|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Пискунов Роман Андреевич  **ОТЧЁТ**  **по технологической практике**  по направлению подготовки *09.03.04 Программная инженерия*  образовательная программа «Программная инженерия»   |  |  | | --- | --- | |  | Руководитель  Доцент  Кафедры информационных технологий в бизнесе  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  О.Л. Викентьева | |

Пермь, 2020 год

# Оглавление

[Введение 4](#_Toc39881334)

[Глава 1. Описание предметной области 7](#_Toc39881335)

[1.1. Описание технологии умный дом 7](#_Toc39881336)

[1.2. Анализ деятельности пользователя 9](#_Toc39881337)

[1.3. Анализ сценариев активности умного дома 13](#_Toc39881338)

[1.4. Анализ литературы 14](#_Toc39881339)

[1.4.1. IoT устройства 14](#_Toc39881340)

[1.4.2. Хранение данных 16](#_Toc39881341)

[1.5. Анализ ресурсов публикаций из интернета 22](#_Toc39881342)

[1.6. Анализ существующих решений разрабатываемой системы 24](#_Toc39881343)

[1.7. Выявление требований к разрабатываемой системе 28](#_Toc39881344)

[Глава 2. Проектирование системы 30](#_Toc39881345)

[1.1. Проектирование архитектуры 30](#_Toc39881346)

[1.2. Проектирование баз данных 34](#_Toc39881347)

[1.3. Проектирование интерфейса системы 35](#_Toc39881348)

[Глава 3. Разработка системы 40](#_Toc39881349)

[Глава 4. Тестирование системы 41](#_Toc39881350)

[Заключение 44](#_Toc39881351)

[Библиографический список 45](#_Toc39881352)

[Приложение А. Термины и обозначения 47](#_Toc39881353)

[Приложение Б. Листинг системы 48](#_Toc39881354)

[Приложение В. Техническое задание 50](#_Toc39881355)

# Введение

Данная работа посвящена проектированию и разработке системы по сбору данных о поведении пользователей умного дома.

Под сценарием поведения пользователя в умном доме понимается специфическая последовательность действий, которая сопровождается временными ограничениями. Под умным домом, как правило, подразумевают множество умных вещей, которые объединены общей системой. Функционал, отвечающий за настройку сценариев поведения пользователей умного дома есть во всех системах умных домов. Настройка сценариев поведения пользователей представляет из себя такое множество действий, как выбор ограничений времени и выбор параметров настройки каждого датчика/устройства умного дома.

Актуальность данной работы связана с тем, что пользователи умного дома постоянно сталкиваются с проблемой сценариев поведения в умном доме. Вероятно, каждый пользователь умного дома пользуется сценариями и у каждого свои привычки поведения. Проблема в том, что привычки со временем меняются, и пользователю умного дома снова приходится менять настройки и сценарии (шаблоны) поведения. На данный момент умный дом – это результат работы программистов, что означает то, что решить данную проблему пользователей может только программист–аналитик, занимающийся анализом, например, средствами машинного обучения или с помощью таких языков программирования, как «R», которые ориентированы на работу со статистикой. Решением данной проблемы является автоматическое формирование сценариев. Решение данной проблемы сводится к 3 последовательным этапам:

1. Сбор данных о поведении пользователей умного дома;
2. Анализа собранных данных;
3. Интеграция созданного решения по анализу поведения пользователя.

Несомненно, качество анализа и реализация математической модели основывается на качестве собранных данных, поэтому процесс сбора данных занимает большую часть времени программиста-аналитика.top Именно для решения этой проблемы необходимо реализовать систему, осуществляющую сбор данных о поведении пользователей умного дома.

Объект исследования: поведение пользователей умного дома.

Предмет исследования: средства для сбора данных о поведении пользователей умного дома.

Цель данной работы — разработка системы, идея которой заключается в сборе данных о поведении пользователей умного дома, используя интернет-источники и методы машинного обучения.

Теоретическая и практическая значимость данной системы заключается в том, что её исходный код и результаты исследования можно использовать для дальнейшего анализа и визуализации данных о поведении пользователей умного дома. Данная система будет полезна, как отдельным разработчикам, так и какого-либо рода организациям.

SMART цели:

1. Изучить методические указания для оформления технологической практики для программных инженеров на 2 курсе.
2. Сформировать требования к разрабатываемой системе.
   1. Проанализировать научную литературу и интернет-источники;
   2. Узнать требования от заказчика;
   3. Провести интервью с владельцами умных домов;
   4. Применить собранные требования к разрабатываемой системе;
3. Выбрать алгоритмы и подходы к разработке.
   1. Изучить алгоритмы машинного обучения, касающиеся математического анализа, линейной алгебры, теории вероятности и математической статистики;
   2. Изучить существующие архитектуры систем (в частности, WEB-приложений) на языке C#;
   3. Изучить методы машинного обучения, применимые к структурированию и генерации данных;
   4. Провести анализ и сравнение по выявлению наиболее эффективных алгоритмов и методов;
4. Спроектировать систему.
   1. Спроектировать БД;
   2. Спроектировать архитектуру разрабатываемой системы;
   3. Спроектировать интерфейс.
5. Разработать систему.
   1. Получить необходимые знания и навыки в области машинного обучения;
   2. Изучить алгоритмы и способы генерации данных;
   3. Разработать запросы к БД;
6. Написать и провести тесты к реализуемой системе.
   1. Изучить интернет-источники по теме разработки интерфейсных тестов;
   2. Разработать интерфейсные и модульные тесты, а также написать тесты по чёрному ящику;
7. Оформить текст технологической практики.
8. Написать руководство пользователя.

Элементы новизны: использование технологий машинного обучения.

# Описание предметной области

## Описание технологии умный дом

В существующем мире, в отличии от недавнего прошлого, люди нередко задумываются о том, как, в частности, машины могут улучшить их повседневную жизнь. В отличии от машин, количество которых постоянно растёт число деятельностей человека остаётся неизменным. В жизни человека довольно много различных бытовых приборов, работу которых возможно зафиксировать и повторить с определённым алгоритмом и последовательностью действий. Концепция «Умный дом» предполагает новый подход в организации жизнеобеспечения здания, при котором за счёт комплекса программно-аппаратных средств значительно возрастает эффективность функционирования и надёжность управления всех систем эксплуатации и исполнительных устройств здания. Под «Умным домом» следует понимать систему, которая должна уметь распознавать конкретные ситуации, происходящие в здании, и в соответствующим образом на них реагировать: одна из систем может управлять поведением других по заранее выработанным алгоритмам. Умная вещь – часть умного дома, которая может быть либо датчиком (например, датчиком температуры или потребления электроэнергии), либо управляемым устройством (например, кондиционером или устройством, которое открывает двери).

На данный момент исследователи совершенствуют систему «Умный дом», ведя работу с отслеживанием работы и состояния всей «начинки» здания с целью создать систему, которая сама будет принимать решения в изменяющихся обстоятельствах.

Систему «Умный дом» можно назвать централизованной системой обслуживания жизни человека. Безусловно, большинство людей постоянно сталкивается с проблемой неразумного потребления различных ресурсов, в частности, энергоресурсов. И решением данной проблемы является централизованная система обслуживания жизни человека. Централизованная система является не только решением проблемы, но и улучшает жизнь человека, автоматизируя рутинные процессы, такие, например, как процесс открытия двери, управление телевизором с помощью голоса или же включение и выключение чайника по таймеру. Каждый умный дом имеет центр обработки и управления информации, от которого идёт адресация на всё оборудование по определённым протоколам и с помощью которого умный дом может быть запрограммирован на выполнение определённых задач, называемых сценариями. Это могут быть простые реакции на воздействия, например включение обогревателя при определённом уровне температуры в помещении, или сложное комплексное управление, например, одновременное закрытие жалюзи, плавное выключение света и включение телевизора или плеера на любимый фильм одной голосовой командой.

Все функции системы «Умный дом» можно разделить на несколько основных категорий:

1. Электричество и освещение;
2. Безопасность;
   1. Техническая;
   2. Личная;
3. Климат-контроль (температура, влажность, отопление, вентиляция);
4. Мультимедиа;
5. Связь (интернет, телефония);
6. Другие инженерные решения.

Перечисленные технологии уже существуют в каждом доме, но никак не согласованы и работают независимо друг от друга. Одной из главных функций умного дома является совместная работа оборудования во избежание нелепостей, например, возможность управления всем домом по заданному сценарию, например, когда хозяин не дома или дома, но в разное время суток и т.д.

Систему «Умный дом» можно разделить на следующие возможные подсистемы:

1. Система микроклимата (отопление, вентиляция, кондиционирование, увлажнение);
2. Система безопасности – аварийных систем (охранно-пожарная сигнализация, система доступа, контроль протечек воды, контроль утечек газа, видеонаблюдение: локальное и удалённое);
3. Система электропитания (резервные системы, аккумуляторы ИБП, контроль перегрузки электросети, система освещения: внутреннее и уличное);
4. Система распределения видео и аудиопотоков по помещениям («мультирум»);
5. Система механизация здания (открытие/закрытие ворот, шлагбаумов, обогрев ливневой канализации – стоков, подогрев ступеней лестниц и дорожек);
6. Система контроля над энергопотреблением (ограничение пиковых нагрузок и распределение нагрузок по фазам питающей сети);
7. Система связи (телефон, локальная сеть, SMS оповещения);
8. Система удалённого управления и слежения (телеметрия, IP-мониторинг, GSM-мониторинг, оповещения);
9. Система управления канализационных насосных станций и систем «автополива» зелёных территорий.

## Анализ деятельности пользователя

Каждый пользователь умного дома имеет уникальные сценарии поведения. В связи с этим важно выделить общие шаблоны поведения пользователей, а именно, проанализировать возможных прецедентов и их поведенческую активность в умном доме. Возможные прецеденты: пользователь, центр управления, система управления голосом, система освещения, датчики движения, система контроля над энергопотреблением и система безопасности. Диаграмма сценариев использования умных вещей пользователем, основанная на результатах анализа содержания умных домов и решений, принятых в результате обсуждения с заказчиком, представлена на Рис. 1.

Основываясь на описанных выше суждениях о системе «Умный дом», множество преимуществ, которыми она обладает в пользу человека возможно представить следующим образом:

1. Объединение отдельных функций в единую систему, как следствие;
2. Возможность экономии энергоресурсов;
3. Комфорт;
4. Надёжность оборудования в целом;
5. Простота мониторинга и управления;
6. Разнообразный выбор устройств для созданной системы.

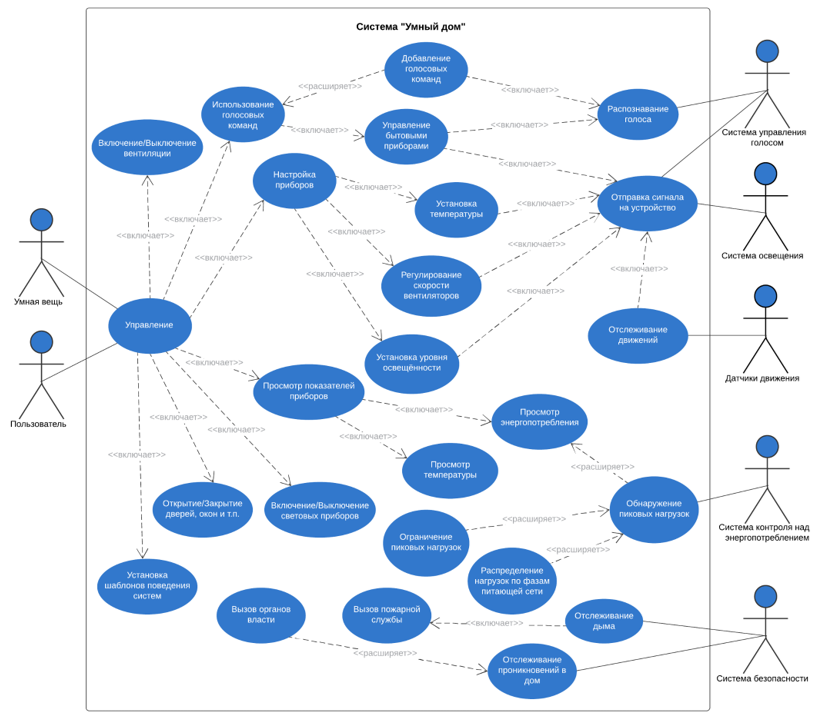


Рисунок Диаграмма сценариев использования

На сегодняшний день технологии позволяют строить домашнюю автоматизацию покомпонентно – выбирать только те функции умного дома, которые действительно нужны. Модульная структура позволяет создавать системы на высокой скорости, с гарантией 100% использования.

Возвращаясь к идее темы данной технологической практики, целесообразно рассмотреть сценарии использования возможностей системы «Умный дом» пользователями. Это позволит сформировать список признаков, а именно, список показаний приборов и датчиков, которые возможно собрать или сгенерировать.

Рассмотрим возможные сценарии использования. Возможные «точки отсчёта» активности каждого пользователя можно соотнести с частями дня, а именно, утром, днём и вечером. Во-первых, когда пользователь просыпается, его может поприветствовать голос или включиться свет. Обыкновенное утро большинства людей, в том числе пользователей системы «Умный дом» состоит и трёх этапов: проснуться, умыться, собраться и позавтракать. Рассматривая данный промежуток активности пользователей, возможны следующие приборы, показатели или данные активности которых можно собрать или сгенерировать:

1. Чайник;
2. Будильник;
3. Центр управления голосовыми командами;
4. Плита;
5. Светильники, лампа, люстра;
6. Холодильник;
7. Шторы, жалюзи;
8. Мультимедийная система (телевизор или музыкальный проигрыватель);
9. Окна;
10. Двери.

