

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

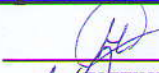
Специальность:	<u>09.05.01</u>	<u>«Применение и эксплуатация</u>
	<u>автоматизированных систем</u>	<u>специального</u>
	<u>назначения»</u>	
Специализация:	<u>«Эксплуатация вычислительных машин, комплексов,</u>	
	<u>систем и сетей специального назначения»</u>	

ДИПЛОМНЫЙ ПРОЕКТ

на тему

Система управления распределенной сетью устройств
интернета вещей

Студент

 Новиков Виктор Викторович
(подпись, дата) (ФИО полностью)

Руководитель

 Федюнин Р.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

 Кучин А.В.
(подпись, дата) (фамилия, инициалы)+

Рецензент, доцент каф.ВМиС Пенз.ГТУ
(должность, место работы)

 Печерский С.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 10.06.2020 г. № 13)

Заведующий кафедрой

 Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от
19.06.2020 г. № 10)

Секретарь ГЭК

 Гурин Е.И.
(фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

«Утверждаю»
Заведующий кафедрой
 Митрохин М.А.
«30» апреля 2020г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Новиков Виктор Викторович гр. 15BO2 факультета вычислительной техники.
2. Тема работы: «Система управления распределенной сетью устройств интернет вещей».

Тема утверждена приказом ПГУ № 0335а/о от 30.04.2020 г.

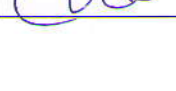
3. Руководитель работы к.т.н., доцент каф. ВТ ПГУ Федюнин Р.Н.
4. Задание на работу (назначение разработки, исходные данные и т.п.).

Система предназначена для мониторинга за контролерами обработки данных.

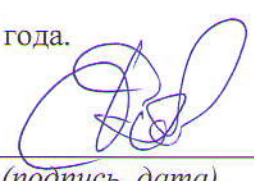
5. Перечень подлежащих разработке вопросов.

В процессе работы определен список требований программного обеспечения, разработаны функции, позволяющие приложению устанавливать соединение с устройством и обрабатывающие входные данные.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	09.05.20	
2.	Выбор стека технологий и компонентов для реализации приложения	10	16.05.20	
3.	Составление UML схемы приложения	20	16.05.20	
4.	Написание программы	10	23.05.20	
5.	Отладка системы	10	26.05.20	
6.	Моделирование и поиск уязвимостей, исправление ошибок в работе системы	10	31.05.20	
7.	Оформление выпускной работы	30	06.06.20	

Дата выдачи задания 30 «апреля» 2020 года.

Руководитель дипломного проекта  Федюнин Р.Н.
(подпись, дата)

Задание к исполнению принял студент  30.04.2020 Новиков В.В.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Работа содержит 115 листов, 51 рисунок, 22 таблицы, 6 источников, 3 приложения.

WEB API, БАЗА ДАННЫХ, ОС ANDROID, HTTP ПРОТОКОЛ, IoT, WEB SOCKET.

Цель работы – на базе распространенных протоколов Интернета вещей реализовать систему мониторинга за контроллерами обработки данных.

Объектом разработки является программное приложение, обеспечивающее управление средой интернет вещей.

В результате проведенной работы должно быть разработано приложение, удовлетворяющее поставленной задаче, а именно обеспечивающее управление средой интернета вещей, состоящей из различных устройств и облачного сервиса.

Данное приложение рекомендуется использовать для массового продвижения товаров и услуг.

					ПГУ 1.09.05.01 010 001 ПЗ						
Изм.	Лист	№ докум.	№	Подпись	Дата	Система управления распределенной сетью устройств интернета вещей. Пояснительная записка			Лит.	Лист	Листов
Разраб.		Новиков В.В.									
Провер.		Федюнин Р.Н.								2	99
Н. Контр.		Кучин А.В.			ФВТ, каф. ВТ гр. 15В02						
..											
Утверд.		Митрохин М.А.									

СОДЕРЖАНИЕ

Введение.....	5
1 Система управления распределенной сетью устройств интернета вещей.....	7
1.1 Анализ предметной области.....	7
1.2 Протоколы взаимодействия в IoT инфраструктуре.....	10
1.3 Выбор базы данных для управления системой.....	15
1.4 Обзор конечных устройств.....	18
1.5 Общая архитектура аппаратно-программной системы.....	20
1.6 Выводы по главе.....	24
2 Реализация системы.....	25
2.1 Реализация backend части приложения.....	25
2.2 Реализация rest api приложения.....	35
2.3 Связь IoT сети и серверной части системы.....	42
2.4 Реализация frontend части приложения.....	48
2.5 Выводы по главе.....	57
3 Технология разработки приложений os android в ide eclipse.....	58
4 Техничко-экономическое обоснование разработки сайта «сервис совместного поиска попутчиков».....	63
4.1 Расчет себестоимости программного продукта.....	63
4.2 Эффективность разработки сайта.....	68
5 Безопасность жизнедеятельности. Организация рабочего места программиста.....	71
5.1 Характеристика условий труда рабочего места программиста.....	71
5.2 Требование к рабочему помещению программиста.....	72
5.2.1 Окраска помещений	72
5.2.2 Освещение рабочего места	73

5.2.3 Параметры микроклимата на рабочем месте.....	75
5.2.4 Требования к шуму и вибрации в рабочих помещениях	76
5.2.5 Режим труда программиста.....	78
5.3 Расчет освещенности рабочего места программиста.....	80
5.4 Расчет уровня шума на рабочем месте.....	82
Заключение.....	85
Список использованных источников.....	86
Приложение А.....	87
Приложение Б.....	100
Приложение В.....	107

ВВЕДЕНИЕ

В сегодняшнем мире, в котором быстрыми темпами набирают свою популярность умные устройства (IoT) «Интернет вещей», выполняющие роль помощников в решении повседневных рутинных задач в различных областях применения.

Все умные устройства, взаимосвязанные друг с другом, работают через выход в глобальную сеть - интернет.

Количество подключенных датчиков и устройств Интернета вещей в мире в 2018 году составила 21 млрд., а к 2022 году превысит 50 млрд. [1].

Интернет вещей — это совокупность встроенных систем, сетей беспроводных датчиков, систем управления и средств автоматизации. Все это позволяет реализовать подключение фабрик, интеллектуальных магазинов, умных домов и городов, а также носимых устройств.

С помощью технологий «Интернет вещей» возможно, преобразовать свой бизнес за счет полученных на основе данных аналитических сведений, усовершенствованных рабочих процессов, новых сфер деятельности и более эффективного использования материалов.

Технологии Интернета вещей продолжают развиваться. Ежегодно появляется бесчисленное количество поставщиков услуг, разнообразные платформы и миллионы новых устройств. Сейчас для начала работы в экосистеме Интернета вещей разработчикам нужно принять множество решений.

Цель работы на базе распространенных протоколов Интернета вещей реализовать систему мониторинга за контроллерами обработки данных.

Интернет вещей — это целая экосистема, которая содержит в себе интеллектуальные устройства оснащенными сенсорами (датчиками), обеспечивающие удаленное управление, хранение, передачу и безопасность

данных. IoT представляет собой решения в различных прикладных областях таких, как здравоохранение, страхование, охрана труда, логистика, экология и т.д.

Для раскрытия полного потенциала использования IoT устройств необходимо решить множество проблем связанные со стандартами обеспечение сохранности данных, архитектуру построение экосистемы, каналы и протоколы подключения устройств. Сегодня в мире крупные организации, такие как NIST, IEEE, ISO/IEC и другие делает огромные усилия в решении вопросов стандартизации, безопасности и архитектуры разрабатываемых устройств. Анализ последних исследовательских работ в области решении вопросов обработки данных IoT показывает, что эффективным способом сбора и хранения данных является гибридный подход, основанный на организации системы в рамках собственных и облачных хранилищ.

Разработка и применение способов обработки, передачи и хранения данных IoT- устройств является сложной задачей.

В связи с этим построение системы распределенной обработки и хранения информации IoT - устройств является целью данной дипломной работы.

1 Система управления распределенной сетью устройств интернета вещей

В данной главе предлагается исследование архитектурных решений для реализации проекта объединения устройств, предназначенных для сбора передачи и обработки данных в единую информационную сеть на базе двух подходов: M2M и IoT.

1.1 Анализ предметной области

Существует три вида реализации аппаратно-программных платформ: M2M-подключения, интернет вещей (Internet of Things, IoT) и концепция интернета сервисов (Internet of Services, IoS).

Понятия «M2M-устройство» и «устройство интернета вещей» зачастую смешивают, хотя они отнюдь не тождественны.

Принципиальная разница состоит в том, что понятие «M2M-устройство» охватывает традиционные проприетарные механизмы объединения устройств в единую сеть — это сетезависимые беспроводные устройства (рисунки 1.1 – 1.2).

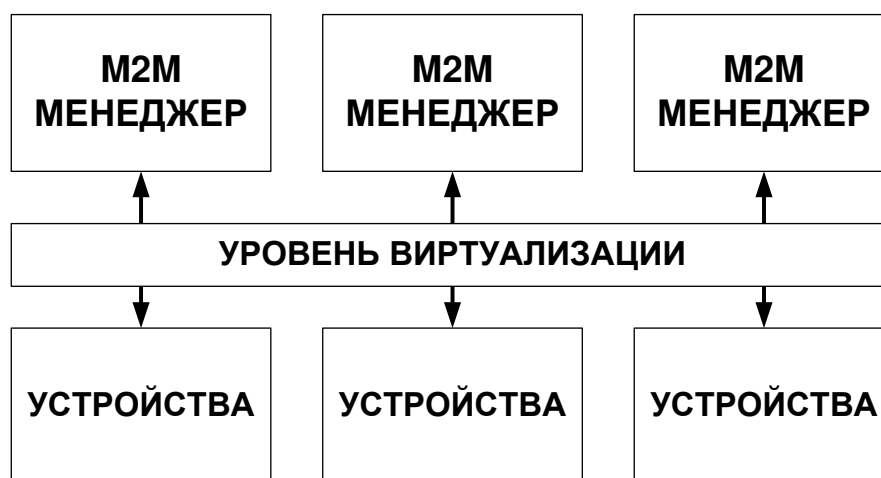


Рисунок 1.1 - Сравнение M2M и IoT подходов

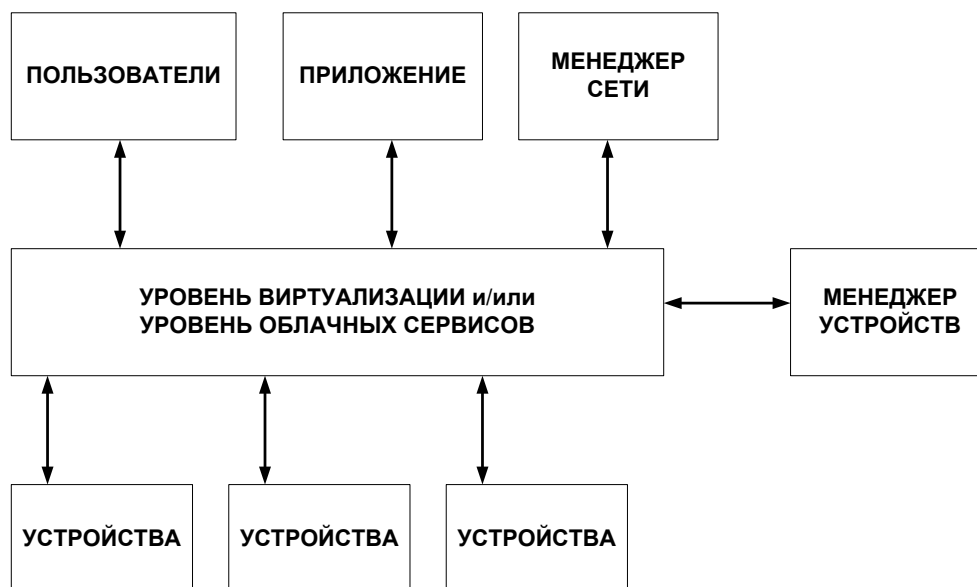


Рисунок 1.2 - Сравнение M2M и IoT подходов

Устройства интернета вещей (IoT) – это устройства, имеющие возможность через свободное IP-подключение (на физическом уровне, как правило Wi-Fi) взаимодействовать с различными системами телеметрии и телеуправления, реализованными как облачные и/или онлайн-сервисы. Интернет вещей — это гибкие решения без привязки к типам устройств и протоколов физического и канального уровня.

Как отмечалось выше, IoT-устройства – это не просто обычные устройства, а устройства с IP-подключением. Однако главное их отличие в том, что функциональные возможности таких устройств в большей степени определяются не аппаратно, а программно (причем аппаратно-независимо), т.е. такие устройства являются «умными». А аппаратная часть реализуется на принципах модульности и открытой архитектуры.

Очевидно, что традиционные M2M-системы ввиду своей низкой эффективности и недостаточной гибкости будут уступать место облачным системам и IoT-устройствам по мере расширения номенклатуры таких устройств, снижения их стоимости (за счет массовости производства) и появления облачных сервисов, способных реализовать оптимизационное управление ими.

Таким образом, IoT нацелен на создание открытой, масштабируемой, основанной на стандартах, ориентированной на обслуживание любого типа устройств сети, в которой очень большое количество узлов может взаимодействовать друг с другом и с единым центром в виде сервера и/или облачного сервиса.

Для этого система должна обладать следующими возможностями:

Возможность подключения здесь и сейчас – конечные устройства в системе имеют ограниченное энергопотребление и очень критичны к скорости передачи данных и не могут долго ждать возможности подключения к серверу.

Поддержка общедоступных протоколов прикладного уровня: IoT в отличие от M2M поддерживает как протоколы низкого уровня, так и протоколы верхнего уровня (RESTAPI и HTTP).

Поддержка IP соединений, возможность устройств взаимодействовать по IPадресам.

Виртуальность конечных объектов, наличие слоя абстракции, который скрывает особенности конкретных серий устройств.

Web API – наличие фреймворков и API по работе облачных, клиентских, серверных слоев с устройствами и/или их виртуальными объектами.

Целью дипломного проекта является построение горизонтальной, универсальной, масштабируемой архитектуры Интернета вещей, в которой большое количество устройств и сервер взаимодействуют друг с другом посредством единого фреймворка (виртуального слоя).

В связи, с чем мониторинг и контроль устройств, мониторинг и управление сетью, идентификация и обнаружение устройств — это задачи, для решения которых проектируется данная инфраструктура IoT.

1.2 Протоколы взаимодействия в IoT инфраструктуре

Система управления устройствами IoT требует беспроводной связи сервера с устройствами и с другими объектами в сети.

Так как планируется реализовать клиент серверное приложение с трехзвенной архитектурой, то ожидается, что сервер будет поддерживать HTTP протокол для бесшовной интеграции с облачными и/или виртуальными объектами сети.

Поскольку HTTP не поддерживает прямую связь в реальном времени, HTTP может быть заменен протоколом WebSocket. Ниже представлены протоколы, которые может поддерживать разрабатываемая система.

HTTP протокол – это проверенная технология построения клиент серверного приложения.

Основной плюс протокола: поддерживается большим количеством клиентов, серверов и устройств. Позволяет передавать файлы, текст, видео и иную информацию.

Основой взаимодействия является URL (например, следующего вида):

- PROTOCOL://HOST_NAME:PORT;
- RESOURCE_PATH?RESOURCE_INPUT.

Достоинства протокола:

- простота. Протокол HTTP позволяет легко создавать необходимые клиентские и серверные приложения.
- расширяемость. Исходные возможности протокола можно расширить, внедрив свои собственные заголовки, с помощью которых можно добиться необходимой функциональности, которая может потребоваться при решении специфических задач. Совместимость с другими серверами и клиентами от этого никак не пострадает: они будут игнорировать неизвестные им заголовки.
- распространённость. Протокол поддерживается в качестве клиента многими программами и есть возможность выбирать среди

хостинговых компаний с серверами HTTP. По этой причине протокол широко используют для решения различных задач. Кроме этого, существует документация на многих языках, что существенно облегчает работу с протоколом.

Для того, чтобы указать серверу на то, какое действие требуется произвести с ресурсом, используется тип HTTP-запроса, который также называется HTTP метод. Существующие методы покрывают все потребности манипуляции конечными устройствами (таблица 1.1).

Таблица 1.1 Методы HTTP протокола

Метод	Описание
GET	используется для получения (или чтения) представления ресурса
PUT	используется для предоставления возможности обновления ресурса
POST	используется для создания новых ресурсов
DELETE	используется для удаления ресурса, идентифицированного конкретным URI (ID)
TRACE	применяется для диагностики, он позволяет видеть клиенту, что происходит в каждом звене цепочки между клиентом, сервером и конечным получателем, для этого существует специальное поле Via

Схема обмена сообщениями по HTTP представлена на рисунке 1.3.

WebSocket протокол - технология, которая позволяет создавать интерактивное соединение между клиентом и сервером для обмена сообщениями в режиме реального времени (рисунок 1.4).

Веб-сокеты, в отличие от HTTP, позволяют работать с двунаправленным потоком данных, что делает эту технологию совершенно

уникальной. Ниже представлено описание, как работает эта технология и чем она отличается от HTTP.

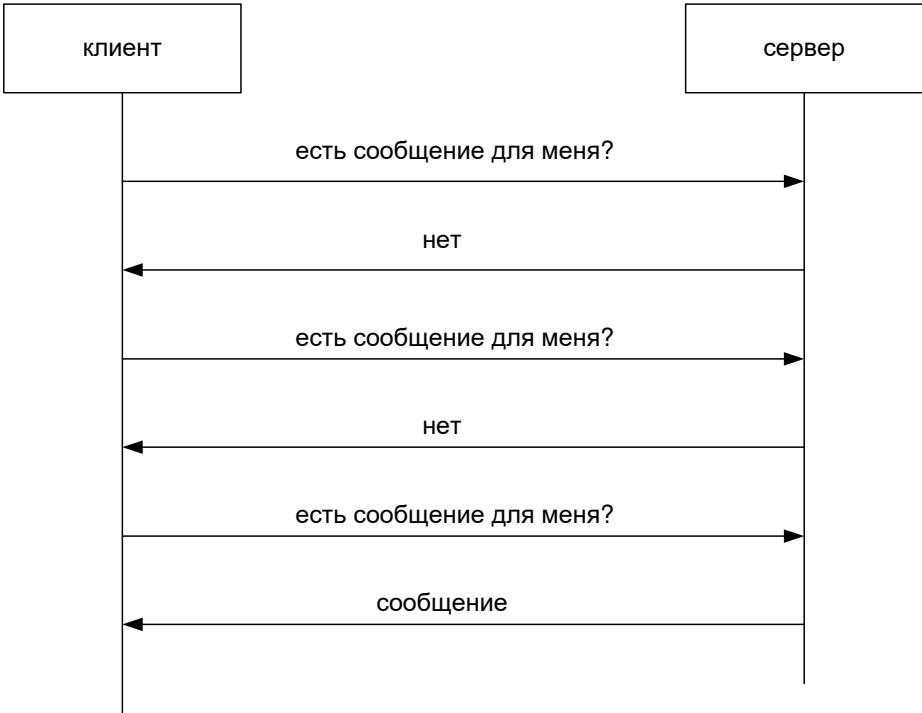


Рисунок 1.3 - Обмен сообщениями по HTTP

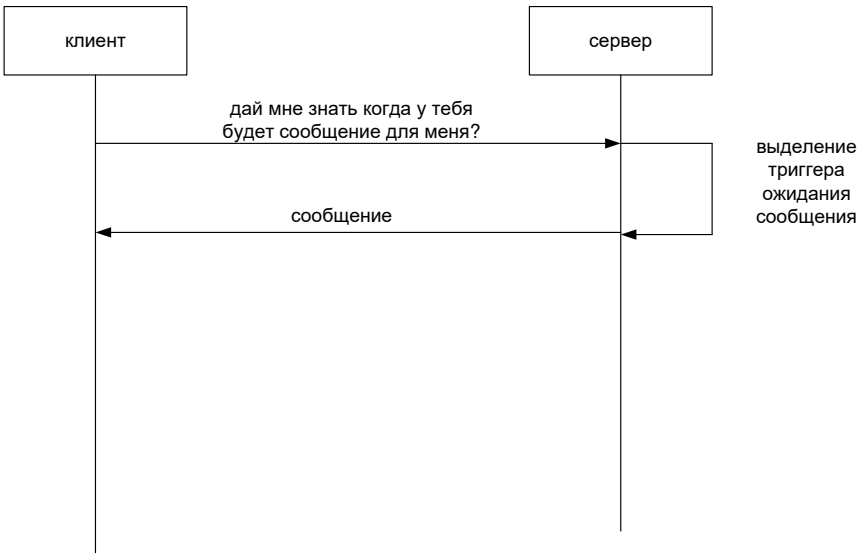


Рисунок 1.4 - Обмен сообщениями WebSocket

Веб-сокеты для ответа не нужны HTTP повторяющиеся запросы. Достаточно выполнить один запрос и ждать отклика. Можно просто слушать сервер, который будет отправлять сообщения клиентам по мере готовности.

Веб-сокеты можно использовать, если разрабатываются:

- приложения реального времени;
- чат-приложения;
- многопользовательские игры.

Протокол CoAP предназначен для взаимодействия простых устройств, например датчиков малой мощности, выключателей, которые управляются или контролируются удаленно через сеть Интернет.

