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

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« _____» _____ 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специальности

Автоматизированная система управления помещением
тема

Руководитель	_____	профессор, канд. техн. наук	Б.И. Борде
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		А.Г. Будажапова
	подпись, дата		инициалы, фамилия
Нормоконтроллер	_____	профессор, канд. техн. наук	Б.И. Борде
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2020

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

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
« ___ » _____ 2020 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту _____Будажаповой Алтане Геннадьевне_____

фамилия, имя, отчество

Группа __КИ16-06Б__ Направление (специальность) __09.03.01_____

номер код

_____Информатика и вычислительная техника_____

наименование

Тема выпускной квалификационной работы Автоматизированная система__ управления помещением_____

Утверждена приказом по университету № _____ от _____

Руководитель ВКР Б. И. Борде, профессор, канд. техн. наук., кафедра ИВТ____

инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР: сформулировать цели и задачи, оценить актуальность, провести анализ существующих систем автоматизации, реализовать систему автоматизированного управления помещением.

Перечень разделов ВКР: анализ предметной области и обзор аналогичных систем, проектирование и обозначение функциональных требований собственной системы, разработка программного обеспечения, инструкция пользователя.

Перечень графического материала: презентация в формате PowerPoint.

Руководитель ВКР _____ _____Б. И. Борде_____

подпись инициалы и фамилия

Задание принял к исполнению _____А. Г. Будажапова__

подпись, инициалы и фамилия студента

« ___ » _____ 2020 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Автоматизированная система управления помещением». Пояснительная записка содержит 47 страниц текстового документа, 17 иллюстраций, 2 таблиц и 17 использованных источников.

ВЕБ-ПРИЛОЖЕНИЕ, СИСТЕМА УПРАВЛЕНИЯ ПОМЕЩЕНИЕМ, МИКРОКОНТРОЛЛЕРНАЯ СИСТЕМА, АВТОМАТИЗАЦИЯ.

Объектом исследования данной работы является разработка и создание системы автоматизации для помещения посредством веб-приложения.

Цель работы – разработать автоматизированную систему управления помещением на базе микропроцессорной системы NodeMCU ESP8266 по беспроводной сети Wi-Fi, с дальнейшей возможностью связи с любыми датчиками и исполнительными устройствами.

В процессе работы был произведен анализ предметной области, после чего были получены представления о системе, которая была бы актуальна в данный момент на основе этого был разработан функциональные возможности системы.

Результатом данной работы является разработка веб-сервера, осуществляющего следующий функционал: прием, обработка и отправка данных датчикам и исполнительным устройствам.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данный момент времени идет тенденция к увеличению качества жизни, так как технологии для этого доступны практически для всех людей. Вы не обязаны разбираться в программировании и электротехнике – готовые программы уже существуют в интернете, а для подключения датчиков можно посмотреть даташит (официальный документ производителя электронных компонентов), где все подробно расписано. Но если у вас нет времени или вам не хочется этим заниматься, подобные системы можно с лёгкостью купить в специализированных магазинах. Подобные системы домашней автоматизации решают следующие задачи:

- дистанционное управление электроприборами;
- обеспечение комфортной среды;
- увеличение свободного времени, которое ранее использовалось на выполнение рутинных задач;
- повышение качества жизни человека.

Подобная система может называть как домашней автоматизацией (англ. home automation), так и умным домом (англ. smart home). Это система домашних устройств, способных выполнять действия и решать определённые повседневные задачи без участия человека. Домашняя автоматизация рассматривается как частный случай интернета вещей, она включает доступные через интернет домашние устройства, в то время как интернет вещей включает любые связанные через интернет устройства в принципе.

Наиболее распространенные примеры автоматических действий в «умном доме» — автоматическое включение и выключение света, автоматическая коррекция работы отопительной системы или кондиционера и автоматическое уведомление о вторжении, возгорании или протечке воды.

Домашняя автоматизация в современных условиях — чрезвычайно гибкая система, которую пользователь конструирует и настраивает самостоятельно в зависимости от собственных потребностей. Это предполагает, что каждый владелец умного дома самостоятельно определяет, какие устройства куда установить и какие задачи они будут исполнять [1].

Цель данного дипломного проекта – разработка программного обеспечения WIFI модуля на базе микропроцессорной системы NodeMCU ESP8266, для реализации автоматизированного управления по беспроводной сети.

Заданием для выпускной квалификационной работы является разработать микропроцессорную систему автоматизации на базе модуля NodeMCU ESP8266 и выполнить следующие пункты:

- провести анализ предметной области, анализ аналогичных решений и их начального конфигурирования;
- разработать способ применения микроконтроллера NodeMCU ESP8266;
- выбрать протокол взаимодействия микроконтроллера с элементами управления;
- разработать структуру программного обеспечения контроллера;
- разработать и протестировать программное обеспечение.

1 Анализ предметной области

1.1 Постановка задачи

Можно выделить несколько типов микроконтроллерных систем управления помещением для взаимодействия с пользователем:

- управляющая панель со встроенным программным обеспечением;
- интерфейс в виде пульта управления;
- приложение или веб-интерфейс для управления системой.

Управляющая панель – один из вариантов реализации, имеет, как и свои преимущества, так и недостатки. Из плюсов можно выделить то, что у управляющих панелей в большинстве случаев удобный и понятный любому пользователю интерфейс, удобство использования и простота эксплуатации. Из минусов – система выполнит определенное действие при выборе его на управляющей панели, то есть это достаточно локальная система.

Пульт управления – это устройство для контроля и управления работой устройств и процессов [1]. Данный вариант более мобилен, чем предыдущий, в чем заключается его несомненный плюс, но также в этом есть свой минус – пульт управления достаточно просто потерять или существует вероятность случайно сломать. Если контроль над устройствами требуется вам на большой площади – возможно, что сигнал не будет достигать точки назначения.

Для последнего варианта все вышеперечисленные минусы не относятся, но есть несколько минусов для некоторых систем подобного типа, что устройства могут работать не так, как запланировано без стабильного интернет-соединения и питания от электричества.

Из всех вышеперечисленных интерфейсов более предпочтителен последний, так как один из главных плюсов заключается в том, что можно осуществлять контроль за домом удаленно.

1.2 Актуальность темы бакалаврской работы

В данный момент времени, в нашем временном промежутке, автоматизация понемногу набирает распространенность по всему миру – от автоматизации отдельных компонентов дома (к примеру, уличное освещение, привязанное к датчику движения), так и до сложных систем автомобильных автопилотов, которые позволяют объезжать препятствия без активного участия водителя. Конечно, это очень сложная и продуманная система, но в данной бакалаврской работе речь будет идти о системе автоматизации помещения.

Система автоматизации помещения позволяет удаленно взаимодействовать с датчиками и устройствами. Посредством внедрения автоматизации в ваш дом, можно установить какое угодно количество датчиков, следить за значениями с датчиков, воспроизводить пользовательские сценарии. В таком случае можно регулировать температуру в доме по алгоритму с прогнозированием: в таком случае, вам будет нужно два датчика, первый должен находиться в доме, а второй – учитывать наружную температуру. При помощи алгоритма с прогнозированием можно постоянно поддерживать комфортную температуру в доме.

Таким образом, актуальность данной темы только начинает набирать обороты, поэтому, на мой взгляд, нужны различные реализации подобных систем для того, чтобы пользователи, которые хотят внедрить себе такую систему могли иметь выбор и выбирать именно под свои нужды.

1.3 Цели и задачи бакалаврской работы

Микроконтроллерная система должна удовлетворять следующим критериям:

- ценовая доступность;
- модульность;

- наращиваемость, т.е. возможность добавления оборудования для расширения функционала системы;

- интуитивно понятный пользовательский интерфейс;

- поддержка оборудования разных производителей;

- использование открытого программного обеспечения.

Далее следует рассмотреть системы с аналогичным функционалом, чтобы понять, в каком направлении лучше развивать собственный проект. Также следует учесть вышеописанные критерии, которые сформулированы на данный момент.

1.4 Обзор аналогов

1.4.1 Комплект Xiaomi Smart Home

Xiaomi Smart Home, который представлен на рисунке 1.1, можно приобрести, как и отдельно, так и единым комплектом. Комплект состоит из 4 устройств: центральный модуль управления, датчик открытия окна или датчик открытия двери, датчик движения и беспроводная кнопка. Средняя стоимость оборудования – 3700 рублей.

Рисунок 1.1 – «Умный дом» Xiaomi

Если говорить о возможности поддержки оборудования разных производителей, то Xiaomi Smart Home поддерживает только собственные датчики и исполнительные устройства. Управление Xiaomi Smart Home происходит через приложение Xiaomi MiHome, которое можно увидеть на рисунке 1.2, доступное как для ОС Android так и для iOS [3].

Рисунок 1.2 – Мобильное приложение для управления «умным домом»

1.4.2 Система Google Home

Мобильное приложение Google Home, представленное на рисунке 1.3, для операционной системы Android и iOS позволяет управлять в одном месте всеми устройствами для умного дома с поддержкой Google Assistant. К тому же управлять можно не только устройствами от Google, но можно добавить различные устройства для умного дома от сторонних производителей: Xiaomi Mi Home, Yeelight, Sonoff eWeLink, Broadlink и другие.

Рисунок 1.3 – Приложение Google Home

Управлять устройствами можно как через приложение, так и через умную колонку Google Home, кроме того, данная колонка понимает команды на русском языке. Цена для минимальной модели Google Home Mini, изображенная на рисунке 1.4 – в среднем идет от 2500 рублей, хотя для автоматизации дома необязательно покупать подобную колонку – достаточно иметь совместимые датчики и исполнительные устройства [4].