Во-вторых, касаемо данных об активности пользователей в дневное время суток, наиболее часто встречающимися видами деятельности являются просмотр фильмов или сериалов и приготовление еды, в чём задействованными могут быть уже перечисленные и приборы.

В-третьих, рассматривая возможные сценарии использования системы «Умный дом» в вечернее время суток, преобладают голосовые команды, которые в свою очередь относятся к вызову уже перечисленных приборов, а также к системам, которые связаны с полом и воздухом, а именно, подогревом и кондиционером.

Помимо уже рассмотренных приборов, пользователи умного дома часто используют сенсорные датчики, а также существуют датчики, которые отслеживают передвижения пользователей системы «Умный дом». На ряду с приборами и датчиками, которые может использовать человек стоят и те системы, которые срабатывают при определённых обстоятельствах или связаны с поддержанием чего-либо в «неувядающем» состоянии. Во-первых, примерами подобных систем являются сигнализация и охранно-пожарная система. Во-вторых, поддержание чего-либо в «неувядающем» состоянии относится к системе автополива газона, сада или оранжереи. Данные об активности подобных приборов и датчиков собирать или генерировать не целесообразно, потому что они всегда заданы по таймеры, и контроль их включения и выключения не зависит от пользователя.

Помимо данных, необходимо учитывать и различные способы управления умным домом:

1. Смартфоны со специальным мобильным приложением;
2. Сенсорные дисплеи;
3. Голосовые команды;
4. Кнопочные панели для регулирования количества чего-либо (температуры, скорости вентиляторов или степени освещённости).

Учитывая возможности каждого из управляющих устройств со стороны пользователя умного дома возможны следующие действия:

1. Нажать кнопку (например, включить/выключить свет);
2. Изменить значение–показатель какого-либо прибора (например, изменить температуру).

Учитывая возможные действия пользователя, диаграмма последовательностей, представленная на Рис. 2 изображает процесс работы умных вещей в системе «Умной дом». В данной диаграмме представлены общие последовательности действий взаимодействия с «умными» вещями.

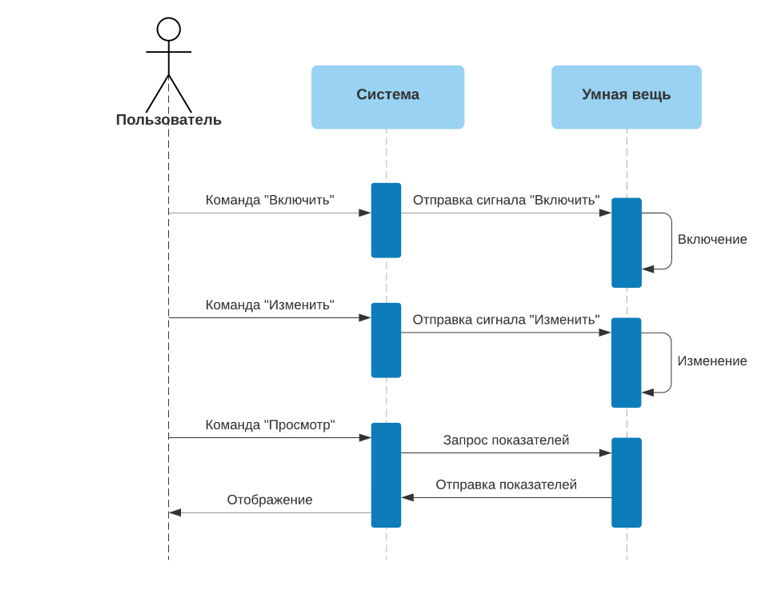


Рисунок Диаграмма последовательностей

В связи с проведённым анализом возможных сценариев пользователя вывод заключается в том, что для хранения собранных данных следует использовать базу данных, ориентированную на работу с временными рядами, причём значения могут быть либо числовыми данными в определённом промежутку, либо бинарными (например, 0 и 1).

## Анализ сценариев активности умного дома

В контексте системы «Умный дом» сценарий является определённой последовательностью действий с временными установками.

Таблица 1.1. Примеры сценария активности умного дома

|  |  |  |  |
| --- | --- | --- | --- |
| **Название деятельности** | **Начало деятельности** | **Конец деятельности** | **Значение/степень** |
| Выключен свет | 00:00:00 (после полуночи) | 07:00:00 (до 7 утра) | — |
| Включён свет | 1. 07:00:00 (с 7 утра) 2. 19:15:00 (с 7 часов 15 минут вечера) | 1. 07:15:00 (до 7 часов 15 минут утра) 2. 21:00:00 (до 9 вечера) | Средняя степень освещения |
| Включён телевизор | 1. 07:40:00 (с 7 часов 40 минут утра) 2. 19:20:00 (с 7 часов 20 минут вечера) | 1. 08:00:00 (до 8 утра) 2. 20:00:00 (до 8 вечера) | — |
| Включён чайник | 07:05:00 (с 7 часов 5 минут утра) | 07:08:00 (до 7 часов 8 минут утра) | — |
| Температура | 07:30:00 (с 7 часов 30 минут утра) | 20:00:00 (до 8 вечера) | Градусы = 18 (из 40) |
| Кондиционер | 16:00:00 (с 4 часов дня) | 16:25:00 (до 4 часов 25 минут дня) | Скорости = 13 (из 20) |

Как уже говорилось во введении, на данный момент пользователь может задать сценарий только собственноручно. Для создания сценария требуется установить следующие параметры:

1. Время начала работы;
2. Время конца работы;
3. Количественная характеристика мощности или значения, относящегося к климатическим изменениям.

Более детально сценарии (поведенческие шаблоны) рассмотрены в разделах «Анализ литературы» и «Анализ интернет-источников».

## Анализ литературы

### IoT устройства

Как уже говорилось ранее, умный дом имеет центр управления комплексом систем. Любая технология умного дома основана на том, что вещи в доме принимают и передают сигналы. Это значит, что все цифровые устройства в умном доме могут иметь данные, которыми можно будет управлять с целью использования в создании сценария или другими словами, поведенческого шаблона.

IOT устройства имеют разнообразный выбор стандартов связи – протоколов. Статья «A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration»[4] широко описывает возможные протоколы, их возможности и выделяет наиболее распространённые. Протоколы делятся на группы в зависимости от концепции:

1. Сенсорный узел – Сенсорный узел (DDS);
2. Сенсорный узел – Сервер (CoAP – REST, MQTT, XMPP, STOMP);
3. Сервер – Сервер (AMQP);
4. Клиент – Сервер (HTTP – REST).

DDS (Data Distribution Service) — протокол, реализующий шаблон публикации-подписки для отправки и приёма данных, событий и команд среди конечных узлов, причём именно узлы-издатели создают темы, содержащие информацию про температуру, местоположение, давление и после чего публикуют шаблон. В качестве «транспорта» используется UPD протокол.

CoAP (Constrained Application Protocol) — протокол, который похож на HTTP, но отличается малым количеством заголовков. Использует архитектуру «Клиент – Сервер» и подходит для передачи информации о состоянии узла на сервер, используя такие виды сообщений, как POST, GET, PUT, HEAD, DELETE и CONNECT. CoAP использует UDP протокол и архитектуру RESTful. Кроме того, данный протокол может использовать как архитектуру «Клиент – Сервер», так и «Публикация – Подписка».

AMQP (Advanced Message Queueing Protocol) — протокол открытого стандарта, который следует парадигме «Публикация – Подписка», предназначенный для обеспечения взаимодействия между широким спектром различных приложений и систем, а также для управления большим количеством обменов сообщениями, которые могут произойти в системе за короткий промежуток времени. Архитектура данного протокола такая же, как и у MQTT, то есть данный протокол передаёт данные через посредника (брокера).

XMPP (Extensible Messaging and Presence Protocol) — протокол, использующий такой формат передачи данных, как XML и работает поверх архитектур «Клиент – Сервер» и «Публикация – Подписка».

MQTT (Message Queue Telemetry Transport) — протокол, осуществляющий сбор данных от множества устройств и их передачу на сервер. Данный протокол основывается на модули «Публикация – Подписка» с использованием промежуточного посредника – брокера (для создания очередей и приоритезацией сообщений) и принимает данные по определённому «топику»(именованному каналу или каналам связи). В качестве «транспорта» используются TCP протокол.

HTTP (Representational State Transfer Hyper Text Transport Protocol) — протокол, который поддерживает такие способы передачи данных, как POST, GET, PUT, DELETE, а также операции CRUD. Данный протокол передаёт и принимает данные в формате JSON, а также взаимодействует по TCP протоколу.

На ряду с рассмотренными протоколами существуют такие стандарты связи, как Wi-Fi, Bluetooth и ZigBee, среди которых Wi-Fi ценится быстротой передачи данных и поддержкой IP, но уступает по такому параметру, как энергопотребление, Bluetooth ценится низким энергопотреблением, но скорость передачи много меньше, чем у Wi-Fi. Что же касается ZigBee, то данный стандарт связи имеет очень малую скорость связи, но экономный по энергопотреблению.

Вывод заключается в том, что наилучшим выбором для IoT приложений являются протоколы MQTT, HTTP – CoAP (REST) из-за достаточно малого объёма трафика, а также низких требований к батарее и памяти. Согласно статье «Протокол MQTT: концептуальное погружение»[5] протокол MQTT работает по принципу «Long Pooling», а также протокол MQTT занял первое место в сравнении со всеми перечисленными выше протоколами по таким параметрам, как время отклика и объём трафика.

Исходя из вышеописанного анализа, в данной технологической целесообразно использовать в разработке протокол MQTT.

### Хранение данных

Для того, чтобы определить формат данных, используемых в качестве связующего звена между устройствами (в том числе датчиками) умного дома и управляющими устройствами необходимо проанализировать статьи и литературу, направленную на решение проблем, связанных с умным домом.

Рассмотрим статью «A Realistic Dataset for the Smart Home Device. Scheduling Problem for DCOPs» [1], в которой говорится об эталонных тестах в оптимизации распределённых ограничений умного дома, используемых в оценке производительности, а также описана проблема координации расписания активности устройств умного дома (проблема согласования интеллектуальных устройств) и подробно излагаются физические модели, применяемые для моделирования интеллектуальных датчиков, интеллектуальных исполнительных механизмов и домашних условий.

Проблема координации расписания активности устройств умного дома является задачей об оптимизации распределённых ограничений из-за наличия как сложных индивидуальных целей устройств, описывающих потребление энергии, так и цель совмещённых устройств, которая заключается в том, чтобы сократить энергетические пики. Данная статья полезна тем, что в ней приводят подробные физические модели, принятых для моделирования датчиков, исполнительных механизмах, а также домашней обстановки, и описывают, как действия исполнительных механизмов влияют на окружающую среду в доме (например, температуру, чистоту и влажность).

Задача оптимизаций распределённых ограничений описана набором из следующих частей:

1. Набор переменных;
2. Набор конечных областей;
3. Множество функций полезности (множество ограничений);
4. Множество областей действия;
5. Набор агентов;
6. Функция, сопоставляющая каждую переменную с агентом.

На ряду с вышеописанной задачей в данной статье описывается и набор данных для задачи о координации расписания активности устройств умного дома. Эта задача определяется следующими данными:

1. Набор умных построек, находящихся в близи друг от друга, способных взаимодействовать между собой;
2. Набор интеллектуальных устройств;
3. Набор местоположений (например, гостиная или кухня);
4. Набор свойств состояния умных домов (например, чистота или температура);
5. Набор свойств состояние устройств (например, заряд батареи для пылесоса);
6. Ограничения планирования проблемы;

Для каждого умного дома набор интеллектуальных устройств разделён на набор исполнительных механизмов и набор датчиков. Каждое устройство описывается в данной статье следующими данными:

1. Соответствующее местоположение устройства;
2. Набор действий, которые устройство может выполнить;
3. Действия устройства в соответствующие свойства состояния (умного дома);
4. Действия устройства в соответствующие свойства состояния (устройства);

Например, рассматривая вакуумный робот (пылесос). У него есть определённое местоположение (например, гостиная или кухня) и такие возможные значения, как «действие», «зарядка», «бездействие». Следовательно, возможными сопоставлениями будут:

1. Действие 🡪 {Чистота};
2. Действие 🡪 {Заряд батарее};
3. Зарядка 🡪 {Нулевое состояние};
4. Зарядка 🡪 {Заряд батареи}.

Цель координации расписания активности устройств умного дома — найти расписание для каждого из устройств, которые достигают определённых пользователем целей (например, достижение определённой температуры в пределах временного окна или же для того, чтобы дом находился на определённом уровне чистоты к определённому сроку). Данные цели можно обозначить, как «правила планирования». В основном, выделяют 2 правила планирования:

1. Активные правила планирования;
2. Пассивные правила планирования.

Активные правила планирования представляют из себя определённые пользователем цели в желаемом состоянии (например, уборка жилой комнаты к 18:00 часам). Пассивные же правила планирования определяют неявные ограничения, т.е. правила, которые соблюдаются каким-либо устройством всегда (например, заряд аккумулятора у робота-пылесоса всегда находится между 0% и 100%).

Сценарий поведения пользователей можно считать успешным, если он удовлетворяет всем пассивным и активным правилам планирования.