Часто подобные устройства называют устройствами с ограниченными ресурсами.

Они обычно имеют ограниченный энергоресурс, небольшой объем памяти и невысокую мощность, поэтому в работе с ними важно обеспечивать низкие энергозатраты, использовать передачу сообщений малого объема.

Существенная особенность протокола CoAP это его совместимость с протоколом HTTP, что обеспечивает при его использовании взаимодействие совокупности устройств IoT, формирующих общую гетерогенную сеть на основе сети Интернет (рисунок 1.5).

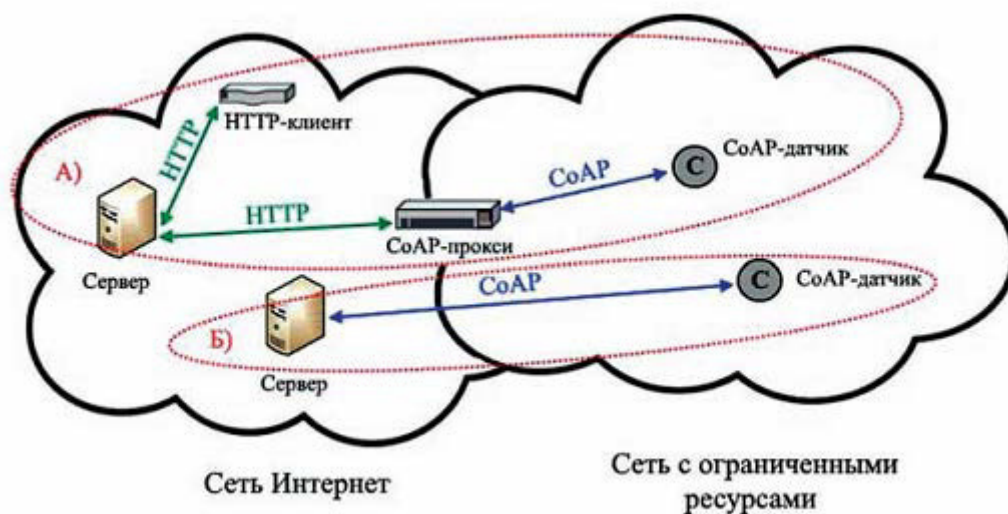


Рисунок 1.5 - Обмен сообщениями CoAP

CoAP методы:

- GET – выполняет поиск ресурсов и предоставляет информацию, которая соответствует ресурсу, указанному в ссылке URI (Uniform Resource Identifier), где ресурс – это источник данных, которым, например, может являться датчик;
- PUT – задает новое действие над ресурсом;
- POST – отвечает за изменения действия над ресурсом;
- DELETE – служит для удаления активированных возможностей ресурса.
- FETCH – выполняет предоставление частичной информации о ресурсе по параметрам в запросе;
- PATCH – отвечает за частичное изменение действия над ресурсом.

Протокол дополнен этими методами, так как возникла необходимость приложениям получать доступ или изменять ресурс не полностью, а взаимодействовать только с определенной его составляющей.

При взаимодействии с датчиком через сеть Интернет приложение на пользовательском устройстве будет работать по протоколу HTTP и использовать методы REST для отправки запросов. Благодаря использованию одинаковых методов в протоколах HTTP и CoAP возможен обмен сообщениями между пользовательским устройством, на котором находится приложение, и датчиком, использующим иные протоколы, ведь сообщения будут легко транслироваться через прокси сервер.

Целевые устройства LWM2M являются устройствами с ограниченными ресурсами.

Следовательно, LWM2M легкий и компактный протокол, а также эффективная модель данных для конечных ресурсов.

Он предоставляет поставщику услуг IoT возможность развернуть систему на основании протокола CoAP.

LWM2M предназначен для:

- управления устройствами через беспроводные и/или сотовые сети;
- передачи служебных данных устройствам;
- поддерживать связь клиентов, серверов и устройств приложений любой сложности.

В случае разрабатываемого приложения выбор сделан в пользу протоколов HTTP и CoAP:

- протоколы апробированы и надежны в реализации;
- поддерживаются всеми фреймворками и известными устройствами;
- позволяют быстро реализовать проект клиент серверного приложения.

1.3 Выбор базы данных для управления системой

В мире технологий баз данных существует два основных направления: SQL и NoSQL, реляционные и нереляционные базы данных. Различия между ними заключаются в том, как они спроектированы, какие типы данных поддерживают, как хранят информацию (рисунок 1.6).

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

15

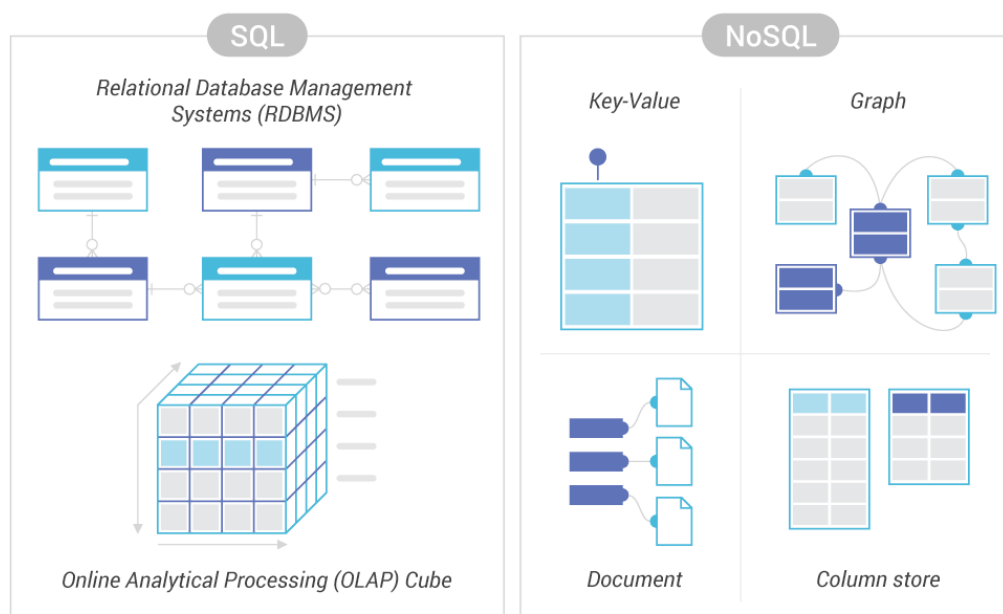


Рисунок 1.6 - Структуры данных SQL и NoSQL

Реляционные БД хранят структурированные данные, которые обычно представляют объекты реального мира. Это могут быть сведения, сгруппированные в таблицах, формат которых задан на этапе проектирования хранилища.

Нереляционные БД устроены иначе. Документо-ориентированные базы хранят информацию в виде иерархических структур данных. Речь может идти об объектах с произвольным набором атрибутов. То, что в реляционной БД будет разбито на несколько взаимосвязанных таблиц, в нереляционной может храниться в виде целостной сущности.

Внутреннее устройство различных систем управления базами данных влияет на особенности работы с ними. Так нереляционные базы лучше поддаются масштабированию.

Плюсы SQL-баз данных - необходимость соответствия базы данных требованиям ACID (Atomicity, Consistency, Isolation, Durability - атомарность, непротиворечивость, изолированность, долговечность).

Это позволяет уменьшить вероятность неожиданного поведения системы и обеспечить целостность базы данных. Достигается подобное

путём жёсткого определения того, как именно транзакции взаимодействуют с базой данных. Это отличается от подхода, используемого в NoSQL-базах, которые требуют гибкость и скорость, а не 100% целостность данных (рисунок 1.7).

Данные структурированы, при этом структура не подвержена частым изменением.

Плюсы NoSQL-баз данных:

- хранение больших объёмов неструктурированной информации.
- база данных NoSQL не накладывает ограничений на типы хранимых данных;
- при необходимости в процессе работы можно добавлять новые типы данных;
- использование облачных вычислений и хранилищ;
- быстрая разработка.

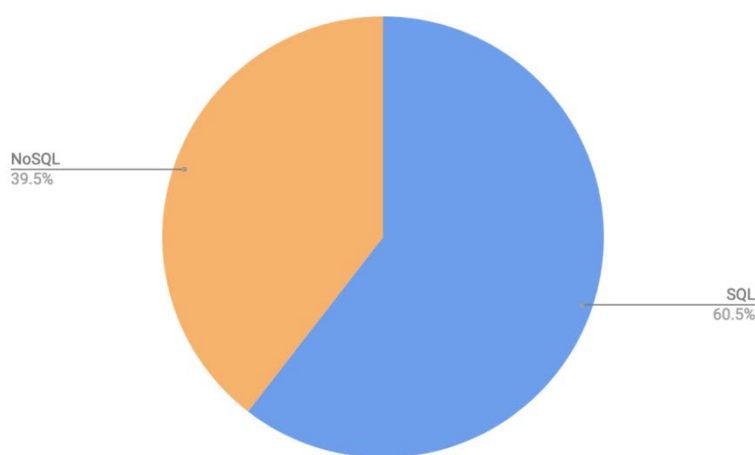


Рисунок 1.7 - Популярность подходов хранения данных на 2019 год

Ниже приведем типы проектов, для которых для которых применим тот или иной подход.

Признаки проектов, для которых подойдут SQL-базы:

- имеются логические требования к данным, которые могут быть определены заранее;
- важна целостность данных;

- нужна основанная на стандартах, зарекомендовавшая себя технология, используя которую можно рассчитывать на большой опыт разработчиков и техническую поддержку.

Свойства проектов, для которых подойдёт что-то из сферы NoSQL:

- требования к данным нечёткие, неопределённые, или развивающиеся с развитием проекта.

- цель проекта может корректироваться со временем, при этом важна возможность немедленного начала разработки.

- одни из основных требований к базе данных скорость обработки данных и масштабируемость.

В данном проекте второй вариант является предпочтительным, т.к. проект является экспериментальным и неизвестны типы API и устройств, с которыми он будет работать.

1.4 Обзор конечных устройств

Плата на базе микроконтроллера ATmega32u4 и Atheros AR9331. Процессор Atheros поддерживает дистрибутив Linux, основанный на базе OpenWrt и называемый OpenWrt-Yun. Плата имеет встроенную поддержку Ethernet и WiFi, порт USB-A, слот для карты micro-SD, 20 (рисунок 1.8).



Рисунок 1.8 - Плата Arduino Yun

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

18

Компактный ПК на базе единственной платы с новым, более мощным, четырехъядерным 64-битным процессором ARMv8 1.2ГГц, а также встроенными модулями Wi-Fi 802.11n и Bluetooth 4.1 с поддержкой BLE (рисунок 1.9).



Рисунок 1.9 - Плата Raspberry Pi

Малопотребляющая и гибкая система на кристалле с поддержкой протоколов Bluetooth Smart, ANT и стеков собственной разработки пользователя в частотном диапазоне 2.4GHz. nRF52832 построен на 32-разрядном процессорном ядре ARM Cortex с 512kB Flash и 64kB RAM памяти. По протоколу работы nRF52832 полностью совместим с предыдущими сериями nRF51, nRF24L и nRF24AP производства Nordic Semiconductor (рисунок 1.10).



Рисунок 1.10 - Плата Nordic (nRF52832)

Данные устройства поддерживают COAP протокол и будут использованы в роли устройств системы.

Изм.	Лист	№ докум.	Подпись	Дата

1.5 Общая архитектура аппаратно-программной системы

Существует три типа существующих архитектур промежуточного программного обеспечения IoT, для построения распределенной системы работы с датчиками, шлюзами, серверами:

- сервис-ориентированная архитектура, позволяющая разработчикам разворачивать разнородные устройства IoT, в качестве служб, таких как хранилище, интерфейсы веб-служб, диспетчер запросов и т. д.
- облачная архитектура, которая ограничивает разработчиков по количеству и типу устройств IoT.
- архитектура на основе аппаратных решений, позволяющая устройствам IoT работать в виде передатчиков и/или серверов сети (рисунки 1.11-13).

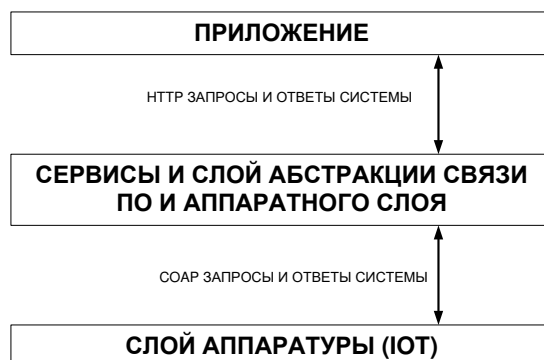


Рисунок 1.11 - Общая модель построения системы

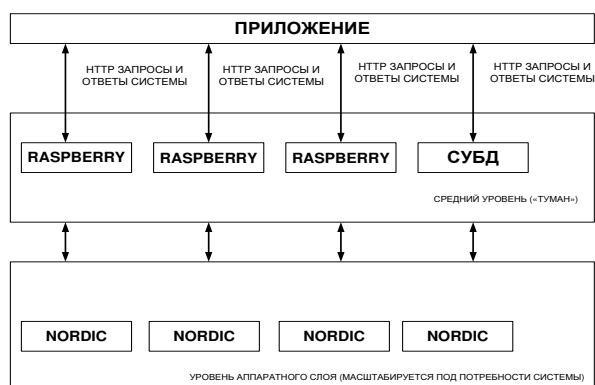


Рисунок 1.12 - Модель построения системы
с несколькими платами типа «Мастер»

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

20

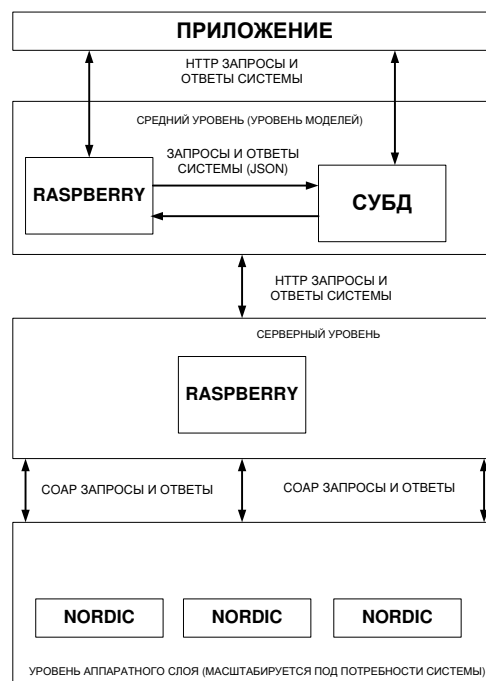


Рисунок 1.13 - Модель построения системы
«Модель – Вид – Контроллер»

Виртуальная прослойка отвечает за управление веб-ресурсами, предоставляя API-интерфейсы REST, доступные пользователям и периферийным устройствам.

Во второй главе на базе REST и на базе унифицированного интерфейса строится распределенная система, представленная на рисунке 1.3.

Предоставление ресурсов осуществляется посредством URI и управляется четырьмя примитивами (CRUD): создание, чтение, обновление, удаление. Данные примитивы сопоставлены с методами HTTP: POST, GET, PUT, DELETE. Таблица 1.2 показывает соответствие между CRUD и HTTP методами.

Таблица 1.2 - Соответствие CRUD и HTTP (REST интерфейс)

№	CRUD	HTTP методы
1	Create	POST
2	Read	GET
3	Update	PUT
4	Delete	DELETE

Конечные точки IoT представлены в виде виртуальных ресурсов, доступные внешним элементам через URI и управляемые унифицированным интерфейсом CRUD.

Эти виртуальные ресурсы формируют службы, которые могут быть промежуточным программным обеспечением.

Таблица 1.3 указывает, что местоположение ресурса может быть идентифицировано URI, а поведение физического и прикладного уровней может управляться CRUD.

Таблица 1.3 - REST API запросы

№	HTTP методы	URI	Значение
1	POST	http://192.168.0.100/IoT/ {board_id}&register	регистрация нового устройства
2	GET	http://192.168.0.100/IoT/ {board_id}&getinfo	получение информации от устройства
3	PUT	http://192.168.0.100/IoT/ {board_id}&update	обновить информацию об устройствах
4	DELETE	http://192.168.0.100/IoT/ {board_id}&delete	удалить устройство из системы

Вычислительный REST (CREST) расширяет возможности предоставления ресурсов путем обмена вычислительными выражениями, так

что любой код может передаваться через Web-сервисы RESTful, обычно с помощью POST и PUT запросов.

Конечное устройство каждую единицу времени отправляет данные в одноплатный компьютер мастер, а затем мастер отправляет запросы POST с полезными нагрузками в точку входа по URL-адресу «http://192.168.0.100/IoT/».

Мастер устройства с технической информацией такими как «BoardName», «Значение», «ProgramVersion», «Location» и другой необходимой информацией в формате JSON отправляет массив данных на сервер. Сервер анализирует строку JSON и сохраняет информацию в базе данных. Пользователи используют веб-приложение и отправляют запросы к СУБД или GET запросы к серверу, например URL-адрес устройства «http://192.168.0.100/IoT/board_id» для получения информации о конечных устройствах.

Исходя из выше приведенного анализа, реализуемая система выглядит следующим образом (рисунок 1.14).

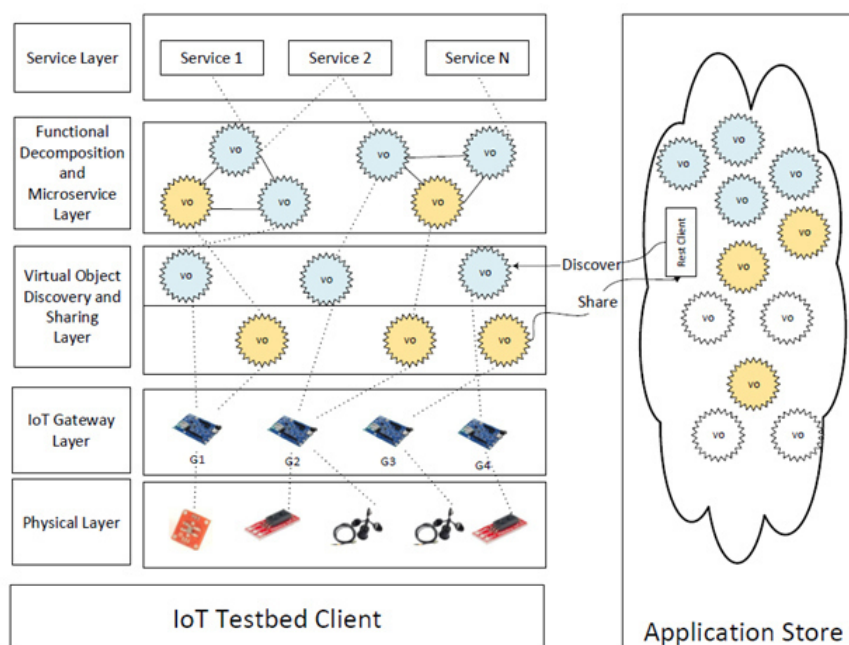


Рисунок 1.14 - Вид реализуемой системы

Средний уровень будет реализован с помощью библиотек, которые представляют уровень абстракции, позволяющий подключать и взаимодействовать разным типам устройств.

1.6 Выводы по главе

В данной главе был сделан анализ предметной области. Было определено, что создание программной системы управления слоями аппаратуры экосистемы Интернета вещей является актуальной тематикой.

Определено что разрабатываемая система будет работать в модели: Модель - Вид-Контроллер по протоколам HTTP и SOAP.

Инструментарием разработки системы выбран язык PHP и фреймворк Yii PHP Framework.

2 Реализация системы

В этой главе будет описана реализация системы управления IoT устройствами с использованием требований, определенных в предыдущей главе.

Архитектура системы состоит из трех разделов:

- описание frontend зоны. Дизайн пользовательского интерфейса и графики;
- описание backend зоны. Проектирование бизнес логики, ее компонентов и связи с устройствами и NoSQL хранилищем;
- описание зоны IoT устройств (рисунок 2.1).

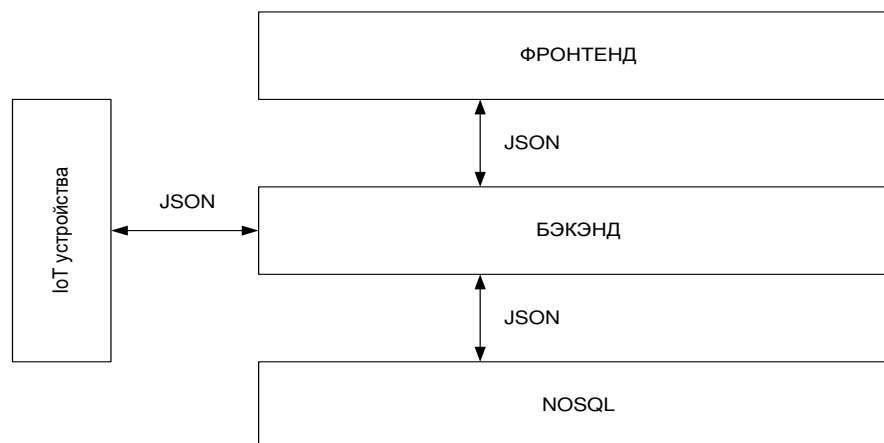


Рисунок 2.1 - Архитектура разрабатываемого приложения.