Рисунок 1.4 – Google Home Mini

1.4.3 Система Amazon Echo

Амазон Эхо совместим с большинством популярных производителей устройств для умного дома, например Philips Hue, WeMo, TP-Link. Голосовой помощник Alexa в данный момент еще не поддерживает русский язык. Управление вашими устройствами осуществляется только при помощи умной колонки посредством голоса (при условии, что ваши устройства совместимы с Alexa). Amazon Echo Dot 2nd Gen (умная колонка из минимальной ценовой категории), изображенная на рисунке 1.5, в среднем обойдется вам в 3000 рублей.[5]

Рисунок 1.5 – Умная колонка Amazon Echo Dot 2nd Gen

1.4.4 Система автоматизации Apple HomeKit

Умная колонка Apple HomePod, изображенная на рисунке 1.7, поддерживает русский язык, различные сценарии и тому подобное. Цена на Apple HomePod начинается от 27 000 рублей, но данную систему можно использовать и без этого девайса, главное – найти совместимые устройства. Управление устройством происходит через специальное приложение Apple Homekit, которое изображено на рисунке 1.6. Это программная платформа Apple, которая позволяет пользователям iOS, macOS, watchOS и tvOS управлять умными аксессуарами в доме как через приложение Home, так и через голосовой помощник Siri [6].

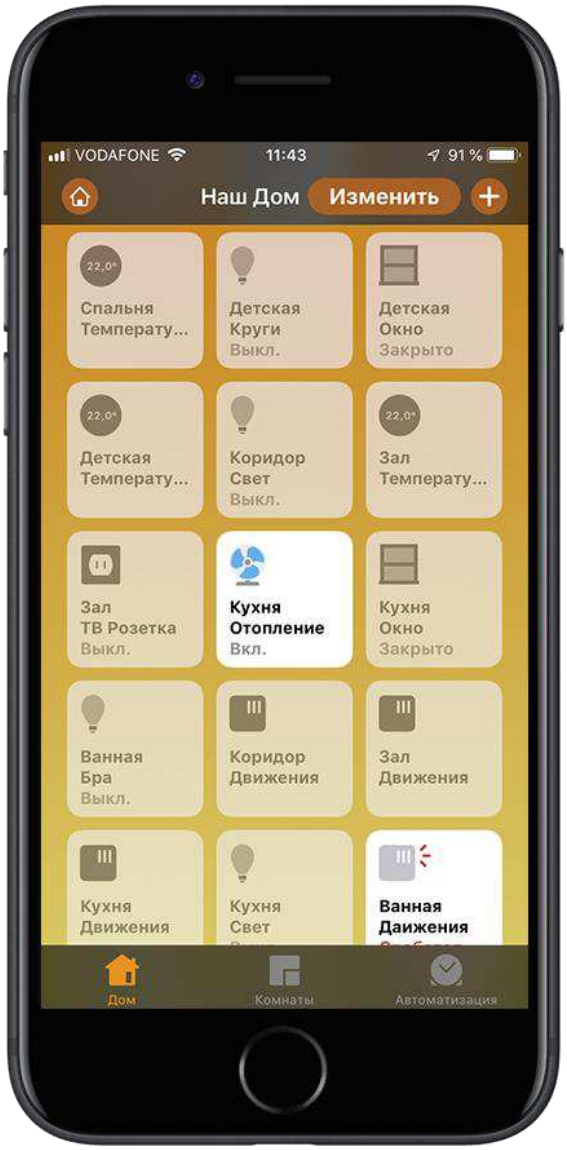

Рисунок 1.6 – Мобильное приложение Apple HomeKit

Рисунок 1.7 – Умная колонка Apple HomePod

На основании полученной таблицы 1 мы можем сделать вывод, что бюджетная система с открытым исходным кодом будет востребована для определенной группы пользователей, которые, к примеру, хотят убедиться, что их домашняя автоматизация не передает личную информацию на удаленные сервера.

Таблица 1 – Сравнение аналогов

Продукт	Цена, р.	Модульность	Наращиваемость	Интерфейс	Разное оборудование	Открытое ПО
Xiaomi	3700	+	+	-	-	-
Google	2500	+	+	+	+	-
Amazon	3000	+	+	+	+	-
Apple	5000	+	+	+	+	-

2 Проектирование системы

2.1 Функциональные требования

После анализа систем, также реализующих различные системы автоматизации, были сформированы следующие функциональные требования:

- система должна уметь принимать информацию от любых датчиков и исполнительных устройств;
- система должна уметь отправлять информацию любым датчикам и исполнительным устройствам;
- система должна уметь корректно обрабатывать любую поступающую на вход информацию;
- система должна уметь сохранять корректно поступившую информацию;
- в системе должен быть предусмотрено наличие сценариев, их добавление, редактирование и удаление;
- система должна быть интуитивно понятная для любого пользователя;
- графический интерфейс системы должен быть приятным для восприятия.

2.1.1 Функциональные требования работы с устройствами

Для работы с устройствами необходимо связать сервер с устройствами. Данные должны быть, как и входящими, так и исходящими для микроконтроллера и сервера. В первую очередь к ним относятся:

- добавление устройств на сервер в соответствующую таблицу в базе данных;

- редактирование устройств в соответствующей таблице в базе данных;

- удаление устройств в соответствующей таблице в базе данных;

- корректная обработка данных по заданным сценариям.

Информация от устройства содержит в себе:

- уникальный идентификатор устройства;

- идентификатор комнаты, в которой находится устройство;

- тип устройства;

- имя устройства;

- поле, отображающее добавление в избранное;

- рисунок устройства;

- текстовое описание устройства;

После подключения устройства в графическом интерфейса отображается само устройство и все его поля, на сервере, в свою очередь, происходит добавление в соответствующую таблицу базы данных.

Следовательно, пользователь может просматривать всю существующую информацию об устройстве, получать полный список всех комнат, получать полный список всех устройств, подключенных к определенной комнате, так же имеет возможность редактирования полей и полного удаления устройства.

Данные, пришедшие с датчиков сохраняются в базе данных и выводятся в удобном для пользователей виде.

2.1.2 Функциональные требования работы со сценариями

Одна из важных функций в данном проекте – это создание сценариев. Пользователь может создать, изменить и удалить сценарий для своих исполнительных устройств.

Сценарии должны выполняться при определенных условиях. При получении определенного значения с датчика, сервер обрабатывает данные и отправляет соответствующую команду для исполнительного устройства.

Сценарии сохраняются в базу данных, таблица в базе данных должна обладать следующими полями:

- уникальный идентификатор сценария;
- топик устройства;
- заданное значение, при котором должно происходить определенное действие;
- текстовое описание сценария.

Таким образом, после анализа аналогов была определена будущая функциональность данного проекта.

2.2 Структурная схема системы

В качестве основы структурной схемы микропроцессорной системы автоматизации помещением возьмем стандартную схему, представленную на рисунке 2.

Блок источников входных сигналов необходим для того, чтобы считывать и собирать всю необходимую информацию с датчиков. Блок обработки получает данные от блока источников входных сигналов и производит работу с ними в соответствии с алгоритмами. В качестве блока ввода используются – датчики входных сигналов, устройства управления – микроконтроллер (в данном случае NodeMCU ESP8266). Блок исполнительных устройств – сервер, получает информацию от блока обработки и выполняет все действия, которые необходимы по алгоритму.

Рисунок 2 – Структурная схема системы

2.3 Выбор средств реализации

2.3.1 Выбор структуры проекта

Выбор структуры проекта был осуществлен в пользу клиент-серверной архитектуры. Клиент-серверная архитектура – это архитектура, которая построена на взаимодействие по схеме «запрос-ответ». В данном случае клиент выполняет активную функцию – инициирует запросы, пассивную же функцию реализует сервер, т.к. его функция заключается в ответах на запросы клиента. Тем не менее, по мере развития системы роли могут меняться: программный блок может одновременно выполнять функции сервера по отношению к одному блоку и клиента по отношению к другому.

Любая информационная система должна обладать минимум тремя основными функциональными частями: модули хранения данных, их обработки и интерфейса с пользователями. Любая из данных частей может быть реализована независимо от двух других. К примеру, можно изменить интерфейс пользователя таким образом, что одни и те же данные, взятые из базы данных, будут отображаться в виде таблиц, графиков, гистограмм и так далее, не меняя целостности программ, используемых для хранения и обработки данных. Также без изменений представления данных можно изменить программы обработки, например, изменив алгоритм поиска. Кроме того, без изменения представлений можно изменить программное

обеспечение для хранения данных переходом на другую файловую систему [7].

Но данные системы обладают также существенным минусом – при отказе одной из функциональных частей полностью остановится работа всего приложения. Такие проблемы решаются дублированием компонентов данной архитектуры.

На рисунке 3 можно увидеть структурную схему проекта.

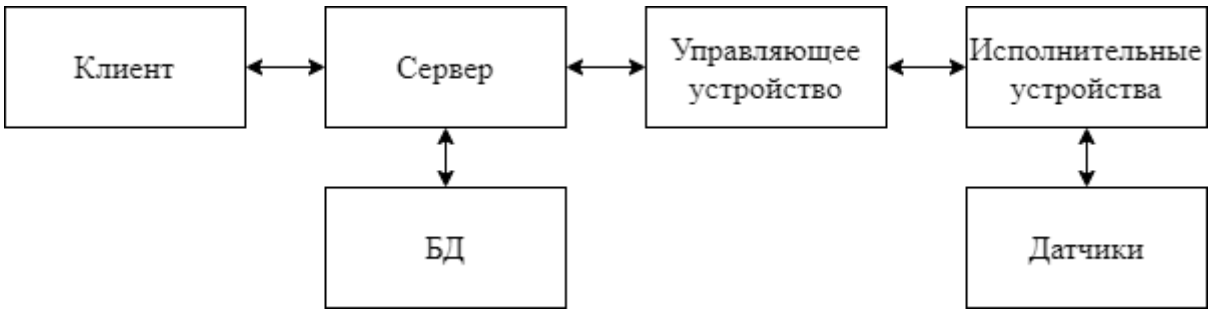

Рисунок 3 – Схема клиент-серверной архитектуры

К клиентской части приложения относится браузер – программное обеспечение для просмотра веб-страниц. При помощи браузера клиент взаимодействует с сервером, отправляя запросы посредством протокола HTTP/HTTPS. В клиентской части реализуется отображение пользовательского интерфейса конечному пользователю, также формируются запросы для сервера и производится обработка данных, полученных от сервера.