Основными параметрами для генерации набора данных являются следующие параметры:

1. Структурные параметры дома;
2. Подробный список интеллектуальных устройств;
3. Влияние каждого устройства на домашнюю среду;

В данной статье рассматривают 3 размера домов (меленький, средний и большой). Размеры домов выражены в метрах (Длина **×** Ширина), а высота составляет 2,4 метра.

Таблица 1.2. Структурные параметры умных домов

|  |  |  |  |
| --- | --- | --- | --- |
| **Структурные параметры** | **Типы домов** | | |
| **Маленький** | **Средний** | **Большой** |
| Размеры домов (м) | 6 × 8 | 8 × 12 | 12 × 15 |
| Площадь стен (м2) | 67,2 | 96 | 129,6 |
| Область окна (м2) | 7,2 | 10 | 16 |
| Коэффициент теплопередачи стен (Uwals,  Вт/ (м2\* ◦C)) | 3,9 | 3,9 | 3,9 |
| Коэффициент теплопередачи окон (Uwals,  Вт/ (м2\* ◦C)) | 2,8 | 2,8 | 2,8 |
| Потребление тепла человеком | 400 | 400 | 400 |

Коэффициенты теплопередачи описывают насколько хорошо строительный элемент (в данном случае бетон с толщиной 150 мм) проводит тепло.

Список датчиков, которые используются в данной статье представлены в таблице 1.3.

Таблица 1.3. Список рассмотренных датчиков

|  |  |
| --- | --- |
| **Действие / Состояние** | **Расположение** |
| Температура воздуха | Комната в доме |
| Чистота пола (пыль) | Комната в доме |
| Температура | Умная вещь |
| Заряд батареи | Умная вещь |
| Выпекание | Умная вещь |
| Стирка белья | Умная вещь |
| Сушка белья | Умная вещь |
| Чистота тарелок | Умная вещь |
| Влажность воздуха | Комната в доме |
| Освещённость | Комната в доме |
| Пребывание («ничего не деленье») | Комната в доме |
| Движение | Комната в доме |
| Детектор дыма | Комната в доме |

Список исполнительных механизмов, их возможные состояния и потребление энергии представлены в таблице 1.4.

Таблица 1.4. Список возможных умных вещей

|  |  |  |
| --- | --- | --- |
| **Умная вещь** | **Действие / Состояние** | **Потребление (кВт/ч)** |
| Подогреватель | Выключен | 0 |
| Вентилятор | 0,5 |
| Подача тепла | 0,5 |
| Охладитель | Выключен | 0 |
| Вентилятор | 0,5 |
| Подача холода | 2,3 |
| Водонагреватель | Включён | 0 |
| Выключен | 4,6 |
| Вакуумный пылесос | Выключен | 0 |
| Работа (чистка) | 0 |
| Зарядка | 0,26 |
| Электромобиль | Выключен | 0 |
| Зарядка | 17,28 |
| Стиральная машина | Выключена | 0 |
| Стирка (Regular) | 0,43 |
| Вращение (Regular) | 0,45 |
| Полоскание (Regular) | 0,15 |
| Стирка (Perm-Press) | 0,39 |
| Вращение (Perm-Press) | 0,43 |
| Полоскание (Perm-Press) | 0,23 |
| Стирка (Delicates) | 0,39 |
| Вращение (Delicates) | 0,33 |
| Полоскание (Delicates) | 0,01 |
| Сушилка для белья | Выключена | 0 |
| Включена (Regular) | 5,76 |
| Включена (Perm-Press) | 5,25 |
| Включена (Timed) | 5,95 |
| Печь | Выключена | 0 |
| Выпекание | 8,46 |
| Обжарка | 10,56 |
| Посудомоечная машина | Выключена | 0 |
| Стирка | 1,12 |
| Полоскание | 1,14 |
| Сушка | 0,63 |

Кроме того, в данной статье рассматриваются физические модели, т.е. формулы, по которым рассчитываются потери и потребление ресурсов, в частности энергоресурсов.

Модель контроля температуры воздуха рассчитывается по стандартному принципу отопления и вентиляции. Таким образом тепловая потери выражаются формулой (1).

|  |  |
| --- | --- |
|  | (1) |

где Uwals – коэффициент теплопередачи стен (Вт / (м2\* ◦C));

Uroof – коэффициент теплопередачи крыши (Вт / (м2\* ◦C));

Uwindows – коэффициент теплопередачи окон (Вт / (м2\* ◦C));

Awals – область (площадь пола) стен (м2);

Aroof – область (площадь пола) крыши (м2);

Awindows – область (площадь пола) окон (м2);

G (вентиляционная проводимость) = Объёмный расход (100) \* плотность воздуха (0,75) \* удельную теплоёмкость воздуха (0,24) = 18.

Тепловая нагрузка же определяется формулой (2).

|  |  |
| --- | --- |
|  | (2) |

где hloss – потери тепла;

TA – текущая температура;

TZ – целевая температура.

Если же при расчёте тепловой нагрузки учитывать температуру наружного воздуха, то используют формулу (3).

|  |  |
| --- | --- |
|  | (3) |

где TN – температура наружного воздуха.

Кроме всего описанного, данная статься демонстрирует набор данных, в котором собраны данные о домах следующими площадями:

1. Asmall  = 48 (м2);
2. Amedium  = 96 (м2);
3. Alarge = 180 (м2).

Помимо статьи про генерацию реалистичного набора данных, описанного выше, одной из многих статей является работа «Anomaly Detection Models for Smart Home Security»[2], которая описывает применение технологий Big Data и машинного обучения для выявлений аномальной активности умного дома. В отличии от ранее рассмотренной статьи, в качестве набора данных в данной статье используются данные в бинарном формате, а именно состояния «On» и «Off». Данная специфика используется из-за того, что в данной статье описывается использование скрытой модели Маркова для анализа и выявления аномалий. Бинарное специфика данных используются так, что каждая строка в наборе данных представляет собой определённое состояние без временны меток. Набор данных в данном случае содержит множество столбцов, каждый из которых представляет отдельно взятое устройство умного дома, например, умную вещь или датчик.

Вывод по рассмотренным статьям заключается в том, что в генерации данных об активности пользователей умного дома следует учитывать то, что показатели умных вещей в умных домах варьируются в зависимости от размеров дома, а также количества проживающих в нём людей и их возраста. Что же касается показателей умных вещей и соответствующих им датчиков, их активность можно представить двумя способами, первый из которых – это представление данных в виде вещественных чисел с временными метками, а второй – использование бинарного формата чисел ряда чисел, как «0, 1», где «0» – умная вещь не используется, а «1» – умная вещь используется, причём данные должны быть без временных меток. Однако, поскольку задача данной практики собрать данные для их дальнейшего анализа, наиболее целесообразно разрабатывать систему и проектировать базу данных для вещественных показателей с временными метками. Следовательно, нужно использовать базы данных, нацеленные на хранение временных рядов и средства работы именно с временными рядами.

## Анализ ресурсов публикаций из интернета

Публикация «Health Smart Home (HIS) datasets»[5] содержит раздел « Activities of Daily Living dataset», в котором описываются несколько датчиков, а также прикреплена ссылка на набор данных. В данной публикации рассматриваются 2 типа представления вещественных данных. Например, данные с датчика температуры могут передаваться, как в виде градусов по Цельсию, так и в качестве количества процентов при условии, если будут установлены границы минимума и максимума (например, минимум = –15°C, а максимум = +40°C). Кроме того, данная публикация содержит ссылки на наборы данных о показателях датчиков влажности, температуры, инфракрасных сенсоров, а также датчики, отслеживающие взаимодействие с комодом, холодильником и шкафом.

Помимо уже проанализированной публикации, были рассмотрены статьи, связанные с таким ресурсом, как «Personalized Retrofit Decision Support Tools for UK Homes Using Smart Home Technology (REFIT)»[6]. Данный ресурс содержит наборы данных и ряд пояснений по данным, которые были собраны с целью модернизации энергопотребления в умных домах Великобритании.

Наиболее важной из всех рассмотренных публикация была публикация «Intel Lab Data»[7]. Данная публикация содержит следующие наборы данных:

1. Координаты датчиков;
2. Показатели датчиков;
3. Данные о подключениях, усреднённые за всё время.

Набор данных о координатах датчиков содержит следующие признаки:

1. ID – целое число
2. Координата X – вещественное число
3. Координата Y – вещественное число

Набор данных о показателях датчиков содержит следующие признаки:

1. Дата – «yyyy-mm-dd»
2. Время – «hh:mm:ss.xxx»
3. Эпоха – целое число
4. Идентификатор режима – целое число (от 1 до 54)
5. Температура – вещественное число (°C)
6. Влажность – вещественное число (относительная величина в диапазоне от 0 до 100%)
7. Освещённость – вещественное число (в Люксах, 1 Люкс = 1 Люмен/м2)  
   1 Люкс – лунный свет  
   400 Люкс – яркий офис  
   100000 Люкс – полный солнечный свет
8. Напряжение – вещественное число (в Вольтах в диапазоне 2 - 3)

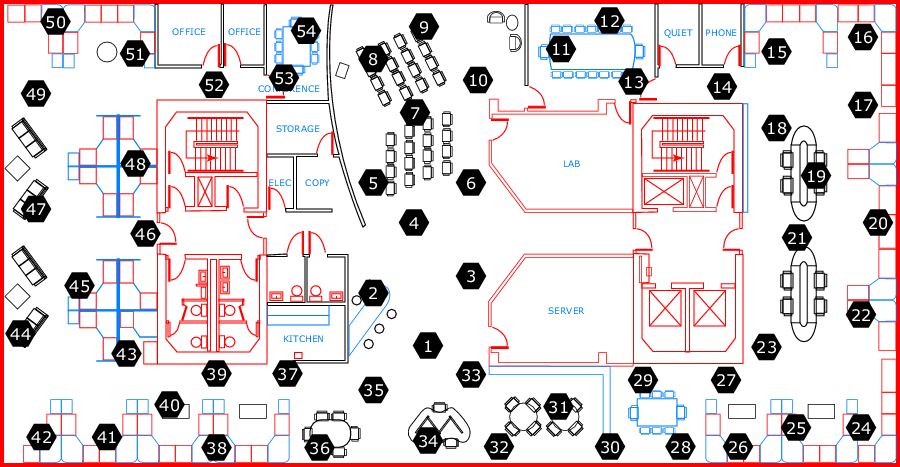


Рисунок Диаграмма дома с датчиками и расположение датчиков

В результате анализа набора данных о показателях датчиков, было выявлено то, что иногда датчики температуры и влажности иногда показывают нереальные значения, в следствии чего показатели датчиков температуры и влажности лучше представить в пределах от 1 до 99 %.

Помимо публикаций, были прочитаны и используются в данной работе публикации автора Rob J Hyndman на тему прогнозирования временных рядов

## Анализ существующих решений разрабатываемой системы

На данный момент не существует систем, нацеленных конкретно на сбор показателей датчиков умных домов. Существуют лишь системы, которые являются разработанными комплексами для использования в качестве системы управления в создаваемых умных домах.

В качестве первого существующего решения было проанализированы программные решения, связанные с темой «Smart Home Device Scheduling»[8], которая была рассмотрена на этапе анализа хранения данных. Рассматриваемые программные решения состояли состоят из 2 последовательных этапов:

1. Генерация данных для решения проблемы планирования — подготовки сценариев.
   1. Создание конфигурационного файла устройств/датчиков;
   2. Создание правил – сценариев;
2. Решение/Оптимизация входных данных.
   1. Создание ограничений для устройств/датчиков;
   2. Собрать список состояний устройств/датчиков;
   3. Создание ограничений для правил – сценариев;
   4. Отправка готового списка состояний с ограниченными правилами;

В одном из решений в репозитории под названием «SHS» используется Байесовская оптимизация.

Кроме того, каждое из решений содержит два JSON файла. Первый файл содержит в себе рассматриваемые устройства и датчики умного дома, которые использовались для анализа в статье «A Realistic Dataset for the Smart Home Device Scheduling Problem for DCOPs»[1], а также их свойства, местоположение и возможные действия/состояния. Второй файл содержит правила – сценарии для конкретных устройств и датчиков (см. подраздел «Хранение данных»).

Вторым на очереди программным решением является сервис «Temboo». Данный сервис позволяет собирать данные следующих типов устройств/датчиков:

1. PLCs & Modbus Devices (PLC через Modbus, Schneider Electric, Yaskawa);
2. Industrial & Enterprise Sensors (NCD, GE, Digi, Iris);
3. Programmable Microcontrollers (Espressif ESP32);
4. Bluetooth 5 Multi-Sensors (Omron, Nordic);
5. Other (Kosmos Starter Kit).

Рассматривая конкретно программирование микроконтроллеров, разработчику необходимо выбрать тип устройства/датчика, данные которого будут собираться. Возможные на выбор типа представлены на Рис.4.

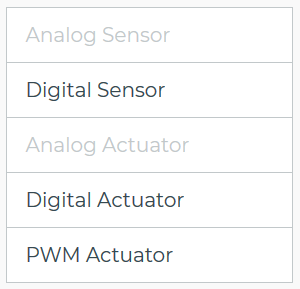


Рисунок Типы устройств/датчиков

Тип «Digital Sensor» включает в себя список цифровых датчиков, которые представлены на Рис.5. При выборе данного типа следующим шагом является выбор формата данные, но в данном решении можно выбрать только «общий»(универсальный) формат.