2.1 Реализация backend части приложения

Разрабатываемая среда управления устройствами и сетями IoT навязывает схемы управления устройствами IoT с функциями IoT-friendly.

Например, частый опрос данных с устройств с низким энергопотреблением не является оптимальным для энергопотребления, и ожидается, что централизованное управление устройствами IoT позволит избежать таких ситуаций. В этом разделе описываются функции управления устройствами IoT, которые направлены на оптимизацию использования

ограниченных сред и максимизацию считывания данных IoT на стороне сервера.

Для работы с множеством устройств с низким энергопотреблением воспользуемся паттерном Publish/Subscriber для организации протокола общения между клиентом, бизнес логикой и устройствами. Данный паттерн реализуется на основе трех сущностей: клиент, брокер, устройство.

База данных брокера (в данном случае NoSQL БД Redis) содержит таблицу со всеми полученными пакетами с индексацией по топикам этих пакетов. Получив пакет, брокер рассылает его всем устройствам в сети согласно их подписке. Чтобы устройство что-то получило от брокера оно должно подписаться на топик. Топики возникают динамически по факту подписки или по факту прихода пакета с данным топиком. От подписки на топик можно и отказаться. Таким образом топики представляют собой удобный механизм организации связей разных видов: один ко многим, многие к одному и многие ко многим. Если у пакета нет подписчика, то он отбрасывается. Если подписчик не на связи, то пакет либо сразу удаляется из базы брокера, либо ждет подключения подписчика некоторое заданное в конфигурации время. Вариант поведения определяется атрибутом QoS пакета.

Основные особенности библиотек паттерна publisher/subscriber (в данном случае библиотека Mosquitto в интеграции с Yii PHP Framework):

- асинхронный протокол передачи данных между сервером и шлюзами устройств;
- компактные сообщения;
- работа в условиях нестабильной связи на линии передачи данных;
- поддержка нескольких уровней качества обслуживания (QoS);
- легкая интеграция новых устройств.

Библиотека Mosquitto работает на прикладном уровне поверх TCP/IP и использует по умолчанию 1883 порт (8883 при подключении через SSL) (рисунок 2.2).

OSI/TCP 5-7	Mosquitto
OSI/TCP 4	TCP
OSI/TCP 3	IP

Рисунок 2.2 - Стек протоколов передачи данных

Обмен сообщениями в API Mosquitto осуществляется между клиентом (client), который может быть издателем или подписчиком (publisher/subscriber) сообщений, и брокером (broker) сообщений.

Издатель отправляет данные брокеру, указывая в сообщении определенную тему, топик (topic). Подписчики могут получать разные данные от множества издателей в зависимости от подписки на соответствующие топики.

Устройства системы используют определенные типы сообщений для взаимодействия с брокером, ниже представлены основные:

- Connect – установить соединение с брокером;
- Disconnect – разорвать соединение с брокером;
- Publish – опубликовать данные в топик на брокере;
- Subscribe – подписаться на топик на брокере;
- Unsubscribe – отписаться от топика.

Схема простого взаимодействия между подписчиком, издателем и брокером в системе Yii PHP Framework + Mosquitto приведена на рисунке 2.3.

Топики от издателя к брокеру представляют собой символы с кодировкой UTF-8. Иерархическая структура топиков имеет формат «дерева», что упрощает их организацию и доступ к данным. Топики состоят

из одного или нескольких уровней, которые разделены между собой СИМВОЛОМ «/».

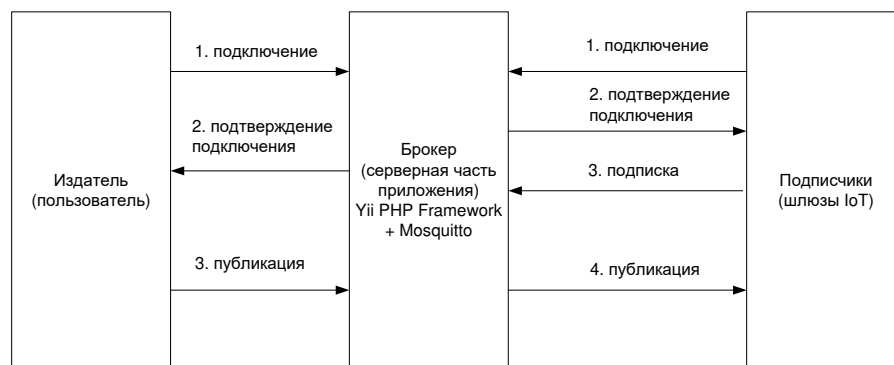


Рисунок 2.3 - Общая схема реализации связи между частями приложения

Пример топика, в который датчик температуры, расположенный в жилой комнате, публикует данные брокеру: `/build_1/living-space/living-room1/temperature`.

Подписчик может так же получать данные сразу с нескольких топиков, для этого существуют wildcard. Они бывают двух типов: одноуровневые и многоуровневые. Для более простого понимания рассмотрим в примерах каждый из них:

Одноуровневый wildcard. Для его использования применяется символ «+». К примеру, необходимо получить данные о температуры во всех жилых комнатах здания:

`/build_1/living-space+/temperature`

В результате получаем данные с топиков:

`/build_1/living-space/living-room1/temperature`

`/build_1/living-space/living-room2/temperature`

`/build_1/living-space/living-room3/temperature`

Многоуровневый wildcard. Для его использования применяется символ «#». К примеру, чтобы получить данные с различных датчиков всех жилых помещений в здании: `/build_1/living-space/#` (рисунок 2.4).

В результате получаем данные с топикиов:

/build_1/living-space/living-room1/temperature

/build_1/living-space/living-room1/light1

/build_1/living-space/living-room1/light2

/build_1/living-space/living-room1/humidity

/build_1/living-space/living-room2/temperature

/build_1/living-space/living-room2/light1

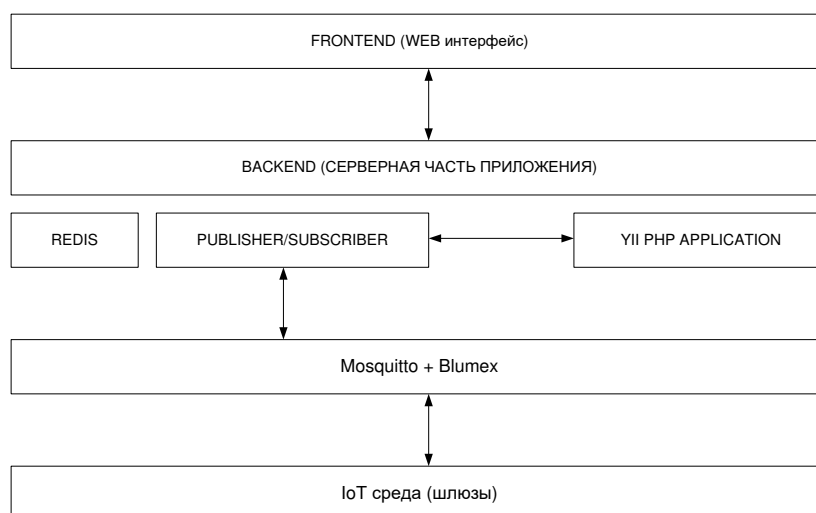


Рисунок 2.4 - Детализация модулей приложения

Сообщения между брокером и шлюзами устройств имеют свой четкий формат:

- фиксированный заголовок (присутствует по всех сообщениях);
- переменный заголовок (присутствует только в определенных сообщениях);
- данные, «нагрузка» (присутствует только в определенных сообщениях) и/или команды (таблица 2.1).

Фиксированный заголовок

БИТЫ	7	6	5	4	3	2	1	0
БАЙТ 1	Тип сообщения				Флаг каждого пакета			
БАЙТ 2	Remaining Length							

где: Message Type — это тип сообщения, например: CONNECT, SUBSCRIBE, PUBLISH и другие.

Flags specific to each packet - эти 4 бита отведены под вспомогательные флаги, наличие и состояние которых зависит от типа сообщения.

Remaining Length - представляет длину текущего сообщения (переменный заголовок + данные), может занимать от 1 до 4 байта.

Таблица 2.1 - Виды сообщений в системе publisher/subscriber

Тип сообщения	Значение	Направление передачи	Описание
Reserved	0000 (0)	нет	Зарезервирован
CONNECT	0001 (1)	IoT шлюз -> сервер	Запрос клиента на подключение к серверу
CONNACK	0010 (2)	IoT шлюз <- сервер	Подтверждение успешного подключения
PUBLISH	0011 (3)	IoT шлюз <- сервер, IoT шлюз -> сервер	Публикация сообщения
PUBACK	0100 (4)	IoT шлюз <- сервер, IoT шлюз -> сервер	Подтверждение публикации
PUBREC	0101 (5)	IoT шлюз <- сервер, IoT шлюз -> сервер	Публикация получена
PUBREL	0110 (6)	IoT шлюз <- сервер, IoT шлюз -> сервер	Разрешение на удаление сообщения
PUBCOMP	0111 (7)	IoT шлюз <- сервер, IoT шлюз -> сервер	Публикация завершена
SUBSCRIBE	1000 (8)	IoT шлюз -> сервер	Запрос на подписку
SUBACK	1001 (9)	IoT шлюз -> сервер	Запрос на подписку принят
UNSUBSCRIBE	1010 (10)	IoT шлюз -> сервер	Запрос на отписку

Продолжение таблицы 2.1

UNSUBACK	1011 (11)	IoT шлюз -> сервер	Запрос на отписку принят
PINGREQ	1100 (12)	IoT шлюз -> сервер	PING запрос
PINGRESP	1101 (13)	IoT шлюз -> сервер	PING ответ
DISCONNECT	1110 (14)	IoT шлюз -> сервер	Сообщение об отключении от сервера
Reserved	1111 (15)	нет	Зарезервирован

Таким образом, наша система запросов от пользователя к устройствам приобретает следующий вид (рисунок 2.5).

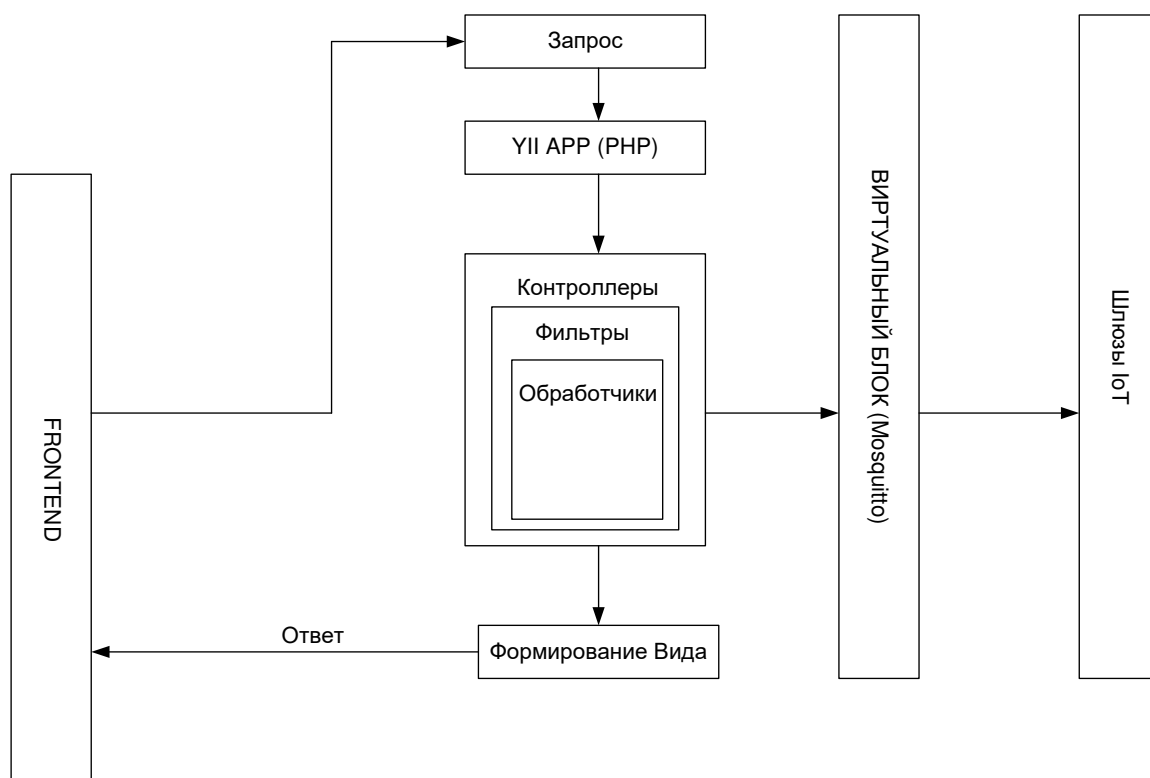


Рисунок 2.5 - Взаимодействие модулей системы

Опишем более подробно действие контроллеров (рисунок 2.5) приложения.

Контроллеры являются частью MVC архитектуры приложения. Это объекты классов, унаследованных от yii\base\Controller, отвечающие за

обработку запроса и генерирование ответа. В сущности, после обработки запроса приложениями, контроллеры проанализируют входные данные, передадут их в модели, вставят результаты модели в представления, и в конечном итоге сгенерируют исходящие ответы.

Чтобы разделить рабочий процесс на логические компоненты, реализация контроллеров, то есть разделение классов контроллеров, осуществляется в соответствии с дизайном внешнего интерфейса, изображенным на рисунке 2.1.

Для взаимодействия трех слоев приложения было реализовано пять основных классов контроллеров:

- SiteController - отвечает за все действия, выполняемые на странице, такие как загрузка информации о IoT шлюзов, обновление значений данных IoT шлюзов, обновление политик IoT шлюзов, отображение подробной информации об устройстве и многое другое. Как упомянуто в рабочем процессе выше, каждая из этих функций соответствует различным действиям в SiteController.

- ElementsController: этот контроллер отвечает за страницу «Сетевые элементы» интерфейса. Класс извлекает список сетевых элементов из CNF и представляет его пользователю.

- NetworkController: этот контроллер собирает журналы и уведомления от CNM и CNF и представляет их пользователю. Пользователь также может устанавливать уровни журналов или фильтровать уведомления.

- SearchController - контроллер ищет с помощью IoT Framework по заданному поисковому запросу элементы IoT сети и возвращает результаты поиска пользователю на отдельной странице.

- UserController - контроллер управляет доступом к пользовательскому интерфейсу на основе учетных данных имени пользователя и пароля.

Полный список связей контроллеров и action функций для них представлено на рисунках 2.6 и 2.7

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

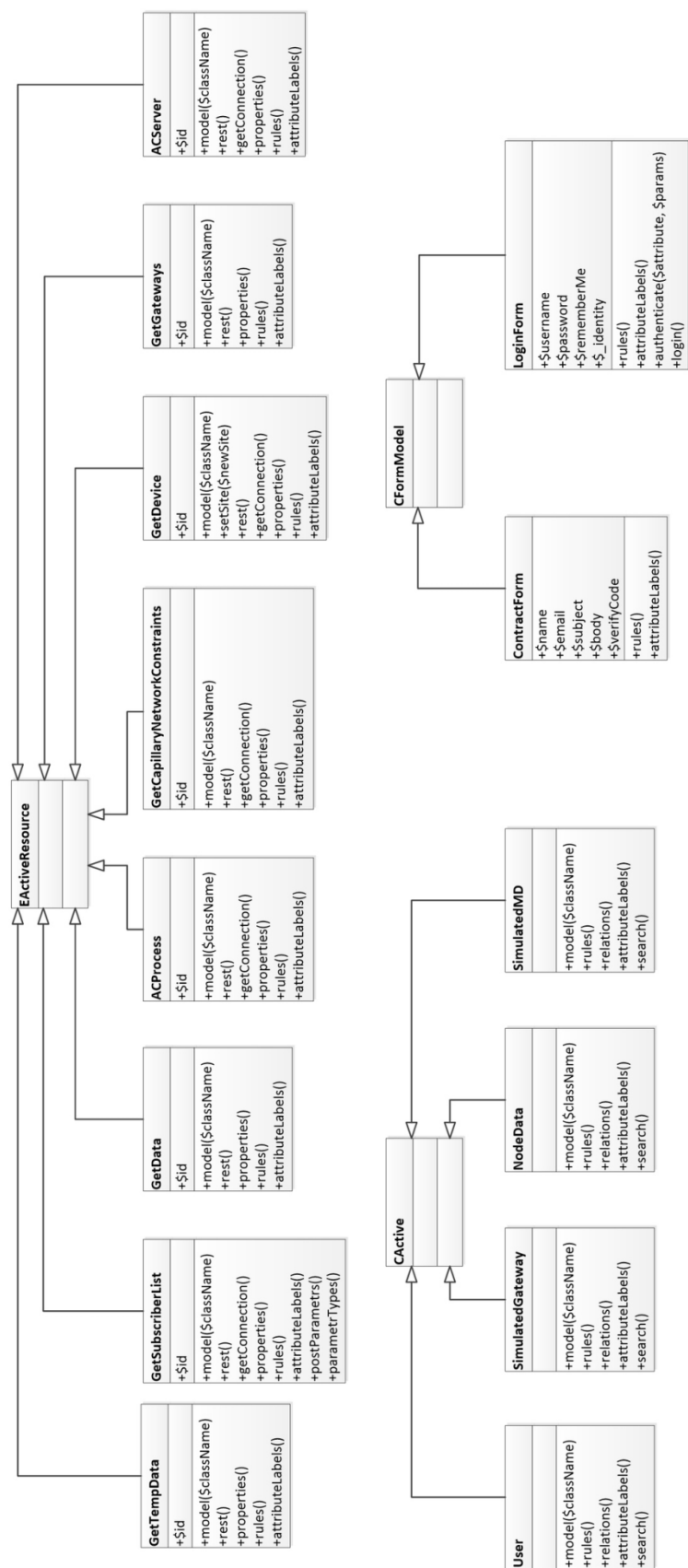


Рисунок 2.6 - Диаграмма классов обработчиков запросов

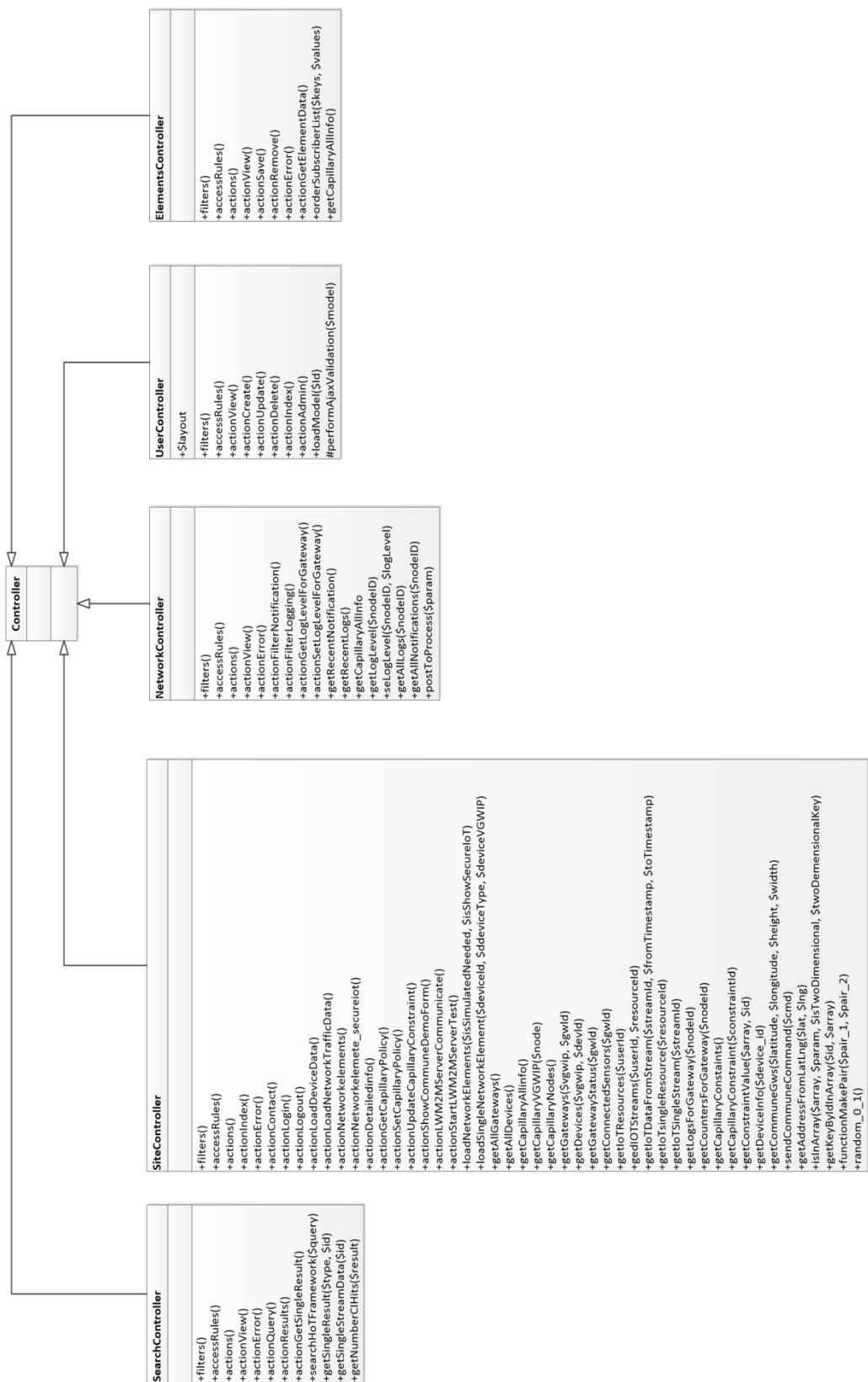


Рисунок 2.7 - Диаграмма классов контроллеров

В результате работы контроллеров система получает информацию от устройств в виде сообщений JSON. Полученная информация содержит поля с названием устройства, его IP адрес, его MAC адрес, а также дополнительную информацию об устройстве.

```
[{"node": "1", "devices": "103:e199:2:2b9a", "vGW_1": "193.234.218.188,
2a00:1d50:2:1001:f816:3eff:fe94:4a3d", "online": "True", "level": "5",
"params": {"1": {"lat": 59.40442269, "lng": 17.95359135}}},
```

```
 {"node": "2", "devices": "280:e103:1:2a9c", "vGW_2": "193.234.217.173,2001:14b8:400:131:f816:3eff:fe4f:de50",
"online": "True", "level": "0", "params": {"2": {"lat": 41.35342516, "lng":
2.13135839}, "280:e103:1:2a9c": {"lat": 41.35361845, "lng": 2.13043571}}}]
```

2.2 Реализация rest api приложения

В предыдущих разделах была описана связь между приложением PHP и сущностями облака IoT.