К серверной части приложения относятся:

- сервер – принимает HTTP-запросы от клиентов. Сервер получает запрос от клиента, обрабатывает его в соответствии с алгоритмами, после чего формирует веб-страницу и отправляет его клиенту с использованием протокола HTTP/HTTPS.

- база данных – компьютеризированная система хранения записей, основное назначение которой – хранить информацию, предоставляя

пользователям средства ее извлечения и модификации [8]. В базе данных хранится вся информация, которая предварительно обработана и проверена сервером на корректность.

- микроконтроллерная система – устройство, предназначенное для управления электронными устройствами. Данное устройство взаимодействует с датчиками и исполнительными устройствами, кроме того, является посредником между аппаратной и программной части комплекса.

2.3.2 Выбор способа реализации серверного приложения

Приложение должно осуществлять прием, обработку, анализирование, сохранение данных с входных устройств и так же предусматривать управление исполнительными устройствами на основе полученных данных.

Разработка программного обеспечения предусматривает два варианта реализации:

- настольное приложение – клиентское программное обеспечение, устанавливаемое локально на рабочую станцию пользователя. Для выполнения данного приложения требуются аппаратные ресурсы компьютера. Такие приложение характеризуются автономностью, имеют высокое быстродействие, не всегда обладают кроссплатформенностью, обладают более сложной реализацией кастомизации.

- веб-приложение – клиентское программное обеспечение, представляющее собой веб-браузер и использующее http/https протоколы. Приложение не требует установки и загрузки каких-либо программных модулей непосредственно на рабочую станцию пользователя. Допускается установка дополнительных общесистемных библиотек и использование защищенных сетевых протоколов. Логика такого приложения распределена

между приложениями сервера и клиента, хранение данных и выполнение основных операция производится преимущественно за счет сервера. Передача данных происходит по сети.

Для реализации данной бакалаврской работы больше всего подойдет веб-приложение, т.к. для реализации функций передачи, хранения и обработки данных больше подойдет удаленный веб-сервер, так же для реализации мобильности, масштабируемости и кроссплатформенности системы, то есть просмотр и управление данными можно осуществлять с любого устройства.

2.3.3 Выбор программной платформы для реализации задачи

Рассмотрим одни из самых популярных технологий для написания веб-сервера: Node.js, PHP, ASP.NET Core.

- Node.js – среда выполнения JavaScript, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Большинство Node.js-фреймворков не имеют фиксированной структуры каталогов и может быть непросто с самого начала проекта придерживаться правильной структуры.

- PHP — это скриптовый интерпретируемый язык, созданный специально для разработки серверной части сайтов. Для запуска локального сервера нужно пользоваться специализированным программным обеспечением, к примеру, Open Server, VertigoServ и др. Данный язык более предпочтителен для небольших проектов. Поддержка кода на PHP сложнее, потому что в нём реже используется ООП и нет типизации, а также сложно проводить отладку.

- ASP.NET Core — это opensource-фреймворк для языков из семейства .NET, который позволяет писать серверную часть сайтов. Процесс отладки на ASP.NET Core в значительной мере проще и удобнее, нежели на других рассматриваемых платформах. Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget. ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. Так же можно легко опубликовать приложение в Azure с помощью Visual Studio.

Для реализации поставленной задачи больше всего подойдет ASP.Net, в виду своих разносторонних возможностей: расширяемость, модульность, единый стек веб-разработки и простая публикация в облаке.

2.3.4 Выбор аппаратного обеспечения для реализации задачи

Основные требования к отладочной плате: модуль WiFi для передачи данных датчиков на сервер, теоретическая расширяемость, доступная стоимость. Для более полного развития данной темы в будущем могут использоваться совершенно различные датчики и исполнительные устройства, поэтому в текущем случае не будут рассматриваться конкретные используемые средства. Рассмотрим три самых популярных варианта отладочных плат.

Можно взять любую отладочную плату Arduino, к примеру, Arduino Uno с модулем ESP8266 WiFi. Данная микропроцессорная система подходит по двум параметрам, а именно по расширяемости и наличию модуля WiFi, но оригинальная Arduino стоит около 3 т. р. и модуль WiFi в среднем 200 р.

Для данной задачи можно взять любую отладочную плату из семейства STM, для примера возьмем Nucleo STM32F401RE и WiFi-модуль

SPWF01SA, поддерживающий стандарт 802.11b,g,n и предназначенный для создания приложений «Интернета вещей». Данная микропроцессорная система так же соответствует по двум параметрам, но цена одного модуля – в среднем 1.5 т. р, к тому же нужно прибавить стоимость WiFi-модуля.

Отладочная плата NodeMCU ESP8266 уже обладает модулем WiFi, то есть не нужно ничего докупать, все идет уже «из коробки». Данная микропроцессорная система соответствует нашим требованиям: имеет модуль WiFi, теоретически расширяема и цена в рознице около 400 р.

Можно сделать выводы по таблице 2, что для решения данной задачи больше всего подходит отладочная плата NodeMCU ESP8266 v3.

Таблица 2 – Сравнение функционала по наиболее приоритетным характеристикам

Параметры	Arduino Uno	Nucleo STM32F401RE	NodeMCU ESP8266 v3
Flash-память	32 КБ	512 кБ	4Мб
Рабочий диапазон температур	От -10 С до 50 С	От -10 С до 40 С	От -40С до 125С
Модуль WiFi	-	-	+
Количество портов ввода-вывода	14	90+	11
Питание	6-20В	5 В, 7–12 В	3,3В/5В/18В
Прошивка	Обновление через USB	Обновление через USB	Обновление через USB
Средняя цена	3000 р	1 500 р	400 р

2.3.5 Выбор протокола для реализации задачи

Рассмотрим три различных протокола для передачи данных.

- HTTP (англ. HyperText Transfer Protocol) – протокол передачи гипертекста, широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам). Протокол HTTP предполагает

использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом. [9]

- FTP (англ. File Transfer Protocol) — протокол передачи файлов по сети, является одним из старейших прикладных протоколов, появившихся задолго до HTTP, и даже до TCP/IP. И в наши дни он широко используется для распространения ПО и доступа к удалённым хостам. В отличие от TFTP, гарантирует передачу (либо выдачу ошибки) за счёт применения квотируемого протокола. Протокол построен на архитектуре «клиент-сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно [10].

- MQTT (англ. message queuing telemetry transport) — упрощённый сетевой протокол, работающий поверх TCP/IP, ориентированный для обмена сообщениями между устройствами по принципу издатель-подписчик.

Протокол ориентируется на простоту в использовании, невысокую нагрузку на каналы связи, работу в условиях постоянной потери связи, лёгкую встраиваемость в любую систему. Основное предназначение — работа с телеметрией от различных датчиков, устройств, использование шаблона подписчика обеспечивает возможность устройствам выходить на связь и публиковать сообщения, которые не были заранее известны или предопределены, в частности, протокол не вводит ограничений на формат передаваемых данных [11].

Делая выводы по выбору протокола для данного проекта, можно заметить, что протокол MQTT больше подходит для выполнения задач по техническому заданию: он предназначен для работы с различными устройствами, если у вас низкая пропускная способность интернет-соединения или же внезапно случилась потеря связи – данный протокол обеспечит доставку сообщения.

2.3.6 Выбор интегрированной среды разработки для аппаратного обеспечения

IDE ESPlorer. Язык программирования – LUA. Чтобы запустить скрипт на LUA, нужно написать скетч в ESPlorer, после нужно сохранить скрипт с названием init.lua. Сразу после этого начнется автоматическая загрузка написанного кода в отладочную плату и его выполнение. Если операция выполнена успешно, отладочная плата начнет мигать светодиодом. Важно отметить, что плата самостоятельно выполняет скрипт, подключение к компьютеру нужно только для подачи питания.

Основной недостаток в том, что VM LUA исполняет LUA скрипты лишь размещенные в оперативной памяти кристалла. А этой памяти для скриптов всего лишь 20 Кбайт. Этого объема памяти хватает на исполнение скрипта примерно в 110 строк.

В Arduino IDE используется язык программирования C++, позволяет писать скетчи и загружать их одним кликом в флеш-память ESP8266. Единственный нюанс – в IDE требуется добавить поддержку ESP8266. ArduinoIDE обладает большим количеством различных готовых проектов, можно запустить тестовый проект в пару кликов прямо из IDE.

Среди рассмотренных интегрированных систем разработки для выполнения поставленных целей больше всего подойдет ArduinoIDE, т.к. это более универсальное решение за счет популярности и

распространенности языка программирования и IDE, так же за счет использования флеш-памяти устройства.

2.3.7 Выбор дизайна для создания удобного интерфейса

Выбор дизайна веб-приложения в первую очередь исходил из следующих критериев: удобство пользования, интуитивно понятный интерфейс, лаконичность, приятная для восприятия палитра цветов.

Для подобных критериев больше подходит Flat Design, Material Design и Fluent Design. Можно сказать, что все вышеперечисленные стили интерфейса – это продолжение идеологии Flat Design. Ниже рассмотрим их подробнее.

Плоский дизайн (англ. Flat Design) — дизайн интерфейсов программ и операционных систем, представленный, как противоположность реализму. Это минималистический подход к дизайну, подчеркивающий удобство использования. С 2013 года – новый стандарт в дизайнерском компьютерном направлении, пример можно увидеть на рисунке 4.1 [12].

