

**«Санкт-Петербургский государственный электротехнический университет  
«ЛЭТИ» им. В.И.Ульянова (Ленина)»  
(СПбГЭТУ «ЛЭТИ»)**

<b>Направление</b>	Направление 09.03.01 - Информатика и вычислительная техника
<b>Профиль</b>	Вычислительные машины, комплексы, системы и сети
<b>Факультет</b>	КТИ
<b>Кафедра</b>	ВТ

*К защите допустить*

Зав. кафедрой

Куприянов М.С.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
БАКАЛАВРА**

**Тема: АРХИТЕКТУРНЫЕ РЕШЕНИЯ ДЛЯ ПОСТРОЕНИЯ  
ОБЛАЧНОГО СЕРВИСА «КОМФОРТНАЯ СРЕДА» НА БАЗЕ  
ИНФРАСТРУКТУРЫ ЭЛЕМЕНТОВ ИНТЕЛЛЕКТУАЛЬНОГО  
ЗДАНИЯ**

Студентка		<hr/>	Илюшкина М.Э.
		<i>подпись</i>	
Руководитель	К.Т.Н.	<hr/>	Дорохов А.В.
		<i>подпись</i>	
Консультанты	К.Т.Н., доцент	<hr/>	Зуев И.С.
		<i>подпись</i>	
		<hr/>	Лебедева Т.Н.
		<i>подпись</i>	

Санкт-Петербург

2016

## ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

Зав. кафедрой ВТ

\_\_\_\_\_ Куприянов М.С.

«\_\_\_» \_\_\_\_\_ 20\_\_ г.

Студентка     Илюшкина М.Э.

Группа 2307

Тема работы: Архитектурные решения для построения облачного сервиса «Комфортная среда» на базе инфраструктуры элементов интеллектуального здания.

Место выполнения ВКР: кафедра ВТ, СПбГЭТУ «ЛЭТИ»

Исходные данные и технические требования:

В рамках работы должен быть спроектирован прототип облачного сервиса для автоматизации мониторинга и управления микроклиматом в жилых и офисных помещениях. В состав контролируемых параметров среды должны включаться температура, влажность и освещенность. Поступление информации с датчиков эмулируется с помощью скриптов на языке Python. Облачный сервис должен быть выполнен на базе операционной системы Ubuntu 14.04 LTS с использованием программного обеспечения Oracle VM VirtualBox для виртуализации.

Содержание ВКР:

Предполагается концептуальная проработка архитектурных решений для построения облачного сервиса, интегрирующего существующие программно-аппаратные решения систем интеллектуализации зданий в рамках концепции «комфортная среда», представляющей индивидуальным

пользователям персонализированный интерфейс управления элементами системы автоматизации зданий в границах эксплуатируемых жилых или офисных помещений (системы управления микроклиматом, контроля энерго и водопотребления, обеспечения безопасности и др.).

Перечень отчетных материалов: текст ВКР, иллюстративный материал – прототип облачного сервиса

Дополнительные разделы: Экономическое обоснование

Дата выдачи задания

«\_\_\_»\_\_\_\_\_ 2016 г.

Дата представления ВКР к защите

«\_\_\_»\_\_\_\_\_ 2016 г.

Студентка

\_\_\_\_\_

Илюшкина М.Э.

Руководитель

к.т.н.

\_\_\_\_\_

Дорохов А.В.

## КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю  
Зав. кафедрой ВТ  
\_\_\_\_\_ Куприянов М.С.  
«\_\_\_» \_\_\_\_\_ 20\_\_ г.

Студентка    Илюшкина М.Э.

Группа 2307

Тема работы: Архитектурные решения для построения облачного сервиса «Комфортная среда» на базе инфраструктуры элементов интеллектуального здания.

№ п/п	Наименование работ	Срок выполнения
1	Разработка технического задания	25.01 – 29.01
2	Изучение существующих решений в предметной области	28.01 – 03.02
3	Уточнение технического задания	04.02
4	Сбор материалов и эксперименты с существующими программными решениями	05.02 – 03.03
5	Составление аналитического обзора предметной области	04.03 – 25.03
6	Разработка архитектуры сервиса	28.03 – 05.04
7	Разработка серверной части	06.04 – 18.04
8	Разработка клиентской части	19.04 – 28.04
9	Тестирование сервиса	29.04 – 06.05
10	Оформление пояснительной записки	10.05 – 30.05
11	Анализ проведенной работы	30.05

Студентка

Илюшкина М.Э.

Руководитель

к.т.н.

Дорохов А.В.

## РЕФЕРАТ

Пояснительная записка содержит 67 стр., 21 рис., 4 табл., 21 ист., 3 прил.

ИНТЕРНЕТ ВЕЩЕЙ, УМНЫЕ ДОМА, ОБЛАЧНЫЕ СЕРВИСЫ,  
АВТОМАТИЗАЦИЯ, ПРОЕКТИРОВАНИЕ, АРХИТЕКТУРА, СЕРВЕР

Объект разработки – облачный сервис для мониторинга и управления системами автоматизации зданий.

Цель работы – разработка архитектуры серверной части облачного сервиса и создание прототипа для подтверждения реализуемости спроектированного решения.

В работе рассмотрена архитектура облачного сервиса «Комфортная среда», который осуществляет сбор и отображение информации о параметрах окружающей среды, таких как температура, влажность и освещенность в помещении, производит обработку полученной информации и оповещает пользователя о критических изменениях выбранных параметров.

Особенность разработанного архитектурного решения является использование средств виртуализации для изолированного хранения данных пользователей, а также для упрощения масштабирования сервиса. В работе приведены варианты усовершенствования спроектированной архитектуры.

## ABSTRACT

This work is devoted to creation of a cloud service for monitoring of environmental parameters in premises and offices which uses the systems of buildings' intellectualization.

In the document the modern problems in creation of the services used for the Internet of things and systems of the Smart buildings are investigated. As a result of research the following problems are revealed: safety at the level of transfer and storage of personal data, lack of a possibility to view the collected information in real time because of congestion of servers, and also lack of graphic interface personalization on the websites of services.

Considering these problems the architectural concept was developed for server part of service. Work contains the description of prototype cloud service "Comfortable workplace" which carries out collecting and display information: temperature, humidity and illuminating. This service is constructed using PCs based on OS Ubuntu 14.04 LTS Server, the Apache 2.0 web-servers and MySQL database-servers. Data of users are stored in databases on personal virtual machines (XEN hypervisor for OS Ubuntu's core is used) that ensures safety of storing of information and rational use of machine resources. Also this can help to create personal graphical interfaces if we use virtual machines as a cloud-server (PaaS model of cloud computing).

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	9
ВВЕДЕНИЕ.....	10
1 Интеллектуальные здания: концепции, технологии и обслуживание.....	12
1.1 Интернет вещей.....	12
1.1.1 Концепция. Определения .....	12
1.1.2 Архитектура и используемые технологии.....	14
1.1.3 Достоинства и недостатки концепции Интернет вещей .....	17
1.2 Интеллектуальные здания .....	20
1.2.1 Определение и архитектура «Интеллектуальных зданий».....	20
1.2.2 Дистанционное управление «Умным домом» .....	21
1.3 Облачные вычисления .....	24
1.3.1 Определение облачных вычислений, модели и основные характеристики .....	24
1.3.2 Применение облачных вычислений в системах умных домов .....	27
1.4 Проблемы проектирования облачных сервисов для интеллектуальных зданий и пути их решения.....	28
1.4.1 Обзор существующих облачных сервисов.....	29
1.4.2 Основные проблемы .....	32
1.4.3 Пути усовершенствования сервисов.....	33
2 Разработка архитектуры серверной части облачного сервиса «Комфортная среда».....	34
2.1 Проектирование взаимодействия клиентов и сервиса «Комфортная среда».....	34
2.1.1 Требования пользователя к системе сервиса .....	34
2.1.2 Структура разрабатываемого облачного сервиса.....	36
2.1.3 Подключение устройств к сервису.....	38
2.1.4 Подключение пользователей к сервису .....	39
2.2 Внутренняя организация серверной части сервиса .....	39
2.2.1 Настройка сети под выбранную архитектуру .....	39
2.2.2 Структура баз данных для хранения информации о пользователях, виртуальных машинах и получаемых данных .....	41
2.2.3 Способы настройки виртуализации .....	43
2.3 Анализ разработанной архитектуры .....	43

3 Проектирование прототипа облачного сервиса «Комфортная среда» .....	44
3.1 Серверная часть сервиса «Комфортная среда» .....	44
3.1.1 Упрощение организации сети .....	44
3.1.2 Реализация серверной части сервиса .....	45
3.1.3 Считывание данных устройства .....	47
3.2 Клиентская часть сервиса «Комфортная среда» .....	48
3.2.1 Отображение информации устройств .....	48
3.2.2 Инструкция по использованию сайта .....	49
4 Технико-экономическое обоснование разработки облачного сервиса .....	53
4.1 Трудоемкость разработки проекта .....	54
4.2 Расчет себестоимости разработки проекта .....	55
4.2.1 Основная и дополнительная заработная плата исполнителей .....	56
4.2.2 Отчисления на страховые взносы .....	56
4.2.3 Затраты на сырье и материалы .....	56
4.2.4 Оплата услуг сторонних организаций .....	57
4.2.5 Затраты на эксплуатацию оборудования .....	57
4.2.6 Амортизационные отчисления .....	58
4.2.7 Накладные расходы .....	58
4.2.8 Себестоимость разработки проекта .....	58
4.2.9 Выводы об экономической целесообразности проекта .....	59
ЗАКЛЮЧЕНИЕ .....	60
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	62
ПРИЛОЖЕНИЕ А_Код РНР. Подключение к базам данных .....	64
ПРИЛОЖЕНИЕ Б_Код РНР. Вывод информации из БД в виде таблицы .....	65
ПРИЛОЖЕНИЕ В_Главная страница сайта .....	66



## **ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

БД – база данных

ВКР – выпускная квалификационная работа

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

IP – Internet Protocol

MAC – Media Access Control

PaaS – Platform as a service

SaaS – Software as a service

## **ВВЕДЕНИЕ**

В современном мире все большую популярность набирает концепция Интернета Вещей (Internet of Things, IoT), которая включает в себя и так называемые «Умные дома» или «Интеллектуальные здания». «Умные» электроприборы, автоматический контроль энерго- и водоснабжения все чаще встречаются в домах обыкновенных пользователей, как за рубежом, так и в России.

На рынке в сфере «Интеллектуальных зданий» представлен широкий выбор готовых решений с различными наборами аппаратно-программных средств от зарубежных и российских производителей. Но установка такого комплекса стоит относительно дорого и, как правило, требует профессионального инженерного планирования на этапе строительства дома или ремонта квартиры.

В связи с вышесказанным, многими создаются самодельные системы с использованием доступных модулей, чаще всего, для контроля климата в помещении и снижения энергопотребления. Для эффективного использования таких систем и дистанционного управления «Вещами» необходим сервис, имеющий удобный интерфейс и поддерживающий все модули, включенные в «умную» сеть. На данном этапе у разработчика могут возникнуть проблемы с хранением, анализом и графической интерпретацией получаемой информации.

Поток информации, поступающей с датчиков температуры, давления, света и подобных им, очень велик, особенно, если требуется непрерывный мониторинг выбранных параметров и анализ статистики за длительный промежуток времени. Соответственно, для хранения такого объема информации требуется большое хранилище. Размещение на физических носителях в домашних условиях потребует денежных затрат и дополнительной настройки, поэтому чаще всего используются облачные сервисы.

Облачные сервисы позволяют собирать данные с датчиков и хранить их длительное время, выводить информацию, если требуется, в реальном времени. Некоторые из них включают возможность анализа полученной информации и событийного оповещения пользователей через социальные сети, по почте или SMS. Следующей проблемой является платность длительного хранения больших объемов информации, ограничение количества каналов для сбора информации с датчиков и стандартное графическое оформление полученных данных без возможности модернизировать его или частично использовать на разработанном самостоятельно Web-сайте.

Большой проблемой, как готовых решений, так и самостоятельно разработанных систем с подключением облачных сервисов, является безопасность. Информация, поступающая с устройств в сеть, а затем и в Интернет, является конфиденциальной и при передаче требует особых мер предосторожности, чем производители систем и ее компонентов на данном этапе развития Интернета вещей могут пренебрегать из-за отсутствия стандартов.

Цель разработки – создать облачный сервис «Комфортная среда» для контроля климата в помещениях, концептуальная проработка архитектурных решений для его построения и создание прототипа сервиса.

Объектом исследований в данной работе является хранение и обработка информации из частных сетей, созданных для мониторинга и управления системами автоматизации зданий. Предмет исследования – облачные сервисы для хранения и интерпретации таких данных.

В первой главе рассмотрены основные концепции, позволяющие понять предметную область разработки; определены проблемы при создании облачных сервисов и методы их устранения. Вторая глава посвящена теоретической проработке архитектуры сервиса. Третья глава – описание созданного прототипа сервиса «Комфортная среда». Четвертая – технико-экономическое обоснование разработки облачного сервиса.

# **1 Интеллектуальные здания: концепции, технологии и обслуживание**

Создание «Интеллектуальных зданий» или «Умных домов» в последние годы набирает популярность в связи с появлением доступных и несложных в установке модулей для их создания. Изначально, этот термин применялся для сложных инженерных систем автоматизации, основанных кабельно-проводном соединении. Но с появлением концепции Интернета вещей (Internet of Things, IoT) «умные дома» стали использовать беспроводные интерфейсы связи между компонентами, а также подключение к сети Интернет для дистанционного управления. Для того чтобы спроектировать сервис обслуживания систем данного типа, включая мониторинг и удаленное регулирование, необходимо изучить основные принципы концепции «Интернета вещей», особенности архитектуры «Интеллектуальных зданий», а также архитектуру облачных вычислений.

## **1.1 Интернет вещей**

Что такое Интернет вещей, из каких уровней он состоит и какие технологии использует? Обзор концепции, ее достоинства и проблемы развития.

### **1.1.1 Концепция. Определения**

Определение концепции Интернета вещей на русском языке можно найти в Рекомендации МСЭ-Т Y.2060 [1].

Термин впервые был сформулирован в 1999 году Кевином Эштоном (англ. Kevin Ashton), работником исследовательской группы в компании Procter & Gamble. Он предложил внедрить радиочастотную идентификацию –

RFID-метки или маркеры (Radio Frequency IDentification), для отслеживания перемещения товаров компании.

Содержание концепции Интернета вещей можно сформулировать следующим образом: для увеличения степени комфорта жизни людей и предоставления сложных комплексных услуг необходимо создание глобальной инфраструктуры, состоящей из множества вещей (виртуальных и физических), которые соединены между собой по средствам существующих, развивающихся и функционально совместимых технологий информационной коммуникации.