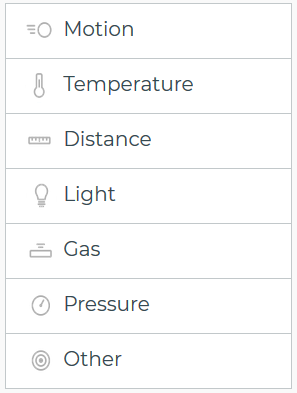


Рисунок Список цифровых датчиков

Тип «Digital Actuator» включает в себя список цифровых устройств (исполнительных механизмов), которые представлены на Рис.6.

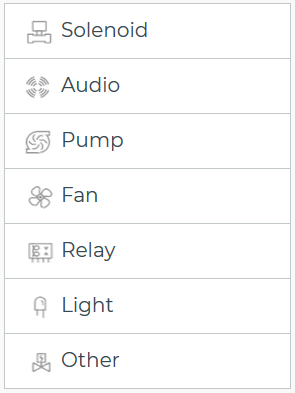


Рисунок Список цифровых исполнительных устройств

При выборе типа «Digital Actuator» следующим шагом будет выбор модели исполнительного устройства, например, при выборе «Solenoid» моделью исполнительного устройства будет «Crouzet USA 81519332», при «Fan» — «Delta Electronics ASB0412LA-A», а при «Light» — «LED».

Тип «PWD Actuator» включает в себя список управляющих устройств, представленных на Рис.7.

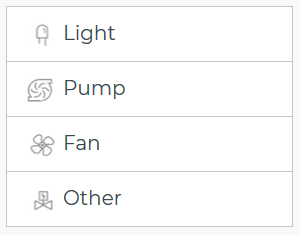


Рисунок Список цифровых исполнительных устройств

После того, как разработчик создал устройство(-а)/датчик(-и), рассматриваемое решение требует выбрать частоту, с которой собирать данные, а также соответствующие параметры, например, при выборе датчика, отслеживающего температуру, нужно будет выбрать формат данных: либо по Цельсию, либо по Фаренгейту.

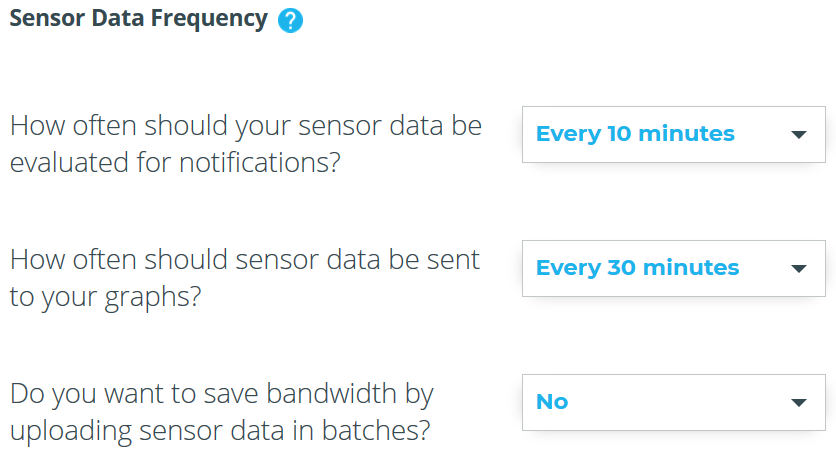


Рисунок Список цифровых исполнительных устройств

Заключительным этапом является то, что разработчик переходит в раздел, в котором генерирует код для интеграции в своё приложение. Для начала взаимодействия с сервисом и сбора данных, сгенерированный код нужно запустить в приложении.

Преимуществом данной системы является то, что генерируемый код может быть на нескольких языках программирования. Однако, данное решение имеет такие следующие недостатки:

1. Минимальная частота отправки данных составляет 10 минут;
2. Отсутствие возможности интеграции результатов;
3. Сложная настройка;
4. Отсутствие экспорта собранных данных.

Помимо уже выше рассмотренных существует решение «Home Assistant At SnM Nest»[14], в котором настройки и описания каждого из устройств/датчиков представлены в качестве YAML файлов. Данное решение основывается на протоколе MQTT.

## Выявление требований к разрабатываемой системе

Сбор данных о поведении пользователей умного дома имеет несколько трактовок.

Первая трактовка заключается в следующем: подразумевается, что есть умный дом, программное обеспечение которого реализовано какой-то компанией и есть возможность взаимодействовать с устройством и программным обеспечением данного умного дома. Для того, чтобы собрать данные с различных датчиков и умных устройств, можно встроить программный код, который будет размещён в части программного обеспечения, в которой происходит логирования данных. Данный программный код будет отправлять данные на определённый сервер, на котором данные будут структурироваться и храниться в базе данных. Альтернативный способ данной трактовки заключается в использовании API к программному обеспечению данного умного дома, с помощью которого можно было бы обеспечить мониторинг активности пользователя с разных точек зрения.

Вторая трактовка подразумевает поиск данных о поведении пользователей умных домов представленных в качестве набора данных. Данный вариант подразумевает следующий сценарий действий:

1. Поиск наборов данных;
   1. Поиск отдельных наборов данных (по каждому датчику или умному устройству отдельно);
   2. Поиск цельных наборов данных;
   3. Поиск «кусочных» наборов данных;
      1. Структурирование и «чистка» (например, уборка таких значений, как «NaN» или сведение классификации типа «Male/Female» к набору «0/1») данных;
2. Содержательный анализ найденных наборов данных;
3. Выделение общих признаков;
4. Генерация новых наборов данных на основе найденных.

Однако, существует вариант, подразумевающий эмуляцию работы умного дома по различным протоколам, которые свойственны для IoT, а поскольку никакая компания просто так не позволит подключиться к какому-нибудь умному дому для мониторинга, задача данной практики сводится к реализации системы по эмуляции работы протоколов, по которым работают IoT устройства.

В результате проведённого анализа литературы, интернет-источников и существующих решений разрабатываемой системы были выявлены следующие требования:

1. Система должна эмулировать один из наиболее популярных протоколов передачи данных между IoT устройствами;
2. Система должна иметь возможность учитывать размеры дома, а также количество и возраст проживающих;
3. Система должна эмулировать разные типы передачи данных, то есть как бинарные данные (например, действия «открыто», «закрыто»), так и вещественные данные (например, показатели температуры, влажности или освещённости);
4. Система должна быть адаптивна к любой системе умного дома;
5. Функциональные возможности системы должны включать в себя возможность изменить параметры сбора данных (например, возможность изменить частоту запросов или адрес сервера, на который будут приходить данные);
6. Функциональные возможности системы должны включать в себя возможность добавить новое устройство/датчик, создать новый дом или изменить параметры существующего умного дома;
7. Система должна иметь возможность экспорта собранных данные как минимум в CSV формат.

# Проектирование системы

Проектирование системы зависит от того, для кого она разрабатывается. Поскольку, данная система разрабатывается для программистов-аналитиков, стоит выбрать архитектуру, которая позволит работать с MQTT протоколом очень быстро, так как важна точность результатов. Результат проектирования представлен в качестве технического задания в приложении В.

## Проектирование архитектуры

Проектирование разрабатываемой системы основывается на том, как работает протокол MQTT. Обратимся очередной раз к статье «Протокол MQTT: концептуальное погружение»[5]. Данный протокол работает по принципу «Публикация – Подписка». Данную связь и общий принцип работы данного протокола объясняет иллюстрация, представленная на Рис.9.

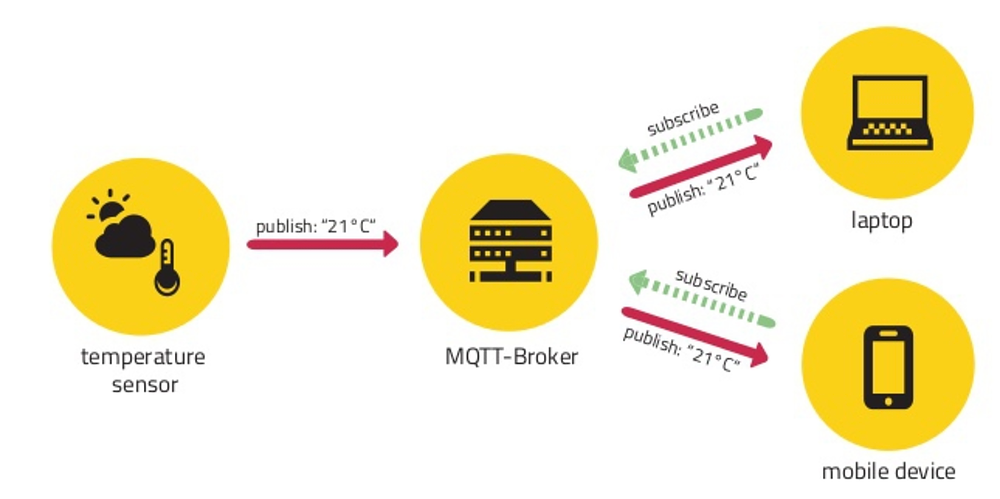


Рисунок Схема работы MQTT протокола

Частниками в передачи данных по MQTT протоколы являются сами датчики/устройства, MQTT Broker (посредник) и клиенты (смартфон, планшет, компьютер, ноутбук). Для работоспособности данного протокола важны следующие данные:

1. Topic (например, «home/2/room/kitchen/devices/temp/T2»);
2. Payload (например, JSON объект, хранящий информацию о датчике или устройства и его показатели, например, показатели температуры на кухне);
3. QoS (уровень качества взаимодействия брокера с различными «клиентами» – 0, 1, 2);
4. Retain (показатель, означающий должно ли сохраняться последнее опубликованное сообщение – True или False).

Каждый датчик или устройства должен иметь название (Topic), по которому можно было бы отследить то, что данные пришли с определённого датчика в конкретной комнате. Для того, чтобы отправить данные на MQTT Broker, датчику или устройству нужно следующее:

1. Подключиться к MQTT Broker с указанием URL адреса MQTT Broker, порта, никнейма и пароля;
2. Подключиться к Topic, который соответствует данному датчику или устройству;
3. Составить отправляемое содержимое по протоколы (Payload);
4. Отправить (опубликовать) содержимое по протоколу.

Поскольку протокол MQTT работает по принципу «Long Pooling», MQTT Broker получит данные автоматически. MQTT Broker выступает своего рода посредником, который хранит в себе множестве ветвей, каждая из которых представляет Topic. Обратимся к части программной обработки клиента(-ов) – смартфон, планшет, компьютер или ноутбук. Чтобы клиенту получить данные, он должен подписаться (подключиться для «прослушки») на Topic, который был указан при отправке сообщения с датчика или устройства. После того, как клиент подписался, он сможет автоматически получать сообщения с датчика или устройства, т.е. датчик отправил данные, MQTT Broker эти данные направил на Topic, а поскольку клиент настроен (подписан на определённый Topic) постоянно «прослушивать» все сообщения, которые приходят по указанному Topic, клиент получает отправленное сообщение с датчика или устройства.

Поскольку, нет доступа к реальному умному дому, потребуется создать сервер, отвечающий за эмуляцию датчиков и устройств, а также WEB приложение. Наиболее целесообразно создать сервер с удалённой базой данных. Что касается WEB-приложения, исходя из требований к данной практике, приложение будет написано с использованием технологии ASP.NET Framework на C#.

В качестве архитектуры сервера был выбран тип REST API.

Архитектура сервера должна основываться на том, что каждый датчик и устройство будет представлять отдельный поток данных, которые отправляются на MQTT Broker с определённым интервал времени (например, 1 минутой). Кроме того, все потоки должны работать в фоновом режиме, чтобы, когда сайт, на котором находится сервер закроют, потоки не закрылись и скрипт не закончил свою работу. Реализацию сервера лучше разработать на Python из-за того, что Python содержит наибольшее количество библиотек для обработки данных и работой с машинным обучением для анализа наборов данных с временными рядами датчиков и устройств, которые доступны в интернете.

Разрабатываемая система должна иметь 2 базы данных:

1. База данных для хранения собираемых данных;
2. База данных для хранения данных, способствующих управлению датчиков и устройств, а также настройке сервера, который их эмулирует.

Сервер, эмулирующий работу датчиков и устройств должен иметь реляционную базы данных, а поскольку сервер должен быть написан на Python, то учитывая то, что наиболее важным фактором в разработке данной системы является скорость передачи данных, а в данном случае скорость обработки запросов, то лучшим выбором будет использовать базу данных под названием MySQL, причём Python содержит дополнительную библиотеку PyMySQL.

Что касается хранения собранных данных, то лучше использовать базу данных под названием InfluxDB, которая предназначена конкретно для хранения временных рядов. Данный выбор обусловлен тем, что данная база данных способна на интеграцию с сервисом Grafana, с помощью которого можно визуализировать собираемые данные.

Поскольку разрабатываемая система требует точности передачи данных, то при передаче данных, параметр «QoS» будет иметь значение «2», что означает то, что данные, отправляемые с датчика или устройства, будут отправлены ровно один раз без потери и дублирования.

Что касается генерации временных рядов уже с учётом параметров комнат, а также количества, возраста и пола проживающих в умном доме людей — возможны 2 варианта: с использованием стандартных средств разработки, которые включают в себя использование результатов математического анализа и с использованием машинного обучения (то есть моделей машинного обучения), которые содержит в себе не одно решение. Возможные решения генерации данных можно разделить на следующие группы:

1. Создание общей модели машинного обучения, которая будет принимать на вход несколько параметров, которые будут характеризовать метками, которые в свою очередь будут символизировать какие-либо существенные изменения в данных на протяжении всего временного ряда;
2. Создание моделей машинного обучения для каждого датчика/устройства;
3. Использование готовых моделей данных для работы с временными рядами.
4. Создание подсистемы, которая будет генерировать временной ряд в зависимости от среднего значения, дисперсии, а также коэффициента вариации, выбросов и размаха временного ряда.

