internet-technologies.ru/articles/vyberite-luchshiy-javascript-freymvork-dlya-razrabotki-na-storone-servera.html (дата обращения: 19.12.2022).

22. React – A JavaScript library for building user interfaces [Электронный ресурс]. – Режим доступа: https://reactjs.org/ (дата обращения: 19.12.2022).

23. Vue.js - The Progressive JavaScript Framework [Электронный ресурс]. – Режим доступа: https://vuejs.org/ (дата обращения: 19.12.2022).

24. Лучший JavaScript-фреймворк 2021: React или Vue? [Электронный ресурс]. – Режим доступа: https://evrone.ru/react-vs-vue (дата обращения: 19.12.2022).

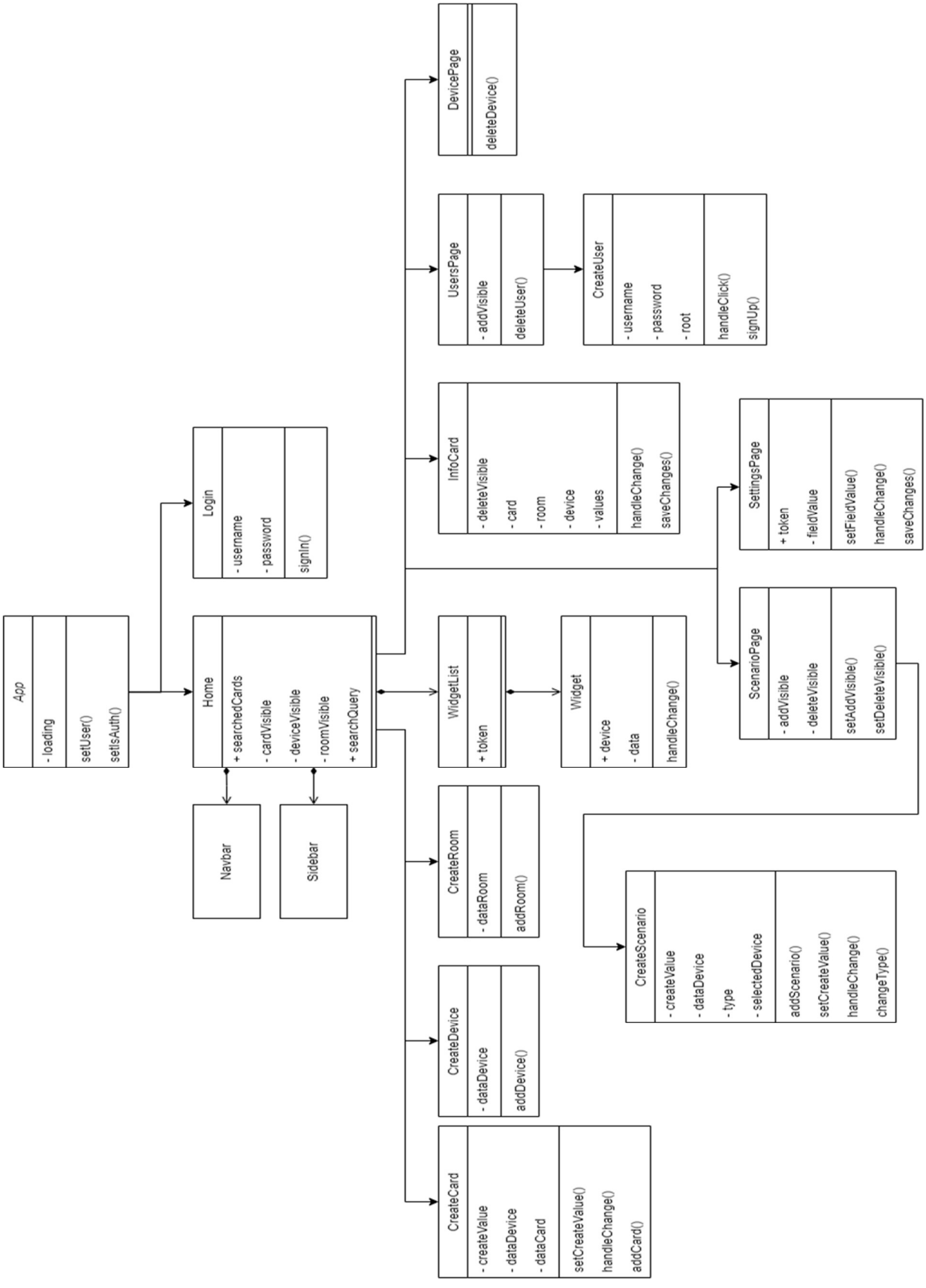
25. Stack Overflow Developer Survey 2021 [Электронный ресурс]. – Режим доступа: https://insights.stackoverflow.com/survey/2021 (дата обращения: 19.12.2022).

26. Mqtt-dashboard_v2 [Электронный ресурс]. – Режим доступа: https://github.com/JYRAVLENOK/mqtt-dashboard_v2 (дата обращения: 19.03.2023).

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов клиентской части



Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись
«22» 06 2023 г.

БАКАЛАВРСКАЯ РАБОТА

Веб-приложение для управления умным домом

09.03.01 – «Информатика и вычислительная техника»

Руководитель	19.06.23 подпись, дата	ст. преподаватель	К. В. Пушкарев
Выпускник	19.06.23 подпись, дата		А.С. Васильев
Нормоконтролер	19.06.23 подпись, дата		К. В. Пушкарев

Красноярск 2023