Физические вещи в данной концепции – это вещи реального физического мира (датчики и различные устройства), а виртуальные – вещи информационного мира (например, виртуальные деньги, и все, что имеет реальную цену, но не имеет физического обличия). Каждая из таких вещей может быть идентифицирована или интегрирована в сеть.

Если посмотреть с практической точки зрения, то концепция Интернета вещей направлена на автоматизацию процессов в различных сферах деятельности, исключения из них человека, и, как следствие, повышение эффективности экономических и общественных процессов.

Основатель европейского совета по «Интернету вещей» Роб Ван Краненбург (бельг. Rob van Kranenburg) в своих работах предлагает интересное представление Интернета вещей в виде «четырёхслойного пирога» [2].

Первый слой включает в себя идентификацию объектов, например, с помощью датчиков или RFID-маркеров. На этом этапе каждая Интернет-вещь получает средство связи с окружающим миром и уникальные данные.

Второй слой – обслуживание потребителя или сервис. Здесь объекты объединяются в сети для выполнения определенной функции в рамках поставленной задачи. Самыми распространенными примерами являются системы мониторинга окружающей среды с помощью беспроводных сенсорных сетей и, конечно, «Умные дома».

Третий слой основан на тенденции урбанизации городской жизни, так называемые «Умные города», которые подразумевают исследование и сбор информации на конкретной территории (квартале, районе и т.д.) и предоставление всей информации ее обитателям.

Четвертый и самый высший уровень – это сенсорная планета, когда все существующие сети объединяются в глобальную информационную инфраструктуру.

Иными словами, Интернет вещей – это сеть сетей.

### 1.1.2 Архитектура и используемые технологии

Архитектуру Интернета вещей можно разделить на четыре функциональных уровня (рисунок 1.1 из [3]), рассмотрим каждый из них подробнее.

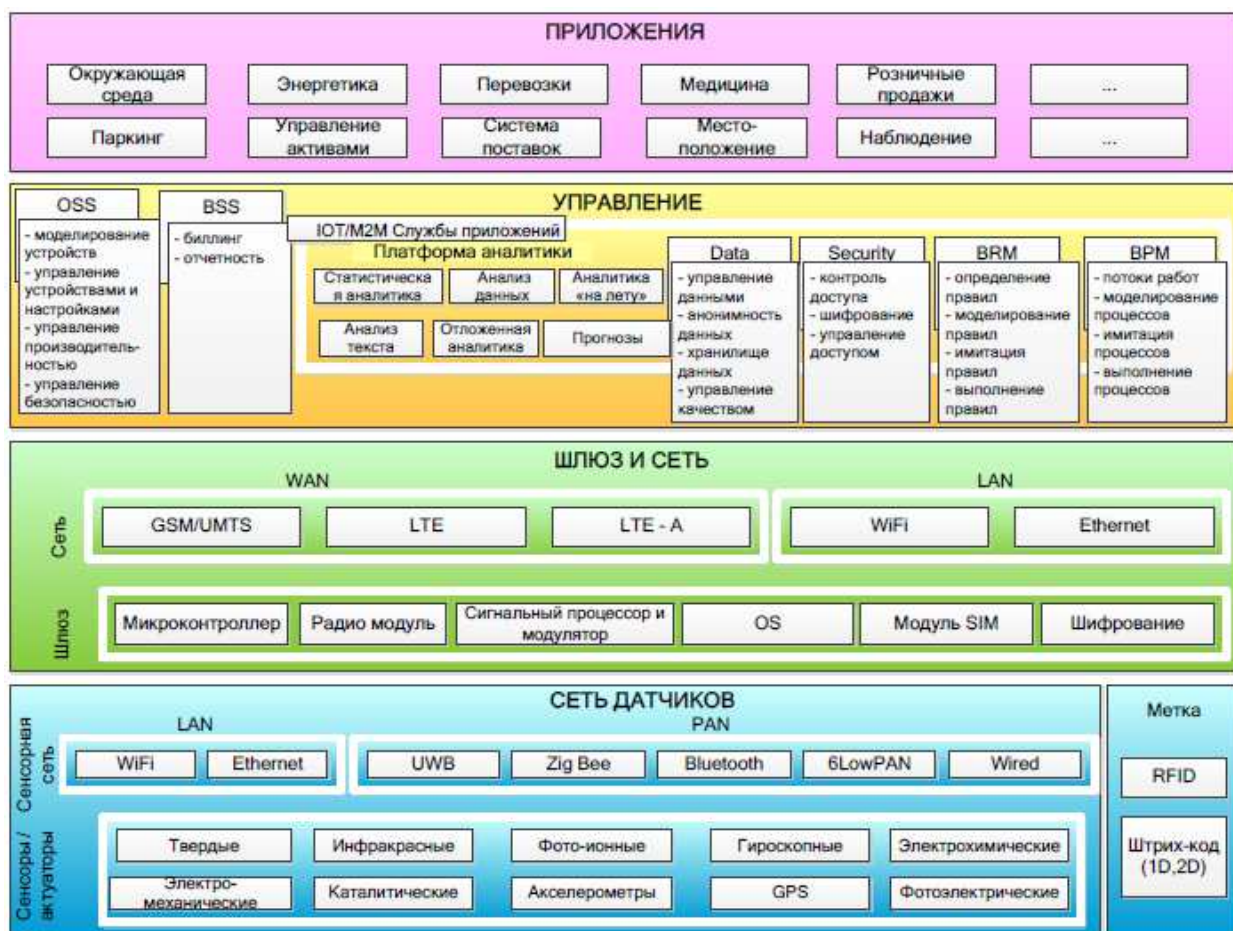


Рисунок 1.1 – Архитектура IoT. Разделение на функциональные уровни

Самым нижним в структуре Интернета вещей, но в тоже время и самым важным, является уровень сенсоров. Если сопоставить с представлением Роба ванн Краненбурга, то он является первым слоем «пирога». Основывается этот уровень на модулях с подключаемыми датчиками и сенсорами, а также иных объектах, обеспечивающих сбор информации, например, на RFID-метках упоминаемые ранее или штрихкодах. Такие объекты могут интегрироваться друг в друга и в иные физические объекты, и даже в тело человека.

Широкие исследования в области сенсоров проводит Институт инженеров электротехники и электроники — IEEE (англ. Institute of Electrical and Electronics Engineers). На данный момент анонсирован стандарт обмена информацией IEEE 802.15.6-2012, позволяющий передавать информацию в маломощной беспроводной сети в непосредственной близости к человеческому телу без помех.

Для передачи информации внутри домашних или офисных сетей используются менее чувствительные и более энергоемкие стандарты обмена, наиболее известными из которых является Ethernet, Wi-fi и Zig Bee.

Большая часть датчиков неинформативна сама по себе, но, объединяясь в сетевые структуры, они несут значительный поток полезной информации. Для объединения и передачи информации в мир большинству датчиков требуется наличие шлюза или агрегатора сенсоров. В зависимости от задачи объединения объектов и рода самих объектов, возможно использование одного из нижеописанных типов сетей.

1) Беспроводные сенсорные сети (WSN, Wireless Sensor Network) – самоорганизующиеся сети датчиков и устройств. Как правило, устройства в сети соединены радиоканалом, поэтому этот тип сетей отличается низким энергопотреблением, малым радиусом покрытия и низкой скоростью передачи. Радиус покрытия корректируется ретрансляцией сообщений (MultiHop Networking или Ad Hoc сети). Основным стандартом передачи является 6LoWPAN (англ. IPv6 over Low power Wireless Personal

Area Networks), который позволяет подключать сенсорные устройства к широко распространенным IP сетям.

2) Сети ультра широкополосной беспроводной связи на малых расстояниях (UWB, Ultra-Wide Band) –используются для подсоединения периферийных устройств в сетях малого радиуса охвата, например, принтеров или мультимедийных систем в пределах дома. Характеризуются высокой скоростью передачи на малых расстояниях.

3) Персональные сети (PAN, Personal Area Network) – сети, создаваемые «вокруг» человека, объединяющие персональные компьютеры, телефоны и иные устройства. В сетях данного типа используются спецификации сетевых протоколов верхнего уровня, такие как ZigBee и Bluetooth.

4) Локальные вычислительные сети (LAN, Local Area Network) – Ethernet и Wi-Fi; данный тип сетей на сегодняшний момент известен большинству людей, так как интернет прочно вошел в современное общество.

5) Глобальные беспроводные сети (WAN, Wide Area Network) – сети, обеспечивающие связь устройств с серверами/приложениями напрямую, если не требуется подключение к агрегатору. В категорию глобальных беспроводных сетей входят GSM, GPRS и LTE.

Реализации сервиса для обслуживания IoT требует обеспечить совмещение большого количества сетей с различными протоколами доступа и технологиями передачи данных. Причем конфигурация таких сетей может быть разнообразной. Второй уровень – уровень шлюзов и сетей как раз и предполагает объединение разнородных сетей в единую сетевую инфраструктуру.

Объединенные сети требуют особого соответствия установленным стандартам качества передачи информации: задержки не должны превышать допустимых норм, пропускная способность и безопасность передачи данных



должны так же соответствовать стандартам. Пользователи должны иметь доступ к ресурсам независимо и совместно без потерь производительности.

Следующий уровень – сервисный. Обеспечивает управление бизнес-правилами и бизнес-процессами (BRM, Business Rule Management и BPM, Business Process Management соответственно). Позволяет автоматизировать операции, проводимые над информацией – хранение и анализ, обеспечивает поддержку операционной и бизнес деятельности (OSS/BSS, Operation Support System/Business Support System).

Самый верхний уровень – уровень приложений. Приложения делятся на «горизонтальные», применимые для разных сфер деятельности и «вертикальные», создаваемые под определенную направленность деятельности.

Из указанных уровней в работе будут использоваться уровень сенсоров и уровень шлюзов, так как именно эти уровни обеспечивают сбор информации о комфортности условий в помещении. А также уровень приложений, так как сервис можно считать в определенной мере приложением.

### **1.1.3 Достоинства и недостатки концепции Интернет вещей**

Достоинства развития технологий Интернета вещей вытекают из практического применения концепции: это автоматизация процессов в различных сферах деятельности, и, как следствие, повышение их экономической эффективности. Интернет вещей призван сделать наше существование комфортней, а производство более выгодным. Вряд ли кто-то не мечтает об автоматически вынесенном мусоре и подогревом ко времени прихода домой с работы чае или свежее сваренном кофе с утра.

Но это лишь малая часть возможностей Интернета вещей в сфере личного пользования. Для работы в офисах колоссальную пользу может принести контроль климата в помещении, ведь комфортная рабочая среда

улучшает самочувствие, повышает работоспособность и даже настроение. А уведомление о незакрытой после рабочего дня входной двери может уберечь от неприятностей. Автоматизация на производстве так же поможет избежать непредвиденных ситуаций, что особенно важно на опасных химических и атомных предприятиях. Постоянный контроль рабочей зоны и сигнал при превышении параметрами установленных норм может помочь избежать катастрофы.

Интернет вещей развивается стремительно, но и без сложностей не обойтись. На сегодняшний день есть несколько причин замедления в развитии столь популярной концепции: дефицит IPv4 адресов, уменьшение энергопотребления датчиков и отсутствие общепринятых стандартов.

1) Дефицит адресов IPv4:

Каждый новый датчик нуждается в уникальном IP-адресе. Адреса IPv4 закончились еще в феврале 2010 года. Следовательно, встает вопрос о переходе к протоколу новой версии с расширенным количеством адресов – IPv6. Помимо большего количества адресов, IPv6 упрощает управление сетями, так как в нем существует функция авто-конфигурирования адресов, а так же подключение по данному протоколу является более безопасным, чем подключение по IPv4.

2) Энергопотребление датчиков:

Энергоснабжение датчиков – очень важный вопрос, так как реализация основной идеи концепции Интернета вещей требует их автономности. Невозможно обеспечить батарейками огромное количество модулей без вреда для окружающей среды. Необходим другой подход. Датчики должны получать энергию из окружающей среды или вырабатывать ее самостоятельно. Простейший способ получения энергии – солнечные батареи, но он не пригоден для всех видов датчиков, лишь для тех, что располагаются близко к источникам света. Большой шаг в развитии был сделан учеными Американского химического общества в 2010 году. Анонсирован наногенератор – гибкий чип, вырабатывающий энергию из

человеческих телодвижений. Несомненно, ученые будут разрабатывать все новые и новые способы получения энергии.

### 3) Стандарты:

В области стандартизации были достигнуты значительные результаты за последние несколько лет, но проблемы безопасности, защиты личной информации и установления единой архитектуры решений по-прежнему актуальны. Уже упоминавшаяся ранее международная ассоциация IEEE (Институт инженеров электротехники и электроники) – одна из немногих организаций, способствующих развитию единых стандартов. Текущим решением является стандартизированная передача пакетов IPv6 в различных видах сетей.

Начиная с 2012 года, Европейская комиссия по вопросам информационного общества проводит консультации по теме регулирования рынка устройств, подключаемых к частным беспроводным сетям. Такая обеспокоенность вызвана тем, что устройствами Интернета вещей собирается, хранится и передается информация личного характера, которую злоумышленники могут использовать против владельцев. Еврокомиссия пытается найти оптимальное решение, сочетающее защиту личных данных и удобство использования модулей и их взаимную совместимость.

Над созданием универсальных спецификаций и соответствующей сертификации в сфере «умной» электроники на данный момент работают несколько организаций, в том числе альянс Open Connectivity Foundation(OCF), включающий в себя Intel, Samsung Electronics и Dell.

Одним из решений проблемы безопасности является аппаратная поддержка полупроводниковыми компонентами протокола *TLS* (Transport Layer Security — безопасность транспортного уровня). Данный протокол аналогичен криптографическому протоколу *SSL* (Sockets Layer — уровень защищенных сокетов), так как использует симметричное шифрование и ассиметричную криптографию, но защищает данные на более низком уровне. Недавно SSL признан небезопасным и советуется заменять TLS.

## **1.2 Интеллектуальные здания**

Определение «Умный дом», из чего состоят системы автоматизации зданий и почему на сегодняшний день это направление очень популярно.

### **1.2.1 Определение и архитектура «Интеллектуальных зданий»**

Автоматизация зданий начала появляться еще в 60-70-х годах прошлого века, тогда и было сформулировано понятие «умный дом». Изначально оно формулировалось как «здание для эффективного использования рабочего пространства», но на сегодняшний день его смысл гораздо шире.

«Умный дом» для современного человека – не просто система рационального использования рабочего пространства, это интеллектуальная система, которая объединяет в себе как инженерные коммуникации и системы безопасности, так и информационные системы здания. Такие объединённые решения призваны повысить комфортность помещений и обеспечить их безопасность. Во многих случаях поводом для установки систем умного дома является желание повысить степень комфорта за счёт автоматизации рутинных или постоянно повторяющихся задач.