Для того, чтобы при прогнозировании учитывать дополнительные переменные, например, определённое местоположение датчика (комната), в построении лучше всего использовать модель ARIMAX, которая поддерживает использование дополнительных переменных в создании модели по прогнозированию временных рядов.

В решении проблема, которая поставлена в данной технологической практике, функциональным звеном между запуском датчиков и записыванием (сбором) данных в базы данных являются два случая – генерация данных и эмуляция загружаемых временных рядов. Генерация данных включает в себя следующую последовательность действий:

1. Загрузка временного ряда, свойственного активности какого-либо датчика/устройства;
2. Выбор диапазон, на который прогнозировать временной ряд;
3. Применение модели (-ей) для прогнозирования;
4. Загрузка данных в эмуляцию датчиков/устройств.

Эмуляция же предлагает хранение и последовательную загрузку данных для отправки, то есть на FTP сервере будут храниться файлы, а поскольку, сервер работает по принципу CronTab, то файлы будут периодически перезаписываться. Для хранения временных рядов лучше всего использовать библиотеку Feather, поскольку, судя по анализу библиотек для хранения данных, который демонстрирует публикация «The Best Format to Save Pandas Data»[16], данная библиотека наиболее оптимальна для записи и чтения, как маленьких по размеру, там и больших файлов.

В рамках данной технологической практики часть архитектуры, отвечающая за генерацию временных рядов, будет находиться на стеке между отправкой данных в базу данных InfluxDB и получением каких-либо запросов от пользователей из WEB-приложения (например, создание нового датчика). Поэтому, пользователь сможет выбрать для себя любой способ генерации.

## Проектирование баз данных

Схема база данных, реализуемая для сервера, эмулирующего работу датчиков и устройств, должна содержать следующие таблицы:

1. Таблица домов;
2. Таблица параметров и количества проживающих в каждом доме;
3. Таблица комнат;
4. Таблица типов комнат;
5. Таблица датчиков/устройств;
6. Таблица типов датчиков/устройств;
7. Таблица действий, свойственных определённым датчикам/устройствам;
8. Таблица с проанализированными данными по устройствам (для генерации новых временных рядов – активности новых умных домов);
9. Таблица для логирования активности MQTT протокола (подключения, остановки).

Значения в таблице с типами комнат должны быть фиксированными, потому что уже с давних времён известны всевозможные комнаты, причём некоторые названия комнат обобщают большое количество других комнат.

Диаграмма отношения сущностей базы данных сервера, который эмулирует датчики представлена на Рис. 10.

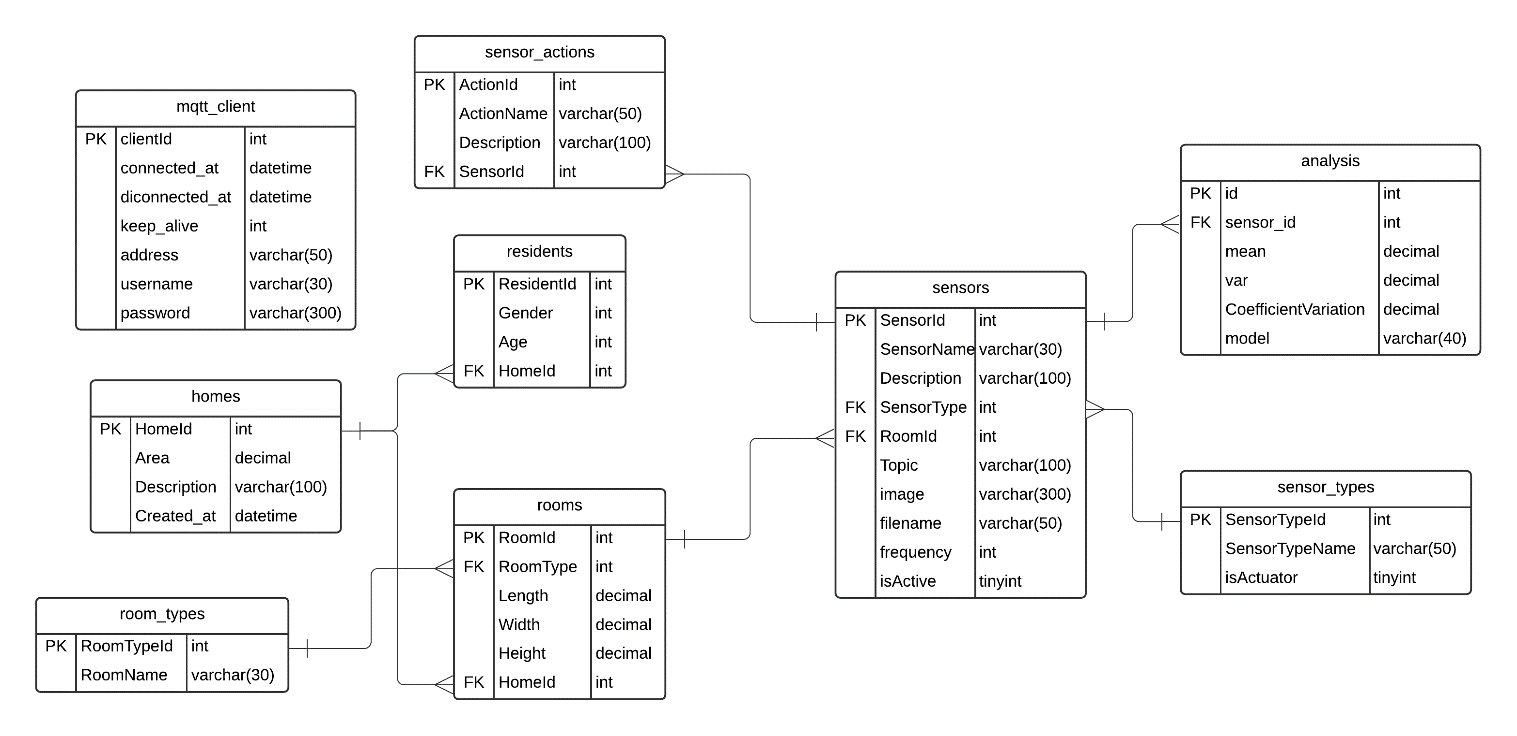


Рисунок ERD диаграмма базы данных REST сервера

Что касается базы данных InfluxDB – после развёртывания и настройки InfluxDB локально, нужно подключиться к InfluxDB Cloud – платформе, которая предоставляет интерфейс для работы с InfluxDB. При создании нового датчика/устройства в разрабатываемой системе в базе данных InfluxDB будет создаваться новая таблица. Если пользователь системы решит удалить датчик/устройство и подтвердит удаление данных в базе данных InfluxDB, то InfluxDB удалит таблицу, связанную с указанным датчиком/устроством.

## Проектирование интерфейса системы

Исходя из требований, на главной странице должна быть информация о датчиках, кнопки редактирования, создания и настроек датчиков/устройств.

Первой страницей, которое увидит пользователь, который впервые запустил систему, будет страница настройки данных, связанных с MQTT подключением, в котором нужно будет ввести адрес MQTT брокера, никнейм, пароль и порт, по которому подключаться системе. Интерфейс страницы первоначальной настройки системы приведён на Рис. 11.

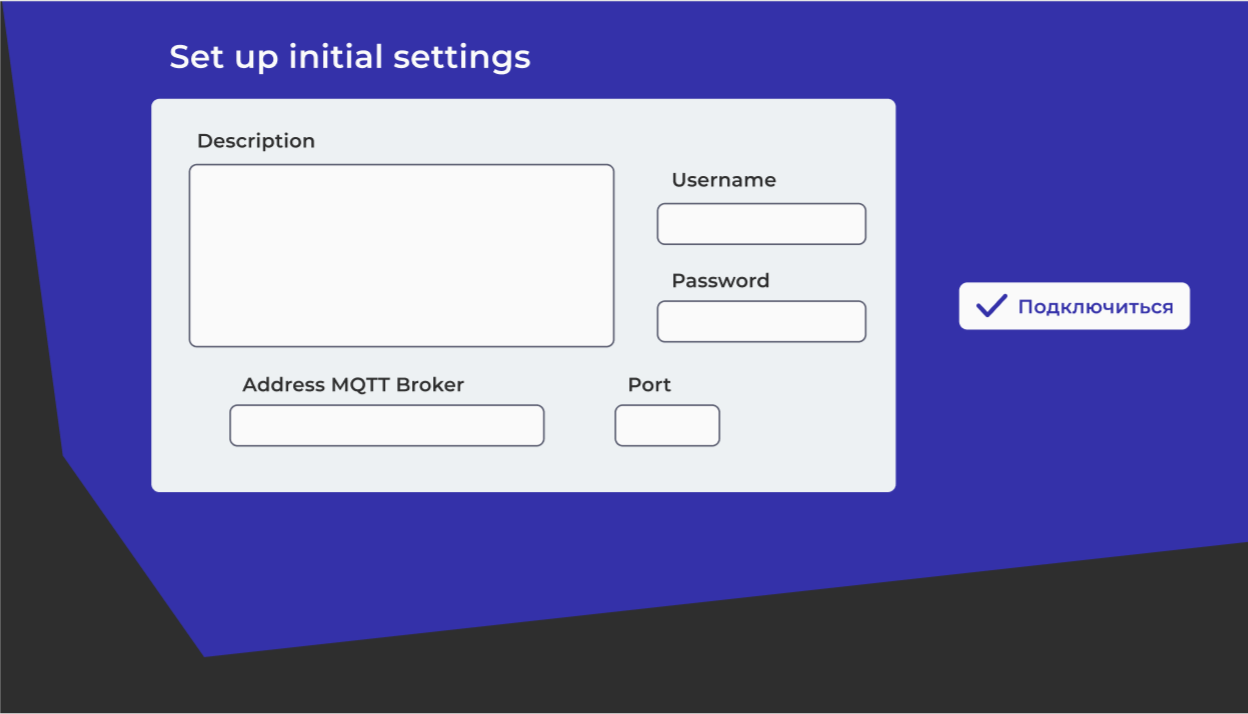


Рисунок Интерфейс страницы первоначальной настройки системы

После настройки MQTT подключения, на сервер отправится запрос на инициализацию датчиков, после чего пользователь видит страницу, где выведены все имеющиеся датчики, а также необходимые элементы управления системой, указанные в требованиях. Главная страница системы представлена на Рис. 12.

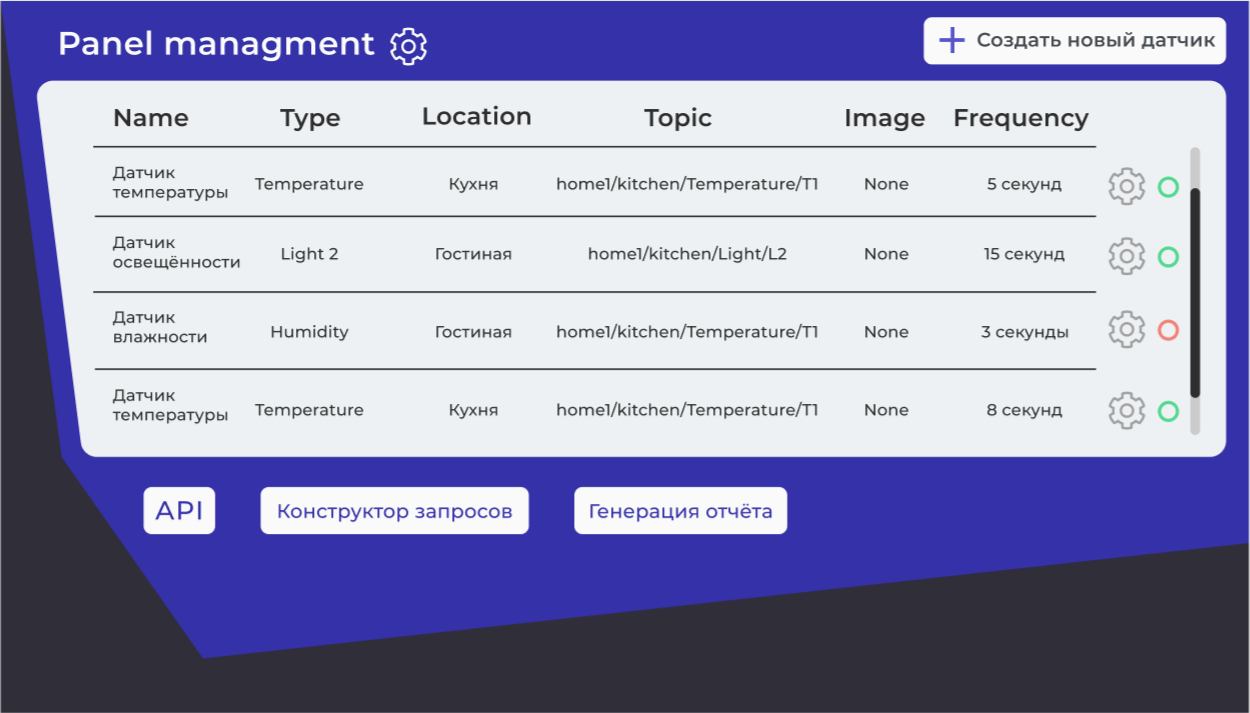


Рисунок Интерфейс главной страницы WEB-приложения



Рисунок Интерфейс модального окна изменения настроен системы

Что касается добавления новых датчиков – для создания новых датчиков требуется ввести следующие данные:

1. Название датчика (прибора);
2. Местоположение датчика (комната и т.п.);
3. Тип датчика (Температура, влажность, освещённость и т.д.);
4. Topic (уникальный адрес);
5. Описание датчика (прибора);
6. Частота публикации сообщений (в секундах);
7. CSV файл с серией данных, каждая строка которой характеризует одно состояние;
8. Фотография, характеризующая датчик (прибор).