Чтобы установить связь между интерфейсом приложения и физическими устройствами, реализованы REST API для запросов к сущностям (шлюзам) в формате JSON.

Таким образом, интерфейс управления, то есть приложение PHP и сущности IoT, могут располагаться отдельно в любом месте сети, и связь между этими двумя частями системы поддерживается соответственно с помощью слоя бизнес логики.

Решение о том, какие API-интерфейсы REST реализовать, основывалось на требованиях бэкэнда и фронтенда, а также на ограничениях, связанных с сущностями IoT.

Таблица 2.2 - Действия контроллеров системы

№	Контроллер	Функция
1	SiteController	Получить список подключенных к сети шлюзов и их конечных устройств
2	SiteController	Получить список подключенных к сети шлюзов
3	SiteController	Получить IP адреса включенных в сеть шлюзов
4	SiteController	Получить онлайн статус шлюзов
5	SiteController NetworkController	Получить последние лог данные от шлюзов
6	NetworkController	Получить статус файлов логов шлюзов (установлены или нет)
7	NetworkController	Установить статус файлов логов шлюзов
8	NetworkController	Получить уведомление от шлюзов (в зависимости от задачи – значение температуры или иные полезные значения)
9	SiteController	Получить значения политики безопасности устройств
10	SiteController	Установить значения политики безопасности устройств
11	LocationCollector	Получить GPS координаты устройств
12	SiteController SearchController	Получить список устройств доступных пользователю
13	SiteController SearchController	Получить информацию о ресурсе сети по ID
14	SiteController SearchController	Получить информацию о ресурсе сети по IP

Далее опишем более подробно основные запросы контроллеров с описанием запроса и ответа на него в виде таблиц запросов.

Таблица 2.3 - Получение полного списка устройств IoT сети и их конечных устройств

POST	/process.php	Получение полного списка устройств IoT сети
Положительный ответ системы	200	
Код ошибки	unauthorized (401), badRequest (400)	
Пример: cmd=get_info		
<pre>[{"node":"2","devices":"","vGW": "193.234.217.173,2001:14b8:400:131:f816:3eff: fe4f:de50", "online":"True", "level":"0", "params":{"2": {"lat": 41.35342516, "lng": 2.13135839}, "280:e103:1:2a9c": {"lat": 41.35361845, "lng": 2.13043571}} }]</pre>		

Таблица 2.4 - Получение полного списка устройств IoT сети

POST	/process.php	Получение полного списка устройств IoT сети
Положительный ответ системы		200
Код ошибки		unauthorized (401), badRequest (400)
Пример: cmd= get_nodes		
["1", "2", "2911", "3", "4", "5"]		

Таблица 2.5 - Получение IP адреса устройств IoT сети

POST	/process.php	Получение IP адресов устройств IoT сети
Положительный системы	ответ	200
Код ошибки	unauthorized (401), badRequest (400)	
Пример: cmd=get_vgw_ip&node=1		
{ "node": "1", "ip4": "193.234.218.188", "ip6": "2a00:1d50:2:1001:f816:3eff:fe94:4a3d" }		

Таблица 2.6 - Получение онлайн статуса устройств IoT сети

POST	/process.php	Получение онлайн статуса устройств IoT сети
Положительный системы	ответ	200
Код ошибки		unauthorized (401), badRequest (400)
Пример: cmd=get_node_status&node=1		
{ "node": "1", "status": "Online" }		

Таблица 2.7 - Получение логи устройств IoT сети

POST	/process.php	Получение логи устройств IoT сети
Положительный системы	ответ	200
Код ошибки	unauthorized (401), badRequest (400)	
Пример: cmd=get_log&node=1		
[[{ "id": "1694774", "level": "4", "entry": "2014-04-23 07:36:40 WARNING: GW 1 disconnected from the network (failed 3 times)" }, { "id": "1694775", "level": "5", "entry": "2014-04-23 07:37:17 GW 1 connected to the network" }, { "id": "1694776", "level": "5", "entry": "2014-04-23 07:37:26 Device 280:e103:1:3829 (type temperature) connected to GW 1" }]]		

Таблица 2.8 - Установить разрешение логгирования для устройств IoT сети

POST	/process.php	Получение логи устройств IoT сети
Положительный ответ системы	200	
Код ошибки	unauthorized (401), badRequest (400)	
Пример: cmd=set_logging_level&node=1&level=4		
Success.		

Таблица 2.9 - Получить политики безопасности для устройств IoT сети

POST	/access.controller/m-nf?cmd=getPolicy	Получить политики безопасности для устройств IoT сети
Положительный ответ системы	200	
Код ошибки	internalServerError (405)	
Пример: http://192.168.0.1:8080/access.controller/m-nf?cmd=getPolicy		
{ "xxalarm": { "load": -1 }, "default": { "battery": 2, "load": -2, "connections": -1 }, "xxxdefault": { "battery": 1, "load": -1 }, "xxdefault": { "connections": -1 } }		

Таблица 2.10 - Установить политики безопасности для устройств IoT сети

POST	/access.controller/m-nf?cmd=setPolicy&policy=X	Установить политики безопасности для устройств IoT сети
Положительный ответ системы	200	
Код ошибки	internalServerError (405)	
Пример: http://0.0.0.0:8080/access.controller/m-nf?cmd=setPolicy		
&policy={"xxalarm":{"load": -5 } }		
{ "status": "ok" }		

Таблица 2.11 - Очистить список устройств IoT сети

POST	/access.controller/emulation? cmd=clear	Очистить список устройств IoT сети
Положительный ответ системы		200
Код ошибки		badRequest (400), internalServerError (405)
Пример: cmd=clear		
Success.		

Таблица 2.12 - Поиск устройств IoT сети

POST	/_search	Поиск устройств IoT сети
Положительный ответ системы		200
Код ошибки		badRequest (400), internalServerError (405)
Пример: { "query" : { "query_string" : { "query" : "vGW1" } } }		
{ "node": "2", "devices": "", "vGW1": "193.234.217.173,2001:14b8:400:131:f816:3eff: fe4f:de50", "online": "True", "level": "0", "params": { "2": { "lat": 41.35342516, "lng": 2.13135839 }, "280:e103:1:2a9c": { "lat": 41.35361845, "lng": 2.13043571 } } }		

Таблица 2.13 - Добавить устройство в IoT сети

POST	/process.php	Добавить устройство в IoT сети
Положительный ответ системы		200
Код ошибки		badRequest (400), internalServerError (405)
Пример: cmd= add&device=vGW1		
<pre>{ "node": "2", "devices": "", "vGW1": "193.234.217.173,2001:14b8:400:131:f816:3eff: fe4f:de50", "online": "True", "level": "0", "params": { "2": { "lat": 41.35342516, "lng": 2.13135839 }, "280:e103:1:2a9c": { "lat": 41.35361845, "lng": 2.13043571 } } }</pre>		

Таблица 2.14 - Удалить устройство в IoT сети

POST	/process.php	Добавить устройство в IoT сети
Положительный ответ системы		200
Код ошибки		badRequest (400), internalServerError (405)
Пример: cmd= delete&device=vGW1		
Success.		

Таблица 2.15 - Проверить устройство в IoT сети

POST	/process.php	Проверить устройство в IoT сети
Положительный ответ системы		200
Код ошибки		badRequest (400), internalServerError (405)
Пример: cmd= add&check=vGW1		
Success.		

2.3 Связь IoT сети и серверной части системы

Для создания IoT сети нам потребуется уже готовое и апробированное пространство, которое готово к интеграции с приложением на базе Yii PHP Framework. Анализ готовых решений привел к выбору решения IBM Watson IoT Platform (Bluemix).

Для обеспечения сопряжения и безопасной передачи данных в серверную часть приложения в раздел библиотеки добавлены сторонние библиотеки:

- OpenSSL – библиотека шифрования данных
- Pthreads – библиотека потокового выполнения операций
- Libwebsockets – подключение асинхронного выполнения команд между сервером и пространством IoT.
- Далее требуются следующие компоненты:
- учетная запись в Bluemix (зарегистрироваться для бесплатного использования);
- загруженный и установленный интерфейс командной строки Cloud Foundry, необходим до момента связи;
- приложение или система на базе Yii PHP Framework с необходимым набором библиотек.

Далее в состав шаблона Internet of Things Platform Starter входит механизм Node-RED, который используется для обработки сообщений от сервера приложения к IoT-шлюзам и обратно.

Первым делом требуется создать службу Internet of Things для отправки сообщений на разрабатываемый сервер приложения и получения сообщений с сервера. Для чего войдите в Bluemix под учетной записью (или зарегистрируйтесь для бесплатного использования).

Выберите раздел Catalog. Щелкните по вкладке Internet of Things Platform Starter.

Укажите имя своего приложения. Это имя будет также использоваться как имя хоста, поэтому оно должно быть уникальным в Bluemix, например, `iot<ИМЯ_СЕРВЕРА_ПРИЛОЖЕНИЙ>`. В данном примере используется имя `iotdemodaniel4` (данному имени соответствует сервер с реальным IP адресом) (рисунке 2.8).

Create an app:

Space:

Name:

Host:

Domain: mybluemix.net

Selected Plan:

SDK for Node.js™:

Cloudant NoSQL DB:

Internet of Things Platform:

CREATE

Рисунок 2.8 - Создание IoT среды

Оставьте значения по умолчанию и нажмите кнопку CREATE. В появившемся сообщении щелкните по кнопке Restage. Ваше приложение будет обновлено.

После того как среда создана ее необходимо наполнить устройствами IoT.

На панели Bluemix Dashboard выберите свое приложение и щелкните по новой службе (рисунках 2.9 - 2.10). На странице службы щелкните по кнопке Launch Dashboard.

Нажмите кнопку Create. На панели инструментов выберите Dashboard и щелкните по созданному приложению (Cloud Foundry или CF APP).

Щелкните по кнопке Add a Service or API. В левой навигационной области выберите Internet of Things для фильтрации списка доступных служб. Выберите службу Internet of Things Platform.

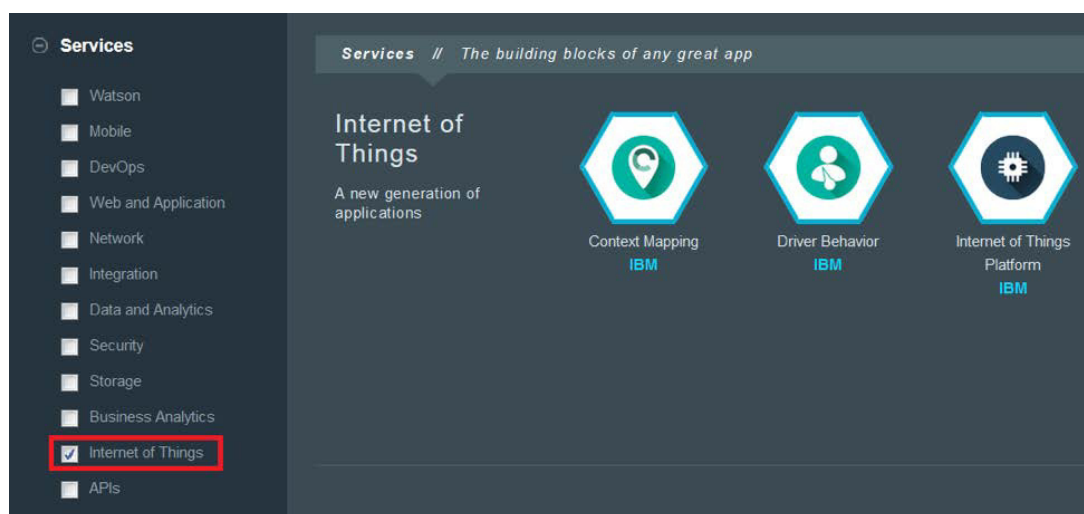


Рисунок 2.9 - Создание IoT среды. Выбор пункта меню

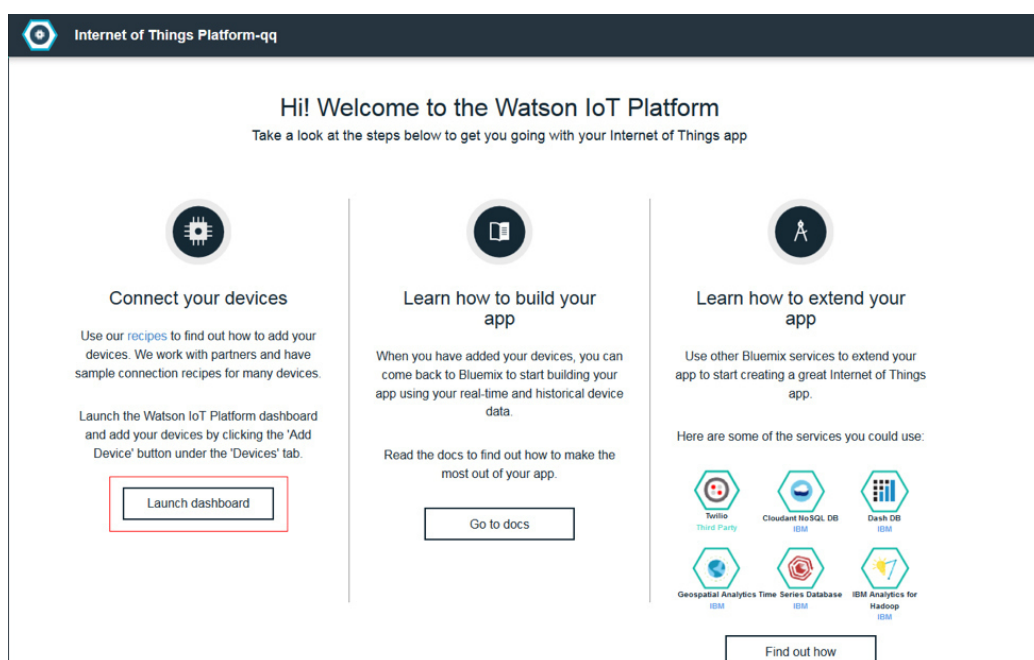


Рисунок 2.10 - Выбор элемента Launch Dashboard

Появится панель сервера IBM Internet of Things — службы, независимой от Bluemix. Приложению присваивается идентификатор организации Organization ID, который потребуется позже при стыковке нашей системы и пространства IoT. На следующем рисунке идентификатор

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		44

организации имеет значение 1xuhbp и отображается под логином в правом верхнем углу панели (рисунке 2.11).

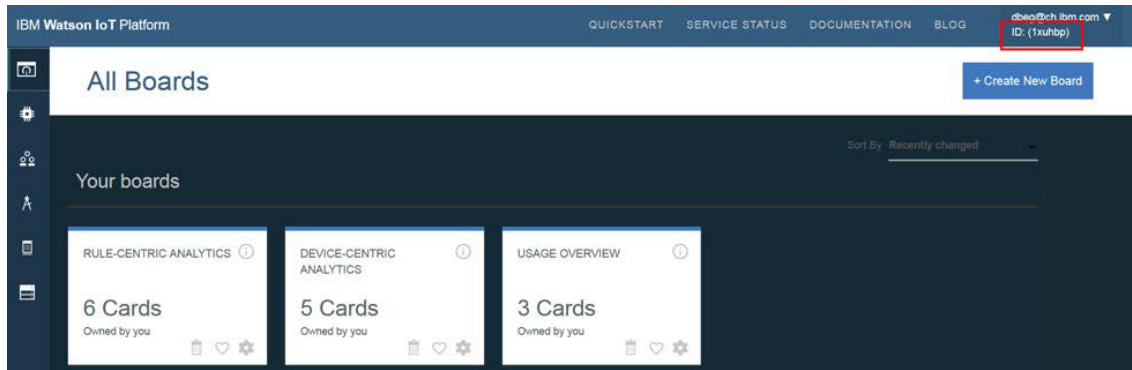


Рисунок 2.11 - ID приложения сформирован

В левом меню, которое всплывает при наведении на него указателя мыши, выберите Devices. Затем щелкните по Add a device и нажмите кнопку Create device type. В организации может быть много типов устройств, в каждом из которых может быть много устройств. Тип устройства — это группа устройств с общими характеристиками; например, у них могут быть одинаковые датчики. В нашем случае тип устройства должен иметь имя Arduino (этот тип устройств необходим приложению, которое будет его использовать для манипуляций) (рисунке 2.12).

Рисунок 2.12 - Выбор типа устройств,
с которыми будет работать система

Нажмите кнопку Next. Появится страница, на которой можно указать такие сведения о своем устройстве, как серийный номер или модель. В этой

демонстрации вводить такую информацию не нужно, просто нажмите кнопку Next. Также нет необходимости указывать какую-либо метаданную. Нажимайте кнопку Next, пока не вернетесь в диалоговое окно Add Device.

Нажмите кнопку Next. Укажите идентификатор устройства. Идентификатором устройства может быть, например, MAC-адрес устройства. Он должен быть уникальным в рамках системы. Поэтому можно указать, как в этом примере, что-нибудь типа "112233445566". Оставьте поле Meta Data пустым и нажмите кнопку Next. Оставьте поле Authentication token пустым и нажмите кнопку Next. На экране появится сгенерированное устройство (рисунке 2.13).

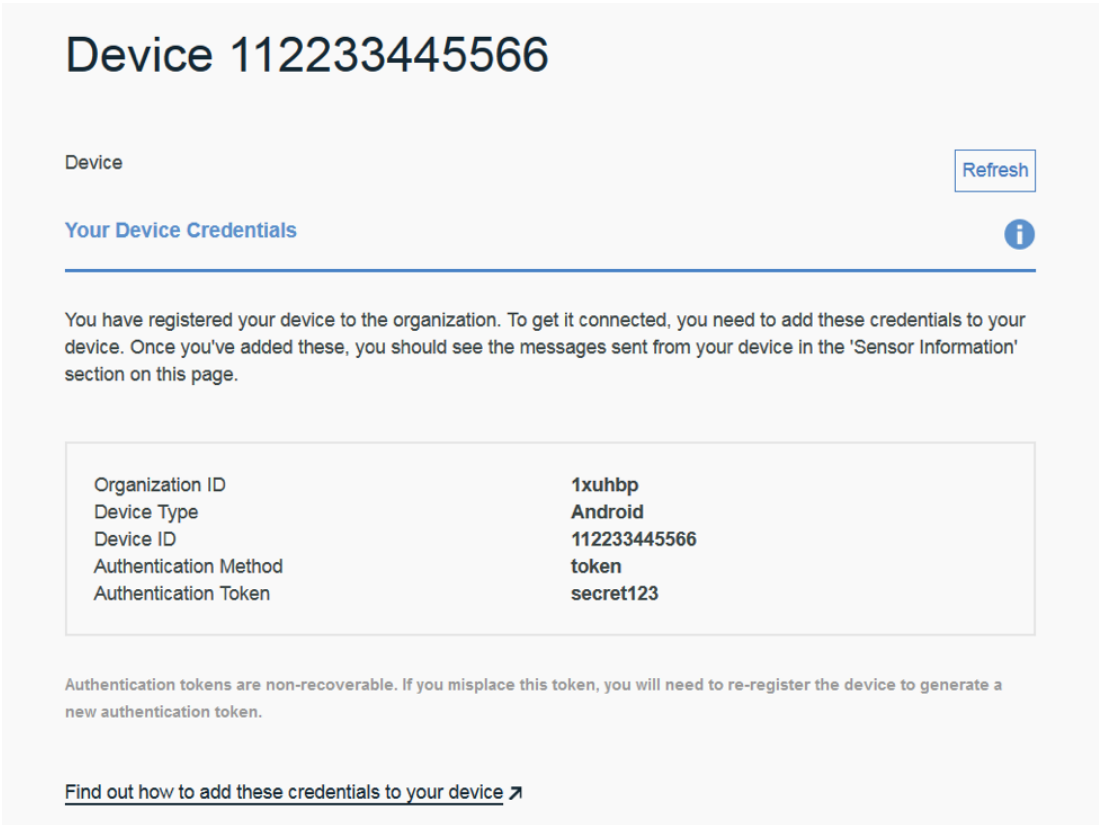


Рисунок 2.13 - Виртуальное устройство,
с которым будет работать система

Закройте это диалоговое окно. Теперь можно отправлять JSON сообщения с устройства на сервер и обратно. Проверка отправки сообщений с сервера на IoT шлюзы и обратно. Укажите значение токена

аутентификации. Запомните это значение, оно понадобится позже. Нажмите Next (рисунке 2.14).