Реализация умного дома делится на две части: аппаратную и программную. На рисунке 1.2 [4] представлена одна из возможных и наиболее часто используемых схем аппаратного обеспечения умного дома. Аппаратная часть, как правило, состоит из контроллера, модулей расширения и конечного оборудования.

Контроллером может выступать ПК, планшет, смартфон, на которые устанавливается программное обеспечение для управления системами умного дома внутри домашней (рабочей) сети или через сеть Интернет.

Модулями или платами расширения называют специальные устройства с подключёнными датчиками различного типа и управляемыми частями системы.

В категорию конечного оборудования входят датчики для отслеживания различных параметров среды и состояния устройств, которые необходимо регулировать.

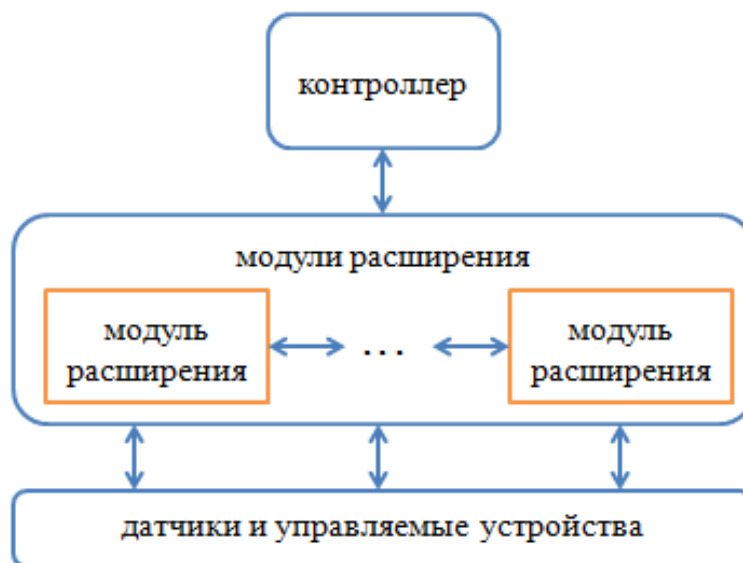


Рисунок 1.2 – Типовая схема аппаратного обеспечения умного дома

Программное обеспечение может быть реализовано многими способами: от обыкновенного пульта управления до сложного синхронизированного комплекса программного обеспечения, устанавливающегося на любое количество гаджетов и ПК владельца интеллектуального здания и полностью автоматических систем, включающих в себя элементы интеллектуализации.

### 1.2.2 Дистанционное управление «Умным домом»

Развитие беспроводных интерфейсов связи и расширение концепции Интернета вещей привело к выходу систем умных домов за пределы помещений и зданий, в которых они установлены. Взаимодействие автоматизирующих комплексов с сетью Интернет дало возможность управлять ими в режиме удаленного доступа.

Дистанционное управление имеет ряд преимуществ перед управлением исключительно автоматическим и через системы, работающие не далее

установленного радиуса. Прежде всего, это повышение уровня безопасности и комфорта.

Основное преимущество, естественно, – повышение комфорта использования пространства дома или офиса, что и является основной идеей для создания умного дома. При наличии функции удаленного управления пользователь может включить, выключить или настроить требующиеся ему устройства (освещение, бытовые приборы и иные системы), где бы он ни находился. Например, подогреть чай перед приходом домой с работы или заранее включить отопление, если по показаниям датчиков в помещении некомфортная температура.

Но главным преимуществом стоит считать повышение уровня безопасности. При отсутствии людей в помещении могут произойти ситуации, угрожающие сохранности имущества и самого здания. Для предотвращения подобных инцидентов возможно подключение камер для наблюдения за обстановкой в помещении или удаленный мониторинг с помощью анализа информации, поступающей с различных датчиков, которые используются в системах безопасности (датчики огня, датчики открытия/закрытия дверей и т. д.). В том числе, автоматическое отключение электроприборов и света поможет не только сэкономить потребление электроэнергии, но и уменьшить риск самовозгорания электропроводки в пустом помещении;

Система дистанционного управления в большинстве случаев проста: пользователь, с использованием гаджетов или ПК, отправляет команды системе через веб-приложение или со страницы веб-сайта, система анализирует полученную команду и с помощью контроллера выполняет указанное действие.

Так как дистанционная работа с датчиками, камерами и иным оборудованием предполагает хранение, обработку и анализ большого количества информации, и повсеместный доступ к результатам несколькими пользователями, целесообразно применение облачных вычислений. При

таком подходе в системе умного дома появляется облачный сервис, позволяющий вынести заботы об обслуживании серверной части системы, с помощью которой происходит управление всей системой, за пределы ведений пользователя.

Для подключения облака существует два варианта. Первый – использование облака в качестве контроллера. Все устройства подключаются непосредственно к облаку и управляются из него напрямую. Управляющая часть системы может быть полностью вынесена за пределы здания. Вторым вариантом – сбор устройств на контроллерах и подключение их к облаку. В этом случае облако будет управлять действиями контроллеров, передавая информацию между несколькими модулями. Контроллер размещается внутри здания, но все программное обеспечение вынесено на сервис. Рисунок 1.3 из [4] иллюстрирует подключение облака к системе умного дома вторым способом.

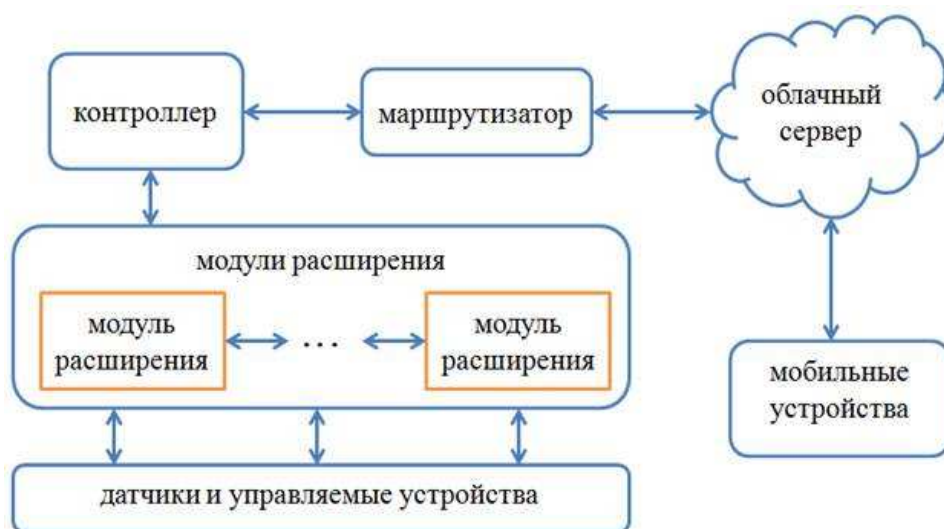


Рисунок 1.3 – Схема аппаратного обеспечения умного дома с удаленным управлением

Оба варианта предполагают вынос анализирующей части системы в облако, что позволяет снизить требования к контроллеру, главное отличие в том, что в первом случае управление устройствами автономно друг от друга, а во втором через общий контроллер.

Многие современные модули работают по собственным протоколам передачи данных и, взаимодействуют с Интернет-сервисами только через

свои API, что создаёт сложности обмена информацией между устройствами непосредственно, а также мешает расширению системы умного дома (наглядный пример проблемы стандартизации, рассмотренной в пункте 1.1.3). Применение облачных сервисов делает систему умного дома более гибкой и позволяет взаимодействовать модулям разных производителей с различными протоколами передачи данных, через облако, тем самым решая одну из проблем.

## **1.3 Облачные вычисления**

Какими бывают облачные вычисления, и какие модели применяются для дистанционного управления и мониторинга умных домов, и почему их стоит применять?

### **1.3.1 Определение облачных вычислений, модели и основные характеристики**

Определение облачных вычислений было дано в 2011 году сотрудниками Национального института стандартов и технологий США [5].

*Облачные вычисления* (англ. cloud computing) — это модель для предоставления возможности повсеместного, удобного в использовании сетевого доступа к общему пулу (англ. pool) конфигурируемых вычислительных ресурсов (например, сетей, серверов, хранилищ, приложений и сервисов), который может быть предоставлен и освобожден быстро, по требованию и с минимальными затратами на управление и взаимодействие с провайдером услуг.

От других типов вычислений (Интернет-ресурсов) облачные вычисления отличает ряд характеристик.

Первое важное свойство – самообслуживание по требованию. По мере необходимости потребитель может самостоятельно выбирать



вычислительные мощности, например, серверное время, объем хранилища данных. Данный процесс происходит без участия провайдера.

Второй характеристикой является широкий (универсальный) сетевой доступ. Существует возможность использования различных типов клиентских платформ (тонких или толстых) для терминальных устройств. Эта возможность существует благодаря наличию стандартных механизмов передачи информации на большие расстояния.

Третий принцип – объединение ресурсов. На стороне поставщика существует единый пул вычислительных ресурсов, который можно использовать для совместного доступа множеством пользователей. Т.е. все машины поставщика работают как единый механизм.

Мгновенная масштабируемость. Размеры предоставления облачных вычислений конфигурируются в реальном времени. Они могут быть предоставлены или освобождены по требованию пользователя.

И последнее, но не последнее по значимости, – измеряемый сервис. Оплата предоставления услуг облачного сервиса производится только за реально использованные объемы, а не в рамках постоянной оплаты. Измерение ресурсов возможно за счет автоматического учета и распределения нагрузки по всему пулу ресурсов.

Рассмотренные характеристики будут учитываться в работе по созданию архитектуры серверной части, так как именно они определяют, является ли сервис по предоставлению вычислительных ресурсов облачным или нет. Сервис должен соответствовать всем пяти указанным пунктам.

У облачных вычислений существует три сервисных модели:

1) *Software as a Service (SaaS)* – ПО как услуга.

То есть поставщик предоставляет доступ к некоторому сервису, развернутому внутри облака, пользователь может получить доступ по средствам веб-интерфейса (через браузер) или с помощью веб-приложения. В данной модели не предусмотрено изменение структуры облака

пользователем. Хранилища, операционные системы, сети и иные настройки доступны исключительно поставщику услуги.

2) *Platform as a Service (PaaS)* – платформа как услуга.

В этой модели пользователь может управлять частью вычислительных ресурсов, выбирать операционные системы, СУБД, средства разработки и тестирования ПО, а также получает право устанавливать требующиеся прикладные программы. Т.е. потребитель арендует виртуальную машину и может ее конфигурировать по своему усмотрению. Данная модель, как и первая, не дает пользователю управлять самой инфраструктурой облака.

3) *Infrastructure as a Service (IaaS)* – инфраструктура как услуга.

Только в этой сервисной модели потребитель может управлять вычислительными ресурсами, но опять же не инфраструктурой всего облака. Он получает в аренду уже не виртуальную машину, а пользуется арендованными абстрактными вычислительными мощностями – сетевым временем, дисковым пространством и другими. Здесь возможна самостоятельная установка операционных систем, настройка сетей, установка ПО не из списка предоставляемых поставщиком программ.

По типу развертывания выделяют три основных модели и одну комбинированную:

- *Private cloud (частное облако)* – облачные вычисления в рамках одной организации.
- *Community cloud (облако сообщества)* – облачные вычисления, настроенные под решение конкретной задачи в рамках одной организации.
- *Public cloud (публичное облако)* – предоставляет свободный доступ к вычислительным ресурсам любым пользователям.
- *Hybrid cloud (гибридное облако)* – комбинированная модель развертывания облачных вычислений, представляет собой структуру из соединенных между собой облаков трех предыдущих моделей, при этом не нарушается целостность входящих в нее моделей.

Итак, исходя из рассмотренных сервисных моделей и моделей развертывания, можно сделать вывод о многослойности облачных структур как по уровням доступности вычислительных ресурсов поставщика, так и по уровню доступа к облачной структуре в целом.

Фундаментом облачных сервисов является физическая инфраструктура, т.е. серверные стойки с оборудованием, хранилища данных, сетевая инфраструктура и системное программное обеспечение облачного дата-центра или нескольких взаимосвязанных дата-центров, так как крупные облачные сервисы могут не ограничиваться одной физической локацией.

### **1.3.2 Применение облачных вычислений в системах умных домов**

Почему для хранения информации из систем автоматизации удобнее всего использовать облачные сервисы?

Во-первых, как указывалось ранее во втором разделе второго параграфа, облачные вычисления позволяют частично решить проблемы стандартизации, являющиеся одной из основных причин замедления развития Интернета вещей. Так, подключая все устройства к облаку, можно настраивать виртуальное взаимодействие между ними независимо от марки производителя и используемого протокола передачи.

Во-вторых, облачные вычисления дают возможность масштабирования домашней или офисной сети. Достаточно подключить новое устройство к облаку, настроить параметры взаимодействия с другими объектами и продолжить использование системы.

В-третьих, работа с системами автоматизации требует хранения большого объема данных, поступающих с устройств. В этом случае также выигрывает вариант использования облачного сервиса, так как на стороне поставщика сосредоточены большие вычислительные мощности, и он может предоставить к ним доступ мгновенно и без видимой реконфигурации уже существующей настроенной системы, в то время как дополнение

самодельного сервера, находящегося у самого клиента, не редко приводит к возникновению простоя в работе системы.

Четвертый плюс использования облачных сервисов – нет необходимости в настройке сервера в домашних условиях, поскольку без определенного набора знаний невозможно настроить сеть с высоким уровнем безопасности.

Из рассмотренных сервисных моделей при использовании облачного сервиса для хранения данных из частных сетей автоматизации зданий и удаленного управления ими больше всего подходит модель *public SaaS* (программное обеспечение как услуга предоставляемого всем желающим того пользователям). Для конечного пользователя преимуществом использования сервиса с такой моделью является получение обработанных результатов без настройки серверов и баз данных вручную.

Но модель *SaaS* подходит не во всех случаях. Облачные сервисы для обработки информации предоставляют их в стандартном виде – графиков или таблиц с данными, что не всегда удовлетворяет запросы пользователя. Если потребитель желает развернуть собственноручно написанный сайт или веб-приложение, то ему необходимо предоставить сервисную модель *PaaS* (платформа как услуга), в рамках которой пользователю предоставляется машина с установленной операционной системой, веб-сервером, сервером баз данных и остальными серверами по требованию.

## **1.4 Проблемы проектирования облачных сервисов для интеллектуальных зданий и пути их решения**

Так как разработка облачного сервиса, который позволяет принимать, хранить и интерпретировать данные из частных сетей пользователей, не является уникальной в своем роде, следует провести анализ существующих систем, выявить их недостатки и предположить возможные пути устранения.