Интерфейс страницы, которая характеризует создание нового датчика/устройства представлена на Рис. 14.

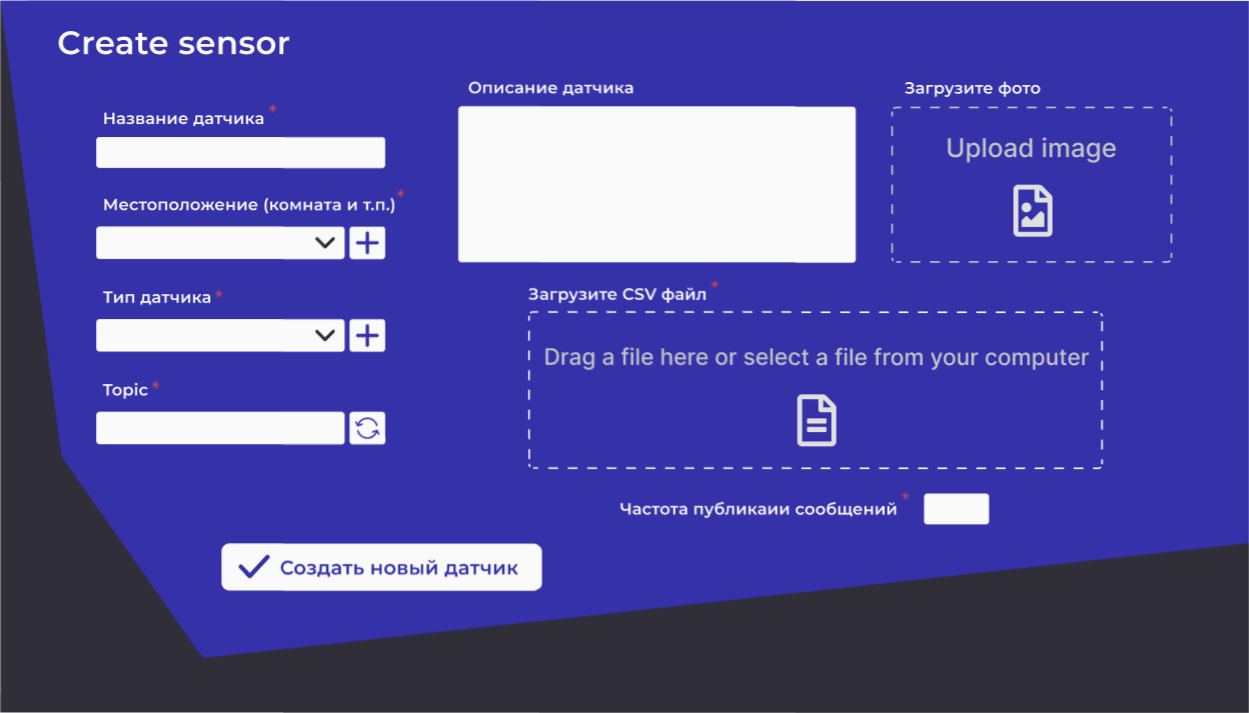


Рисунок Интерфейс страницы добавления нового датчика/устройства

Добавление нового датчика и редактирование датчиков реализованы на отдельных страницах, поскольку количество обрабатываемых элементов велико.

Что касается интерфейса редактирования данных о каком-либо датчике/устройстве – интерфейс представлен на Рис. 15.

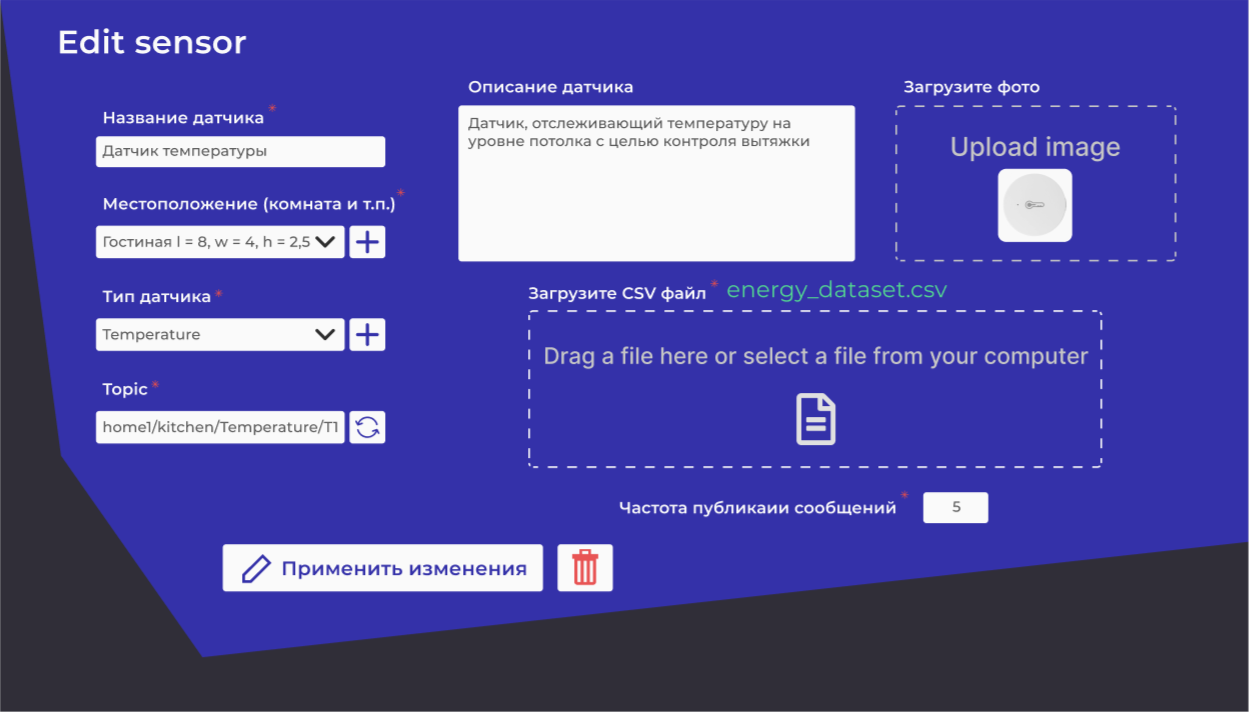


Рисунок Интерфейс страницы редактирования датчика/устройства

Интерфейс окна создания новой комнаты представлен на Рис. 16.

Такие элементы, как комната, тип комнаты и тип датчика должны быть реализованы в качестве модальных окон, поскольку действия, осуществляемые пользователем, будут характеризовать лишь запрос на создание.

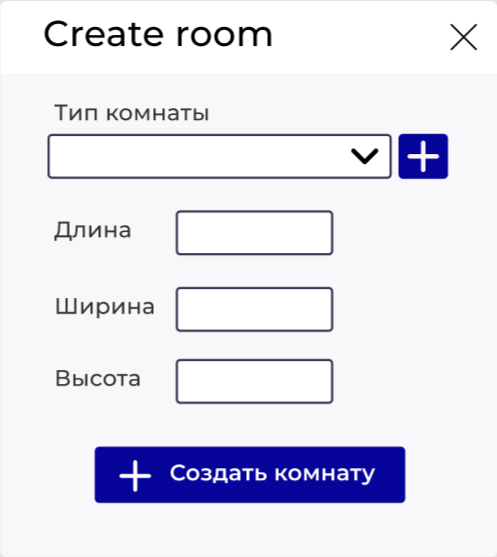


Рисунок Интерфейс модального окна создания новой комнаты

Интерфейс модального окна создания нового типа комнат представлен на рис. 17.

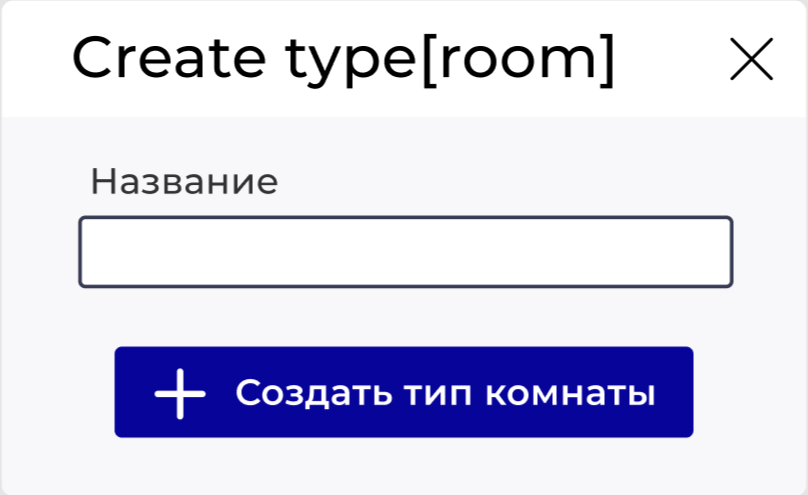


Рисунок Интерфейс модального окна создания нового типа комнаты

Помимо окон создания новых типов комнат есть и окно создания новых типов датчиков, интерфейс которого представлен на Рис. 18.

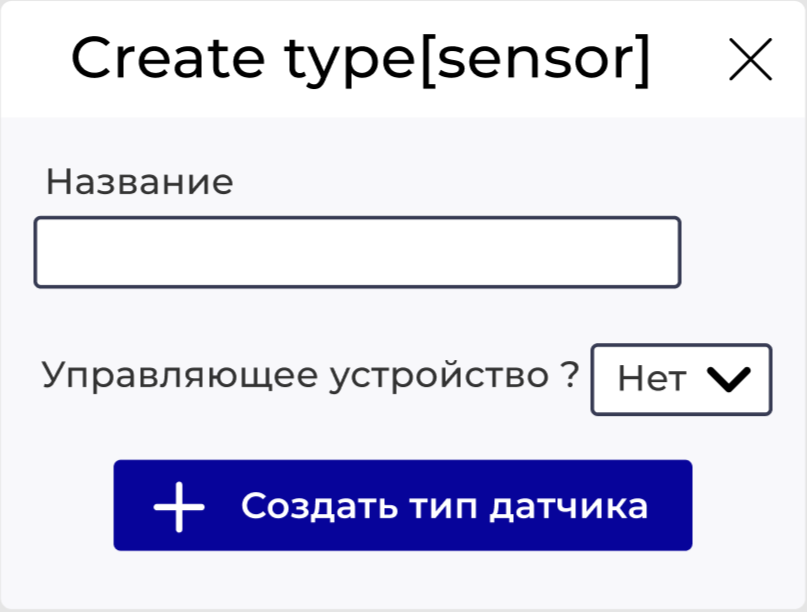


Рисунок Интерфейс модального окна создания нового типа комнаты

# Разработка системы

Поскольку, в рамках данной технологической практики не доступно использованием выделенных серверов, было принято решение реализовать сервис на бесплатном хостинге в качестве примера. Данный способ подразумевает не только экономию ресурсов, но и возможность использовать разрабатываемую систему без углубления в сферу развёртывания каких-либо решений.

Поскольку, WEB-приложение основывается на сервере, который эмулирует работу датчиков/устройств, сначала был разработан REST сервер. Как уже было описано на этапе проектирования, каждый датчик должен работать одновременно и постоянно, причём независимо друг от друга. Для решения данной задачи была выбрана библиотека Advanced Python Scheduler, которая имеет функциональность, схожую с потоками, которые содержат параметр «daemon», отвечающий за автономную работу потока до тех пор, пока его принудительно не закроют. Библиотека Advanced Python Scheduler работает по принципу работы CRON Tab. Алгоритм первоначального запуска сервера возможно представить следующей последовательностью действий:

1. Подключение к MQTT брокеру;
2. Открытие бесконечного соединения;
3. Запуск центра обработки и запросов данных с помощью библиотеки APScheduler.

Программный код реализации взаимодействия с MQTT брокером через библиотеку Advanced Python Scheduler представлен на Рис. 25.

REST API разработано с помощью библиотеки Flask-RESTful, особенности реализации которой представлены на Рис. 19 – Рис. 22.

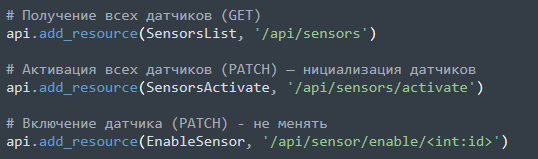


Рисунок Программный код добавления API модели в общую модель фреймворка Flask

Что касается модели, которая продемонстрирована на Рис. 19 – в основу её реализации легла модель стандартного объектно-ориентированного класса, программный код которого представлен на Рис. 20.



Рисунок Программный код добавления API модели в общую модель фреймворка Flask

Что касается обращения к базе данных, использование которого видно в начале метода *get* – программный код части класса для работы с базой данных представлен на Рис. 21.