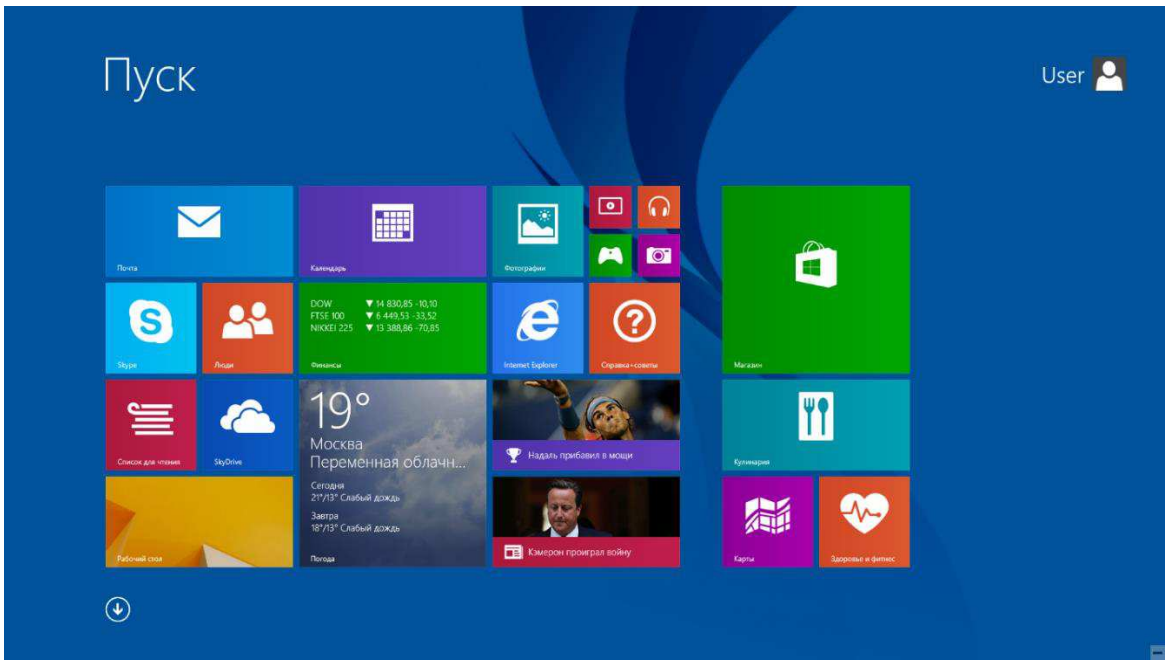

Рисунок 4.1 – Плоский дизайн

Material Design также относится к Flat Design. Это визуальный язык, представлен в 2014 году Google, используется чаще всего в мобильных приложения. Пример использования Material Design'a можно увидеть во многих мобильных приложения Google (Play, Music, Books и т.д.), а также в Chrome OS.

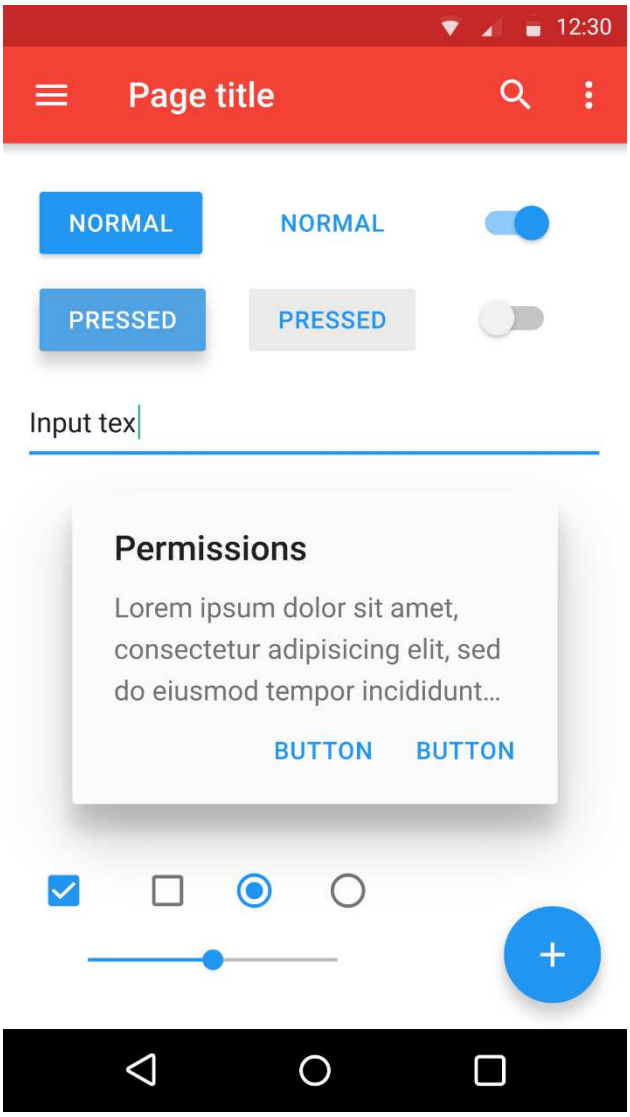

Рисунок 4.2 – Материальный дизайн от компании Google

Fluent Design – стиль оформления графического интерфейса, представленный компанией Microsoft в 2017 году. Данный стиль оформления фокусируется на 5 принципах: свет, глубина, движение, материал, масштаб.

- свет помогает пользователю ориентироваться в интерфейсе и показывать важные и ранее незаметные функции программы;

- иллюзия глубины, которая учитывает расположения компонентов в пространстве, приближает важные события, а также устанавливает большие размеры для важных элементов интерфейса;

- движение является индикацией динамического взаимодействия пользователя с интерфейсом;

- материал внешний вид элементов определяется не только палитрой, но и физическими свойствами материалов;

- масштаб элементы интерфейса адаптируются в зависимости от текущего использования – будь то экран мобильного телефона или система виртуальной реальности [13].

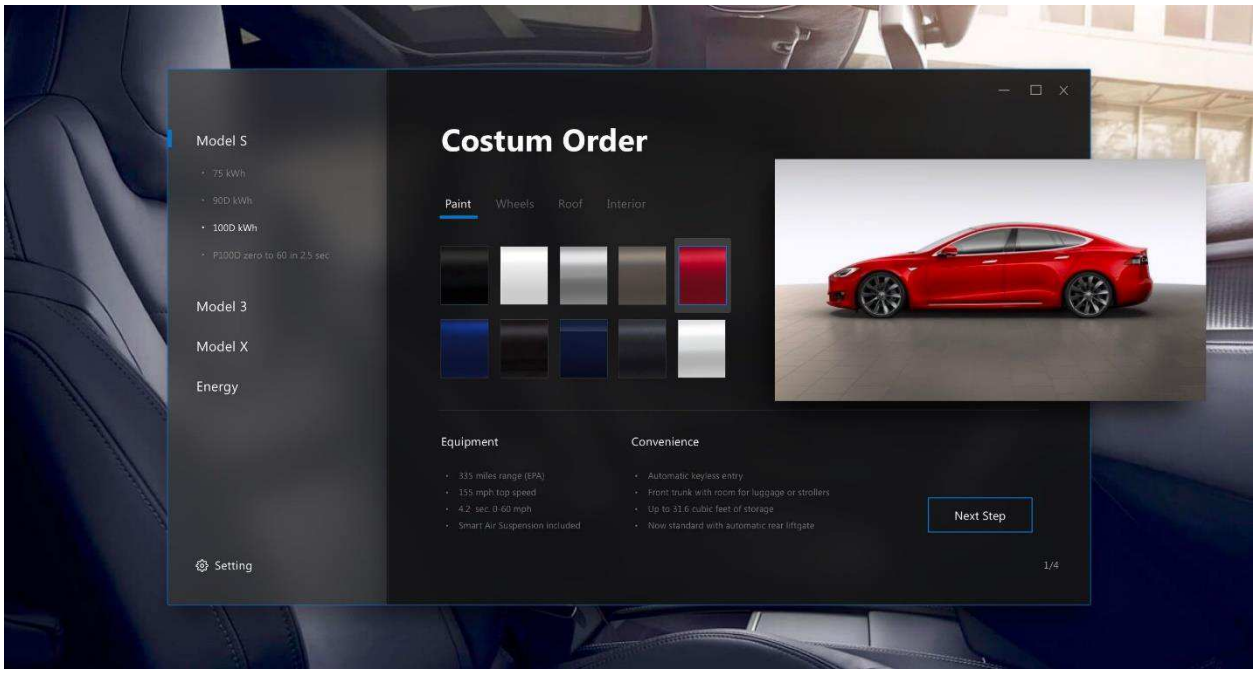

Рисунок 4.3 – Пример оформления в стиле Fluent Design

Ниже будут перечислены различные фреймворки, доступные для внедрения дизайна в проект.

Material Design Lite – официальный фреймворк Google для Web. Этот фреймворк предлагает только основные компоненты материального дизайна, здесь нет карусели, сетки, эффекта Параллакса и так далее [14].

MaterializeCss – фреймворк от компании Google. Данный фреймворк предлагает набор готовых к использованию компонентов в стиле

материального дизайна, содержит в себе сетку Bootstrap и крайне несложно внедряющийся в проект [15].

Fluent UI React – официальный фреймворк с открытыми исходным кодом для создания приложений в стиле Microsoft. Компоненты стиля настраиваются в соответствии CSS-in-JS.
Fabric Core – еще один официальный фреймворк от компании Microsoft. Это коллекция CSS классов и SCSS миксинов с открытым исходным кодом, которые предоставляют вам доступ к цветам, анимации, шрифтам, значкам и сетке в стиле Fluent Design.
В данном проекте был выбран фреймворк MaterializeCss из-за своей легкости подключения к проекту и остальных вышеперечисленных критериев, а именно: удобство пользования, интуитивно понятный интерфейс, лаконичность, приятная для восприятия палитра цветов.