### 1.4.1 Обзор существующих облачных сервисов

Для изучения было взято три облачных сервиса: Intel IOT Analytics, Ubidots и ThingSpeak. Проводился анализ работы с точки зрения пользователя. Аналогов на русском языке не найдено.

#### 1. Intel IOT Analytics (Beta version) [6]

Сервис находится на стадии бета-версии. Имеют ограничения по подключаемым устройствам: работает только с устройствами Intel IOT Platform. Так как устройства данного типа не было в наличии, рассмотрение сервиса происходило по статьям [7] и [8]. Сервисом предоставляется графический интерфейс отображения информации в виде графиков и набора стикеров. Отправка и получение данных устройства происходит в *JSON* (англ. JavaScript Object Notation) формате – текстовый формат обмена данными, основанный на JavaScript. Оповещения об аномальных значениях с датчиков может быть прислано только на специальный адрес по HTTP.

Такой сервис в виду ограничения платформ пользователям будет неинтересен, за исключением обладателей Intel IOT устройств, и разработчиков, собирающихся создавать сложную профессиональную сеть.

#### 2. Ubidots [9]

Данный сервис тестировался в рамках исследовательского проекта «Comfortable Workplace», проводимого группой студентов в исследовательской лаборатории INRIA института INSA Lyon, Франция. В данном проекте был создан прототип беспроводной сенсорной сети, осуществляющий мониторинг температуры, влажности и освещенности окружающей среды. Аппаратной основой для проекта служили модули Zolertia Z1 [10] и датчики температуры, влажности и освещения, входящие в комплект разработчика, и операционная система – ContikiOS [11].

Пример графического отображения данных представлен на рисунке 1.4.

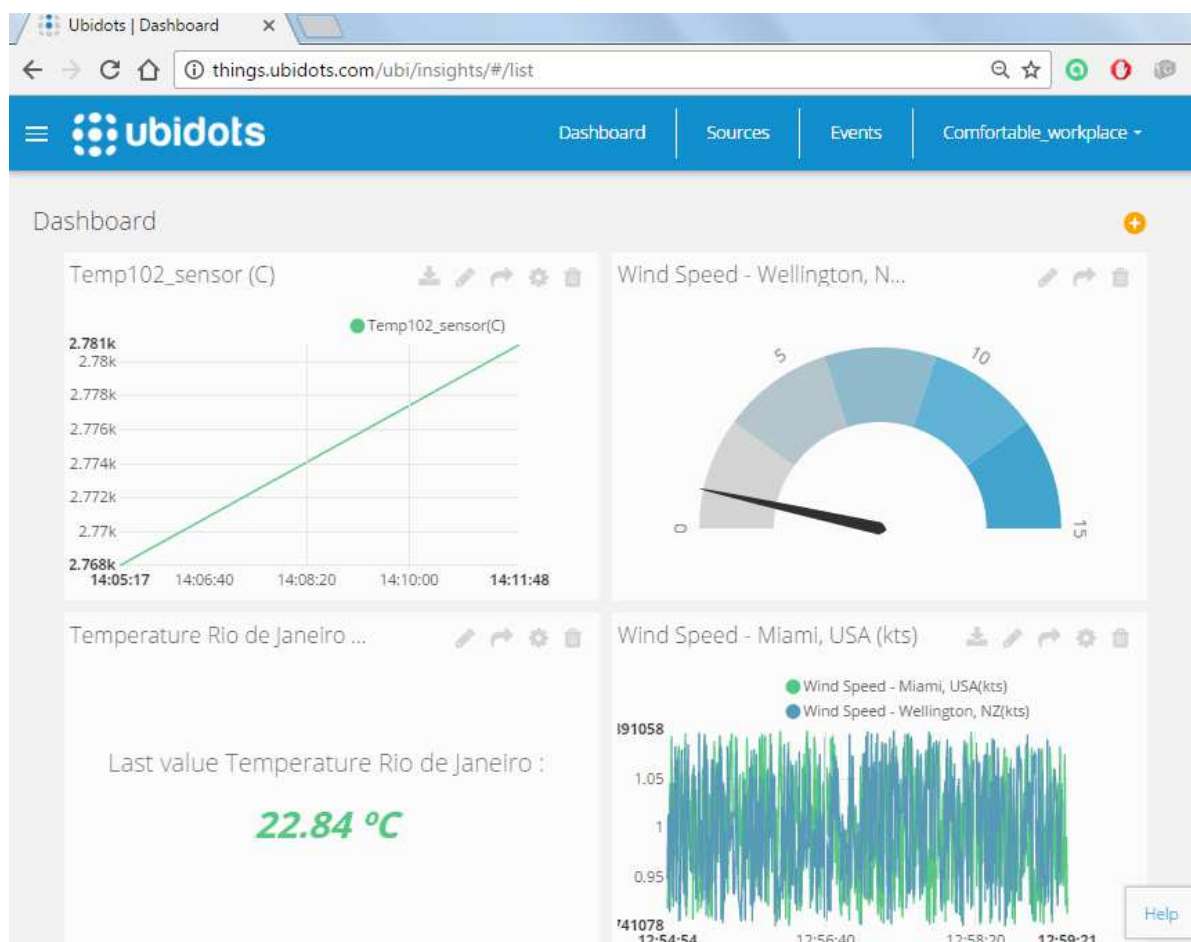


Рисунок 1.4 – Пример пользовательского интерфейса сервиса Ubidots

Положительными моментами использования данного сервиса является удобный, дружелюбный графический интерфейс, наличие скриптов для отправления данных с устройств на нескольких языках: Python, Java, C, PHP, Node и Ruby. Так же существуют специальные библиотеки для операционных систем под устройства в беспроводных сенсорных сетях, включая изучаемый Z1 модуль.

Ubidots располагает широкими возможностями по обработке данных и оповещению пользователей, но эти услуги предоставляются за отдельную плату. Также существует ограничение на количество подключаемых устройств. Еще одним минусом сервиса оказалось несвоевременное обновление информации на сайте или же потеря данных.

В связи с указанными серьезными недостатками системы Ubidots в рамках того же проекта был рассмотрен еще один облачный сервис.

### 3. ThingSpeak [12]

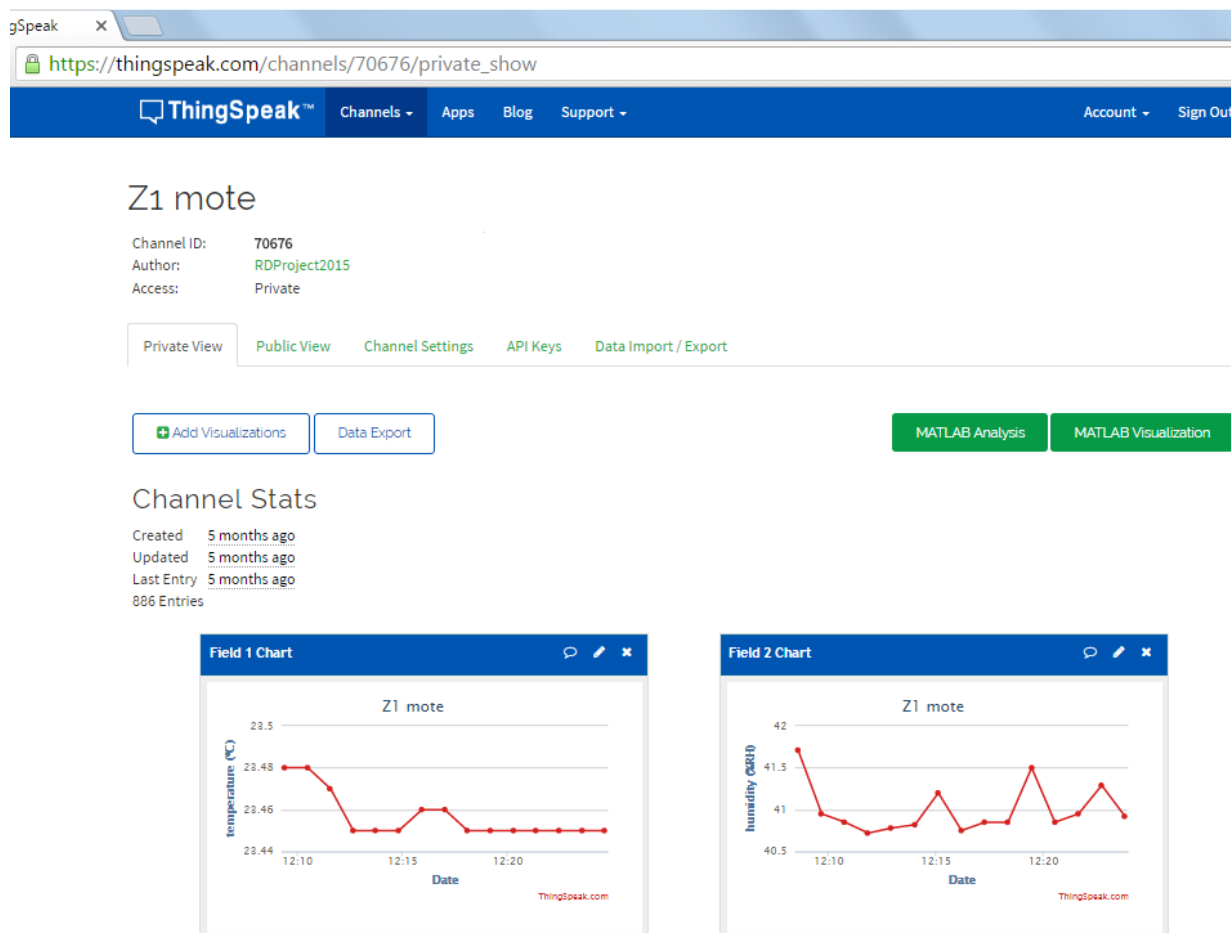


Рисунок 1.5 – Пример работы сервиса ThingSpeak

Для подключения к данному portalу устройств с сенсорами был использован Python скрипт, с использованием POST и GET запросов, так как все взаимодействие с сервисом происходит через HTTP-запросы напрямую. На портале прописаны все необходимые функции для обращения с сервисом.

Этот сервис предназначен для приема большого потока данных и его анализа, в том числе математического с помощью Matlab надстройки. Графический интерфейс не подстроен под создание систем автоматизации «Умных домов», но имеет функции для размещения полученных внутри этого сервиса данных на ваш личный сайт. Система оповещения пользователей об аномальных значениях параметров настроена на отправку сообщений в Twitter.

### 1.4.2 Основные проблемы

Из рассмотренных примеров облачных сервисов можно сделать вывод о том, что системы с хорошим графическим интерфейсом, предназначенные для сбора информации именно из сетей автоматизации зданий не являются полностью бесплатными и имеют проблемы с производительностью вычислительных мощностей. Системы, имеющие хорошую производительность, предназначены только для хранения информации и ее математической обработки, но обладают простым пользовательским интерфейсом. Часть сервисов по автоматизации домов настраиваются под определенную систему.

Если учесть выводы, которые были сделаны в первом параграфе главы о проблемах безопасности, и соотнести их с полученными экспериментальными данными, получим следующую ситуацию.

Основными проблемами проектирования облачных сервисов для систем умных домов являются:

- безопасность – проектирование облака, надежно хранящего личные данные, полученные от пользователя; изоляция данных пользователей;
- надежная передача данных по защищенным соединениям;
- отсутствие гибкой настройки взаимодействия устройств внутри самого облака;
- графическое представление данных для удобного визуального восприятия
- распределение потоков данных от пользователей для обеспечения оптимальной пропускной способности каналов связи;
- подбор аппаратного обеспечения для гарантии бесперебойного обслуживания клиентов;
- надежное хранение данных – резервное копирование, использование RAID-ов;



### 1.4.3 Пути усовершенствования сервисов

Первую проблему – безопасности хранения данных пользователя – можно решить использованием средств виртуализации и применением принципа *мультитенантности* – возможности изолированно обслужить разных пользователей, используя разделение физических и логических ресурсов. Например, для каждого пользователя создается своя виртуальная машина, на которой хранится вся информация, полученная с его устройств.

Защищенная передача данных возможна с использованием протоколов шифрования данных на аппаратном и сетевом уровне: TSL и SSL, упоминавшихся в части третьей части первого параграфа настоящей главы.

Для устранения проблемы взаимодействия между устройствами следует создать расширенный список команд, возможных для исполнения после срабатывания установленного пользователем условия для конкретного датчика. А также возможность добавления пользовательских функций для конкретного устройства в проекте.

Графический интерфейс сервиса должен быть максимально удобным для пользователя и подстраиваться под существующие задачи. Желательно вынесение большей части обработки компонентов на сторону клиента, на стороне сервера обрабатывать исключительно SQL-запросы для уменьшения нагрузки на серверы.

Уменьшение нагрузки на серверы можно добиться распределением запросов по нескольким IP-адресам с дополнительной настройкой DNS-сервера.

Проблема аппаратной части будет возникать снова и снова по мере увеличения количества обслуживаемых пользователей, задача состоит в том, чтобы подобрать оптимальное начальное решение для реализации аппаратной части сервера, которое максимально долго не потребует расширений и при этом будет приемлемо по цене.

## **2 Разработка архитектуры серверной части облачного сервиса «Комфортная среда»**

В данном разделе описаны теоретические модели, разработанные в рамках проекта, для создания полноценного облачного сервиса «Комфортная среда», который будет использоваться для получения и хранения информации с датчиков, значения которых можно представить в виде таблиц или графиков. В первом параграфе предполагается описание общих принципов взаимодействия клиентов с сервисом, второй параграф посвящен организации серверной части, в третьем параграфе содержится анализ полученной архитектуры.

### **2.1 Проектирование взаимодействия клиентов и сервиса «Комфортная среда»**

В этом параграфе рассматривается организация взаимодействия клиента и сервиса сначала на уровне абстракций, а затем на уровне сетей, проводится анализ полученной структуры.

#### **2.1.1 Требования пользователя к системе сервиса**

С точки зрения пользователя облачный сервис должен предоставлять ресурсы для хранения пользовательских данных, обрабатывать их и выводить в удобном для клиента виде.

Так как задание выпускной квалификационной работы предполагает разработку сервиса для хранения данных о температуре, влажности и освещенности – а это числовые значения, то требования к графическому представлению данных со стороны пользователя будут минимальны. Сервис должен выводить информацию, записанную в базу данных пользователя, в виде таблиц со значениями или графиков.

Такой способ представления информации прост в понимании и реализации. Но пользователю может понадобиться постоянный просмотр данных только за определенный период времени или на текущий момент. Соответственно, клиент должен иметь возможность сконфигурировать панель данных так, как ему это будет удобно, а так же иметь доступ к ресурсам, как с персонального компьютера, так и со смартфонов и планшетов без снижения степени удобства.