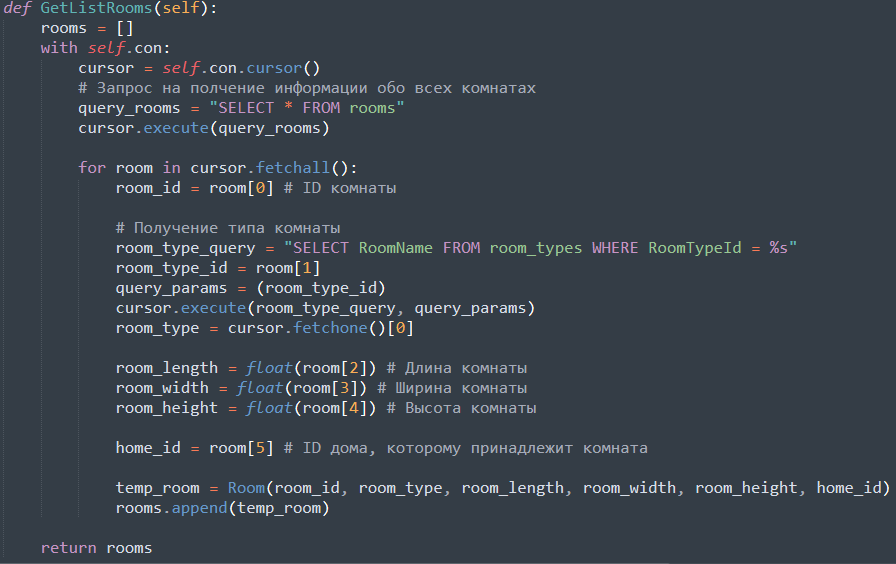


Рисунок Программный код получения списка комнат, которые хранятся в базе данных

Преимущество библиотеки Flask-RESTful в том, что для реализации API к одному адресу можно реализовать сразу несколько методов, по которым можно обращаться к API.

Для того, чтобы грамотно взаимодействовать с классом, который осуществляет запросы к базе данных, были созданы модели данных, список которых и возвращается из главного файла, в котором происходит реализация API. В качестве примера программный код класса модели комнаты представлен на Рис. 22.

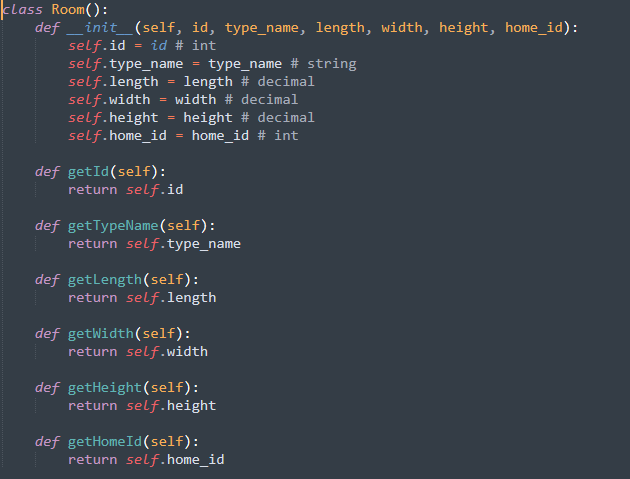


Рисунок Программный код получения списка комнат, которые хранятся в базе данных

Поскольку, при обращении к внешним ресурсам (в данном случае, к API) платформа ASP.NET выдаёт ошибку, REST API сервис реализован с помощью обходного пути через добавление заголовка «Access-Control-Allow-Origin» с помощью библиотеки Flask-CORS.

Что касается того, как данные будут приниматься сервером – для этого была использована утилита RequestParser, причём, в случае с загрузкой на сервер CSV файла и изображения использовалась библиотека Request, которая принимает на вход поток данных, который идёт с WEB-приложения.

Изначально, в качестве WEB-проекта был выбран пустой шаблон для ASP.NET приложения на платформе Framework, после чего была произведена настройка проекта под HTTPS.

Стилизация приложения была реализована с помощью CSS-фреймворка Bootstrap версии 4.5.0, что позволила использовать современные возможности «на полную». Для того, чтобы CSS и JS файлы подгружались быстрее, в качестве оптимизации загрузки подобных файлов использовались Bundle объекты, которые ссылаются на файлы в проекте. Программный код реализации Bundle объектов представлен ан Рис. 23.



Рисунок Программный код реализации Bundle объектов

Что касается программного кода использования данных объектов – он продемонстрирован на Рис. 24.

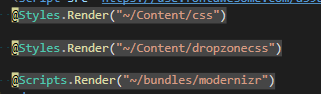


Рисунок Программный код использования Bundle объектов

Для того, чтобы загружать CSV файлы и изображения в WEB-приложение использовалась библиотека DropzoneJS, но для подключения данной библиотеки, данные подгружались не через пакеты Nuget, а через добавление новых элементов, поскольку пакеты Nuget содержат в себе устаревшую версию данной библиотеки.

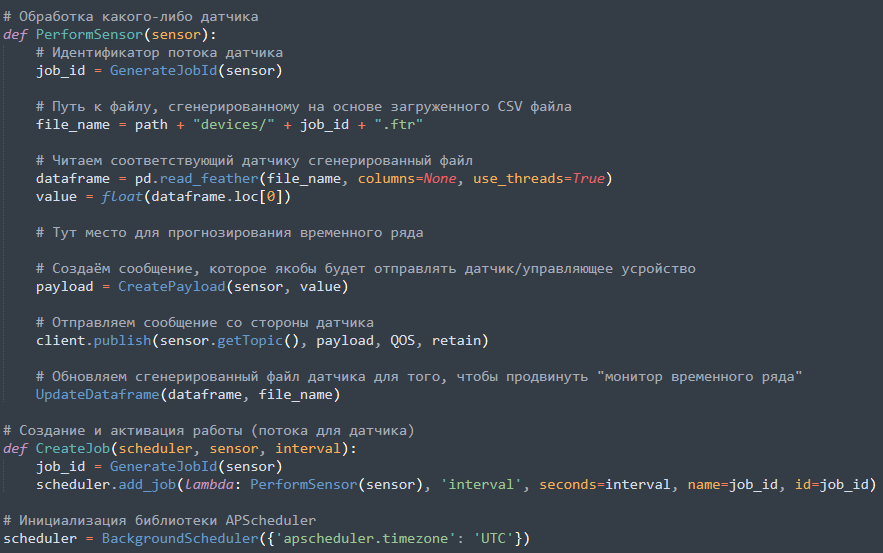


Рисунок Программный код работы библиотеки APScheduler

Для подключения к MQTT брокеру использовалась библиотека paho-mqtt, а в качестве MQTT брокера был взят сервис Cloud MQTT.

Что касается того, как REST архитектура была реализована в WEB-приложении, то использовалась библиотека Refit, которая работает по тому же принципу, что и библиотека Retrofit, которая предназначена для работы с API на мобильных устройствах.

Для того, чтобы осуществлять запросы к API был создан интерфейс, который представлен на Рис. 26.

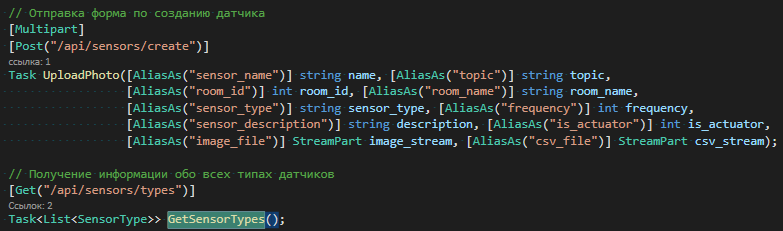


Рисунок Программный код интерфейса для работы с библиотекой Refit.

Для создания модальных окон использовалась модель PartialView для того, чтобы разделить логические типы моделей данных, которые используются на странице. Что касается обработки модальных окон – их обработка реализована с помощью фреймворка AJAX, с помощью которого возможно выполнять запросы асинхронно и без перезагрузки всей страницы. После того, как запрос на добавления датчика был отправлен модальное окно закрывается, выводится Popup сообщение о том, что добавлен новый датчик и опять же с помощью библиотеки AJAX, в блоке таблица, в которой отображаются датчики появляется новы датчик. Рис. 27 демонстрирует то, как работает модальное окно.

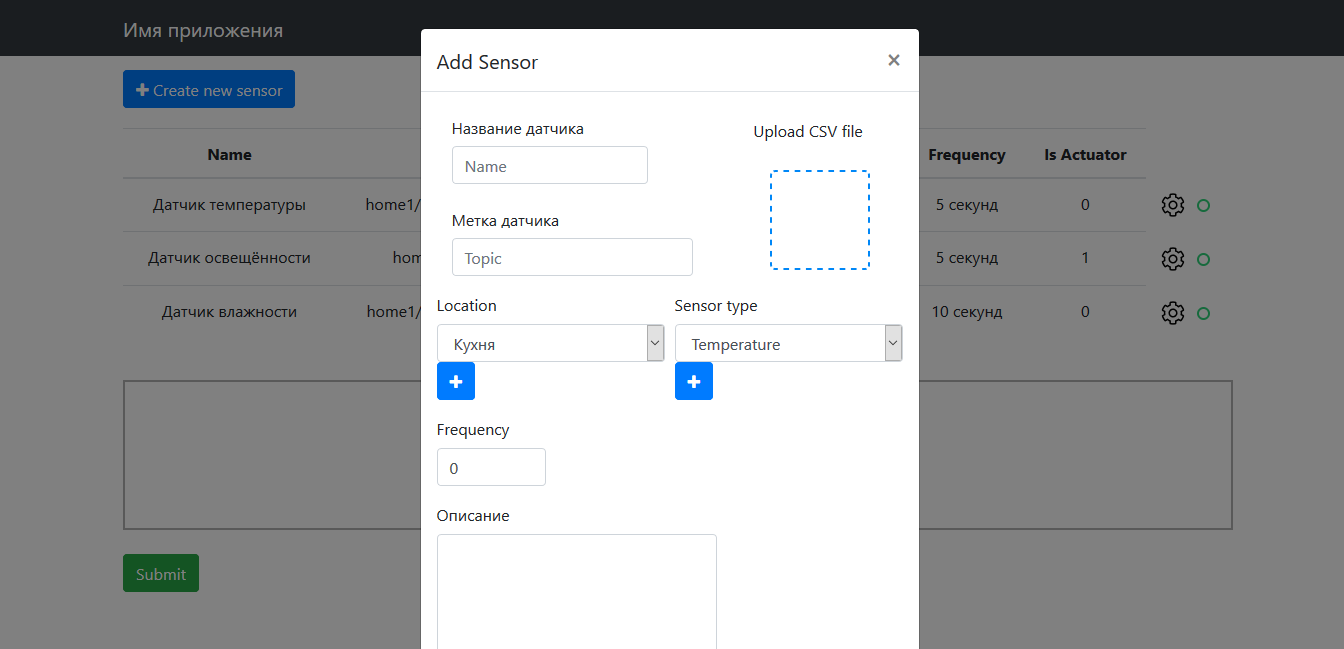


Рисунок Демонстрация того, как работает модальное окно

# Тестирование системы

Результатом разработки данной технологической практики является сервер, база данных и WEB-приложение. WEB-приложение будет лучше всего протестировать с помощью чёрного ящика, поскольку, всю главную функциональность содержит в себе сервер с REST архитектурой. Для тестирования сервера лучше использовать программу Postman, которая предназначена для тестирования всевозможных запросов со множеством вариантов и типов входных параметров. Поскольку, в разработанной системе используется блокировка вводы отрицательных чисел в поля ввода, во всех нижеописанных таблицах с тестами не учитываются ситуации, которые предусматривают отрицательные числа в различных полях. Таблицы со всевозможными тестами представлены в таблицах 4.2.

Таблица 4.1. Таблица тестов для стартовой страницы настроек и окна редактирования этих настроек

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** | **Описание** |
| 1 | MQTT Adress = squekl.cloud.com  Port= 13666  Username = inmovery  Password= Pablo228 | Переход на главное окно приложения | Верный набор данных |
| 2 | Какое-то из полей пустое | Вывод модального окна с сообщением о том, что введены не все данные | Неправильный набор данных из-за пустого поля |
| 3 | В каком-либо поле находится значение, которое не прошло проверку | Вывод модального окна с сообщением о том, что данные не валидны с указанием того параметра, который не прошёл проверку | Набор данных с не валидными данными |

Таблица 4.2. Таблица тестов для главной страницы системы

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** | **Описание** |
| 1 | Не создано ни одного датчика | Вывод текста «Вы ещё не создавали датчиков» со ссылкой на создание датчика и документацию того, как его создавать и что для этого нужно | Проверка того, что ни один датчик ещё не создан |
| 2 | Какой-либо датчик не активирован, то есть данные с какого-то датчика не доходят до MQTT брокера | Возле не активированного датчика выведено изображение с красным цветом | Проверка того, что пользователь намеренно остановил сбор данных с какого-то датчика |
| 3 | Все датчики включены | Возле каждого датчика выведено изображение с зелёным цветом | Проверка того, что все датчики, которые добавил пользователь – активированы |

Таблица 4.3. Таблица тестов для модального окна добавления нового датчика

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** | **Описание** |
| 1 | Название датчик состоит только из цифр или только из пробелов и табов | Вывод модального окна с предупреждением о недопустимых названиях | Проверка правильности заполнения поля «Название» |
| 2 | Не выбрано какое-либо поле | Вывод модального окна с сообщением о том, что пользователь ввёл не все данные | Проверка «невнимательности» пользователя |
| 3 | В какое-либо поле для загрузки файлов загружен файл не подходящего формата | Вывод модального окна об ошибке загрузки файла с указанием на то, какого формата и какой структуры должен быть загружаемый файл | Проверка на загрузку файлов не подходящего формата |
| 4 | Поле «Topic» содержит значение, которое уже есть в базе данных у какого-то датчика | Вывод модального окна с сообщением о том, что пользователю нужно проверять актуальность параметра «Topic» с указанием картинки, на которой подробно описан процесс валидации параметра «Topic» | Проверка валидации параметра «Topic» |
| 5 | Поле «Частота публикаций сообщений» содержит в себе значение |  |  |

Что касается тестов для окон, связанных с редактированием данных, то для них тесты такие же, как и тесты для окон, которые предназначены для добавления датчиков и настроен.