3 Разработка системы

3.1 Структура приложения

3.1.1 Выбор шаблона проектирования

Выбор шаблона проектирования базировался на уже имеющихся и поддерживаемых средствах на выбранной платформе ASP.Net, поэтому был выбран шаблон проектирования MVC. На схеме, отображенной на рисунке 5 показаны основные компоненты и связи между ними.

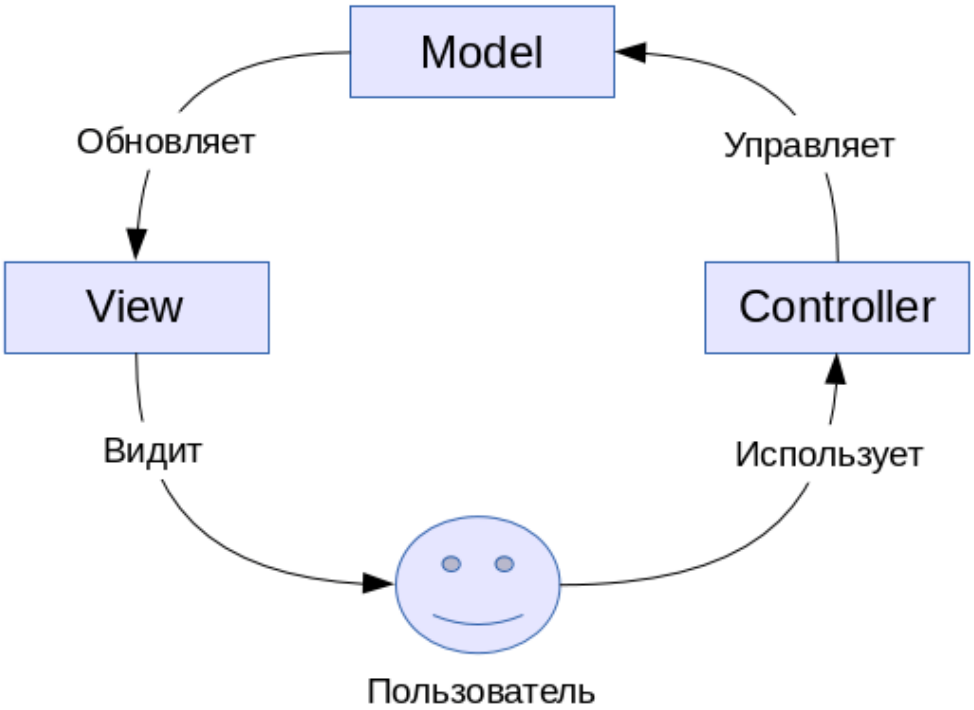

Рисунок 5 – Структура паттерна проектирования MVC

Преимущества данной архитектуры заключаются в том, что такое распределение обязанностей позволяет эффективно масштабировать приложение так как проще писать код, выполнять отладку и тестирование компонента (модели, представления или контроллера) с одним заданием. Гораздо работать с кодом, зависимости которого находятся в двух или более областях. К примеру, логика пользовательского интерфейса, как правило, подвергается изменениям чаще, чем бизнес-логика. Если код представления и бизнес-логика объединены в один объект, содержащий бизнес-логику, объект необходимо изменять при каждом обновлении пользовательского интерфейса. Это часто приводит к возникновению ошибок и необходимости повторно тестировать бизнес-логику после каждого незначительного изменения пользовательского интерфейса.

К тому же, представление и контроллер зависят от модели, но сама модель не зависит ни от контроллера, ни от представления. Это является одним из ключевых преимуществ разделения. Такое разделение позволяет создавать и тестировать модели независимо от их визуального представления [16].

3.1.2 Разработка контроллеров

Контроллер – это класс на языке C# с методами, свойствами и другими членами класса. Контроллер принимает входные данные, получает HTTP запрос и предоставляет правильный вывод на пришедшее сообщение.

Для распределения логики приложения с использованием модели MVC были разработаны следующие контроллеры:

HomeController – контроллер, отвечающий за представление домашней страницы.

Данный контроллер состоит из метода, который отображает главную страницу веб-сервиса с подсказками и инструкцией по пользованию сервисом.

RoomController – контроллер, отвечающий за отображение, добавление, изменение и удаление комнат вашего помещения.

RoomController состоит из следующих методов:

- метод отображения страницы по умолчанию Room. Данный метод возвращает все существующие в базе данных комнаты;

- метод создания нового элемента CreateRoom. Данный метод принимает на вход объект типа Room, тем самым позволяя создать новый элемент и добавляя его в таблицу Room при помощи метода db.Rooms.Add() будет генерироваться SQL-выражение CREATE, которое будет выполнено вызовом db.SaveChangesAsync(). Возвращает отображение всех комнат;

- метод отображения подробностей о комнате DetailsRoom. Данный метод принимает на вход идентификатор требуемой комнаты, возвращает объект Room представлению и выводит на экран все устройства, принадлежащие текущей комнате;

- метод для редактирования комнаты EditRoom. Таких методов два, т.к. один метод GET, а второй – POST. GET метод принимает на вход идентификатор объекта и возвращает форму с данными объекта, которые

пользователь может отредактировать. Метод POST получает отредактированные данные в виде объекта Room и с помощью обращения к базе данных при помощи метода db.Rooms.Update() будет генерироваться SQL-выражение UPDATE, которое будет выполнено вызовом db.SaveChangesAsync();

- метод удаления комнаты ConfirmDeleteRoom. Два метода, созданных для удаления обусловлены тем, что это GET метод и удаление через него небезопасно (можно написать все параметры в адресной строке и элемент будет удален из базы данных). Данный метод принимает на вход идентификатор комнаты, обращается к базе данных при помощи метода db.Rooms.FirstOrDefaultAsync() и извлекает первый элемент с входным идентификатором или значение по умолчанию в асинхронном режиме. Возвращает объект Room;

- метод удаления комнаты DeleteRoom. Данный метод на вход принимает идентификатор, ищет удаляемый объект в базе данных и удаляет его с помощью метода db.Rooms.Remove(), метод генерирует SQL-выражение DELETE, которое выполняется вызовом db.SaveChangesAsync(). Возвращает страницу по умолчанию.

DeviceController – контроллер, отвечающий за отображение, создание, изменение и удаление устройств из базы данных.

Данный контроллер состоит из таких методов, как:
- отображение страницы по умолчанию с выводом всех устройств, которые принадлежат выбранной комнате;

- метод создания нового элемента CreateDevice. Данный метод принимает на вход объект типа Device при помощи метода db. Devices.Add() будет генерироваться SQL-выражение CREATE, которое будет выполнено вызовом db.SaveChangesAsync(). Возвращает отображение всех устройств;

- метод отображения подробностей о комнате DetailsDevice. Данный метод принимает на вход идентификатор требуемого устройства,

возвращает объект Device представлению и выводит на экран все данные об устройстве, принадлежащие текущему устройству.

- метод для редактирования устройства EditDevice. Таких методов два, т.к. один метод GET, а второй – POST. GET метод принимает на вход идентификатор объекта и возвращает форму с данными объекта, которые пользователь может отредактировать. Метод POST получает отредактированные данные в виде объекта Device и с помощью обращения к базе данных при помощи метода db.Devices.Update() будет генерироваться SQL-выражение UPDATE, которое будет выполнено вызовом db.SaveChangesAsync().

- метод удаления комнаты ConfirmDeleteDevice. Два метода, созданных для удаления обусловлены тем, что это GET метод и удаление через него небезопасно (можно написать все параметры в адресной строке и элемент будет удален из базы данных). Данный метод принимает на вход идентификатор устройства, обращается к базе данных при помощи метода db. Devices.FirstOrDefaultAsync() и извлекает первый элемент с входным идентификатором или значение по умолчанию в асинхронном режиме. Возвращает объект Device.

- метод удаления комнаты DeleteDevice. Данный метод на вход принимает идентификатор, ищет удаляемый объект в базе данных и удаляет его с помощью метода db.Devices.Remove(), метод генерирует SQL-выражение DELETE, которое выполняется вызовом db.SaveChangesAsync(). Возвращает страницу по умолчанию.

ScriptController – контроллер, отвечающий за пользовательские сценарии.

- метод отображения страницы по умолчанию Script. Данный метод возвращает все существующие в базе данных сценарии

- метод создания нового элемента CreateScript. Данный метод принимает на вход объект типа Script, тем самым позволяя создать новый

элемент и добавляя его в таблицу Script, возвращает отображение всех сценариев.

- метод отображения подробностей DetailsScript. Данный метод принимает на вход идентификатор требуемого сценария, возвращает объект Script представлению и выводит на экран всю расширенную информацию, принадлежащую текущему сценарию.

- метод для редактирования сценария EditScript. Таких методов два, т.к. один метод GET, а второй – POST. GET метод принимает на вход идентификатор объекта и возвращает форму с данными объекта, которые пользователь может отредактировать. Метод POST получает отредактированные данные в виде объекта Room и с помощью обращения к базе данных при помощи метода db.Scripts.Update() будет генерироваться SQL-выражение UPDATE, которое будет выполнено вызовом db.SaveChangesAsync().

- метод удаления сценария ConfirmDeleteScript. Два метода, созданных для удаления обусловлены тем, что это GET метод и удаление через него небезопасно (можно написать все параметры в адресной строке и элемент будет удален из базы данных). Данный метод принимает на вход идентификатор сценария, обращается к базе данных при помощи метода db.Scripts.FirstOrDefaultAsync() и извлекает первый элемент с входным идентификатором или значение по умолчанию в асинхронном режиме. Возвращает объект Script.