Устройства, подключаемые к облачному сервису, не должны зависеть от фирмы-производителя, поддерживаемых стандартов и протоколов передачи данных. Необходим простой механизм отправления данных с устройства. Этот критерий очень важен при создании сервиса, так как многие клиенты выбирают облачные решения из-за потребности обмена данными между аппаратно-несовместимыми устройствами.

Пользователю необходим доступ к личным данным удаленно, с использованием веб-браузера или веб-приложения.

Из указанного перечня требований пользователя можно сформировать архитектуру взаимодействия сервиса и клиентской части (рисунок 2.1).



Рисунок 2.1 – Взаимодействие пользователя и сервиса

Пользователь через сеть Интернет получает доступ к облачному сервису, регистрируется и создает новый проект. В этом проекте он указывает количество устройств и типы датчиков в описании каждого из них. Далее пользователь получает коды для передачи данных с устройств в

облачное хранилище, запускает соответствующие скрипты на устройствах и снова заходит на портал сервиса. Пройдя аутентификацию, он получает доступ к графическому представлению отправленных данных.

Так выглядит сообщение между клиентом и сервером с точки зрения пользователя.

### 2.1.2 Структура разрабатываемого облачного сервиса

Любое хранение потока данных на серверной стороне клиент-серверного взаимодействия предполагает использование баз данных. А создание портала для удаленного просмотра отправляемых данных неизбежно связано с настройкой веб-серверов.

В самой простой конфигурации структура выглядит как на рисунке 2.2. Сервер баз данных и веб-сервер располагаются на одной хост-машине.



Рисунок 2.2 – Взаимодействие пользователя и сервиса. Одна хост-машина

Такая конфигурация сервера пригодна для небольших нагрузок. При увеличении количества пользователей работа с базой данных (далее БД) будет занимать значительную часть вычислительных ресурсов, что может отрицательно сказаться на работе остального программного обеспечения.

Если обратиться к характеристикам, определяющим облачные вычисления, указанным в третьем параграфе первой главы, то сервис с такой конфигурацией нельзя считать облачным. Эту структуру невозможно

мгновенно масштабировать, пул вычислительных ресурсов состоит из одного сервера.

Как добавить масштабируемость сервису и расширить внутреннюю структуру?

В целях обеспечения безопасности данных клиента целесообразно вынести базу данных с его информацией во внутреннюю сеть. Так как предполагается дальнейшее расширение облачного сервиса и предоставление услуг по модели РaaS, следует не просто отделить базу данных от центрального сервера, а установить ее на виртуальную машину, запускаемую на самом сервере или на других серверах в той же сети.

Где разворачивать виртуализацию?

При большом количестве подключаемых клиентов разворачивание виртуальных машин на одном сервере не целесообразно, так как это существенно снизит производительность системы и сделает шлюз сети узким местом – так называемым бутылочным горлышком. Одним из вариантов является запуск виртуальных машин на дополнительном сервере в локальной сети и предоставление доступа к хранящимся на них данным.

Если рассмотреть инфраструктуру с указанными дополнениями, то получим сервис с конфигурацией на рисунке 2.3:



Рисунок 2.3 – Архитектура облачного сервиса с выделением БД пользователя на виртуальную машину на дополнительном хосте

Следующий этап проектирования облачного сервиса – определение структуры внутренней сети сервера, определение иерархии серверов. Но для начала необходимо рассмотреть подключение устройств и клиентов к сервису с указанной архитектурой.

### **2.1.3 Подключение устройств к сервису**

После создания учетной записи пользователя и получения ключей для устройств создается виртуальная машина, на которую будет поступать информация с аппаратуры клиента. Соединение для передачи данных от устройств должно быть настроено не с центральным сервером, на котором базируется сайт, а с сервером, на котором развернута виртуальная машина пользователя. Необходима маршрутизация пакетов внутри локальной сети и передача пакетов во внутреннюю сеть (*IP forwarding*).

Отправка данных с устройств по заданию выпускной квалификационной работы должна происходить с помощью скрипта на языке Python. Это обусловлено тем, что необходимо генерировать поток данных, эмулирующий работу реального устройства.

Передачу данных следует вести с использованием метода POST, так как он обеспечивает больший объем передаваемой информации и, в отличие от метода GET, не кэширует передаваемые данные и не выставляет их в адресную строку.

Для приема данных на сервере создается обрабатывающий скрипт, анализирующий полученные данные и направляющий их в БД на соответствующую машину в виртуальной сети.

В связи с наличием нескольких таких машин необходима таблица соответствия указываемых пользователем в форме запроса данных и IP-адресов виртуальных машин.

Задача меняется, если количество пользователей будет превышать 253, так как в этом случае необходим еще один или более физических серверов в основной сети, алгоритм поиска машины клиента будет усложнен.

### **2.1.4 Подключение пользователей к сервису**

Подключение пользователей к облачному сервису следует вести по защищенному протоколу HTTPS, использующему протокол SSL для шифрования данных.

Пользователь будет подключен к серверу, на котором разворачивается информационный портал в виде сайта, а информация из БД пользователя, находящейся на виртуальной машине, будет предоставляться центральному серверу через локальную сеть.

## **2.2 Внутренняя организация серверной части сервиса**

Для того чтобы разработка архитектуры обрела законченную форму, необходимо спроектировать определенную структуру сетевого подключения элементов инфраструктуры, а так же определить хранимую информацию и форму ее размещения в виде таблиц баз данных.

### **2.2.1 Настройка сети под выбранную архитектуру**

За основу построения сети принята архитектура, разработанная в пункте 2 первого параграфа текущей главы (рисунок 2.3).

Для публикации портала облачного сервиса необходим статический IP, он предоставляется Интернет-провайдером на платной основе и присваивается центральному серверу сети, на котором будет размещен сайт облачного сервиса. Так как сервис имеет сложную архитектуру (рисунок 2.4), необходимо создание двух и более локальных сетей. Шлюзом из внешней сети в локальную будет служить центральный сервер.

Первая сеть включает в себя реальные серверные машины, на которых установлены веб-сервер, база данных для учета существующих виртуальных машин и соотношения их адресов с именами пользователей, и средства

виртуализации. Адреса сети выдаются DHCP-сервером, установленным на шлюзе во внешнюю сеть, и имеют маску подсети 255.255.255.0.



Рисунок 2.4 – Сетевые настройки разрабатываемого облачного сервера

Второй тип локальных сетей используется для виртуальных машин. Адреса выдаются из диапазона 192.168.[номер машины физического хоста].х. То есть на физической машине с IP-адресом 192.168.100.2 будут выдаваться адреса 192.168.2.х для виртуальных машин.

Для открытия сообщения между сетями необходимо прописать пути в таблице маршрутизации и разрешить пересылку пакетов между сетями:

1. На центральном сервере: путь достижения компьютеров в сети 192.168.у.х через шлюз 192.168.100.у.
2. На машинах с гипервизорами: разрешить IP-форвардинг и прописать шлюз в глобальную сеть.

Кроме уже выполненных настроек необходимо прописать так называемый «проброс портов» или трансляцию порт-адрес (DNAT – Destination NAT). Этот механизм позволяет перенаправить пакеты запросов на внутренние адреса сети в соответствии с указанным портом. Эта настройка нужна для отправки данных с устройств пользователя не на центральный сервер, а на физическую машину, на которой стоит нужная виртуальная.



## **2.2.2 Структура баз данных для хранения информации о пользователях, виртуальных машинах и получаемых данных**

В архитектуре облачных сервисов будут выделены следующие типы баз данных:

- БД информации о пользователях – установлена на центральный сервер; хранит данные о пользователе и информацию о том, на каком локальном физическом хосте находится виртуальная машина пользователя на данный момент.
- БД запуска виртуальных машин – установлена на центральном сервере; содержит информацию о последнем идентификаторе в IP-адресе виртуальной машины, которая доступна для создания.
- БД соответствия IP-адресов пользователям – установлена сервер, если на нем планируется развернуть виртуальные машины; хранит логин пользователя и IP-адрес соответствующей ему виртуальной машины.
- БД данных пользователя – установлена на виртуальную машину; содержит таблицу проектов, устройств и датчиков пользователя, персональные настройки интерфейса, данные, полученные от аппаратуры пользователя.

Первые две базы данных можно объединить в одну.

Описание таблиц баз данных на серверах и виртуальных машинах:

### **1. Центральный сервер**

а. Таблица информации о пользователе – содержит уникальный идентификатор пользователя (ключ), уникальное имя пользователя (он же логин), пароль, адрес электронной почты, отметка о согласии сбора персональной информации, IP-адрес, закрепленный за виртуальной машиной пользователя.

б. Таблица запуска виртуальных машин – содержит IP-адрес реальной хост-машины (ключ), на которой возможен запуск виртуальных

машин, и последний идентификатор свободного для создания виртуальной машины IP-адреса.

## 2. Сервер для виртуальных машин

Таблица соответствия логина пользователя (ключ) и IP-адреса виртуальной машины

## 3. Виртуальная машина

a. Таблица проектов пользователя – уникальный идентификатор проекта (ключ), уникальное название проекта, сгенерированный ключ проекта для записи данных в проект.

b. Таблица устройств внутри проекта – уникальный идентификатор устройства (ключ), уникальное имя устройства в рамках проекта, сгенерированный ключ устройства для записи полученных данных в базу, идентификатор проекта, в котором находится устройство.

c. Таблица датчиков устройств – уникальный идентификатор датчика (ключ), уникальное имя датчика в пределах устройства, идентификатор устройства, в котором находится датчик, тип графического интерфейса для отображения на портале в последний визит пользователя.

d. \*(необязательна) Таблица типов графического интерфейса – уникальный идентификатор графического интерфейса (ключ), название графического интерфейса.

e. Таблицы значений датчиков (наименование таблицы в соответствии с уникальным именем датчика и устройства, в которое он включен) – уникальный идентификатор записи (ключ), значение, дата записи.

Количество устройств в одном проекте целесообразно ограничить 10. Количество датчиков, подключенных к устройству – 8. Такие ограничения не позволят переполнить базу данных клиента, и не потребуется расширения памяти виртуальной машины.

Таблица стилей отображения графических элементов необязательна и может быть создана для удобства работы.

### **2.2.3 Способы настройки виртуализации**

Так как в основу создания проекта закладывались технические требования по реализации серверной платформы на базе операционной системы Ubuntu 14.04 LTS server [13], то в качестве средства виртуализации предлагается использование гипервизора XEN [14]. Данный монитор виртуальных машин позволяет добиться высокой производительности всех запущенных систем, в отличие от систем эмуляции Oracle VM VirtualBox [15] или VMWare [16] и им подобных. XEN поддерживает возможность клонирования, что необходимо для гибкости использования вычислительных ресурсов облачного сервиса, переноса виртуальных машин между физическими хостами. Также он удобен для создания виртуальных машин с заданными характеристиками через командную строку.

## **2.3 Анализ разработанной архитектуры**

Выбранная конфигурация сети удобна при возникновении поломок серверной части облачного сервиса или проведения профилактических работ без прекращения работы системы. В случае выхода из строя одного из узлов оборудования, возможно клонирование виртуальной машины пользователя и ее перенос на другую физическую машину. В этой ситуации необходима настройка дополнительной сети в гипервизоре и внесение изменений в таблицы пользователей в центральной базе данных и базе данных физических хостов, настройка таблиц маршрутизации.

В случае проведения профилактических работ на одной из физических машин, также клонируются все образы виртуальных машин пользователей, переносятся на другую машину и делается проброс порта на новый адрес.

Разделение серверов для обслуживания клиентов и устройств должно распределить нагрузку среди вычислительных узлов и обеспечить необходимую пропускную способность.

### **3 Проектирование прототипа облачного сервиса «Комфортная среда»**

Этот раздел посвящен разработке прототипа облачного сервиса, основанного на архитектурных решениях, предложенных во второй главе выпускной квалификационной работы. Описано создание, как серверной части облачного сервиса, так и создание портала для отображения данных.

Для создания серверной части прототипа сервиса «Комфортная среда» использовались персональный компьютер под управлением ОС Windows 7 (Процессор – ADM Phenom™ II N930 Quad-Core 2.00 GHz, ОЗУ 4.00 ГБ) и средства виртуализации Oracle VM VirtualBox и VMWare.

#### **3.1 Серверная часть сервиса «Комфортная среда»**

В качестве операционной системы для серверов использовался дистрибутив системы Ubuntu 14.04 LTS server. Выбранный веб-сервер – Apache2 [17], сервер баз данных – MySQL [18], серверная часть сайта написана на PHP5.5 [19].

##### **3.1.1 Упрощение организации сети**

В связи с тем, что создание прототипа выполняется на виртуальных машинах, а не на физических хостах, было принято решение об упрощении задачи и исключении виртуализации с вспомогательного сервера. Целью практического проектирования является настройка сети вручную между всеми участниками процесса для отладки архитектуры и проверки ее работоспособности. При дальнейшем развитии проекта выполнение аналогичных действий на средствах виртуализации должны быть автоматизированы и внесены в скрипты для выполнения при установке всей системы в целом и запуске виртуальных машин для каждого пользователя, а

образы виртуальных машин будут использованы для ускоренного воссоздания системы на реальных машинах.

#### Скорректированная задача:

Необходимо создать два сервера в единой сети – один центральный, второй дополнительный для работы с подключаемыми устройствами. Оба сервера имеют сетевые карты для подключения в глобальную (через роутер с настроенной трансляцией адресов – NAT) и общую локальную сеть. Дополнительный сервер также имеет сетевую карту для подключения машин пользователей. Создать виртуальные машины пользователей и подключить их ко второму серверу. Создать машину подключаемого устройства, задать сетевые настройки так, чтобы она находилась в глобальной сети. Создать сайт для регистрации, получения ключей запросов и отображения полученной информации, доступный из глобальной сети.

### **3.1.2 Реализация серверной части сервиса**

Сервис строится в домашней сети за роутером, поэтому адреса серверов в глобальной сети по eth0 выдаются автоматически DHCP-сервером роутера. Для того чтобы адреса были статическими, в ARP-таблице роутера прописываются соответствия MAC- и IP-адресов.

Реализация серверной части начинается с создания двух виртуальных машин со следующими параметрами (рисунок 3.1, 3.2):

Первый сервер (центральный) получает по eth0 адрес 192.168.100.10, по интерфейсу eth1 – адрес 192.168.184.1.