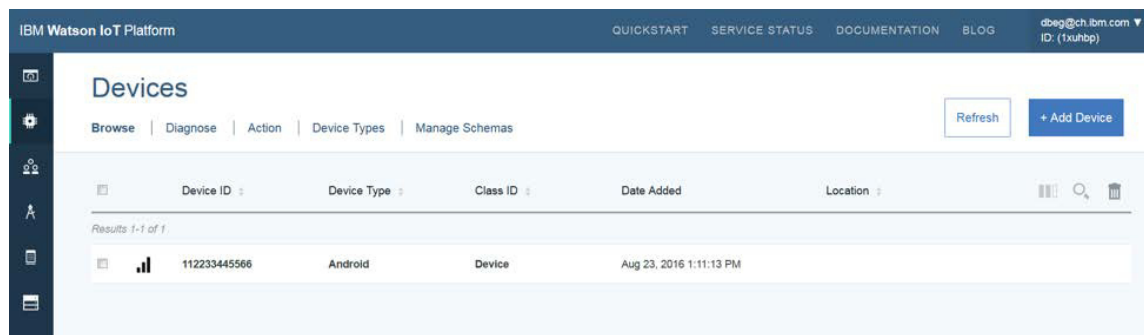


Рисунок 2.14 – Устройство зарегистрировано в системе

Проверка отправки сообщений с сервера на IoT шлюз. Откройте на компьютере страницу сервера IBM IoT для организации (см. начало этапа Добавление устройства, которое будет отправлять сообщения на сервер IoT). На вкладке Devices отображается устройство для работы в системе.

Щелкните по идентификатору устройства. Появится всплывающее окно с сообщениями, поступающими с вашего устройства. Щелкните по событию, чтобы увидеть подробные сведения (рисунке 2.15).

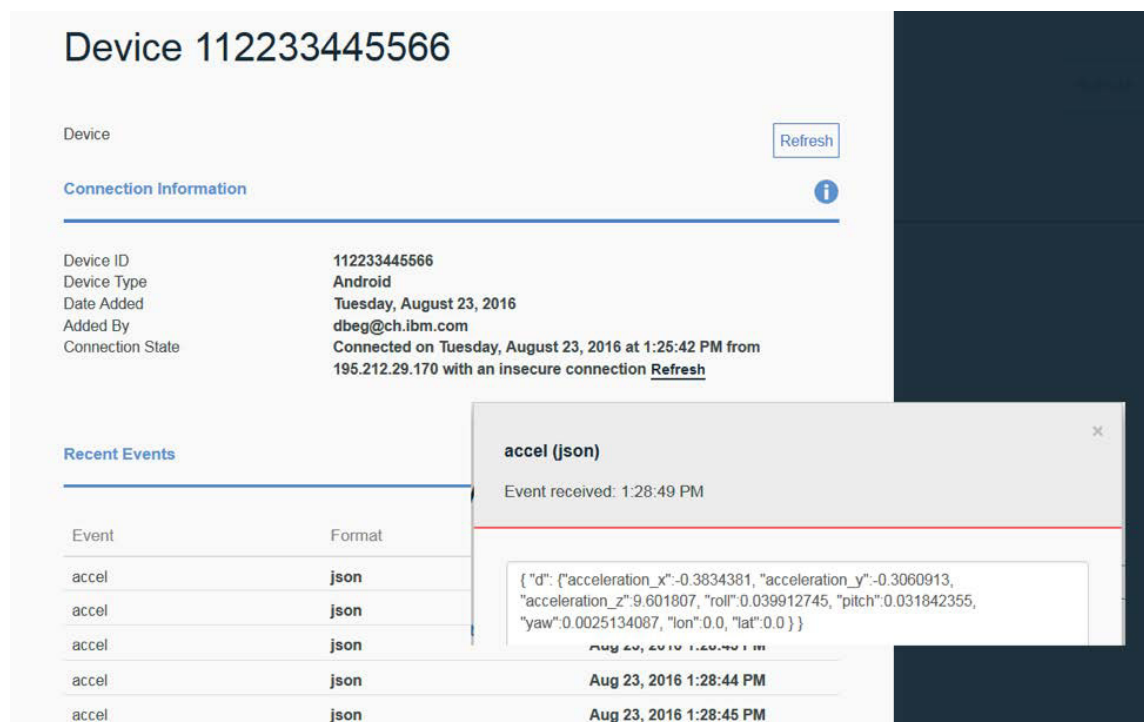


Рисунок 2.15 – Проверка работоспособности устройства

Сообщения с сервера отправляются в формате JSON. Они содержат данные об ускорении и позиции.

Теперь можно работать с данными сообщений в системе.

2.4 Реализация frontend части приложения

Требования к функционалу frontend части системы сформулируем в виде таблицы (таблица 2.16).

Таблица 2.16 - Проверить устройство в IoT сети

№	Функционал	Описание функционала
1	Подключение к сети новых устройств	Интерфейс должен отображать автоматическую настройку подключаемых модулей. Любое новое устройство или любое изменение в существующих устройствах должно быть зафиксировано в списке устройств. Пользователь должен выполнить минимальное количество действий, чтобы увидеть эти обновления на главной странице. Изменения топологии сети необходимо обновить в интерфейсе, не влияя на взаимодействие с пользователем. Связь между платами среднего звена и конечным устройством должна быть понятной пользователю.
2	Управление сервером (бизнес логикой)	Пользователь имеет возможность получать информацию о виртуальных машинах, созданных в качестве промежуточного программного обеспечения в облаке (например, в Yii PHP Framework или IBM Blumex).
3	Выборка шлюзов	Просмотр информации о шлюзах, с возможностью просмотра информации о состоянии шлюза (заряд батареи, состоянии периферийных портов)

Продолжение таблицы 2.16

4	Управление политикой сети шлюзов	Возможность контролировать доступ к шлюзам, а также шлюзов к конкретным устройствам
5	Функция поиска	Многофункциональный поиск по критериям
6	Логгирование ошибок	Функции приема сообщений об ошибках от шлюзов и формирование логов.

Работа системы начинается с запуска приложения в браузере по адресу <https://127.0.0.1:5555>. Для работы системы требуется работать по протоколу https, т.к. сервер Blumex позволит работать интерфейсу только по защищенному каналу.

Для начала работы требуется выбрать ссылку «Список устройств», после чего незарегистрированный пользователь попадает на страницу работы с устройствами (рисунки 2.16 - 2.18).

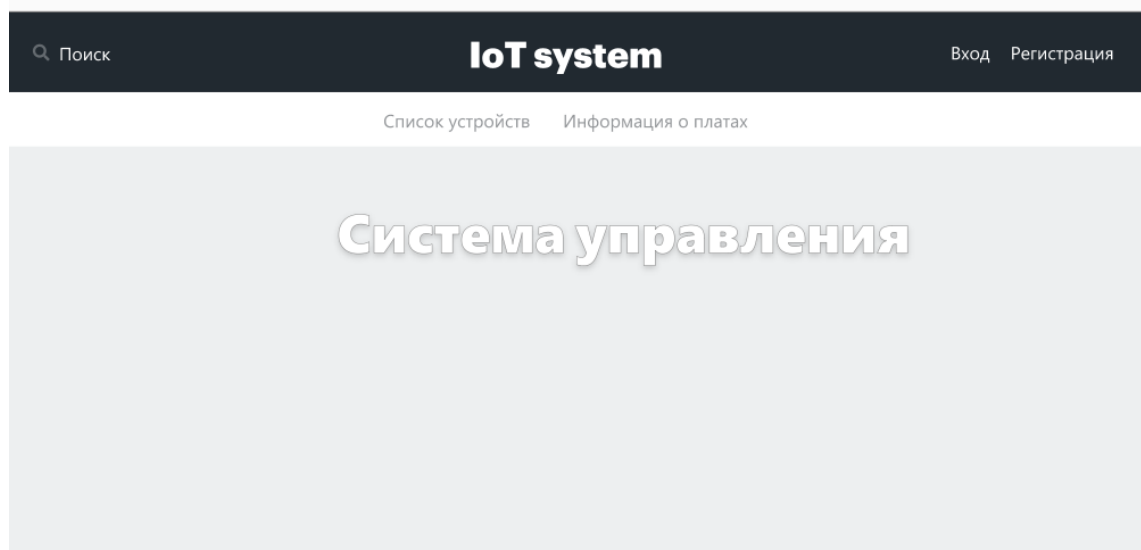


Рисунок 2.16 – Приветственная страница приложения

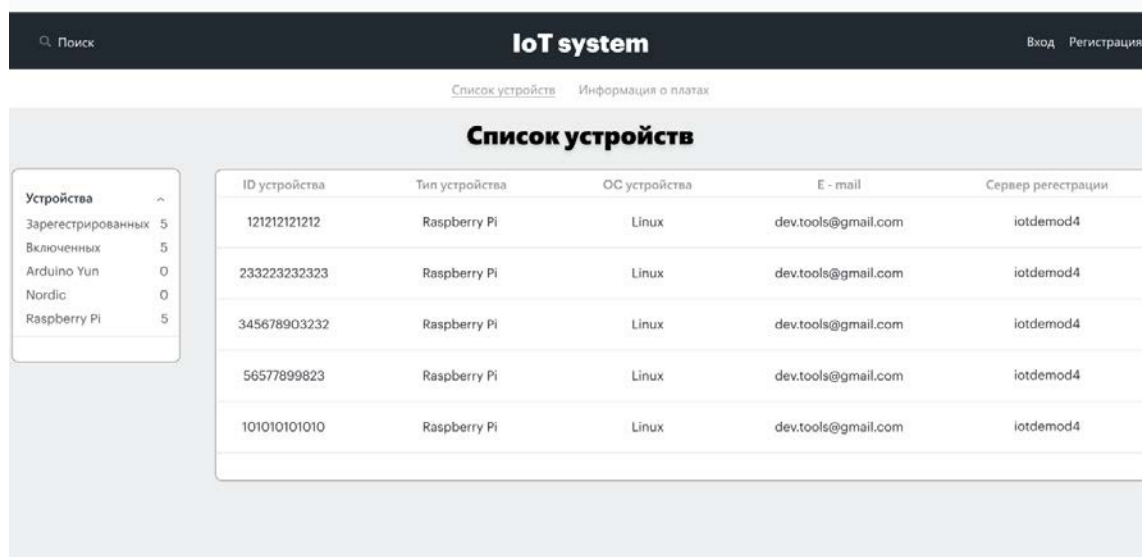


Рисунок 2.17 – Страница работы с устройствами



Рисунок 2.18 – Страница информации об устройствах, размещенных в системе

На странице «Список устройств» незарегистрированный пользователь видит список доступных системе устройств, каждый элемент является ссылкой, кликнув на которую, пользователь попадает на страницу работы с устройством (рисунках 2.19 - 2.21).

Поиск

IoT система

Вход Регистрация

[Список устройств](#)
[Информация о платах](#)

ID устройства: 1212121212

Тип устройства: Raspberry Pi

ОС устройства: Linux

E - mail: dev.tools@gmail.com

Сервер устройства: iotdemon4

Состояние: Включено

Обновить информацию об устройстве: Применить

Получить информацию об устройстве: Нет прав

Удалить устройство: Нет прав

Команда устройству: Нет прав

Для полной возможности работы с устройством необходимо Войти или Зарегистрироваться.

Рисунок 2.19 – Страница работы с устройством,
размещенным в системе

Поиск

IoT система

Вход Регистрация

[Список устройств](#)
[Информация о платах](#)

Регистрация

E - mail

Имя

Фамилия

Пароль

Повторите пароль

Подтвердить

Отмена

Рисунок 2.20 – Страница регистрации пользователя в системе

Поиск

IoT система

iot.sample58@gmail.com

Список устройств

Информация о платах

ID устройства: 121212121212

Тип устройства: Raspberry Pi

ОС устройства: Linux

E - mail: dev.tools@gmail.com

Сервер устройства: iotdemod4

Состояние: Включено

Обновить информацию об устройстве:

Применить

Получить информацию об устройстве:

Применить

Удалить устройство: Нет прав

Команда устройству: Нет прав

Полученная информация об устройстве:

×

IP устройства: 172.16.55.88

Свободная память: 4 Мб

SSD всего: 32 Мб

SSD свободно: 4 Мб

Температура процессора: 54 °C

Рисунок 2.21 – Страница работы с устройством
в профиле пользователя.

Зарегистрированный пользователь имеет определенные права на взаимодействие с устройством (рисунок 2.20). Пользователь имеет возможность обновить информацию и увидеть информацию о нем на информационной панели страницы.

Административные права дают возможность обновить информацию об устройстве, получить информацию об устройстве, удалить устройство и выполнить команду в отношении устройства.

Список команд на исполнении в рамках программного интерфейса ограничен функциональными возможностями библиотеки melibcom (php модуль позволяющий исполнить 10 команд в системе IoT устройств) (рисунках 2.22 – 2.25).

Поиск

IoT система

iotadm.sample58@gmail.com

Список устройств

Информация о платах

Пользователи

ID устройства: 1212121212

Тип устройства: Raspberry Pi

ОС устройства: Linux

E - mail: dev.tools@gmail.com

Сервер устройства: iotdemod4

Состояние: Включено

Обновить информацию об устройстве:

Применить

Получить информацию об устройстве:

Применить

Удалить устройство:

Применить

Команда устройству:

Применить

Удаление устройства:

×

Устройство будет удалено из виртуальной системы. Восстановление системы будет возможным только через сервис Blumex.

Подтвердить

Рисунок 2.22 – Удаление устройства в профиле администратора

IoT система

iotadm.sample58@gmail.com

Список устройств

Информация о платах

Пользователи

ID устройства: 1212121212

Тип устройства: Raspberry Pi

ОС устройства: Linux

E - mail: dev.tools@gmail.com

Сервер устройства: iotdemod4

Состояние: Включено

Обновить информацию об устройстве:

Применить

Получить информацию об устройстве:

Применить

Удалить устройство:

Применить

Команда устройству:

Применить

Команда устройству:

×

Ping 172.16.55.88

Отправить

Рисунок 2.23 – Выполнение команды в профиле администратора

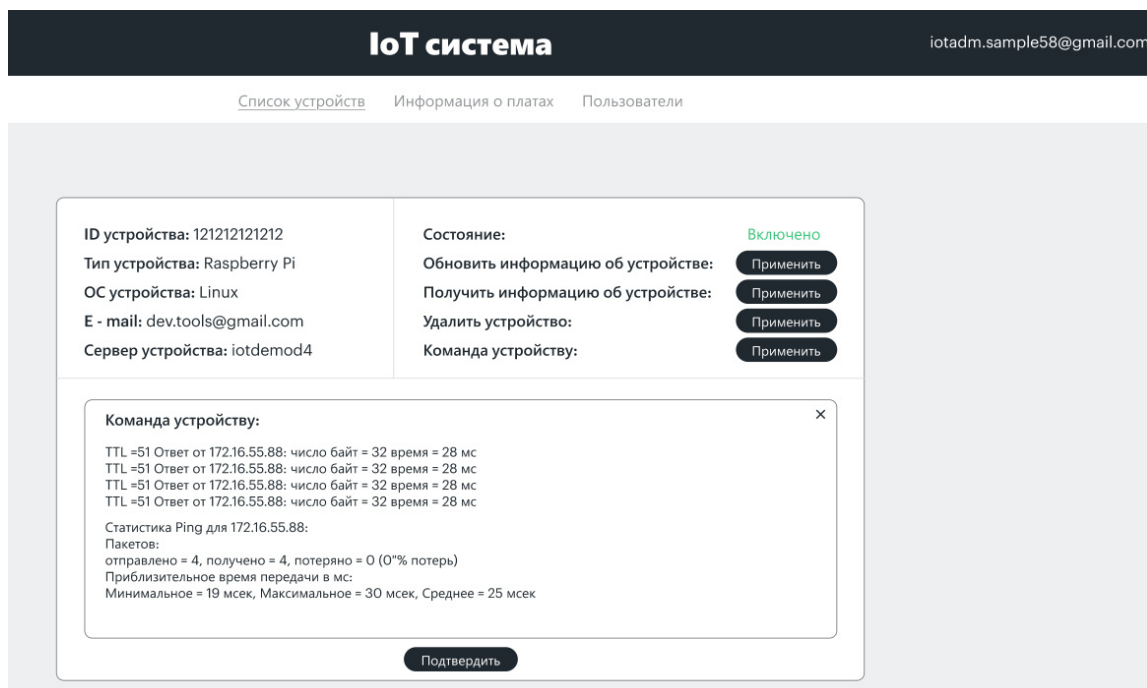


Рисунок 2.24 – Результат выполнения команды
в профиле администратора



Рисунок 2.25 – Результат выполнения команды
в профиле администратора

После возвращения на основную страницу администратор видит список устройств и кнопку добавить устройство, которая позволяет ему зарегистрировать в системе новое устройство без входа пользователя в систему Blumex (рисунках 2.26 – 2.30).

IoT system

iotadm.sample58@gmail.com

[Список устройств](#)
[Информация о платах](#)

Список устройств

Добавление нового устройства

ID устройства

1212121212

Тип устройства

Raspberry Pi

E - mail

dev.tools@gmail.com

IP адрес устройства

172.16.55.45

Сервер регистрации

lotdemod4.blumex.com

Добавить

Отменить

ID устройства	Тип устройства	IP адрес устройства	E - mail	Сервер регистрации
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4

Рисунок 2.26 – Добавление устройства в профиле администратора

IoT system

iotadm.sample58@gmail.com

[Список устройств](#)
[Информация о платах](#)

Список устройств

ID устройства	Тип устройства	IP адрес устройства	E - mail	Сервер регистрации
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4
1212121212	Raspberry Pi	172.16.55.45	dev.tools@gmail.com	lotdemod4

!

В систему добавлено новое устройство. Сервер Blumex запустит данное устройство. Это может занять несколько минут. После чего устройство будет отображаться в списке. Вся информация о состоянии устройства отправляется на e-mail: dev.tools@gmail.com

Рисунок 2.27 – Процесс регистрации устройства в системе

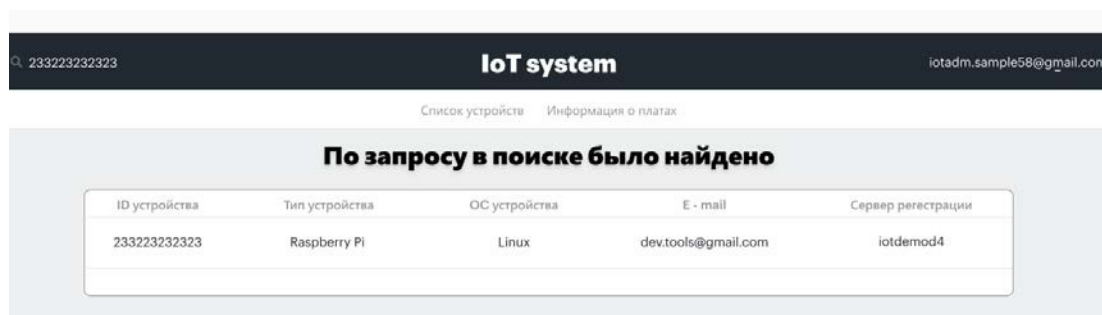


Рисунок 2.28 – Процесс поиска устройства в системе.

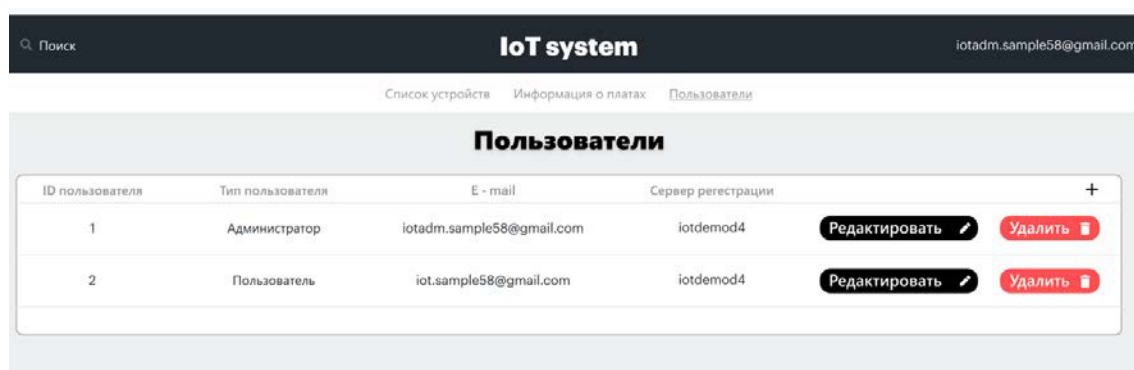


Рисунок 2.29 – Процесс поиска устройства в системе

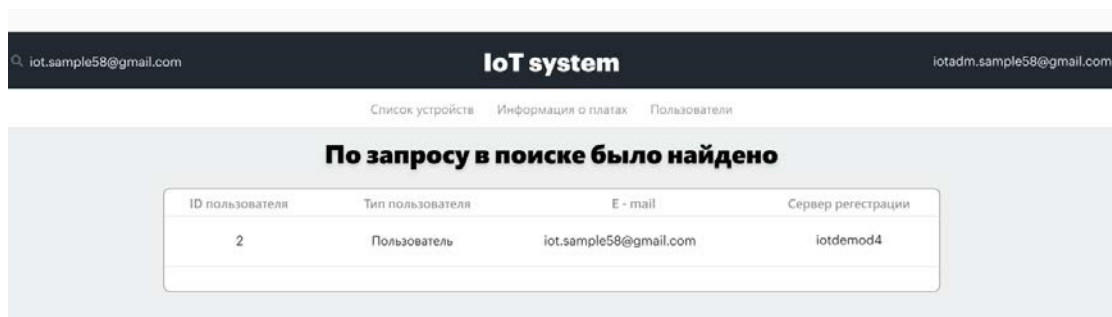


Рисунок 2.30 – Процесс поиска устройства в системе

Администратор имеет возможность управления не только устройствами. Но и пользователями системы. Администратор имеет возможность редактировать профиль пользователя и/или удалить его.