- метод удаления комнаты DeleteScript. Данный метод на вход принимает идентификатор, ищет удаляемый объект в базе данных и удаляет его с помощью метода db.Scripts.Remove(), метод генерирует SQL-выражение DELETE, которое выполняется вызовом db.SaveChangesAsync(). Возвращает страницу по умолчанию.

- метод выполнения сценария ExecuteScript. Данный метод на вход принимает идентификатор, ищет объект в базе данных и отправляет через MQTT протокол сообщение в топик устройства. Возвращает объект Script.

3.1.3 Разработка представлений

Представление в ASP.NET MVC может содержать не только стандартный код HTML, но и также вставки кода на языке C#. Для обработки кода, содержащего как элементы HTML, так и конструкции C#, используется синтаксис разметки для внедрения серверного кода на веб-страницы представлений.

При вызове метода View контроллер не производит рендеринг представления и не генерирует разметку HTML. Контроллер готовит данные и выбирает, какое представление надо возвратить в качестве объекта ViewResult. Затем уже объект ViewResult обращается к механизму представления для рендеринга представления в выходной ответ.

По умолчанию в ASP.NET MVC Core используется один механизм представлений - Razor. Цель Razor – определить переход от разметки HTML к коду C#.

Каждый из разработанных в системе контроллеров содержит, как правило, несколько представлений, передаваемых пользователю в зависимости от запрашиваемого метода контроллера.

В разработанных представлениях благодаря технологии представлений Razor мы можем комбинировать язык C# с языком гипертекстовой разметки HTML, динамически перестраивать представление в зависимости от входных данных, выводить информацию из базы данных в удобном для пользователей виде [17].

Первое представление создается автоматически с проектом – это Index.cshtml в также созданной по умолчанию папке Home. В данном

представлении содержится информация, объясняющая пользователю работу с системой. Данное представление относится к контроллеру Home.

В представлении Room отображены все внесенные в базу данных комнаты и функции, которые вы можете с ними сделать: изменить и удалить комнату. Так же мы можем добавить комнату в базу данных на этой же странице. Данное представление относится к контроллеру Room. В свою очередь, для каждой функции создано собственное представление, для вывода более подробной информации – это представление DetailsRoom.cshtml, для редактирования комнаты – EditRoom.cshtml, для создания – CreateRoom.cshtml и для удаления – ConfirmDeleteRoom.cshtml.

Представление Device полностью повторяет структуру Room. В представлении Device отображены все внесенные в базу данных устройства и функции, которые вы можете с ними сделать: изменить и удалить устройство. Так же мы можем добавить устройство в базу данных на этой же странице. Данное представление относится к контроллеру Device. В свою очередь, для каждой функции создано собственное представление, для вывода более подробной информации – это представление DetailsDevice.cshtml, для редактирования комнаты – EditDevice.cshtml, для создания – CreateDevice.cshtml и для удаления – ConfirmDeleteDevice.cshtml.

Аналогично и с представлением Script, в котором отображаются все функции, относящиеся к сценариям.

3.1.4 Разработка программного кода аппаратной части

Приложение для связи аппаратной и программной частей реализуется на микроконтроллерной системе NodeMCU ESP8266 при помощи Arduino IDE на языке C++. Приложение использует следующие основные методы:

В методе setup назначаются все начальные параметры, такие как: подключенные датчики и исполнительные устройства, подключение к сети Wi-Fi при помощи функции setupWifi и подключение к MQTT брокеру. Функция setupWifi принимает SSID и пароль Wi-Fi сети.

Метод loop – основной цикл программы, содержит в себе вызов функций reconnectWifi, callback, sendData. Функция reconnectWifi осуществляет переподключение к сети, если произошло отключение от выбранной сети Wi-Fi, callback осуществляет прием сообщений из определенного MQTT топика для выполнения их на исполнительных устройствах и sendData выполняет отправку данных с датчиков на сервер в формате JSON.

4 Руководство пользователя

На главной странице отображены подсказки для пользователя относительно интерфейса, что можно увидеть на рисунке 6.1. Сверху, на панели навигации расположены названия страниц, кликнув по которым, можно перейти на определенные представления, которые обрабатывают одноименные контроллеры.

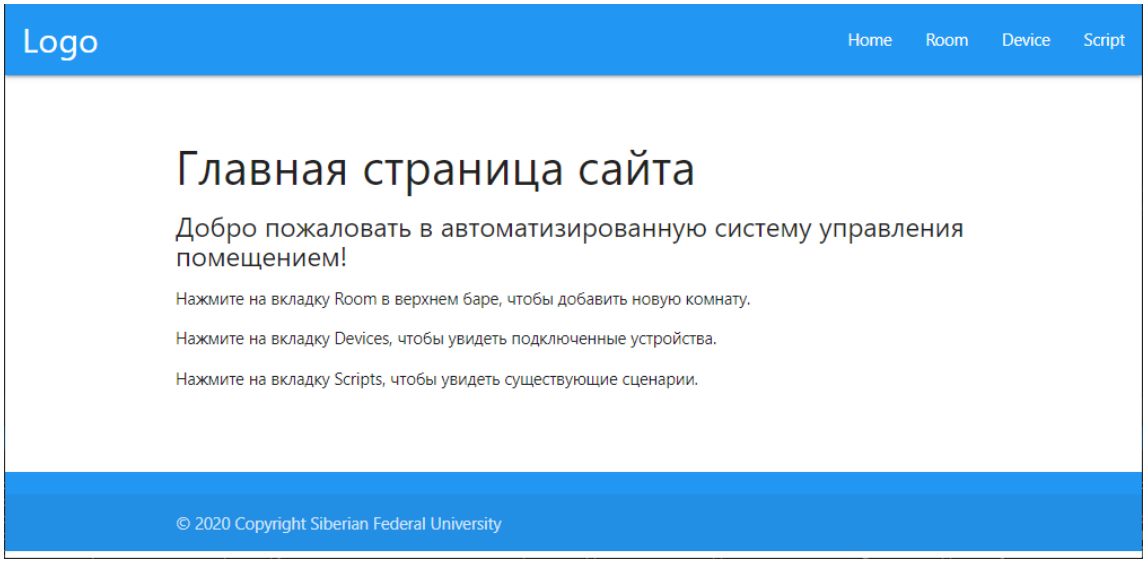

Рисунок 6.1 – Представление Home

Веб-интерфейс также оптимизирован и для мобильных устройств, что отображено на рисунке 6.2.

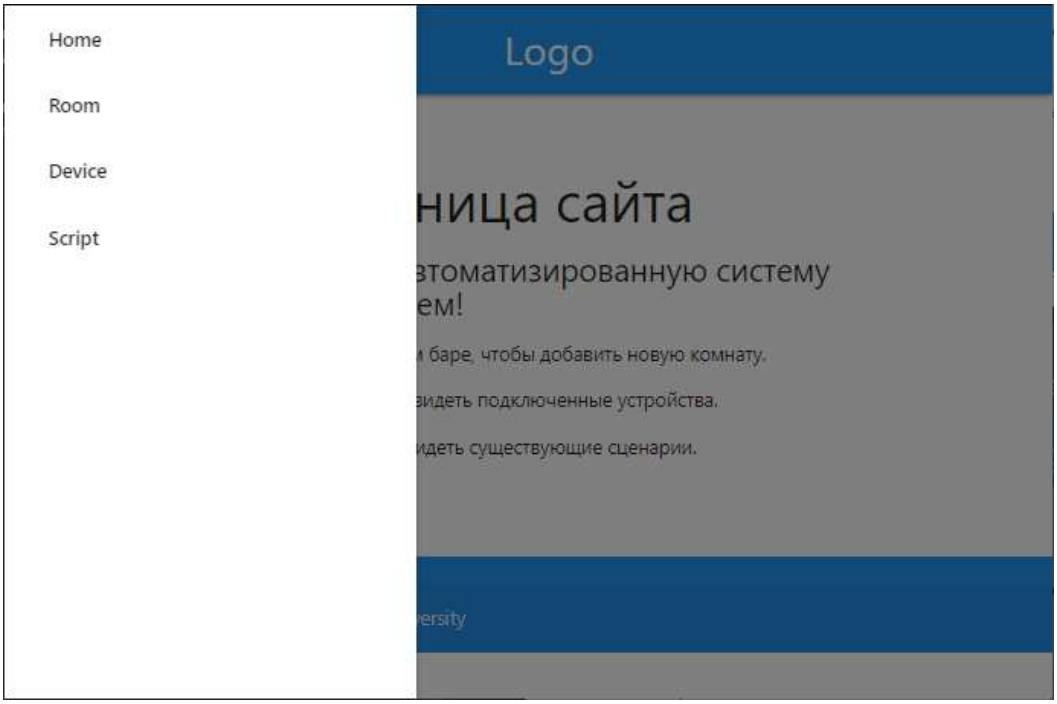

Рисунок 6.2 – Адаптивное представление Home

Представление комнат выглядит как на рисунке 6.3. На данном представлении можно создать новую комнату, показать подробности о существующих комнатах, изменить информацию о существующей комнате и, наконец, удалить выбранную комнату. Создание новой комнаты находится в верхней части страницы. При нажатии на иконку плюса зеленого цвета на карточке «Добавить» открывается соответствующее представление, которое отвечает за добавление новой комнаты.

На каждой карточке есть три кнопки. При нажатии на первую кнопку происходит действие, которое отображает представление с устройствами, которые относятся к данной комнате. При нажатии на вторую кнопку можно изменить информацию о данной комнате и произойдут изменения в соответствующей таблице базы данных. Если нажать на последнюю кнопку, можно удалить комнату навсегда.