Второй сервер (дополнительный) имеет аналогичную настройку, и получает по eth0 адрес 192.168.100.11, по интерфейсу eth1 – адрес 192.168.184.2. Дополнительно подключается третий интерфейс – eth2 с адресом 192.168.2.1

Далее необходимо разрешить доступ к базам данных серверам и настроить маршрутизацию, выполнив следующие команды:

## 1. Настройка IP-форвардинга для серверов 1 и 2.

```
$ grep -i forward /etc/network/options  
ip_forward=yes
```

## 2. Настройка таблиц маршрутизации:

route add -net 192.168.100.0 netmask 255.255.255.0 gw 192.168.11.1 – в случае Виртуальной машины. -net 192.168.2.0 и gw 192.168.184.2 – в случае центрального сервера.

```
ifcmain-admin@central-server:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 08:00:27:1b:3f:cd  
          inet addr:192.168.100.10  Bcast:192.168.100.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe1b:3fcd/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:43 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:4622 (4.6 KB)  TX bytes:2906 (2.9 KB)  
  
eth1      Link encap:Ethernet  HWaddr 08:00:27:ee:8d:31  
          inet addr:192.168.184.1  Bcast:192.168.184.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:feee:8d31/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B)  TX bytes:648 (648.0 B)  
  
lo        Link encap:Локальная петля (Loopback)  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
main-admin@central-server:~$
```

Рисунок 3.1 – Конфигурация сервера 1

```
eth1      Link encap:Ethernet  HWaddr 08:00:27:00:3b:28  
          inet addr:192.168.184.2  Bcast:192.168.184.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe00:3b28/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B)  TX bytes:648 (648.0 B)  
  
eth2      Link encap:Ethernet  HWaddr 08:00:27:50:e4:27  
          inet addr:192.168.11.1  Bcast:192.168.11.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe50:e427/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B)  TX bytes:648 (648.0 B)  
  
lo        Link encap:Локальная петля (Loopback)  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:1184 (1.1 KB)  TX bytes:1184 (1.1 KB)  
  
main-admin@central-server:~$
```

Рисунок 3.2 – Конфигурация сервера 2

### 3.1.3 Считывание данных устройства

Отправление данных на сервер осуществляется с помощью Python-скрипта, приведенного ниже. Переменные var1, var2, var3 – данные с трех датчиков устройства с ключом genKey, указанным в поле 'key' параметров для пересылки данных.

```
import httplib, urllib
import time
import sys
import random

#data = sys.stdin.readline()
def doit():

    #print field
    var1=random.uniform(19,27)
    var2=random.uniform(70,90)
    var3=random.uniform(300,1000)
    params = urllib.urlencode({'sens1':var1,'sens2':var2,'sens3':var3,'key':'fe456kfdt6029455'})
    headers = {"Content-type": "application/x-www-form-urlencoded","Accept": "text/plain"}
    conn = httplib.HTTPConnection("192.168.100.11:80")
    conn.request("POST", "/get_sensor_data.php", params, headers)
    response = conn.getresponse()
    print response.status, response.reason
    data = response.read()
    conn.close()

#sleep for 16 seconds (api limit of 15 secs)
if __name__ == "__main__":
    while True:
        doit()
        time.sleep(16)
```

Считывание данных устройства происходит с помощью глобального массива \$\_REQUEST следующим образом:

```
<?php
    $ip = $_SERVER['REMOTE_ADDR']; //получаем IP адрес клиента
    //получаем идентификатор HTTP клиента
    $client = $_SERVER['HTTP_USER_AGENT'];
    $today = date("Y.m.d H:i:s"); //получаем текущие дату и время
    include ('db_users/user_db.php');
    $param1 = $_REQUEST['sens1'];//получаем значение посланной переменной "sens1"
    $param2 = $_REQUEST['sens2'];//получаем значение посланной переменной "sens2"
```

```

$param3 = $_REQUEST['sens3']; //получаем значение посланной переменной "sens3"
//выполняем запрос к БД
$result = mysql_query ("INSERT INTO data_z (data1, data2, data3,
date) VALUES ('$param1', '$param2', '$param3', '$today')") or
die(mysql_error);
mysql_close($db);
?>
<p>Data read page</p>

```

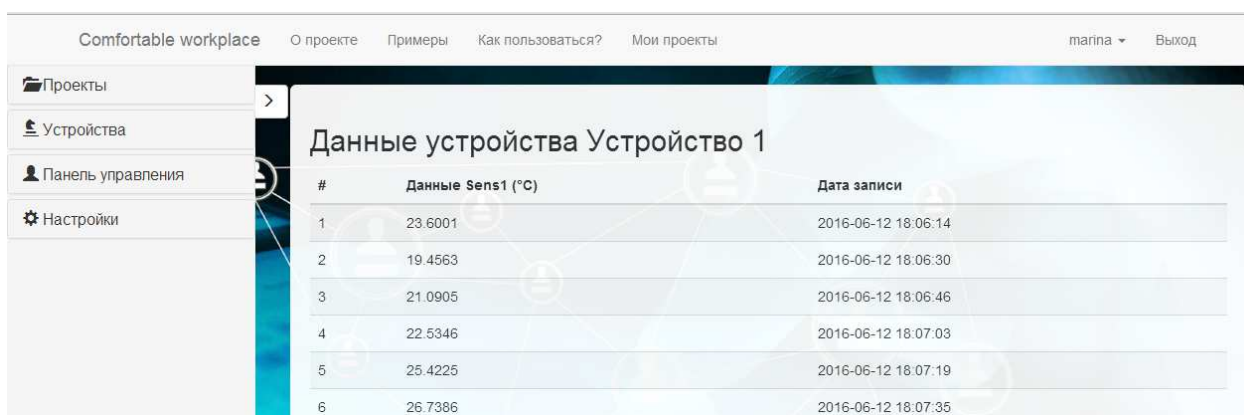
## 3.2 Клиентская часть сервиса «Комфортная среда»

Клиентская часть разрабатывалась на основе фреймворка Bootstrap 3.3.6 [20] и готовых элементов, представленных на официальном сайте с использованием JQuery библиотеки JavaScript и PHP5.5 в качестве скриптового языка на стороне сервера.

### 3.2.1 Отображение информации устройств

Отображение данных с устройств пользователя происходит в двух вариантах: первый – таблица с данными (рисунок 3.3), второй – график (рисунок 3.4).

Обновление таблиц и графиков реализовано при помощи технологии Ajax, встроенной в библиотеку JQuery. Для графиков используется библиотека для рисования графиков Highcharts [21].



#	Данные Sens1 (°C)	Дата записи
1	23.6001	2016-06-12 18:06:14
2	19.4563	2016-06-12 18:06:30
3	21.0905	2016-06-12 18:06:46
4	22.5346	2016-06-12 18:07:03
5	25.4225	2016-06-12 18:07:19
6	26.7386	2016-06-12 18:07:35

Рисунок 3.3 – Отображение данных устройства в виде таблицы



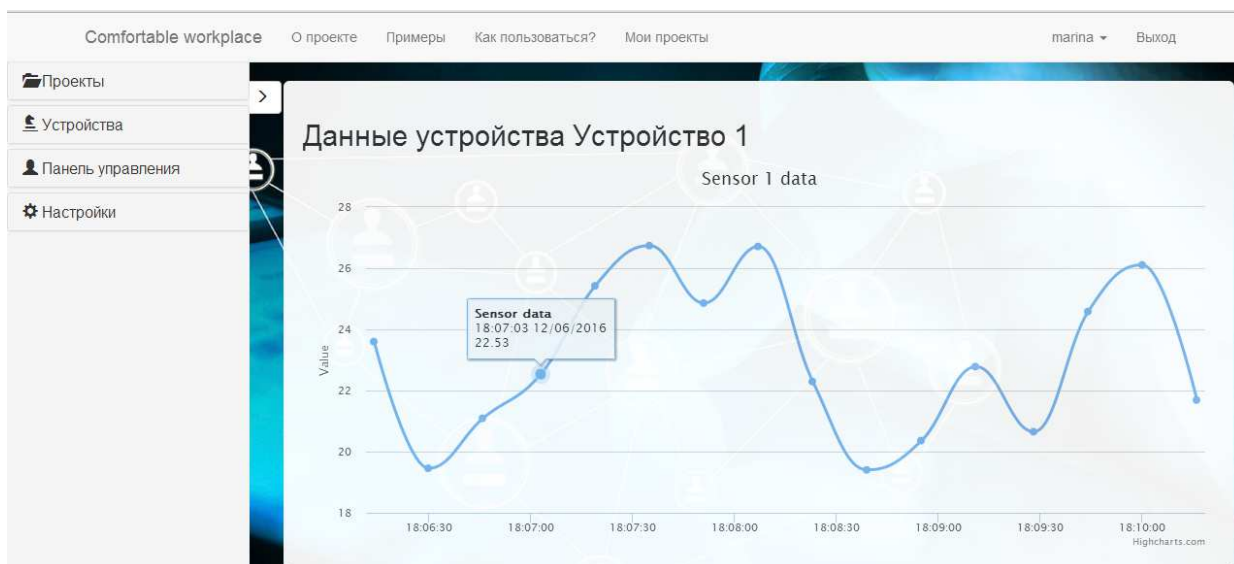


Рисунок 3.4 – Отображение данных устройства в виде графика

### 3.2.2 Инструкция по использованию сайта

Заходя на портал облачного сервиса «Комфортная среда» пользователь видит информацию, представленную на рисунке 3.5

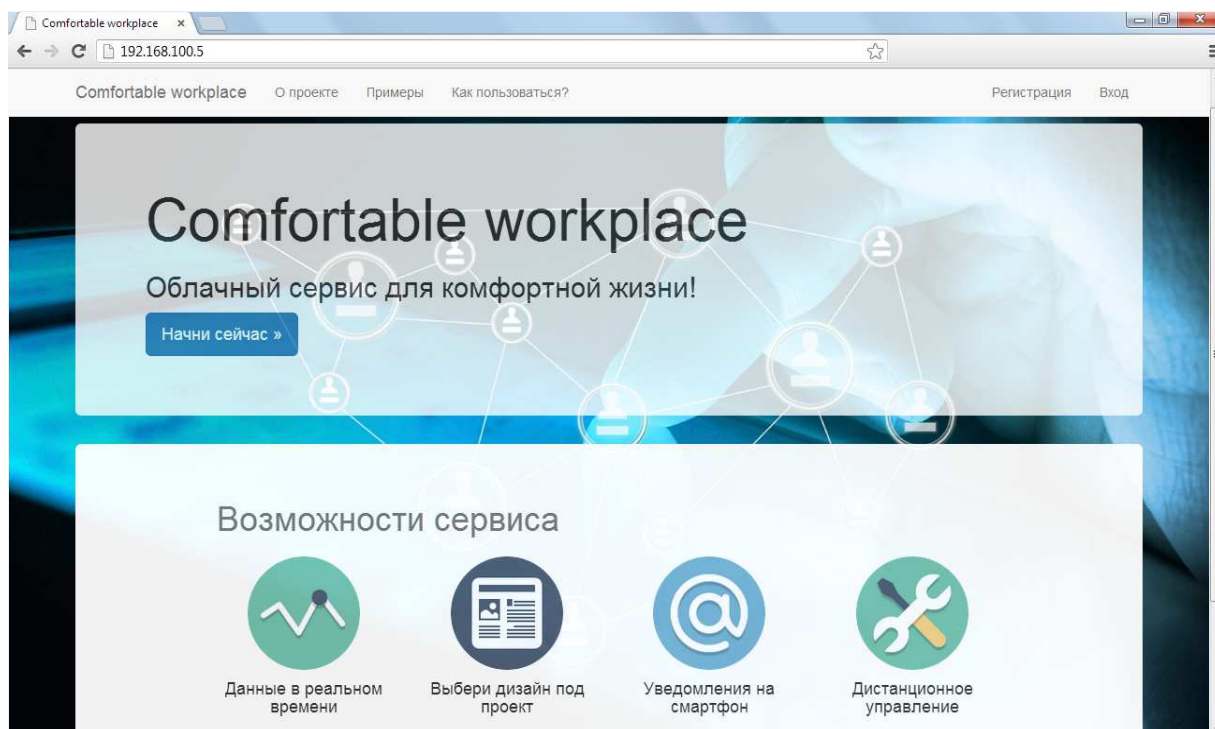


Рисунок 3.5 – Главная страница портала «Комфортная среда»

Если пользователь еще не зарегистрирован, то он может попасть только на три страницы, не включая главную: на страницу описания проекта, страницу примеров или инструкции по использованию сайта. На момент

разработки прототипа эти разделы не заполнены. Позже на них будет находиться соответствующая информация.

При желании пользователь может зарегистрироваться, нажав кнопку «Регистрация» в правой стороне меню шапки сайта. После чего он попадает на страницу регистрации (рисунок 3.6). Данная форма оснащена информированием об ошибках ввода, включая незаполненные поля, несовпадения паролей и введение уже существующего в системе логина (рисунок 3.7).

После успешной регистрации пользователь попадает на страницу проектов (рисунок 3.8). Здесь, после создания, можно будет увидеть таблицу с названиями всех созданных пользователем проектов. Для создания нового проекта пользователь должен нажать на кнопку «Добавить проект», после чего для него будет выведено модальное окно с предложением ввести название проекта (рисунок 3.9).