Восстановление удаленного пользователя возможно только в системе Blumex(рисунке 2.31).

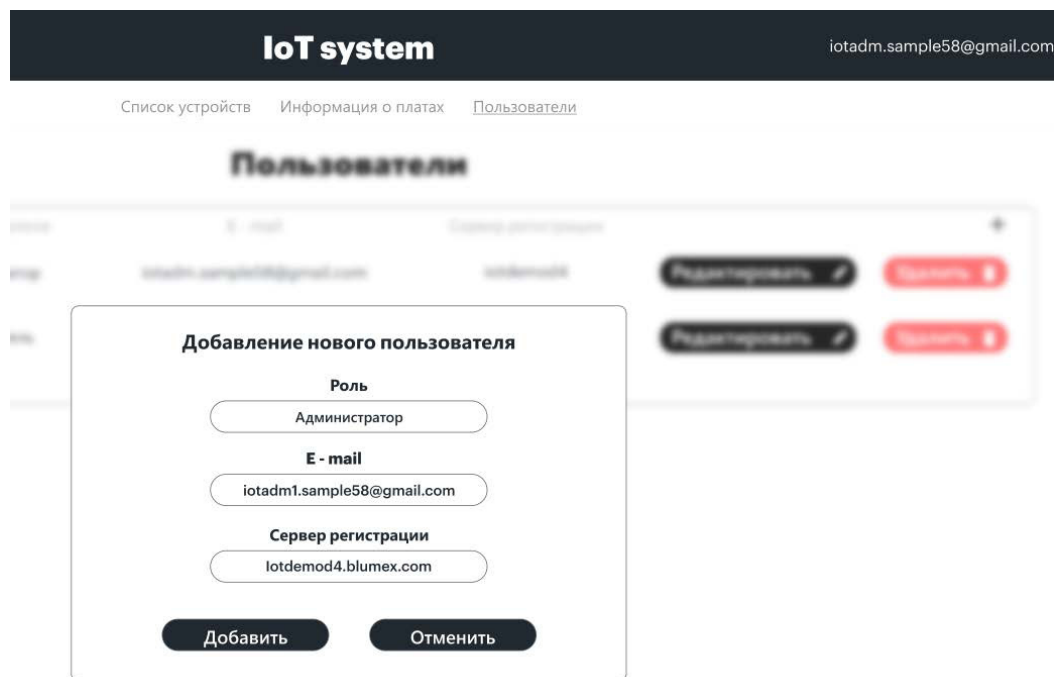


Рисунок 2.31 – Процесс поиска устройства в системе.

Добавление пользователя возможно при наличии у него e-mail адреса, на который система Blumex пришлет уведомление о регистрации в системе, которое пользователь должен будет перейти.

2.5 Выводы по главе

В результате второй главы дипломного проекта разработан web интерфейс взаимодействия с IoT системой Blumex, который:

- реализует функции управления устройствами системы;
- реализует функции управления пользователями системы;
- передает полученные данные на сервер сбора и визуализации данных об устройствах интернета вещей.

3 Технология разработки приложений os android в ide eclipse

Скачиваем архив с расширением rgr с официального сайта по адресу <http://nginx.org/ru/download.html>. Стабильной версией для ОС Windows сейчас 1.16.1. Рассмотрим установку на примере последней 10 версии.

Распаковываем архив в любую папку, однако для целей правильной установки желательно поместить содержимое архива в директорию локального диска, где размещена сама ОС. Лучшим вариантом папки будет системная папка "Program Files" на системном диске.

Находясь в папке архива веб сервера, найдите файл инсталляции с расширением Nginx.exe. Дайте ОС команду на запуск файла удобным для вас способом, например двойным щелчком кнопки мыши. После необходимо проверить результат, для чего зайдите в браузер по умолчанию и в адресной строке пропишите "localhost" Положительный ответ сервера будет содержать текст как рисунку 3.1.



Рисунок 3.1 – Приветственное окно

Остановить работу веб сервера Nginx можно через диспетчера задач ОС (рисунок 3.2).

Name	Status	11% CPU	48% Memory	10% Disk	0% Network
Microsoft Network Realtime Ins...		0%	2.7 MB	0 MB/s	0 Mbps
Microsoft Office Click-to-Run (...)		0%	4.9 MB	0 MB/s	0 Mbps
SkypeApp (3)		0%	1.4 MB	0 MB/s	0 Mbps
Microsoft Store (2)		0%	2.7 MB	0 MB/s	0 Mbps
Microsoft Text Input Application		0%	6.7 MB	0 MB/s	0 Mbps
Microsoft Windows Search Filte...		0%	1.1 MB	0 MB/s	0 Mbps
Microsoft Windows Search Inde...		0%	22.8 MB	0 MB/s	0 Mbps
Microsoft Windows Search Prot...		0%	1.6 MB	0 MB/s	0 Mbps
Movies & TV (2)		0%	1.4 MB	0 MB/s	0 Mbps
nginx.exe (32 bit)		0%	1.2 MB	0 MB/s	0 Mbps
nginx.exe (32 bit)		0%	1.6 MB	0 MB/s	0 Mbps
Node.js: Server-side JavaScript		0%	10.2 MB	0 MB/s	0 Mbps
PresentationFontCache.exe		0%	0.6 MB	0 MB/s	0 Mbps

Рисунок 3.2 – Просмотр запущенных процессов

Запуск сервера на ОС Windows 10 имеет свои особенности. Нам потребуется задействовать родной для ОС веб сервер ISS. Через панель управления перейдите в установку и удаление программ и справа найдите кнопку Turn Windows Features On or Off.

Вам необходимо найти опции и включить процесс, отметив соответственно поля текущего инструмента веб разработки (IIS Management Console в группе Web Management Tools) (рисунок 3.3).

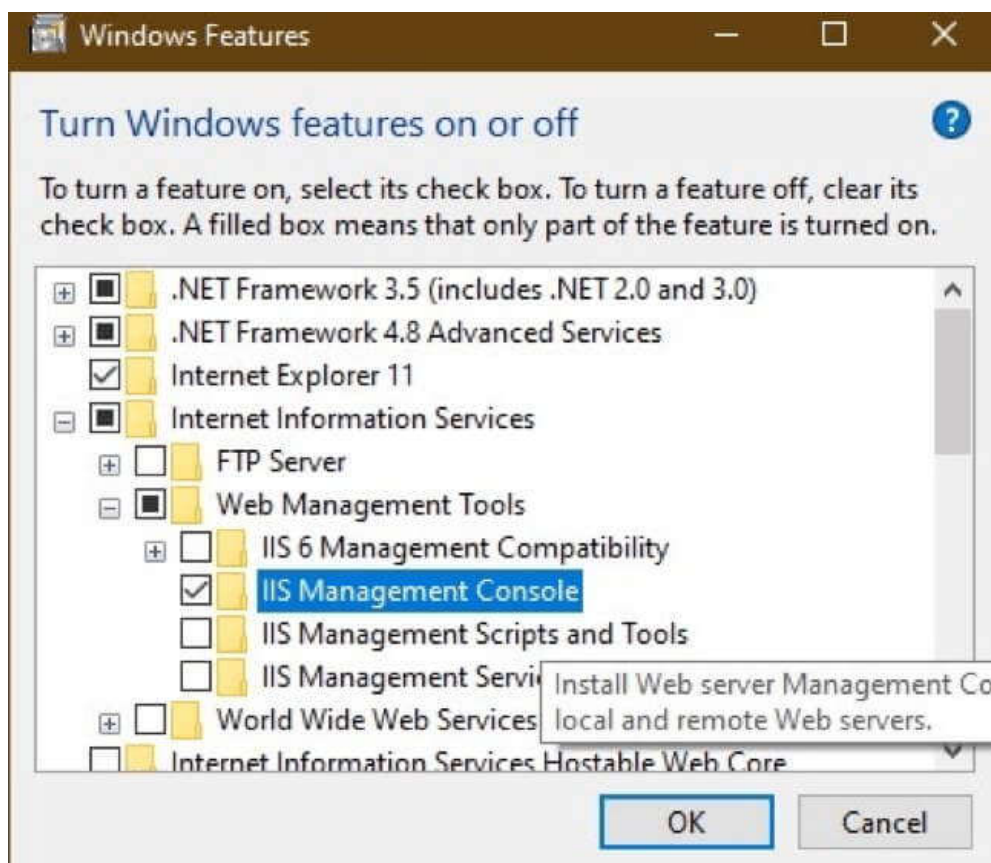


Рисунок 3.3 – Настройка консоли

Через несколько минут сервер будет доступен для использования. Консоль менеджера сервера ISS, в котором будет находиться сайт по умолчанию можно открыть непосредственно из меню "Пуск". Для удобства пользования измените путь к нему с "inetpub wwwroot" на C: \ "Имя вашей папки", например C: \ Work (рисунок 3.4).

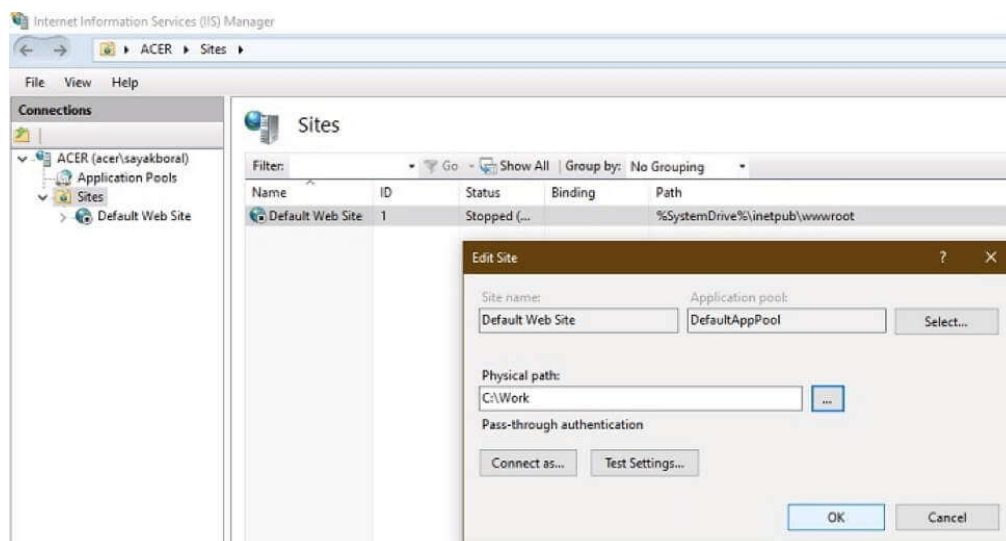


Рисунок 3.4 – Выбор и настройка рабочего каталога

В каталоге с установленным сервером Nginx найдите папку conf, а в ней файл nginx.conf, откройте его любым текстовым редактором (рекомендуется Notepad). Отредактируйте строку в корне локальной ветки пути к сайту по умолчанию, заменив предыдущий путь на созданный нами. Для корректной работы Nginx необходимо настроить firewall, открыть порты для подключения к серверу (рисунок 3.5). Чтобы открыть доступ для подключения к серверу по HTTPS, выполните команды:

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables-save
```

Так как планируется использовать SSL-сертификат, для подключения по HTTPS выполните команду:

```
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables-save
```

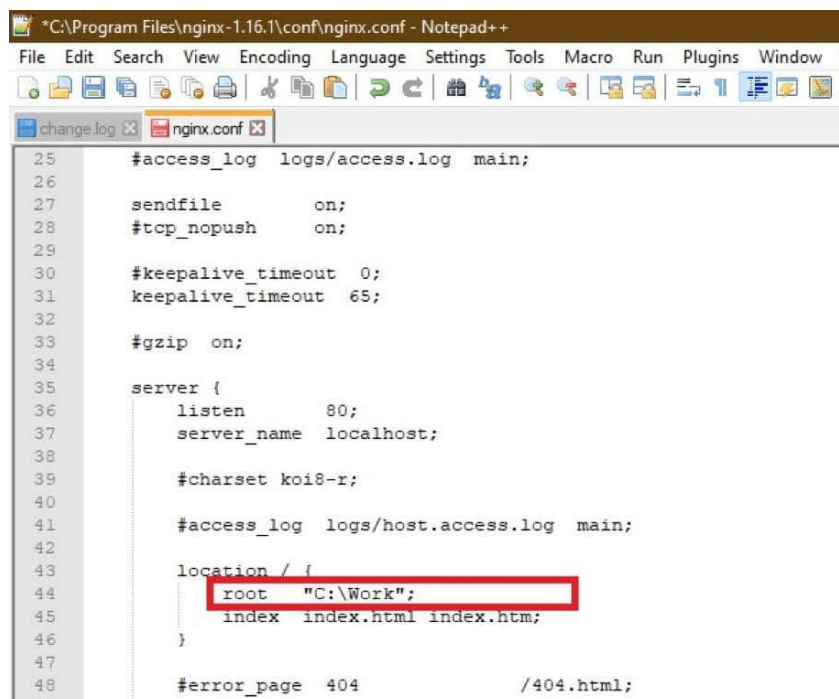


Рисунок 3.5 – Конфигурационный файл

Используя отдельную вкладку редактора, также можно изменить приветственный текст на главной странице веб сервера Nginx, например, "Hello". При очередном рабочем запуске веб сервера он будет отображать ее в рабочем окне браузера. Однако снова запустите исполняемый файл веб сервера nginx.exe, и в адресной строке браузера введите его локальный адрес (localhost) (рисунок 3.6).

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Рисунок 3.6 – проверка работоспособности SSL

4 Технико-экономическое обоснование разработки сайта «сервис совместного поиска попутчиков»

4.1 Расчет себестоимости программного продукта

Процесс разработки Интернет-сайта включает в себя следующие основные этапы:

- анализ требований к сайту;
- проектирование, составление алгоритма;
- написание самой программы;
- тестирование и отладка.

При этом первые два этапа представляют собой интеллектуальный труд разработчика, который не может быть автоматизирован, а время, затрачиваемое на выполнение этих этапов, зависит только от накопленных знаний и квалификации программиста. Следующие этапы представляет собой работу при переводе разрабатываемой программы на язык программирования и устранения всех ошибок.

При определении затрат времени на разработку Интернет-сайта был использован метод экспертной оценки. В качестве экспертов выступают разработчик программы и руководитель проекта.

На основании экспертных оценок затрат времени определяется средняя оценка, которая рассчитывается по формуле 4.1:

$$t_{cp} = (3t_p + 2t_c) / 5, \quad (4.1)$$

где t_p – оценка, данная руководителем проекта;

t_c – оценка, данная автором проекта или другим специалистом;

t_{cp} – средняя оценка.

Экспертные оценки времени делаются для трех ситуаций:

- наименее возможная величина затрат, a_i ;
- наиболее вероятная величина затрат, m_i ;
- наиболее возможная величина затрат, b_i .

Результаты расчета средней оценки затрат времени приведем в таблице 4.1.

Таблица 4.1 - Экспертные и средние оценки затрат времени в днях

Этапы разработки программы	Наименее возможная величина затрат, A_i			Наиболее вероятная величина затрат, M_i			Наиболее возможная величина затрат, b_i		
	tp	tc	Tcp	tp	Tc	tcp	tp	tc	tcp
Анализ требований	2,0	2,0	2,0	3,0	4,0	3,4	4,0	4,5	4,2
Проектирование, составление алгоритма	3,0	3,5	2,6	4,0	4,5	4,2	5,0	6,0	5,4
Написание программы	6,5	7,0	6,7	7,0	8,0	7,4	9,0	10,0	9,4
Тестирование, отладка	6,0	7,5	6,6	7,0	8,0	7,4	12	12	12

Далее определяется математически ожидаемая величина затрат времени и стандартное отклонение для каждого этапа по формулам 4.2-4.3:

$$MO_i = (a_i + 4m_i + b_i)/6, \quad (4.2)$$

$$G_i = (b_i - a_i)/6, \quad (4.3)$$

где a_i , b_i , m_i – средние оценки.

Общая величина математического ожидания и общая величина стандартного отклонения рассчитываются по формулам 4.4-4.5:

$$MO = \sum MO_i, \quad (4.4)$$

$$G = \sqrt{\sum G_i^2} \quad (4.5)$$

Коэффициент вариации:

$$V_i = G/MO * 100\%, \quad (4.6)$$

где i – меняется от 1 до n (количество этапов разработки).

Таблица 4.2 - Математически ожидаема величина затрат времени и стандартное отклонение

Этапы разработки программы	Математически ожидаемая величина затрат времени, MO_i	Стандартное отклонение G_i	Коэффициент вариации, V_i
Анализ требований, t_1	3,3	0,4	12,1
Проектирование, t_2	4,2	0,5	11,9
Написание программы, t_3	7,7	0,5	6,5
Тестирование, t_4	9,8	0,9	9,2

Себестоимость разработки программного продукта включает в себя затраты на зарплату разработчика, а также отчисления во внебюджетные фонды и затраты, связанные с использованием машинного времени. Тогда формула для расчета себестоимости выглядит следующим образом по формуле 4.7:

$$C = (3/m) * k * (t_1 + t_2) * (1 + k_n) + 8 * t_3 * C_m + 8 * t_4 * C_1 \quad (4.7)$$

где 3 – среднемесячная зарплата программиста (примем $3 = 20000$ руб.);

m – количество рабочих дней в месяце (примем $m = 22$ дня);

t_1 – время, затраченное на анализ требований к программному продукту, чел.дн.

t_2 – время, затраченное на отладку программы, чел.дн.

t_3 – время, затраченное на разработку программы с использованием машинного времени;

k – коэффициент, учитывающий начисления во внебюджетные фонды, $k = 1,302$;

k_n – коэффициент, учитывающий накладные расходы, $k_n = 2$.

C_m – стоимость 1 часа машинного времени;

C_1 – стоимость 1 часа работы в сети интернет.

Произведем расчет стоимости одного часа работы машины, для этого воспользуемся формулой 4.8:

$$C_m = \frac{Z_{\text{эПК}}}{T_{\text{эПК}}} \quad (4.8)$$

где $Z_{\text{эПК}}$ – годовые текущие затраты на эксплуатацию персонального компьютера

$T_{\text{эПК}}$ – годовой фонд времени полезной работы персонального компьютера.

Годовой фонд времени полезной работы ПК рассчитывается по формуле 4.9:

$$T_{\text{эПК}} = n - (\pi + P) \quad (4.9)$$

где n - количество рабочих часов в год. ($n = 1987$ часов).

π - профилактика ($5 \text{ ч/мес} * 12 \text{ мес} = 60$ часов);

P – ремонты в год ($6 \text{ дн} * 8 \text{ ч/дн} = 48$ часов);

$T_{\text{эПК}} = 1879 \text{ ч}$.

В свою очередь, затраты на эксплуатацию компьютера, вычисляются по формуле 4.10:

$$Z_{\text{эПК}} = Z_{\text{ЗП}} + Z_{\text{Э/Э}} + Z_{\text{гпр}} \quad (4.10)$$

где $Z_{\text{ЗП}}$ – годовые затраты на заработную плату обслуживающего персонала;

$Z_{\text{Э/Э}}$ – годовые затраты на электроэнергию;

Згпр – прочие затраты в год (включают ремонт, расходные материалы и т.п.).

Годовые затраты на заработную плату обслуживающего персонала можно определить по формуле 4.11:

$$\text{ЗгЗП} = \text{ЗОП} \cdot \text{кр} \cdot \text{кст} \cdot 12 \quad (4.11)$$

где ЗОП – заработная плата обслуживающего персонала в месяц. ЗОП = 1000 руб;

кр – районный коэффициент (1,2);

кст – страховые взносы (1,302).

$$\text{ЗгЗП} = 1000 \cdot 1,2 \cdot 1,302 \cdot 12 = 18748.8 \text{ руб.}$$

Годовые затраты электроэнергии можно определить по формуле 4.12:

$$\text{ЗгЭ/Э} = \text{МуПК} \cdot \text{ТгЭПК} \cdot \text{ЦЭ/Э} \quad (4.12)$$

где МуПК – установочная мощность ПК (0,4 кВт/час);

ЦЭ/Э – цена электроэнергии в руб./кВт-ч (1,76 руб./кВт-ч)

$$\text{ЗгЭ/Э} = 0,4 \cdot 1879 \cdot 1,76 = 1322.8 \text{ руб.}$$

Прочие затраты можно принять равными 5% от стоимости компьютера:

$$\text{Згпр} = 20000 \cdot 0,05 = 1000 \text{ руб.}$$

Используя формулу (4.8) можно получить стоимость одного часа работы с компьютером: $\text{См} = 21071.6 / 1879 = 11,21 \text{ руб.}$

Зная все составляющие формулы, рассчитаем затраты на разработку программного продукта: $\text{С} = (22000/22) \cdot 1,302 \cdot (3.3 + 4.2) \cdot (1+2) + 8 \cdot 7.7 \cdot 15 + 8 \cdot 9.8 \cdot 0.98 = 30295.83 \text{ руб.}$

Сравним полученную стоимость со стоимостью доработки программного обеспечения организацией разработчиком либо силами сторонних разработчиков.

Исходя из среднерыночной стоимости часа разработки программного обеспечения на рынке 100 у.е. за час, исходя из минимальных 14 часов необходимых на разработку, получаем, что минимальная стоимость может

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

составить 1400 у.е., при средних расчетных сроках разработки сумма составит 2170 у.е. Таким образом, можно сделать вывод, что более выгодным для организации является разработка операции собственными силами, без привлечения сторонних организаций.

Отсюда программный продукт может быть теоретически продан, а значит для оценки по формуле 4.13:

$$\text{Ц} = \text{С} * (1 + (\text{Р}/100)), \quad (4.13)$$

где С – себестоимость данного программного продукта;

Р – рентабельность (примем Р = 30%);

$$\text{Ц} = 30295.83 * (1 + (30/100)) = 39384.58 \text{ руб.}$$

Цена с налогом на добавленную стоимость рассчитывается по формуле 4.14:

$$\text{Цс НДС} = \text{Ц} * \text{КнДС}, \quad (4.14)$$

где КнДС – коэффициент НДС, равный 1,1; Ц - цена без НДС;

$$\text{Цс НДС} = 39384,58 * 1,20 = 46473,81 \text{ руб.}$$

4.2 Эффективность разработки сайта

К основным задачам, которые могут быть решены с помощью Интернет-сайта, можно отнести:

- предоставление информации о себе, о своих товарах и услугах;
- выход на новые рынки;
- формирование постоянной целевой аудитории;
- получение отклика от клиентов.

Имея свой сайт в сети Интернет, появляется возможность существенно сэкономить на вложении средств в другие виды рекламы. Указав ссылку на телеэкране в ходе видеоролика или добавив ее в макет графического блока в журнале или газете, исчезнет необходимость детального освещения сфер

деятельности, размещения исчерпывающей информации о предлагаемых товарах и услугах.

Интернет-сайт доступен 24 часа в сутки, а значит, в любое время сможет предоставлять необходимую информацию.

Правильно спроектированный сайт обладает возможностью безграничного расширения информационной структуры – разделы, подразделы, страницы и т.д. Если сайт легче управляется, значит и развивается быстрее, следовательно – быстрее окупается его разработка.

На сайте реализуется:

- предоставление информации о себе, об услугах;
- формирование постоянной целевой аудитории;
- получение отклика от пользователей;
- оповещение о важных событиях организации;
- демонстрация результатов работы организации.

В данный момент продукцию организации рекламируют в журналах, на радио, других сайтах. Годовые затраты на рекламу приведены в таблице 4.3.

Таблица 4.3 – Годовые затраты на рекламу

Виды рекламы	До появления сайта, тыс. руб.	После появления сайта, тыс. руб.
Журналы	180	-
Радио	280	280
Другие сайты	200	80
Итого	560	360

Затраты на рекламу были предоставлены организацией. После создания сайта для организации появится возможность отказаться от какого-то вида рекламы или сократить количество рекламодателей, как в случае с другими сайтами, и получить экономию, которая пойдет на окупаемость сайта.

Рассчитаем годовую экономию по рекламе по формуле 4.15:

$$\Delta \text{Эгод} = A - B, \quad (4.15)$$

где $\Delta \text{Эгод}$ – годовая экономия по рекламе;

A - затраты на рекламу до появления сайта;

B - затраты на рекламу после появления сайта.

$$\Delta \text{Эгод} = 660 - 360 = 300 \text{ тыс. руб.}$$

Период окупаемости затрат на приобретение веб-сайта компанией вычисляется по формуле 4.16:

$$\text{По} = C / \Delta \text{Эгод} \quad (4.16)$$

где C – себестоимость разработанного программного продукта;

$\Delta \text{Эгод}$ – годовая экономия по рекламе.

$$\text{По} = 39384,58 / 300000 = 0,10 \text{ года}$$

Таким образом, после внедрения данного программного продукта он окупится через 0,10 года.

5 Безопасность жизнедеятельности. Организация рабочего места программиста

5.1 Характеристика условий труда рабочего места программиста

Научно-технический прогресс внес серьезные изменения в условия производственной деятельности работников умственного труда. Их труд стал более интенсивным, напряженным, требующим значительных затрат умственной, эмоциональной и физической энергии. Это потребовало комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

В настоящее время компьютерная техника широко применяется во всех областях деятельности человека. При работе с компьютером, программист подвергается воздействию вредных электромагнитных полей (диапазон радиочастот ВЧ, УВЧ и СВЧ), инфракрасного и ионизирующего излучений, статического электричества и т.д.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы программиста.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

Данный раздел дипломного проекта посвящен рассмотрению следующих вопросов по организации рабочего места программиста:

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

- определение оптимальных условий работы программиста;
- расчет освещенности рабочего места программиста;
- расчет уровня шума на рабочем месте программиста.

5.2 Требование к рабочему помещению программиста

5.2.1 Окраска помещений

Окраска помещений и мебели должна способствовать созданию благоприятных условий для зрительного восприятия, хорошего настроения.

Источники света, такие как светильники и окна, которые дают отражение от поверхности экрана, значительно ухудшают точность знаков и влекут за собой помехи физиологического характера, которые могут выразиться в значительном напряжении, особенно при продолжительной работе. Отражение, включая отражения от вторичных источников света, должно быть, сведено к минимуму. Для защиты от избыточной яркости окон могут быть применены шторы и экраны.

В зависимости от ориентации окон рекомендуется следующая окраска стен и пола:

- окна ориентированы на юг: - стены зеленовато-голубого или светло-голубого цвета; пол - зеленый;
- окна ориентированы на север: - стены светло-оранжевого или оранжево-желтого цвета; пол - красновато-оранжевый;
- окна ориентированы на восток: - стены желто-зеленого цвета; пол зеленый или красновато-оранжевый;
- окна ориентированы на запад: - стены желто-зеленого или голубовато-зеленого цвета; пол зеленый или красновато-оранжевый.

В помещениях, где находится компьютер, необходимо обеспечить следующие величины коэффициента отражения: для потолка: 60-70%, для

стен: 40...50%, для пола: около 30%. Для других поверхностей и рабочей мебели: 30...40%.

5.2.2 Освещение рабочего места

Правильно спроектированное и выполненное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на программиста, повышает безопасность труда и снижает травматизм.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

Существует три вида освещения - естественное, искусственное и совмещенное (естественное и искусственное вместе).

Естественное освещение - освещение помещений дневным светом, проникающим через световые проемы в наружных ограждающих конструкциях помещений. Естественное освещение характеризуется тем, что меняется в широких пределах в зависимости от времени дня, времени года, характера области и ряда других факторов.

Искусственное освещение применяется при работе в темное время суток и днем, когда не удастся обеспечить нормированные значения коэффициента естественного освещения (пасмурная погода, короткий световой день). Освещение, при котором недостаточное по нормам

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

естественное освещение дополняется искусственным, называется совмещенным освещением.

Искусственное освещение подразделяется на рабочее, аварийное, эвакуационное, охранное. Рабочее освещение, в свою очередь, может быть общим или комбинированным. Общее - освещение, при котором светильники размещаются в верхней зоне помещения равномерно или применительно к расположению оборудования. Комбинированное - освещение, при котором к общему добавляется местное освещение.

Согласно СНиП 23-05-2010 в помещениях вычислительных центров необходимо применить систему комбинированного освещения.

При выполнении работ категории высокой зрительной точности (наименьший размер объекта различения 0,3...0,5 мм) величина коэффициента естественного освещения (КЕО) должна быть не ниже 1,5%, а при зрительной работе средней точности (наименьший размер объекта различения 0,5...1,0 мм) КЕО должен быть не ниже 1,0%. В качестве источников искусственного освещения обычно используются люминесцентные лампы типа ЛБ или ДРЛ, которые попарно объединяются в светильники, которые должны располагаться над рабочими поверхностями равномерно.

Требования к освещенности в помещениях, где установлены компьютеры, следующие: при выполнении зрительных работ высокой точности, общая освещенность должна составлять 300 лк, а комбинированная - 750 лк; аналогичные требования при выполнении работ средней точности - 200 и 300 лк соответственно.

Кроме того, все поле зрения должно быть освещено достаточно равномерно – это основное гигиеническое требование. Иными словами, степень освещения помещения и яркость экрана компьютера должны быть примерно одинаковыми, т.к. яркий свет в районе периферийного зрения значительно увеличивает напряженность глаз и, как следствие, приводит к их быстрой утомляемости.

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

5.2.3 Параметры микроклимата на рабочем месте

Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является поддержание постоянства температуры тела благодаря терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования микроклимата – создание оптимальных условий для теплообмена тела человека с окружающей средой.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата. В СНиП 41-01-2003 установлены величины параметров микроклимата, создающие комфортные условия. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (таблица. 5.1).

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше $19,5 \text{ м}^3/\text{человека}$ с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в таблице 5.2.

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха, отопительная система).

Таблица 5.1 Параметры микроклимата для помещений, где установлены компьютеры.

Период года	Параметр микроклимата	Величина
Холодный	Температура воздуха в помещении	22...24 °С
	Относительная влажность	40...60 %
	Скорость движения воздуха	до 0,1 м/с
Теплый	Температура воздуха в помещении	23...25 °С
	Относительная влажность	40...60 %
	Скорость движения воздуха	0,1...0,2 м/с

Таблица 5.2 Нормы подачи свежего воздуха в помещения, где расположены компьютеры

Характеристика помещения	Объемный расход подаваемого в помещение свежего воздуха, м ³ /на
Объем до 20 м ³ на человека	Не менее 30
20...40 м ³ на человека	Не менее 20
Более 40 м ³ на человека	Естественная вентиляция

5.2.4 Требования к шуму и вибрации в рабочих помещениях

Шум ухудшает условия труда оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д. Такие нарушения в работе ряда органов и систем организма человека могут вызвать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания,

нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением ухудшается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума [выше 80 дБ (А)] на слух человека приводит к его частичной или полной потере. В таблице 5.3 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности.

Таблица 5.3 Предельные уровни звука, дБ, на рабочих местах

Категория напряженности труда	Категория тяжести труда			
	I. Легкая	II. Средняя	III. Тяжелая	IV. Очень тяжелая
I. Мало напряженный	80	80	75	75
II. Умеренно напряженный	70	70	65	65
III. Напряженный	60	60	-	-
IV. Очень напряженный	50	50	-	-

Уровень шума на рабочем месте программиста не должен превышать 65 дБА. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами. Уровень вибрации в помещениях вычислительных центров может быть снижен путем установки оборудования на специальные виброизоляторы.

5.2.5 Режим труда программиста

Как уже было неоднократно отмечено, при работе с компьютером очень важную роль играет соблюдение правильного режима труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

В таблице 5.4 представлены сведения о регламентированных перерывах, которые необходимо делать при работе на компьютере, в зависимости от продолжительности рабочей смены, видов и категорий трудовой деятельности с ВДТ (видео дисплейный терминал) и ПЭВМ (в соответствии с СанПиН 2.2.4 3359-16 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах»).

Таблица 5.4 Время регламентированных перерывов

Категория работы с ВДТ или ПЭВМ	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Суммарное время регламентированных	
	Группа А, количество знаков	Группа Б, количество знаков	Группа В, часов	При 8-часовой смене	При 12-часовой смене
I	до 20 000	до 15 000	до 2,0	30	70
II	до 40 000	до 30 000	до 4,0	50	90
III	до 60 000	до 40 000	до 6,0	70	120

Примечание. Время перерывов дано при соблюдении указанных Санитарных правил и норм. При несоответствии фактических условий труда требованиям Санитарных правил и норм время регламентированных перерывов следует увеличить на 30%.

В соответствии со СанПиН 2.2.4 3359-16 все виды трудовой деятельности, связанные с использованием компьютера, разделяются на три группы:

- группа А: работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;
- группа Б: работа по вводу информации;
- группа В: творческая работа в режиме диалога с ЭВМ.

Эффективность перерывов повышается при сочетании с производственной гимнастикой или организации специального помещения для отдыха персонала с удобной мягкой мебелью, аквариумом, зеленой зоной и т.п.

5.3 Расчет освещенности рабочего места программиста

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

- по спектральному составу света они близки к дневному, естественному свету;
- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
- более длительный срок службы.

Расчет освещения производится для комнаты площадью 15 м^2 , ширина которой 5 м, высота - 3 м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле 5.1:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n}, \quad (5.1)$$

где F - рассчитываемый световой поток, Лм;

E - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет $E = 300 \text{ Лк}$;

S - площадь освещаемого помещения (в нашем случае $S = 15 \text{ м}^2$);

Z - отношение средней освещенности к минимальной (обычно принимается равным $1,1 \dots 1,2$, принимаем $Z = 1,1$);

K - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае $K = 1,5$);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризующих коэффициентами отражения от стен (P_c) и потолка (P_{Π})), значение коэффициентов P_c и P_{Π} были указаны выше: $P_c = 40\%$, $P_{\Pi} = 60\%$. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле 5.2:

$$I = \frac{S}{h(A+B)}, \quad (5.2)$$

где

S - площадь помещения, $S = 15 \text{ м}^2$;

h - расчетная высота подвеса, $h = 2.92 \text{ м}$;

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

А - ширина помещения, А = 3 м;

В - длина помещения, В = 5 м.

Подставив значения получим:

$$I = \frac{15}{2,92 \cdot (3+5)} = 0,64$$

Зная индекс помещения I, по таблице находим n = 0,22

Подставим все значения в формулу для определения светового потока F:

$$F = \frac{300 \cdot 1,5 \cdot 15 \cdot 1,1}{0,22} = 33750 \text{ Лм}$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых F = 4320 Лк.

Рассчитаем необходимое количество ламп по формуле 5.3:

$$N = \frac{F}{F_{\text{л}}}, \quad (5.3)$$

Где N - определяемое число ламп;

F - световой поток, F = 33750 Лм;

F_л - световой поток лампы, F_л = 4320 Лм.

$$N = \frac{33750}{4320} = 8 \text{ шт.}$$

При выборе осветительных приборов используем светильники типа ОД. Каждый светильник комплектуется двумя лампами.

5.4 Расчет уровня шума на рабочем месте

Одним из неблагоприятных факторов производственной среды в ИВЦ является высокий уровень шума, создаваемый печатными устройствами, оборудованием для кондиционирования воздуха, вентиляторами систем охлаждения в самих ЭВМ.

Для решения вопросов о необходимости и целесообразности снижения шума необходимо знать уровни шума на рабочем месте программиста.

Уровень шума, возникающий от нескольких некогерентных источников, работающих одновременно, подсчитывается на основании принципа энергетического суммирования излучений отдельных источников по формуле 5.4:

$$L_{\Sigma} = 10 \lg \sum_{i=1}^{i=n} 10^{0,1L_i}, \quad (5.4)$$

где L_i – уровень звукового давления i -го источника шума;

n – количество источников шума.

Полученные результаты расчета сравниваются с допустимым значением уровня шума для данного рабочего места. Если результаты расчета выше допустимого значения уровня шума, то необходимы специальные меры по снижению шума. К ним относятся: облицовка стен и потолка зала звукопоглощающими материалами, снижение шума в источнике, правильная планировка оборудования и рациональная организация рабочего места программиста.

Уровни звукового давления источников шума, действующих на рабочем месте, представлены в таблице 5.5. Обычно рабочее место программиста оснащено следующим оборудованием: винчестер в системном блоке, вентилятор(ы) систем охлаждения ПК, монитор, клавиатура, принтер и сканер.

Таблица 5.5 Уровни звукового давления различных источников

Источник шума	Уровень шума, дБ
Жесткий диск	40
Вентилятор	45
Монитор	17
Клавиатура	10

Продолжение таблицы 5.5

Принтер	45
Сканер	42

Подставив значения уровня звукового давления для каждого вида оборудования в формулу, получим:

$$L_{\Sigma} = 10 \cdot \lg (10^4 + 10^{4,5} + 10^{1,7} + 10^1 + 10^{4,5} + 10^{4,2}) = 49,5 \text{ дБ}$$

Полученное значение не превышает допустимый уровень шума для рабочего места оператора, равный 65 дБ (ГОСТ 12.1.003-83). И если учесть, что вряд ли такие периферийные устройства как сканер и принтер будут использоваться одновременно, то эта цифра будет еще ниже.

Кроме того, при работе принтера непосредственное присутствие необязательно, т.к. принтер снабжен механизмом автоподачи листов.

Заключение

В ходе выполнения дипломной работы было разработано web приложение, обеспечивающее управление средой интернет вещей, состоящей из различных устройств и облачного сервиса. Данный проект — это приложение на основе клиент серверного подхода, созданное на основе данных, собранных пользователем в облачной среде. Приложение интегрируется со сторонними службами для обеспечения безопасности работы с данными и конвертирования полученной информации на облачный сервер. Приложение построено с использованием технологий, которые являются производительными, гибкими и масштабируемыми.

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		84

Список использованных источников

1. Галанин А.Л. Сервисы IBM. Программирование устройств; БХВ-Петербург - Москва, 2018. - 148 с.
2. Нильсон Лорен, Вуден Шейн РНР за 24 часа. Программирование приложений; Рид Групп - Москва, 2017. - 445 с.
3. Примасов Д.У. Программирование для Yii2. Самоучитель; БХВ-Петербург - Москва, 2016. - 272 с.
4. Майер Рето Интернет вещей. Программирование приложений; Эксмо - Москва, 2015. - 672 с.
6. Медов А.Н., Гуревич Л.Ф., Миканкин Б.И. WEB программирование; Питер - Москва, 2018. - 560 с.

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		85

ПРИЛОЖЕНИЕ А
(обязательное)
ЛИСТИНГ ПРОГРАММЫ

					<i>ПГУ 1.09.05.01 010 001 ПЗ</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		<i>86</i>

```

index.php

<?php
require 'Slim/Slim.php';

\Slim\Slim::registerAutoloader();

$app = new \Slim\Slim();

$app->get( '/', function () use ($app)
{

    if(!isset($_SESSION))
    {
        session_start();
    }
    if ( !isset($_SESSION['username']) )
    {
        // header('Location:public/index.html');
        echo "<script language=\"javascript\">";

        echo "document.location=\"public/index.html\"";

        echo "</script>";
    }
    else
    {
        echo "<script language=\"javascript\">";

        echo "document.location=\"public/index.html\"";

        echo "</script>";
    }
    # $app->render('phpinfo.php',array('title'=>'PHP'));
});

```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

```

// GET /switches?page=1&rows=10
$app->get('/devices', function () use ($app)
{
    include 'conn.php';

    $sql = "select * from devices ";
    $rs = mysql_query($sql);
    $items = array();

    while($row = mysql_fetch_object($rs))
    {
        $row->description = urlencode($row->description );
        array_push($items, $row);
    }

    $app->response()->header('Content-Type', 'application/json');
    echo json_encode($items, JSON_NUMERIC_CHECK);
});

// GET /devices/:id
$app->get('/devices/:id', function ($id) use ($app)
{
    include 'conn.php';
    $sql = "select * from devices where id = '$id' limit 1";
    $rs = mysql_query($sql);

    if($row = mysql_fetch_object($rs))
    {
        $app->response()->header('Content-Type', 'application/json');

        echo json_encode( $row, JSON_NUMERIC_CHECK);
    }
    else
    {
        $app->response()->status(404);
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

88

```

    }
});

// GET /devices/:id/value
$app->get('/devices/:id/value', function ($id) use ($app)
{
    include 'conn.php';
    //$sql = 'select * from switches where id =' . $id;
    $sql = "select * from devices where id = '$id' limit 1";
    $rs = mysql_query($sql);
    $row = mysql_fetch_object($rs);

    if(null != $row)
    {

        if ("switch" == $row->type)
        {
            $device_value = "{\"switch\":\".(string)$row->value.\"}";
        }
        elseif ("step" == $row->type)
        {
            $device_value = (string)$row->value;
        }

        $app->response()->header('Content-Type', 'application/json');
        echo json_encode( json_decode($device_value,true), JSON_NUMERIC_CHECK);
    }
    else
    {
        $app->response()->status(404);
    }
});

// POST value
$app->post('/devices/:id', function ($id) use ($app)
{