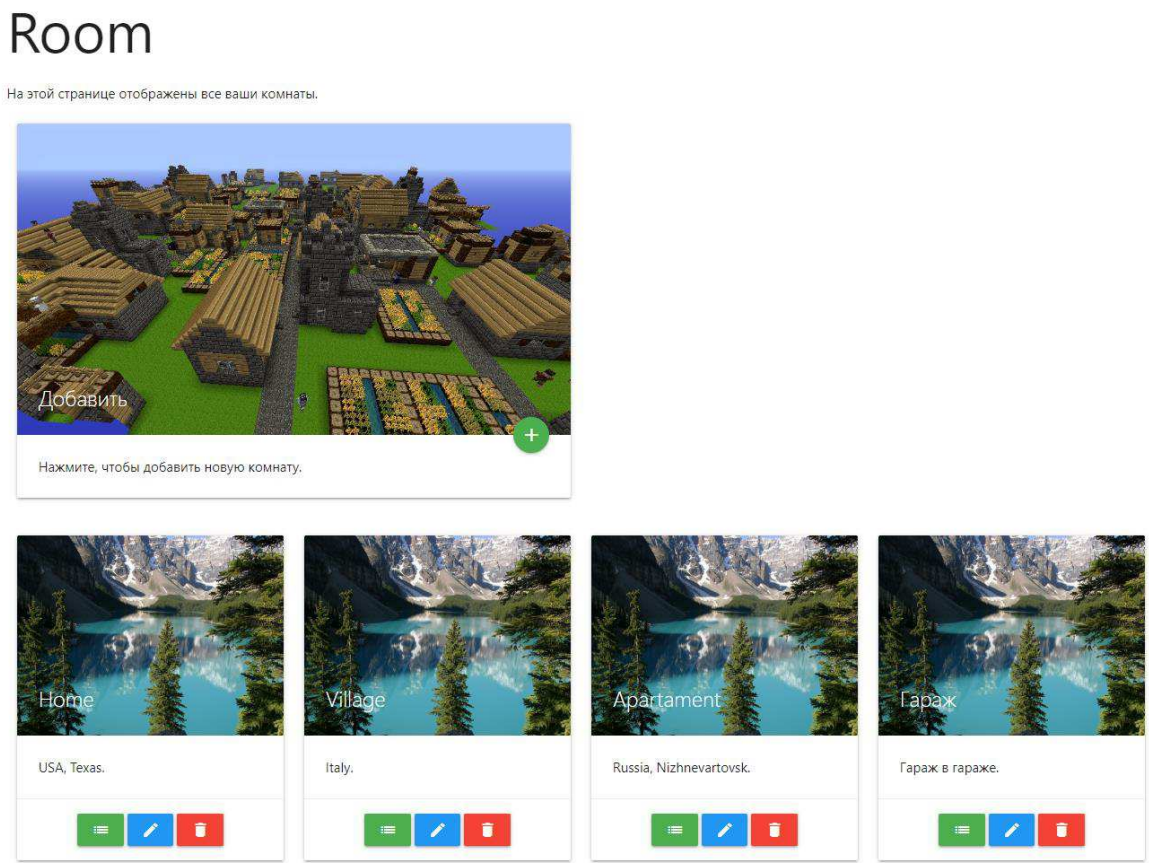

Рисунок 6.3 – Представление Room

Оптимизация под мобильный интерфейс представлена на рисунке 6.4.

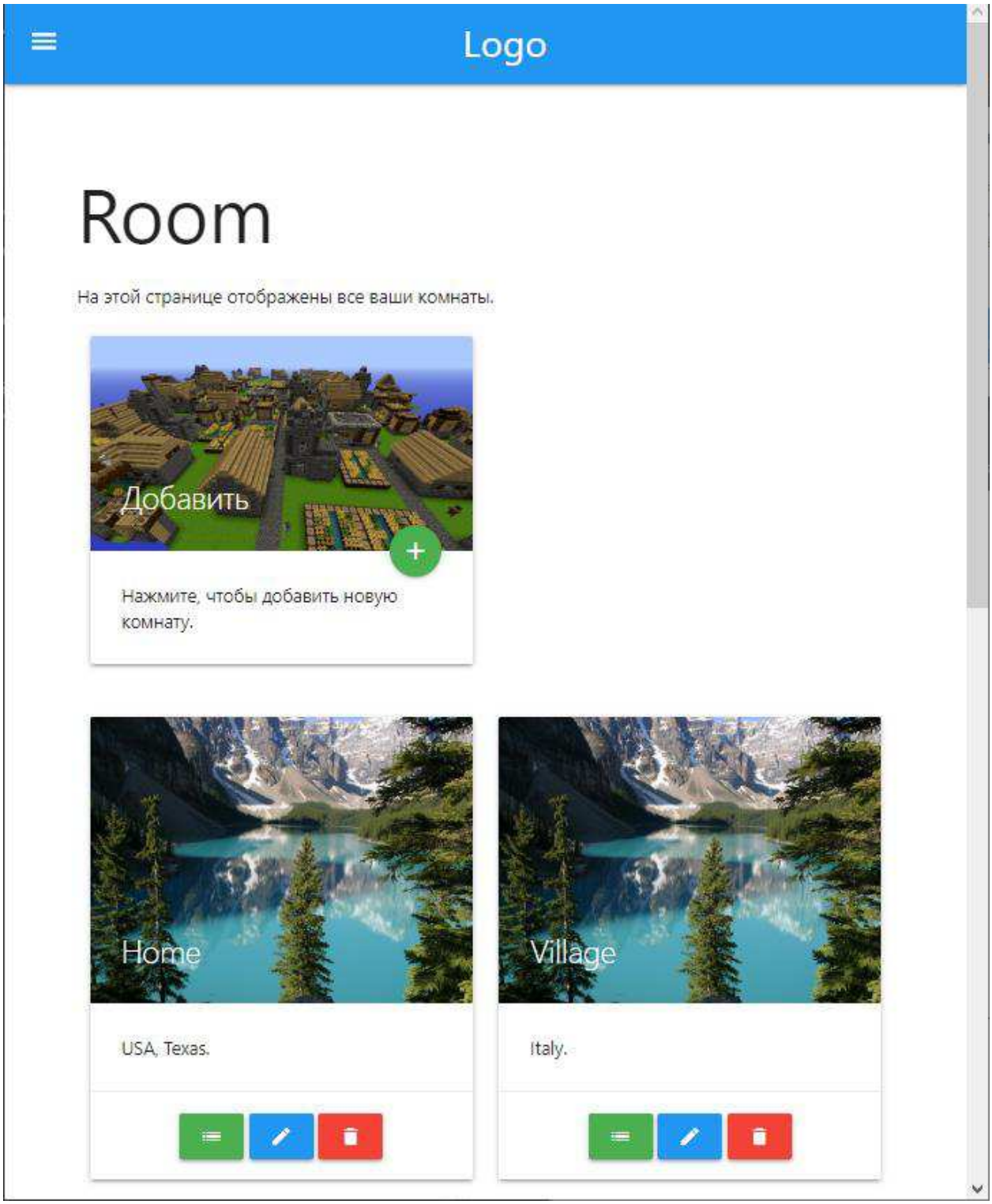

Рисунок 6.4 – Адаптивное представление Room

Представление устройств выглядит как на рисунке 6.5. На данном представлении можно добавить новое устройство в определенную комнату,

показать подробности о существующих устройствах, изменить информацию о существующей устройстве и, наконец, удалить выбранную устройство. Создание нового устройства находится в верхней части страницы. При нажатии на иконку плюса зеленого цвета на карточке «Добавить» открывается соответствующее представление, которое отвечает за добавление нового устройства.

На каждой карточке есть три кнопки. При нажатии на первую кнопку происходит действие, которое отображает представление более подробной информацией об устройствах. При нажатии на вторую кнопку можно изменить информацию о данном устройстве и произойдут изменения в соответствующей таблице базы данных. Если нажать на последнюю кнопку, можно удалить устройство навсегда.

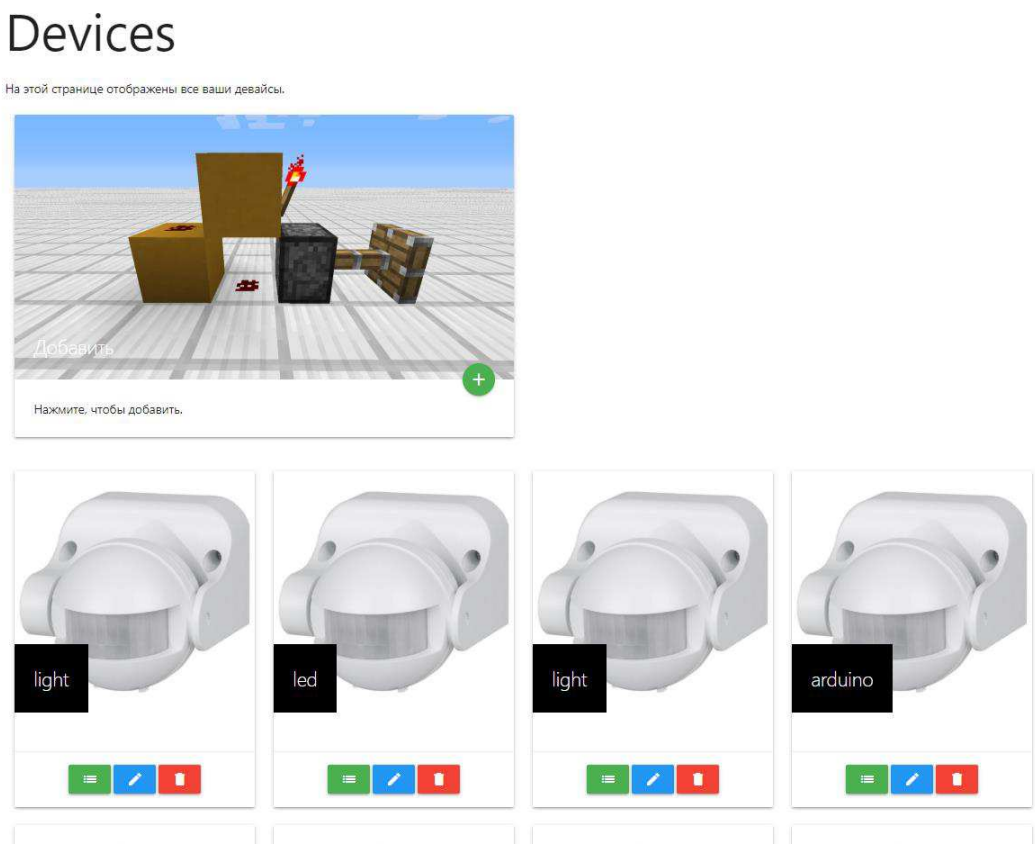

Рисунок 6.5 – Представление Devices

Адаптивное представление отображено на рисунке 6.6.