Рисунок 3.6 – Регистрационная форма портала «Комфортная среда»

Рисунок 3.7 – Регистрационная форма портала «Комфортная среда». Ввод существующего логина

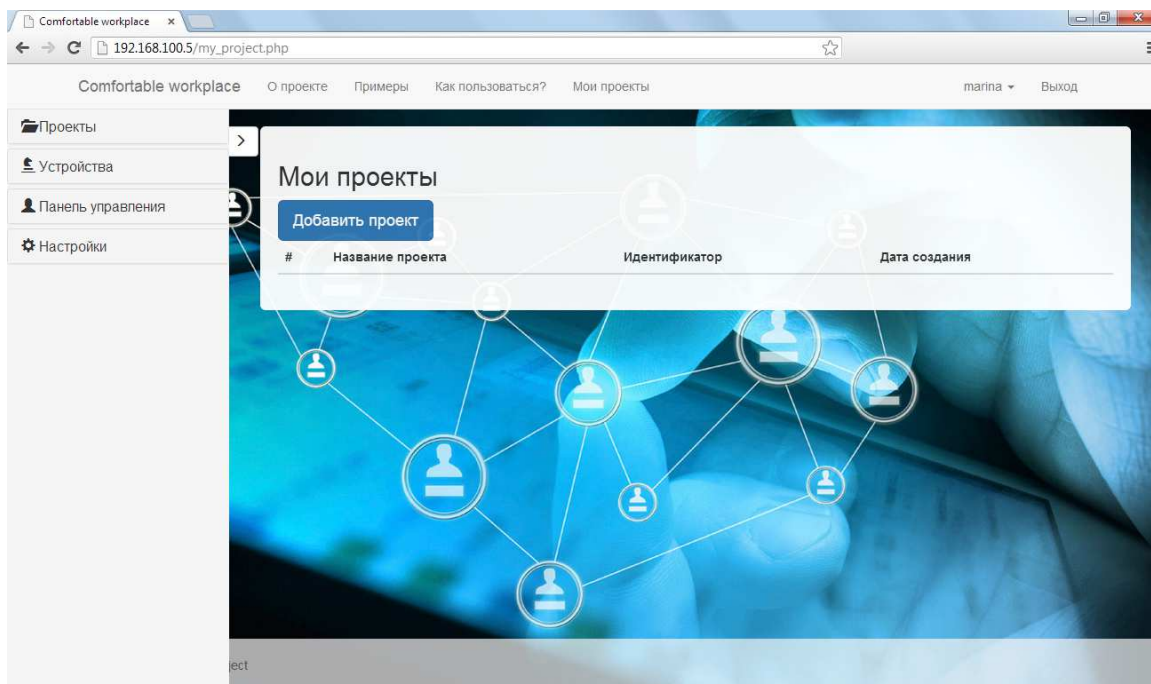


Рисунок 3.8 – Страница проектов портала «Комфортная среда»

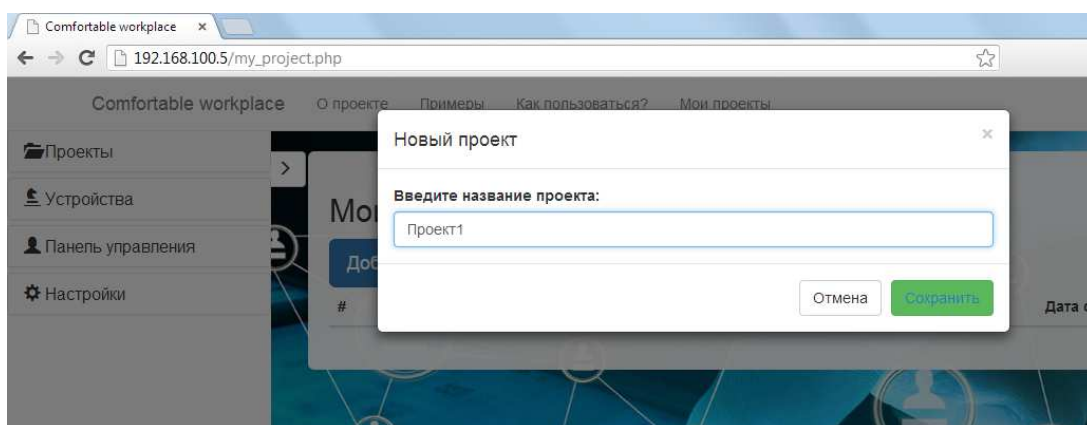


Рисунок 3.9 – Модальное окно добавления проекта пользователем портала «Комфортная среда»

После добавления проекта пользователь может добавить устройства, входящие в него (рисунки 3.10 и 3.11).

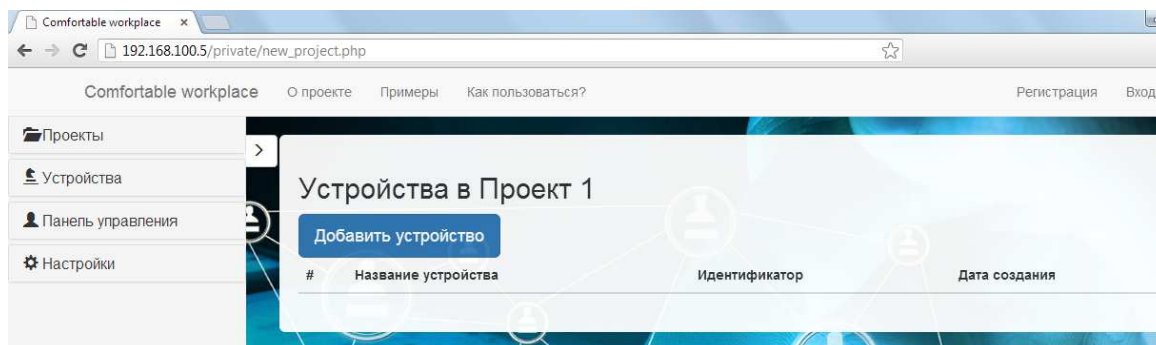


Рисунок 3.10 – Страница устройств, включенных в проект

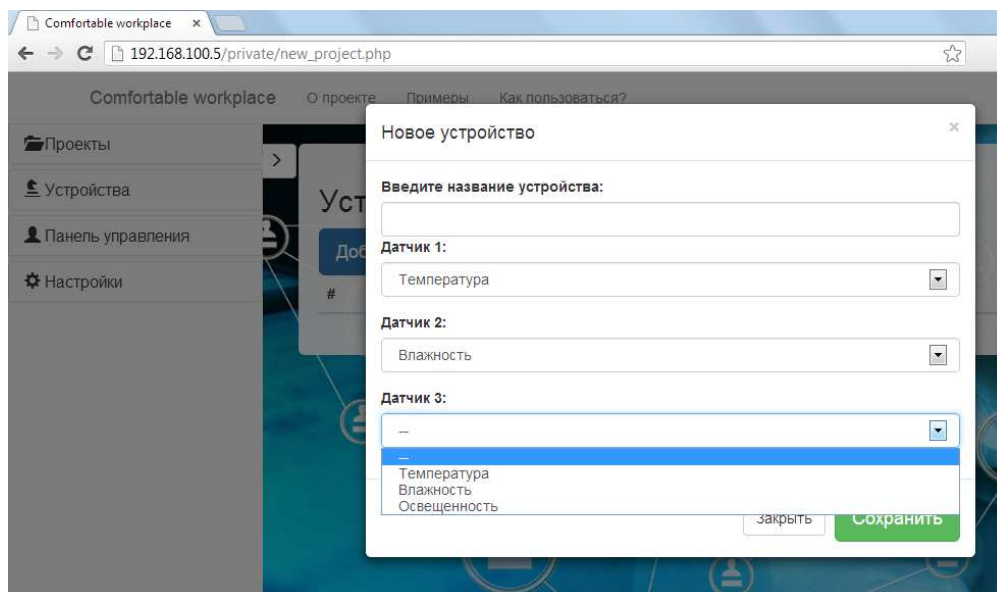


Рисунок 3.11 – Модальное окно для добавления нового устройства

Так как в рамках проекта проектируется только прототип сервиса, устройство может иметь только три датчика одного из трех указанных типов.

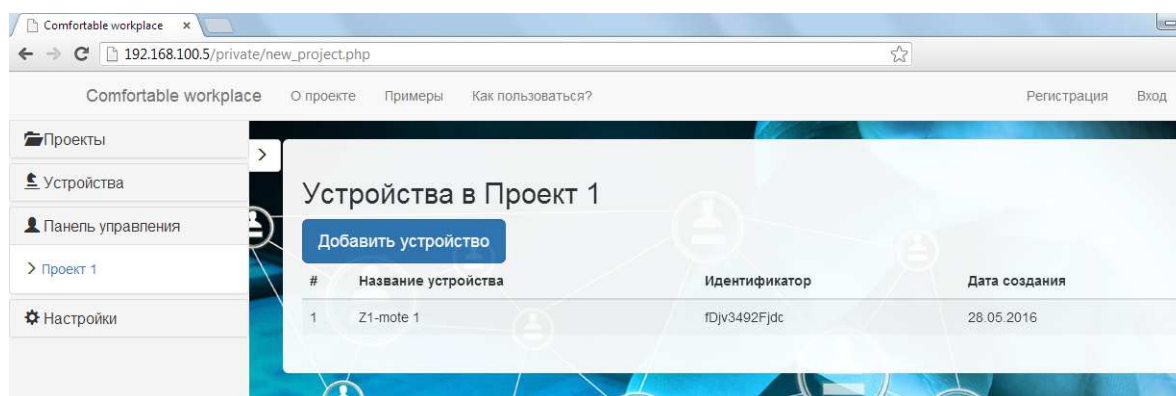


Рисунок 3.12 – Окно портала «Комфортная среда» с добавленным пользователем устройством

После того, как устройство добавлено к проекту (рисунок 3.12), можно приступать к запуску скрипта `script_post.py`, расположенного в корневой папке виртуальной машины устройства.

## **4 Технико-экономическое обоснование разработки облачного сервиса**

### Актуальность проблемы

Концепция «Интеллектуальных зданий» или «Умных домов» в последние годы набирает все большую популярность, что связано с расширением ассортимента устройств и модулей, используемых для автоматизации зданий, мониторинга среды и экономии затрат на потребляемые ресурсы. Продукцию указанного назначения выпускают как зарубежные, так и российские производители, в то время как услуги крупных многофункциональных облачных сервисов, которые позволяют собирать, хранить и выводить полученную информацию, предоставляются исключительно зарубежными фирмами.

Существует множество готовых программно-аппаратных решений для систем «Умный дом», но они, как правило, дорогостоящие и нерасширяемые. В связи с чем, предпринимаются попытки создавать такие системы самостоятельно. Такие решения дешевле готовых продуктов, но сопряжены с расходами на создание сервера для хранения большого объема данных или аренду облачного хранилища, а также с поиском программного обеспечения или интернет-сервисов для обработки информации и удаленного доступа ко всей системе. Облачные сервисы объединяют в себе описанный выше функционал, но являются бесплатными с ограничениями на количество подключаемых устройств, а часть из них имеет ограниченный список обслуживаемых модулей. Также они имеют стандартный интерфейс представления данных без возможности его модификации.

Проект выпускной квалификационной работы направлен на разработку архитектуры серверной части облачного сервиса для систем типа «Умный дом», включая разработку минимального каркаса сайта для его использования.

## 4.1 Трудоемкость разработки проекта

Для подсчета затрат на разработку прототипа сервиса, в первую очередь, необходимо составить детализированный план работ, и определить исполнителя и трудоемкость каждой из них (таблица 1).

Таблица 1 – Трудоемкость проведенных работ

Наименование работ	Трудоемкость (чел./дни)	
	Студент	Руководитель
Разработка технического задания	3	5
Изучение существующих решений в предметной области	6	2
Уточнение технического задания	1	1
Сбор материалов и эксперименты с существующими программными решениями	30	–
Составление (обзор) аналитического обзора предметной области	14	2 (обзор)
Разработка архитектуры сервиса	5	–
Разработка серверной части	9	–
Разработка клиентской части	8	–
Тестирование сервиса	4	2
Оформление пояснительной записки	14	–
Анализ проведенной работы	1	2
<b>ИТОГО</b>	<b>95</b>	<b>14</b>

## 4.2 Расчет себестоимости разработки проекта

Расчет себестоимости производится по следующим статьям:

- основная и дополнительная заработная плата исполнителей;
- отчисления на страховые взносы (социальное, пенсионное и медицинское страхование);
- затраты на сырье и материалы;
- оплата услуг сторонних организаций;
- затраты на эксплуатацию оборудования;
- амортизационные отчисления;
- накладные расходы.

За основу расчетов взяты данные таблицы 2.

Таблица 2 – Исходные данные

Наименование коэффициента	Условное обозначение коэффициента	Студент	Руководитель
Трудоемкость работ (чел./дни)	$T_{ст} / T_{рук}$	95	14
Дневная ставка заработной платы (руб./день)	$C_{ст} / C_{рук}$	533 руб. 33 коп.	738 руб. 10 коп.
Норматив дополнительной заработной платы (%)	$H_{доп}$	14	
Норматив отчислений на страховые взносы (%)	$H_{соц}$	30	
Норма транспортно- заготовительных расходов (%)	$H_{т.з}$	10	



#### 4.2.1 Основная и дополнительная заработная плата исполнителей

Основная и дополнительная заработные платы исполнителей, т.е. студента и руководителя, рассчитываются исходя из трудоемкости выполненных работ, а также дневной ставки заработной платы в соответствии с занимаемой должностью.

Расходы на выплату основной заработной платы исполнителей рассчитываются по формуле:

$$З_{осн.з/н} = T_{ст} \cdot C_{ст} + T_{рук} \cdot C_{рук},$$

и составляют:

$$З_{осн.з/нл} = 95 \cdot 533,33 \text{ руб} + 14 \cdot 738,10 \text{ руб} = 61000 \text{ руб}$$

Дополнительная заработная платы предполагает расчет по формуле:

$$З_{доп.з/н} = З_{осн.з/н} \cdot \frac{H_{доп}}{100} = 61000 \text{ руб.} \cdot 0,14 = 8540 \text{ руб}$$

#### 4.2.2 Отчисления на страховые взносы

К статье отчислений на страховые взносы относятся:

- отчисления в фонд обязательно социального страхования;
- отчисления в пенсионный фонд;
- отчисления в фонд обязательного медицинского страхования.

Выплаты пропорциональны сумме основной и дополнительной заработной платы:

$$З_{соц} = (З_{осн.з/н} + З_{доп.з/н}) \cdot \frac{H_{соц}}{100} = (61000 + 8540) \cdot 0,30 = 20862 \text{ руб}$$

#### 4.2.3 Затраты на сырье и материалы

В статью затрат на сырье и материалы включены расходы на основные и вспомогательные материалы и изделия, которые могут использоваться во время работы над проектом выпускной квалификационной работы. Калькуляция расходов приведена в таблице 3.



Таблица 3 – Затраты на сырье и материалы

Материалы	Кол-во	Цена, руб.	Сумма, руб.
Бумага для офисной техники, пачка	1	210	210
Картридж для принтера Xerox Phaser 3010 (совместимый)	1	700	700
Флешка Transcend 32Gb	1	729	729
Канцелярские товары			200
ИТОГО:			1839
<b>ВСЕГО</b> (с учетом транспортно-заготовительных расходов – Н <sub>т.з</sub> ):			<b>2022 руб. 90 коп.</b>

#### 4.2.4 Оплата услуг сторонних организаций

Для выполнения проекта, в том числе, связи с руководителем, поиска материалов и тестирования сервиса, необходимо использование услуг Интернет-провайдера.

Оплата за месяц составляет 480 руб. (Пл<sub>мес</sub>), проект предполагает порядка 60 часов работы с сетью Интернет (Т<sub>инт</sub>). Расчет услуг провайдера производится по следующей формуле:

$$У_{стор} = \frac{Пл_{мес}}{21} \cdot T_{инт} = \frac{480}{21} \cdot 60 руб = 1371,43 руб$$

#### 4.2.5 Затраты на эксплуатацию оборудования

Эксплуатация оборудования требует затрат на использование электроэнергии. Тариф оплаты электроэнергии 3 руб. 31 коп. / час. Из расчета эксплуатации ПК в течение всего времени выполнения проекта, и длительности рабочего дня – 8 часов, получаем сумму затрат в размере:

$$З_{э.об} = Дни \cdot Раб.день \cdot Тариф = 109 \cdot 8ч \cdot 3,31 руб / ч = 360,79 руб$$

#### 4.2.6 Амортизационные отчисления

Так как в процессе выполнения проекта задействованы основные средства, а именно ПК студента, то необходим учет его амортизации.