```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

```

// the arduino post a string ,then encode it to a json.
$request = $app->request();
$body = $request->getBody();
$input = json_decode($body);
$type = (string)$input->type;
// $type = $_POST["type"];
include 'conn.php';
// classify three types from the json code items.

if ("switch" == $type)
{
    // $value = $_POST['value'];
    $value = $input->value;
}
elseif ("step" == $type)
{
    // $switch = $_POST['switch'];
    // $controller = $_POST['controller'];
    $switch = (string)$input->switch;
    $controller = (string)$input->controller;
    $value = '{"switch":'.$switch.', "controller":'.$controller.'}';
}
$sql = "update devices set value='$value' where id='$id'";
$result = @mysql_query($sql);
if ($result)
{
    //echo (string)$body;
    echo json_encode(array('success'=>true));
}
else
{
    echo json_encode(array('success'=>false));
}
});
// POST value publish

```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

```

$app->post('/mqttdevices/:id', function ($id) use ($app)
{
    // the arduino post a string ,then encode it to a json.
    $request = $app->request();
    $body = $request->getBody();
    $input = json_decode($body);
    $type = (string)$input->type;

    include 'conn.php';

    if("switch" == $type)
    {
        $value = $input->value;
        $device_value = "{\"switch\":\".(string)$value.\"}";
        $mqtt_message = json_encode( json_decode($device_value,true),
JSON_NUMERIC_CHECK);
    }
    elseif ("step" == $type)
    {
        $switch = (string)$input->switch;
        $controller = (string)$input->controller;
        $value = '{"switch":'.$switch.',"controller":'.$controller.'}';
        $mqtt_message = json_encode( json_decode($value, true), JSON_NUMERIC_CHECK);
    }

    $sql = "update devices set value='$value' where id='$id'";

    $result = @mysql_query($sql);

    require("public/phpMQTT.php");

    $mqtt = new phpMQTT("127.0.0.1", 1883, "phpMQTT"); //Change client name to something
unique

    if ($mqtt->connect())

```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						91
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    $mqtt->publish("d".$id, $mqtt_message, 0);
    $mqtt->close();
}

if ($result)
{
    echo json_encode(array('success'=>true));
}
else
{
    echo json_encode(array('success'=>false));
}
});

$app->get('/userall', function () use ($app)
{
    $username = htmlspecialchars($_GET['username']);
    // $password = $_POST['pwd'];
    include 'conn.php';
    if ("admin" === $username)
    {
        $sql = "select * from userlists";
        $rs = mysql_query($sql);
        $items = array();

        while($row = mysql_fetch_object($rs))
        {
            if ($row->username !== "admin")
            {
                array_push($items, $row);
            }
        }
    }

    $app->response()->header('Content-Type', 'application/json');
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

92

```

        echo json_encode($items);
    });

$app->get('/user', function () use ($app)
{
    $username = htmlspecialchars($_GET['username']);
    // $password = $_POST['pwd'];

    include 'conn.php';
    if ("admin" === $username )
    {
        $sql = "select * from userlists where username='$username' limit 1";
        $rs = mysql_query($sql);

        if($result = mysql_fetch_object($rs))
        {
            $app->response()->header('Content-Type', 'application/json');
            echo json_encode($result);
        }
    }

});

$app->post('/user', function () use ($app)
{
    $result = false;
    $request = $app->request();
    $body = $request->getBody();
    $input = json_decode($body);
    $username = (string)$input->username;
    $realname = (string)$input->realname;
    $email = (string)$input->email;
    $qq = (string)$input->qq;
    // $password = $_POST['pwd'];
    include 'conn.php';

```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		93

```

        $sql = "update userlists set realname='$realname', email='$email', qq='$qq' where
username='$username' limit 1";
        $rs = @mysql_query($sql);
        if($rs)
        {
        }
        });
$app->post('/userlogin', function ()
{
    $username = htmlspecialchars($_POST['username']);
    $password = $_POST['pwd'];

    include 'conn.php';

    $check_query = mysql_query("select * from userlists where username='$username' and
password='$password' limit 1");

    if ($result = mysql_fetch_array($check_query))
    {
        echo json_encode(array('username'=>$username));
        if(!isset($_SESSION))
        {
            session_start();
        }
        $_SESSION['username'] = $result['username'];
        $_SESSION['password'] = $result['password'];
    }
    else
    {
        //header('Location:/public/index.html');
    }
});

$app->post('/userdelete', function ()
{

```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

```

$username = htmlspecialchars($_POST['username']);
// $password = $_POST['pwd'];
$deleteName = $_POST['delname'];
include 'conn.php';
if ("admin" === $username)
{
    $sql = "delete from userlists where username='$deleteName' limit 1";
    $result = @mysql_query($sql);
}
});
$app->post('/useredit', function () use ($app)
{
    $result = false;
    $request = $app->request();
    $body = $request->getBody();
    $input = json_decode($body);
    $username = (string)$input->username;
    $oldpwd = (string)$input->oldpwd;
    $newpwd = (string)$input->newpwd;
    include 'conn.php';
    $check_query = mysql_query("select * from userlists where username='$username' and
password='$oldpwd' limit 1");
    if ($checkresult = mysql_fetch_object($check_query))
    {
        $update_sql = "update userlists set password='$newpwd' where username='$username' limit
1";
        $result = @mysql_query($update_sql);
    }
    if ($result)
    {
        echo json_encode(array('password'=>$newpwd));
    }
    else
    {

```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		95

```

});
$app->post('/usercreate', function () use ($app)
{
    $result = false;
    $request = $app->request();
    $body = $request->getBody();
    $input = json_decode($body);
    $username = (string)$input->username;
    $password = (string)$input->password;
    $realname = (string)$input->realname;
    $qq = (string)$input->qq;
    $email = (string)$input->email;
    // $username = $_POST['username'];
    // $oldpwd = $_POST['oldpwd'];
    // $newpwd = $_POST['newpwd'];
    include 'conn.php';
    $query = "INSERT INTO userlists(username, password, realname, qq, email)
VALUE('$username', '$password', '$realname', '$qq', '$email')";
    $result = @mysql_query($query);
    if ($result)
    {
        echo json_encode(array('username'=>$username));
    }
    else
    {
    }
});
$app->post('/feedback/:name', function ($name) use ($app)
{
    $request = $app->request();
    $body = $request->getBody();
    $input = json_decode($body);

    $feedback = $input->code;
    $name = (string)$name;

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

96

```

include 'conn.php';
// To ensure that there is $name or not
$sql = "update feedBackCode set code='$feedback' where name='$name'";
$result = @mysql_query($sql);
if ($result)
{
    echo json_encode(array('updatefeedBack'=>true));
}
else
{
    echo json_encode(array('updatefeedBack'=>false));
}

});
$app->get('/feedback/:name', function ($name) use ($app)
{
    include 'conn.php';
    //$sql = 'select * from switches where id =' . $id;
    $sql = "select code from feedBackCode where name='$name' limit 1";
    $rs = mysql_query($sql);
    if($row = mysql_fetch_object($rs))
    {
        $app->response()->header('Content-Type', 'application/json');

        echo $row->code;//json_encode( $row, JSON_NUMERIC_CHECK);
    }
    else
    {
        $app->response()->status(404);
    }
});
$app->run();

```

Middleware.php

```
final public function getApplication()
```

					ПГУ 1.09.05.01 010 001 ПЗ	Лист
						97
Изм.	Лист	№ докум.	Подпись	Дата		

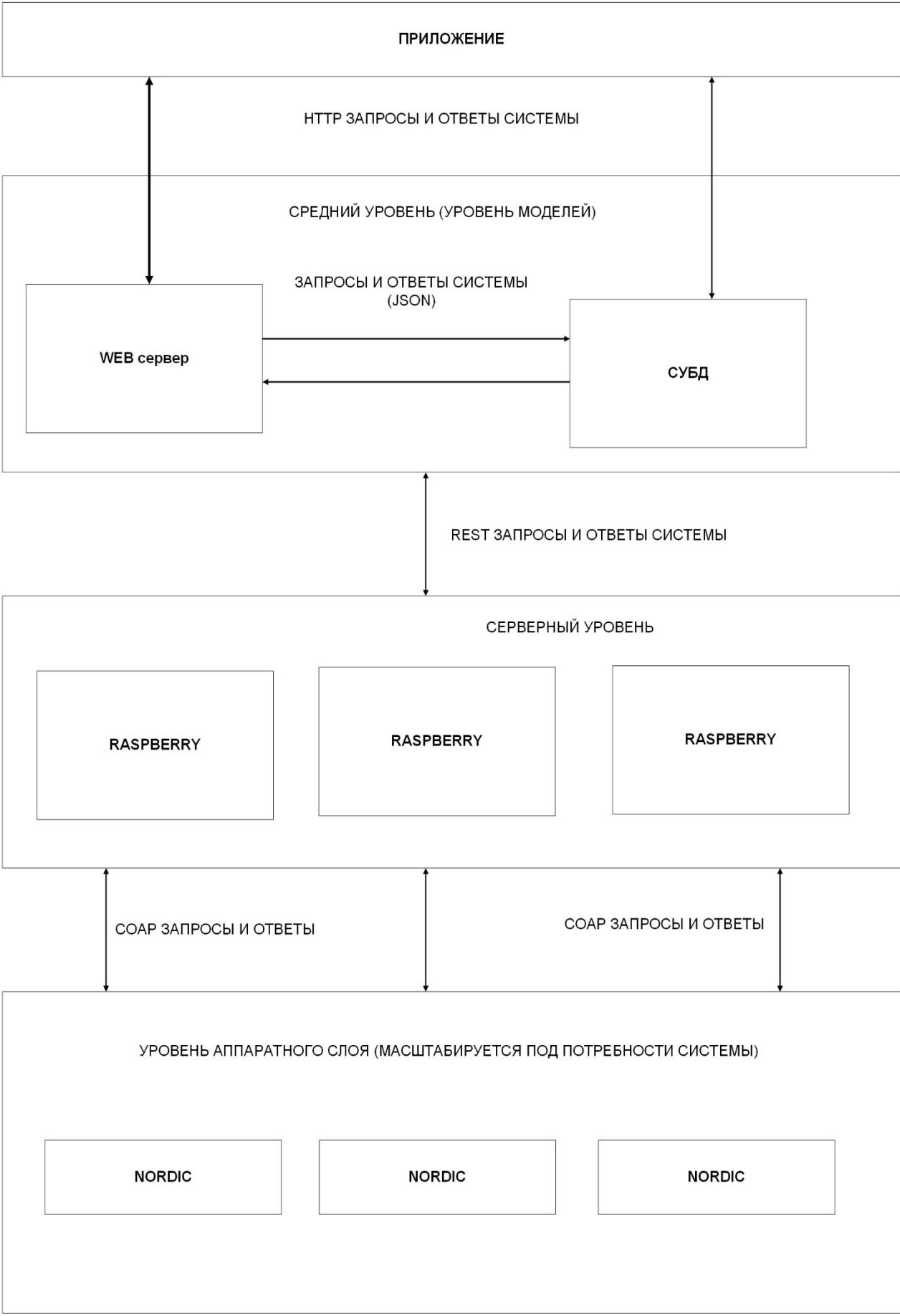
```

{
    return $this->app;
}
final public function setNextMiddleware($nextMiddleware)
{
    $this->next = $nextMiddleware;
}
final public function getNextMiddleware()
{
    return $this->next;
}
abstract public function call();
}

```

ПРИЛОЖЕНИЕ Б
(обязательное)

СХЕМЫ ПРОГРАММЫ



ПРИЛОЖЕНИЕ В
(обязательное)
Презентация

Выпускная квалификационная работа на тему : «Система управления распределенной сетью устройств интернета вещей»

Выполнил:Студент группы 15BO2 Новиков В.В.

Руководитель ВКР: к.т.н., доцент Федюнин Р. Н.

Цели и задачи

- Цель работы на базе распространенных протоколов Интернета вещей реализовать систему мониторинга за контроллерами обработки данных.
- Задачи:
 - Изучить предметную область;
 - Выбрать средства и технологии для разработки;
 - Разработать web приложение ;
 - Протестировать и отладить программу.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

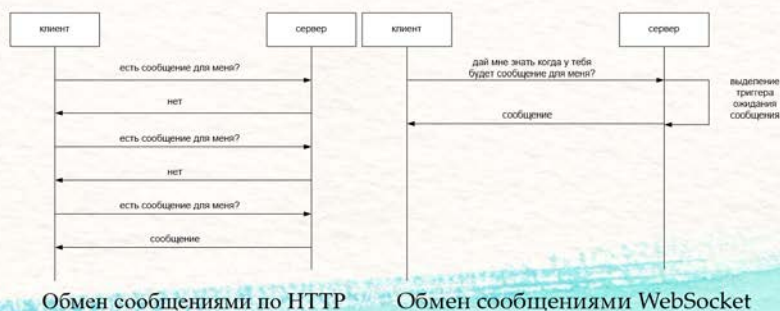
107

Стек технологий

- Yii PHP Framework
- NoSQL
- ide eclipse
- oc android

Основные протоколы взаимодействия в IoT инфраструктуре

- HTTP протокол
- WebSocket протокол
- Протокол CoAP
- LWM2M протокол



Выбор окончечных устройств



Плата Arduino Yun

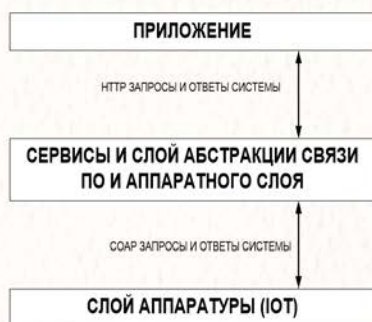


Плата Raspberry Pi

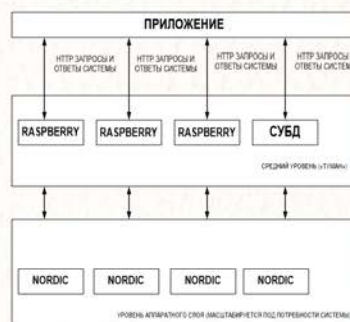


Плата Nordic (nRF52832)

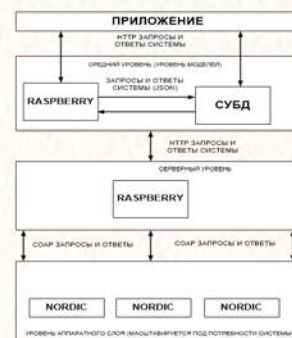
Общая архитектура аппаратно-программной системы



Общая модель построения системы

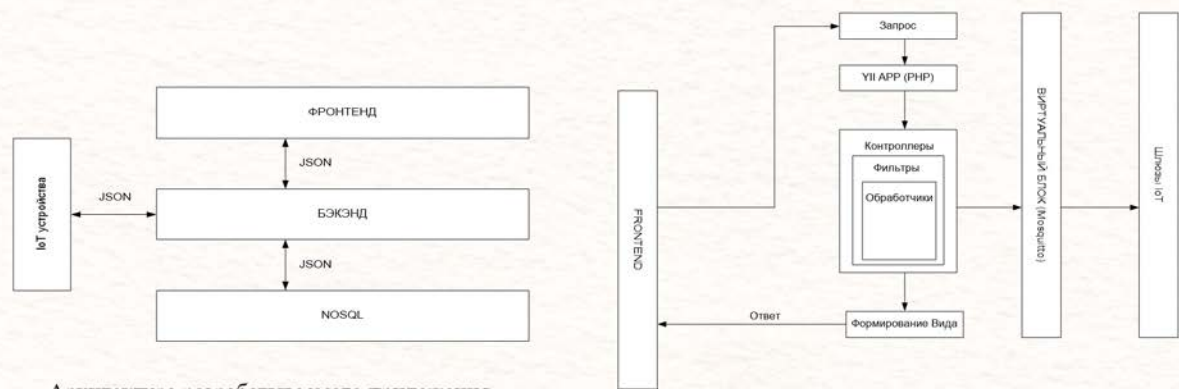


Модель построения системы
с несколькими платами типа «Мастер»



Модель построения системы
«Модель – Вид – Контроллер»

Описание зоны IoT устройств

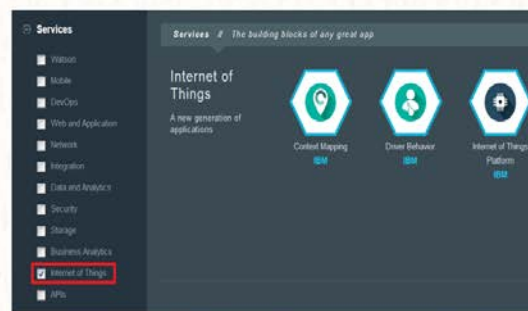


Архитектура разрабатываемого приложения

Взаимодействие модулей системы

The screenshot shows the 'Create an app' form. Fields include: 'Scope' (dev), 'Name' (iottodemoappfield), 'Host' (iottodemoappfield), 'Domain' (mybluemix.net), 'Selected Plan' (Free), 'SDK for Node.js™' (Default), 'Cloudant NOSQL DB' (Shared), and 'Internet of Things Platform' (Free). A green 'CREATE' button is at the bottom.

Создание IoT среды



Создание IoT среды. Выбор пункта меню



Выбор элемента Launch Dashboard

Create Device Type

The screenshot shows the 'Create Device Type' form. Fields include: 'Name' (Arduino), 'Description' (Device for IoT fog), and 'The device type description can be used for a more descriptive way of identifying the device type.'

Выбор типа устройств, с которыми будет работать система

Device 112233445566

The screenshot shows the 'Device 112233445566' page. It displays connection information, device type (Android), and a list of recent events with details like 'Event ID', 'Event Type', 'Event Name', and 'Event Date'.

Проверка работоспособности устройства

Изм.	Лист	№ докум.	Подпись	Дата
------	------	----------	---------	------

ПГУ 1.09.05.01 010 001 ПЗ

Лист
110

Пользовательский интерфейс



Приветственная страница приложения



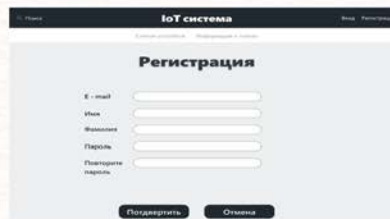
Страница работы с устройствами



Страница информации об устройствах, размещенных в системе



Страница работы с устройством, размещенным в системе



Страница регистрации пользователя в системе



Страница работы с устройством в профиле пользователя.

Пользовательский интерфейс



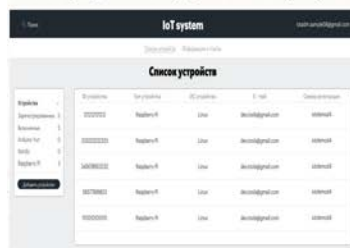
Удаление устройства в профиле администратора



Выполнение команды в профиле администратора



Результат выполнения команды в профиле администратора



Результат выполнения команды в профиле администратора

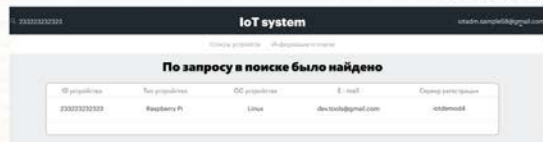


Добавление устройства в профиле администратора

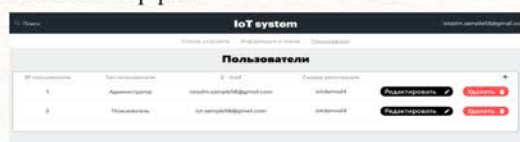


Процесс регистрации устройства в системе

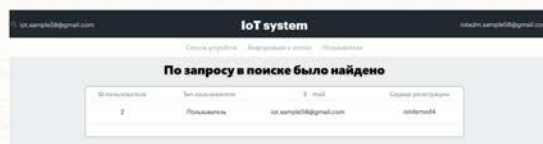
Пользовательский интерфейс



Процесс поиска устройства в системе.



Процесс поиска устройства в системе



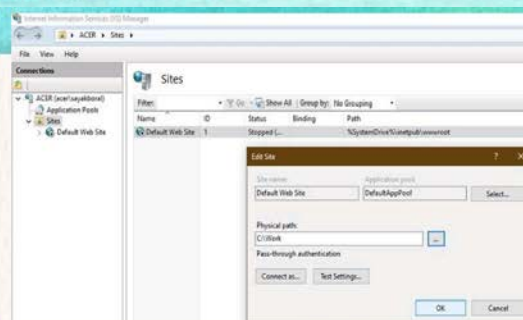
Процесс поиска устройства в системе



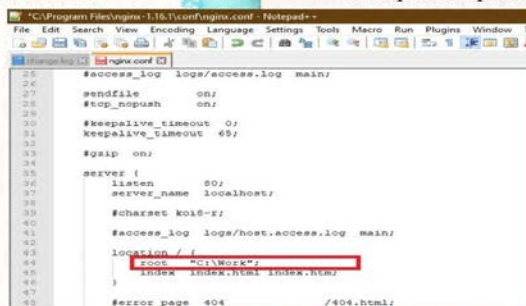
Процесс поиска устройства в системе



Приветственное окно



Выбор и настройка рабочего каталога



Конфигурационный файл

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист
112

Заключение

В ходе выполнения дипломной работы было разработано web приложение, обеспечивающее управление средой интернет вещей, состоящей из различных устройств и облачного сервиса. Данный проект — это приложение на основе клиент серверного подхода, созданное на основе данных, собранных пользователем в облачной среде. Приложение интегрируется со сторонними службами для обеспечения безопасности работы с данными и конвертирования полученной информации на облачный сервер. Приложение построено с использованием технологий, которые являются производительными, гибкими и масштабируемыми.

СПАСИБО ЗА ВНИМАНИЕ!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01 010 001 ПЗ

Лист

113