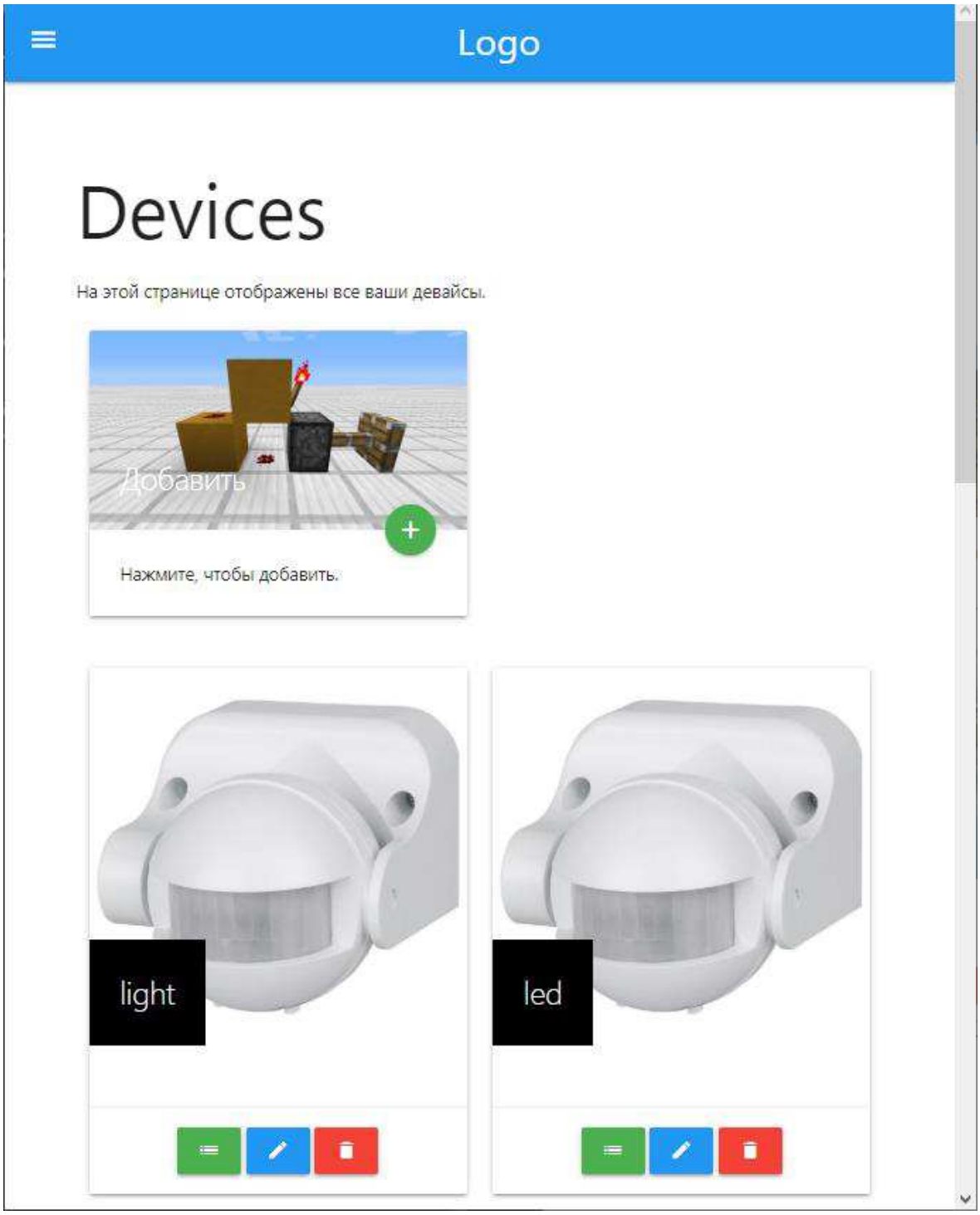

Рисунок 6.6 – Адаптивное представление Devices

5 Практическое применение системы

Данная система имеет потенциальную расширяемость модулей. С помощью такой системы можно оборудовать, как и собственный дом, так и учебную аудиторию различными сценариями поведения. Можно перечислить такие модули как: модуль температуры и влажности, модуль устройств и модуль безопасности.

Модуль температуры и влажности может включать в себя:

- поддержание температурного режима на основе расчета теплопотерь комнаты или помещения;
- использование алгоритмов с прогнозированием для учета ночного снижения температуры и других произвольных факторов для систем отопления и теплого пола;
- поддержание оптимального значения влажности в помещениях;
- организация проветривания для помещения с заданной периодичностью.

Модуль, отвечающий за устройства, может включать в себя:

- удаленное управление устройствами;
- запуск и включение заданных устройств по таймеру.

Модуль освещения может включать в себя:

- удаленное управление освещением помещения;
- уличное освещение ночью или освещение в темное время суток, настроенное на датчик движения.
- автоматическое управление жалюзи для контроля уровня освещения;

Модуль безопасности может включать в себя:

- видеонаблюдение для контроля в критической ситуации;
- срабатывание сигнализации при несанкционированном проникновении в помещение или при других критических ситуациях;

- установка датчика затопления на полу для сигнализирования о критической ситуации;

- датчики закрытия дверей для предотвращения несанкционированного доступа;

- поддержание нормальной концентрации опасных для человека газов в воздухе.

Таким образом, данная система является универсальной и не ограничена для определенных областей применений. Вдобавок к этому, можно придумать большое количество сценариев и взаимодействий датчиков, исполнительных устройств и сервера.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы были рассмотрены фабричные системы автоматизации, на основе этого был выделен общие функциональные требования.

В ходе выполнения работы были спроектированы и разработаны системные модули, отвечающие за взаимодействие управляющего устройства с сервером, взаимодействие пользователя с сервером. В пояснительной записке проведен анализ и обоснование выбора средств реализации, разработана архитектура и функциональные требования к системе.

Подводя итоги, можно сказать, что проект может использоваться как замена готовым системам автоматизации при условии, что у вас есть отладочная плата NodeMCU ESP8266, нужные датчики и исполнительные устройства.

Данная система обладает модульностью и расширяемостью, т.е. ее можно дополнять и улучшать в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 СТО 4.2-07-2014 "Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности". – Взамен СТО 4.2-07-2012; введ. 30.12.2013. – Красноярск: СФУ, 2013. – 60 с

2 Домашняя автоматизация [Электронный ресурс] // «Википедия –свободная энциклопедия». – Режим доступа: https://ru.wikipedia.org/wiki/Домашняя_автоматизация

3 Xiaomi Mi Home [Электронный ресурс] // «Умный дом Xiaomi». – Режим доступа: https://xiaomi-smarthome.ru/xiaomi-mi-home/

4 Приложение Google Home – центр управления устройствами умный дом [Электронный ресурс] // Голосовые помощники и умный дом. – Режим доступа: https://voiceapp.ru/articles/google-home-app

5 Что мы знаем об Amazon Alexa? [Электронный ресурс] // Хабр – сообщество IT-специалистов. – Режим доступа: https://habr.com/ru/company/unet/blog/371435/

6 HomeKit - что это такое? Умный дом от Apple [Электронный ресурс] // Яндекс. – Режим доступа: https://yandex.ru/turbo/s/sprut.ai/client/article/1039

7 Коржов, В. В. Многоуровневые системы клиент-сервер / В. В. Коржов // Сети/Network world – 1997. – №6.

8 Дейт, Дж. К. Введение в системы баз данных: учебное пособие / Дж. К. Дейт. – Москва, Санкт-Петербург, Киев: Вильямс, 2005. – 41 с.

9 Простым языком об HTTP [Электронный ресурс] // Хабр – Режим доступа: https://habr.com/ru/post/215117/

10 Протокол FTP [Электронный ресурс] // «Википедия – свободная энциклопедия». – Режим доступа: https://ru.wikipedia.org/wiki/FTP

11 Протокол MQTT [Электронный ресурс] // MQTT. – Режим доступа: http://mqtt.org/

12 «Плоский» дизайн – Flat Design [Электронный ресурс] // Powerbranding. – Режим доступа: http://powerbranding.ru/design/flat-design-june13/

13 Fluent Design System [Электронный ресурс] // Fluent Design System. – Режим доступа: https://www.microsoft.com/design/fluent/#/

14 Material Design Lite [Электронный ресурс] // Material Design Lite. – Режим доступа: https://getmdl.io/

15 Materialize [Электронный ресурс] // A modern responsive front-end framework based on Material Design. – Режим доступа: https://materializecss.com/

16 MVC ASP.Net [Электронный ресурс] // Microsoft. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-3.1

17 ASP.NET Core Identity [Электронный ресурс] // Метанит – сайт о программировании – Режим доступа: https://metanit.com/sharp/aspnet5/7.2.php

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

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« _____» _____ 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специальности

Автоматизированная система управления помещением
тема

Руководитель	_____	профессор, канд. техн. наук	Б.И. Борде
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	23.06.20		А.Г. Будажапова
	подпись, дата		инициалы, фамилия
Нормоконтроллер	_____	профессор, канд. техн. наук	Б.И. Борде
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2020