Первоначальная стоимость ПК, на котором выполняются работы по проекту – 21800 руб ( $C_{п.н}$ ). Годовая норма амортизации персональных компьютеров составляет 20% ( $H_{аПК}$ ).

Годовые амортизационные отчисления по основному средству составят:

$$A_{ПК} = C_{п.н} \cdot \frac{H_{аПК}}{100} = 21800 \text{ руб.} \cdot 0,20 = 4360 \text{ руб}$$

Амортизационные отчисления по использованию основного средства в подготовке ВКР рассчитываются по формуле:

$$A_{ПК / ВКР} = A_{ПК} \cdot \frac{T_{ПК / ВКР}}{12} = A_{ПК} \cdot \frac{T_{см} / 21}{12} = 4360 \text{ руб.} \cdot \frac{95 / 21}{12} = 1643,65 \text{ руб}$$

#### 4.2.7 Накладные расходы

Размер накладных расходов определяется как 34% ( $H_{накл}$ ) от суммы основной и дополнительной заработной платы исполнителям проекта.

$$P_{накл} = (Z_{осн.з / п} + Z_{доп.з / п}) \cdot \frac{H_{накл}}{100} = (61000 + 8540) \cdot 0,34 = 23643,60 \text{ руб}$$

#### 4.2.8 Себестоимость разработки проекта

На основании расчетов в пунктах 4.2.1 – 4.2.7 производится калькуляция себестоимости проекта, выполненного в рамках ВКР. Итоговая сумма затрат приведена в таблице 4.

Таблица 4 – Калькуляция себестоимости проекта

Статья калькуляции	Сумма, руб.
Основная заработная плата	61000
Дополнительная заработная плата	8540
Отчисления на страховые взносы	20862
Затраты на сырье и материалы	2022,90
Услуги сторонних организаций	1371,43
Затраты на эксплуатацию оборудования	360,79
Амортизационные отчисления	1643,65
Накладные расходы	23643,60
<b>ИТОГО:</b>	<b>119 444 руб. 37 коп.</b>

Итого, себестоимость разработки проекта составила 119 444 руб. 37 коп.

#### 4.2.9 Выводы об экономической целесообразности проекта

По результатам подсчетов, проведенных в данной главе, была проанализирована экономическая составляющая проекта и рассчитана его себестоимость.

Основными затратами на выполнение работы является заработная плата исполнителей.

Конечная себестоимость предусмотренного проектом прототипа составляет 119 444 руб. 37 коп. При коммерческой реализации разработанного облачного сервиса, эта сумма будет взята за основу расчетов оплаты обслуживания для крупных организаций, некоммерческие проекты планируется обслуживать бесплатно. К рассчитанной стоимости добавятся затраты на покупку серверного оборудования и его обслуживание, включая амортизационные отчисления.

## ЗАКЛЮЧЕНИЕ

Выполнение проекта по созданию облачного сервиса «Комфортная среда» – это комплексная проработка идеи о создании полнофункционального, эстетически красивого и конфигурируемого сервиса для мониторинга окружающей среды в помещениях с возможностью управлять модулями дистанционно.

Первая, и самая важная, часть разработки была выполнена в данной работе. Бесперебойное функционирование сервиса, который предназначен для сбора большого количества информации, закладывается правильной архитектурой сетей внутри серверной части.

Решение, предложенное в этой работе, не является оптимальным, но дает возможность легкого масштабирования и быстрого переноса данных в случае технических поломок. Виртуализация, использованная для хранения данных пользователей на начальном этапе разработки, может быть применена для создания облачных вычислений принципиально другой модели – платформы как услуги (PaaS). Такое расширение позволит создать удобную, конфигурируемую под пожелания пользователя, среду для управления умными домами и их мониторинга.

При расширении клиентской базы будет необходимо добавить несколько так называемых «backend»-обработчиков, в рассматриваемом случае, php-приложений. Такая конфигурация позволит ускорить обработку данных, но при этом потребуются балансировка нагрузки – определение весов серверов, для того чтобы более мощные сервера обрабатывали большее количество информации. Также, использование нескольких серверов приведет к необходимости создания хранилищ сессий пользователей, так как в один сеанс информация может обрабатываться разными серверами, в зависимости от настроек и распределения функционала сервиса по обработчикам.

Полученный прототип сервиса – это начальный этап создания полноценной архитектуры. На практике были определены сложные моменты в настройке инфраструктуры. В ходе анализа построения сети и взаимодействия компонентов системы были выявлены основные алгоритмы создания виртуальных машин, их настройки и использования, которые могут быть применены в виде самоисполняющихся скриптов на дополнительных серверах при создании и подключении пользователей. По техническим причинам не была применена и протестирована виртуализация внутри сервера, такое развертывание дает очень большую нагрузку на систему и выполнение более трех виртуальных машин одновременно сопряжено со значительным увеличением времени на разработку.

В дальнейшем планируется развитие системы и ее доработка. Применение виртуализации на серверах, включая автоматизацию создания машин пользователей – на следующем этапе разработки. Третий этап работы – разработка конфигурируемого интерфейса портала облачного сервиса.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рекомендация МСЭ-Т Y.2069 (07/2012), Обзор интернета вещей
2. Internet of Things Роба Ван Краненбурга. Лекция в рамках FuturoDesignLab. – <http://design-union.ru/authors/theory/1490-internet-of-things-futurodesignlab>
3. Росляков, А.В. Р75 Интернет вещей: учебное пособие [текст] / А.В. Росляков, С.В. Ванышин, А.Ю. Гребешков. – Самара: ПГУТИ, 2015. – 200 с.
4. Николаев, П. Л. Применение облачных технологий в системах умного дома // Научный журнал «Молодой ученый». — 2014. — №13. — С. 37-39.
5. The NIST Definition of Cloud Computing / Peter Mell, Timothy Grance. – NIST Special Publication 800-145. – <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>, 2011. – 7 с.
6. Официальная страница регистрации Intel IOT Analytics. – <http://dashboard.us.enableiot.com>
7. «Анализ данных в мире Интернета вещей с использованием сайта Intel IOT Analytics», статья на официальном сайте компании Intel, автор – пользователь ALEXEY K. <https://software.intel.com/ru-ru/articles/the-internet-of-things-analytics-using-the-intel-iot-analytics-website-for-data-mining>
8. «Intel Edison. Работа с облаком Intel IoT Analytics: регистрация и отправка данных», Сердюков Антон, статья на сайте <https://geektimes.ru/post/255578/>
9. Официальный портал сервиса Ubidots. – <http://ubidots.com/home-ubidots.html>
10. Официальный сайт модулей Zolertia Z1. – <http://zolertia.io/z1>
11. Официальный сайт операционной системы Contiki OS. – <http://www.contiki-os.org/>
12. Официальный сайт сервиса ThingSpeak. – <https://thingspeak.com/>
13. Официальный сайт операционной системы Ubuntu. – <http://ubuntu.ru/>

14. Официальный сайт гипервизора XEN. – <http://www.xenproject.org/>
15. Официальный сайт средства виртуализации Oracle VM VirtualBox. – <https://www.virtualbox.org/>
16. Русскоязычная страница официального сайта средства виртуализации VMWare. – <http://www.vmware.com/ru>
17. Официальный сайт Apache HTTP server. – <https://httpd.apache.org/>
18. Официальный сайт MySQL. – <https://www.mysql.com/>
19. Официальный сайт PHP. – <http://php.net/>
20. Официальный сайт Bootstrap. – <http://getbootstrap.com/>
21. Официальный сайт Highchart. – <http://www.highcharts.com/>

## ПРИЛОЖЕНИЕ А

### Код PHP. Подключение к базам данных

Файл для подключения базы данных пользователей на центральном сервере:

```
<?php
/*
*Файл для подключения базы данных пользователей на центральном сервере
*/
//данные о БД: хост, имя базы, пользователь, пароль
$dblocation = "localhost";
$dbname = "db_users";
$dbuser = "mserver-admin";
$dbpasswd = "maincatserver";
//подключение к СУБД
$db = mysql_connect($dblocation, $dbuser, $dbpasswd) or die (mysql_error());
mysql_select_db($dbname, $db) or die (mysql_error());
mysql_query("SET NAMES 'utf8'");
?>
```

Файл для подключения базы данных пользователей на центральном сервере:

```
<?php
/*
*Файл для подключения базы данных информации о проектах на доп.сервере
*/
//данные о БД: хост, имя базы, пользователь, пароль
$dblocation = "192.168.184.6:3306"; //адрес виртуальной машины : порт MySQL
$dbname = "user";
$dbuser = "user";
$dbpasswd = "useruser";
//подключение к СУБД
$db = mysql_connect($dblocation, $dbuser, $dbpasswd) or die (mysql_error());
mysql_select_db($dbname, $db) or die (mysql_error());
mysql_query("SET NAMES 'utf8'");
?>
```



## ПРИЛОЖЕНИЕ Б

### Код PHP. Вывод информации из БД в виде таблицы

```
<?php
//Подключение БД
include ('../db_users/user_db.php');
//выбор данных из таблицы устройств по критерию – имя проекта
$result = mysql_query("SELECT * FROM devices_table WHERE project='$project'", $db) or
    die(mysql_error());
//подсчет результатов
$n=mysql_num_rows($result);
?>
//таблица ответов; заголовок и цикл для вывода данных в ячейки таблицы
<div class="table-responsive">
    <table class="table table-striped">
        <thead>
            <tr>
                <th>#</th>
                <th>Название устройства</th>
                <th>Идентификатор</th>
                <th>Дата создания</th>
            </tr>
        </thead>
        <tbody>
            <?php
                for($i=0;$i<$n;$i++) {
                    echo
                        "<tr><td>",mysql_result($result,$i,id),
                        "</td><td>",mysql_result($result,$i,devName),
                        "</td><td>",mysql_result($result,$i,genKey),
                        "</td><td>",mysql_result($result,$i,date),
                        "</td></tr>";
                }
            //закрытие соединения с БД
            mysql_close($db);
        ?>
    </tbody>
</table>
```

## ПРИЛОЖЕНИЕ В

### Главная страница сайта

index.php

```
<?php
//поддержка текущей сессии
session_start();
//поддержка закрытия сессии при нажатии кнопки «Выход»
include ('logout.php');
?>
<!DOCTYPE html>
<html>
    <head>
        <!--подключение необходимых библиотек-->
        <meta name="viewport" charset="UTF-8"
            content="width=device-width, initial-scale=1.0" />
        <script src="site/js/jquery-2.0.3.min.js" type="text/javascript"></script>
        <script src="site/js/bootstrap.min.js" type="text/javascript"></script>
        <link rel="stylesheet" href="site/css/bootstrap.min.css">
        <link rel="stylesheet" href="site/css/style.css">
        <link rel="stylesheet" href="site/css/index_style.css">
        <link rel="stylesheet" href="site/css/register_style.css">
        <!--заголовок страницы-->
        <title>Comfortable workplace</title>
    </head>
    <body>
        <!--подключение файла с хедером страницы-->
        <?php include('toolbars/header.php')?>

        <!--основное содержание страницы-->
        <div class="container" id="central_body">

            <!--название проекта и кнопка перехода к регистрации-->
            <div class="jumbotron" id="hello">
                <div class="container">
                    <h1>Comfortable workplace</h1>
                    <h2>Облачный сервис для комфортной жизни!</h2>
                    <p><a class="btn btn-primary btn-lg"
                        href="registration.php" role="button">
                            Начни сейчас &raquo;</a></p>
                </div>
            </div>

            <!--описание возможностей проекта: четыре контейнера с картинками и текстом
            оформление элементов содержится в файле index_style.css-->
            <div class="container-fluid">
                <div class="row" id="opport">
                    <div class="col-sm-9 col-sm-offset-2 col-md-10 col-md-offset-1 main">
                        <h1 class="text-muted">Возможности сервиса</h1>

                        <div class="row placeholders">
                            <div class="col-xs-6 col-sm-3 placeholder">
                                
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </body>
</html>
```

```

        <h4>Данные в реальном времени</h4>
    </div>
    <div class="col-xs-6 col-sm-3 placeholder">
        
        <h4>Выбери дизайн под проект</h4>
    </div>
    <div class="col-xs-6 col-sm-3 placeholder">
        
        <h4>Уведомления на смартфон</h4>
    </div>
    <div class="col-xs-6 col-sm-3 placeholder">
        
        <h4>Дистанционное управление</h4>
    </div>
</div>
</div>
</div>
</div>
</div>
</div>
<!--подключение файла с футером страницы-->
<?php include('toolbars/footer.php')?>

</body>
</html>

```

## header.php:

```

<!--хедер содержит панель навигации, состоящую из двух частей-->
<div class="navbar navbar-default navbar-fixed-top">
    <div class="container">
        <!--постоянная составляющая меню -->
        <div class="navbar-header">
            <!--кнопка для сворачивания меню-->
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <!--логотип проекта с переходом к начальной странице-->
            <a class="navbar-brand" href=" ../index.php">Comfortable workplace</a>
        </div>

        <!--сворачиваемое меню -->
        <div class="navbar-collapse collapse" id="navbar-collapse">

            <!--права панель навигации-->
            <ul class="nav navbar-nav pull-right">
                <?php
                    //пользователь не зарегистрирован или не авторизован--вывод кнопки регистрации и входа
                    if (!isset($_SESSION['user'])) {?>

```

```

        <li class="menu-item menu-item-type-post_type menu-item-object-page">
            <a href="../my_project.php?action=reg">Регистрация</a>
        </li>
        <li>
            <a href="../my_project.php">Вход</a>
        </li>
    <?php } else{ ?>
    <!--пользователь зарегистрирован или авторизован--кнопка с проектами и настройками и
        ВЫХОД-->
    <li class="dropdown">
        <a href="#" class="dropdown-toggle" data-toggle="dropdown">
            <?php echo $_SESSION['user']; ?> <b class="caret"></b></a>
        <ul class="dropdown-menu">
            <li><a href="../my_project.php">Мои проекты</a></li>
            <li><a href="../private/settings.php">Настройки</a></li>
        </ul>
    </li>
    <li>
        <a href="?action=logout">Выход</a>
    </li>
    <?php } ?>
</ul>

<!--левая панель навигации: переход к информации о проекте, примерам, инструкции и
    проектам, если пользователь авторизирован-->
<ul class="nav navbar-nav pull-left">
    <li>
        <a href="../About.php">О проекте</a>
    </li>
    <li>
        <a href="../examples.php">Примеры</a>
    </li>
    <li>
        <a href="../Instruction.php">Как пользоваться?</a>
    </li>
    <?php
    if (isset($_SESSION['user'])) { ?>
    <li>
        <a href="../my_project.php">Мои проекты</a>
    </li>
    <?php } ?>
</ul>
</div>
</div>
</div>

```