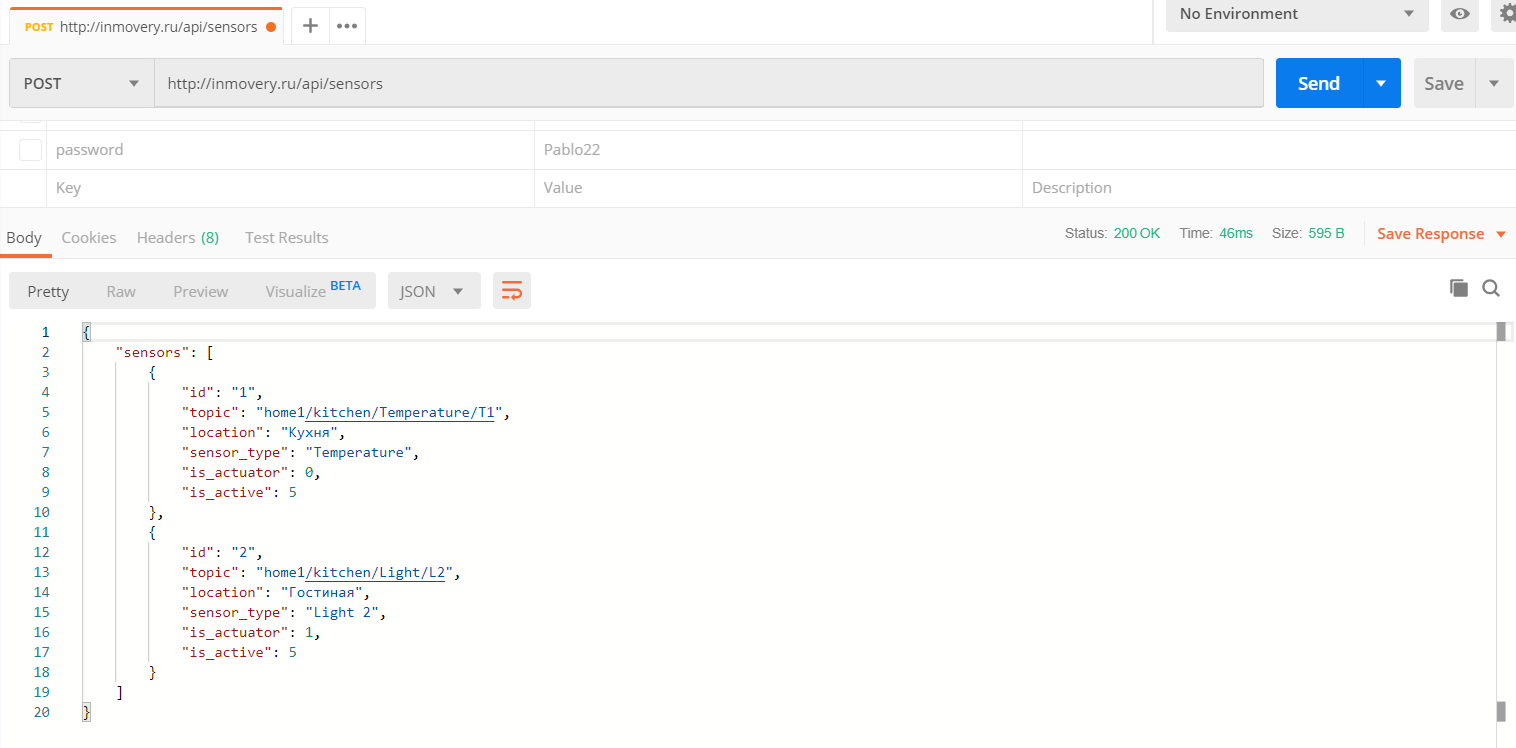


Рисунок Тестирование получения всех датчиков

Кроме того, были разработаны интеграционные тесты с помощью фреймворка Selenium, которые представлены в репозитории по адресу <https://github.com/inmovery/TechnologicalPractice>.

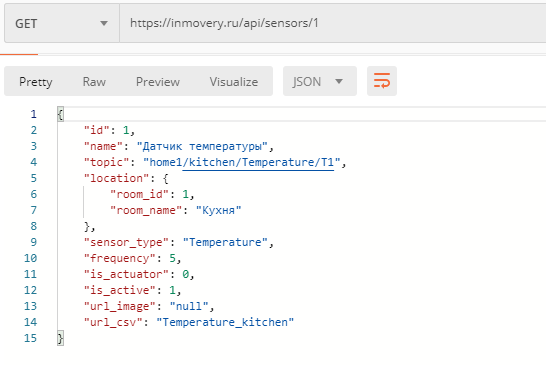


Рисунок Тестирование получения датчика по его ID

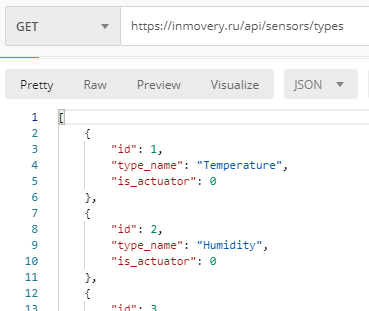


Рисунок Тестирование получения всех типов датчиков

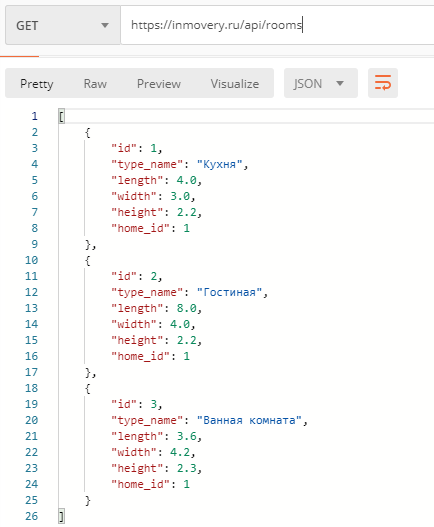


Рисунок Тестирование получения всех комнат

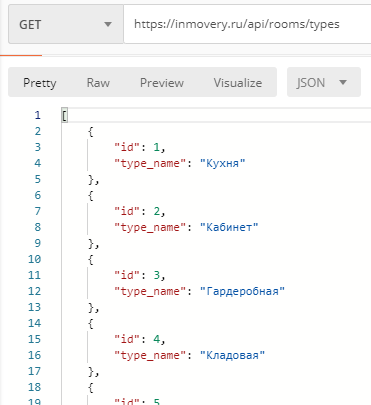


Рисунок Тестирование получения всех типов комнат

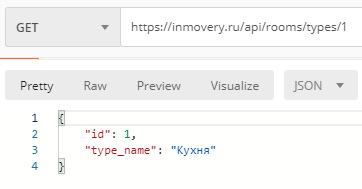


Рисунок Тестирование получения типа комнаты по его ID

# Заключение

Результатом анализа, проектирования и разработки касаемо данной технологической практики стала система, которая состоит из сервера по архитектуре REST API и WEB-приложение, предназначенное для управления системой, базами данных и контроля сбора данных. На протяжении этапа анализа предметной области была выявлена большая часть функциональных требований. Однако, не все требования удалось реализовать из-за нехватки времени, а именно, не удалось реализовать модели машинного обучения, входными данными для которых были бы количество проживающих людей, размеры дома и комнат.

На этапе анализа предметной области были применены знания, полученные на дисциплинах «Теория вероятности» и «Статистические и эмпирические методы компьютинга», а работа с временными рядами. Также, на этапе разработки были применены знания, полученные в ходе обучения на дисциплине «Конструирование программного обеспечения».

Проектирование проводилось с учётом основным интерфейсных недостатков проанализированных аналогов приложения, а именно, с добавлением простых элементов, как на главном, так и на второстепенных страницах WEB приложения. В результате тестами покрыта большая часть программного кода, а также была достигнута цель и решены все задачи, поставленные во введении. На этапе проектирования были применены знания, полученные на дисциплине «Научно-исследовательский семинар», а именно, использование и опыт в работе с UX/UI.

На данный момент планируется изменить функционал, связанный генерацией данных на сервере и сделать мобильное приложение, которое позволит управлять отправкой данных с датчиков по средству добавления (записи) совершённого действия с помощью смартфона и добавления в очередь выполнения отправки запроса на сервер.

# Библиографический список

1. William Kluegel, Muhammad Aamir Iqbal, Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. A Realistic Dataset for the Smart Home Device Scheduling Problem for DCOPs (Дата обращения: 04.03.2020);
2. Sowmya Ramapatruni, Sandeep Nair Narayanan, Sudip Mittal, Anupam Joshi, and Karuna Joshi. Anomaly Detection Models for Smart Home Security (Дата обращения: 09.03.2020);
3. Протоколы передачи данных IoT. // (Электронный ресурс) — URL: <https://iot.ru/wiki/protokoly-peredachi-dannykh-iot> (Дата обращения: 07.03.2020);
4. Jasenka Dizdarevic, Francisco Carpio, Admela Jukan, Xavi Masip-Bruin. A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration (Доступ: 07.03.2020);
5. Протокол MQTT: концептуальное погружение. // (Электронный ресурс) — URL: <https://habr.com/ru/post/463669/> (Дата обращения: 15.03.2020);
6. Health Smart Home (HIS) datasets. // (Электронный ресурс) — URL: <https://lig-getalp.imag.fr/en/health-smart-home-his-datasets-2/> (Дата обращения: 08.03.2020);
7. REFIT Smart Homes. // (Электронный ресурс) — URL: <https://www.refitsmarthomes.org/> (Дата обращения: 08.03.2020);
8. Intel Lab Data. // (Электронный ресурс) — URL: <http://db.csail.mit.edu/labdata/labdata.html> (Дата обращения: 08.03.2020);
9. Программные решения на тему «Smart Home Device Scheduling». Репозитории «SHDS-solver», «SHDS\_dataset», «SHDS\_dataset2», «SHS». // (Электронный ресурс) — URL: [https://github.com/persoon](https://github.com/persoon/SHDS-solver) (Дата обращения: 08.03.2020);
10. Hardware Sensors Indicator. // (Электронный ресурс) — URL: <https://github.com/alexmurray/indicator-sensors> (Дата обращения: 08.03.2020);
11. Create Temperature Sensor Data Graphs With Temboo. // (Электронный ресурс) — URL: <https://dzone.com/articles/create-temperature-sensor-data-graphs-with-temboo> (Дата обращения: 10.03.2020);
12. Modern Complex Smart Home API. // (Электронный ресурс) — URL: <https://e154.github.io/smart-home/api/> (Дата обращения: 10.03.2020);
13. API для передачи показаний датчиков посредством TCP, UDP, GET, POST, MQTT // (Электронный ресурс) — URL: <https://narodmon.ru/#!devdoc> (Дата обращения: 15.03.2020);
14. Home Assistant At SnM Nest // (Электронный ресурс) — URL: <https://github.com/sameeraman/HomeAssistant> (Дата обращения: 10.03.2020);
15. Seasonality, Holiday Effects, And Regressors. // (Электронный ресурс) — URL:<https://facebook.github.io/prophet/docs/seasonality,_holiday_effects,_and_regressors.html> (Дата обращения: 11.03.2020);
16. The Best Format to Save Pandas Data. // (Электронный ресурс) — URL: <https://towardsdatascience.com/the-best-format-to-save-pandas-data-414dca023e0d> (Дата обращения: 12.03.2020);
17. Руководство по ASP.NET MVC 5. // (Электронный ресурс) — URL: <https://metanit.com/sharp/mvc5/> (Дата обращения: 22.05.2020);
18. Selenium C# Tutorial. // (Электронный адрес) — URL: <https://www.toolsqa.com/selenium-c-sharp/> (Дата обращения: 23.06.2020);
19. **Термины и обозначения**

Телеметрия — удалённое слежение за системами.

IP-мониторинг объекта — удалённое управление системами по сети.

GSM-мониторинг — удалённое информирование об инцидентах в доме (квартире, офисе, объекте) и управлении системами дома через телефон (в некоторых системах при этом можно получать голосовые инструкции по планируемым управляющим воздействиям, а также голосовые отчёты по результатам выполнения действий).

Аккумуляторы ИБП — аккумуляторы с источником бесперебойного питания (мотор-генератор).

Протоколы передачи данных IoT — правила, определяющие способы обмена данными между объектами сети Интернета вещей.

CRUD — базовые операции с базами данных и не только (CREATE, READ, UPDATE, DELETE).

PWM (Pulse Width Modulation) — это способ кодирования аналогового сигнала, предназначенная для управления аналоговыми цепями с использованием цифровых выходов. Данная техника применяется, например, при создании аналогового уровня напряжения, регулировании и преобразовании мощности, а также для управления скорости двигателя.

1. **Техническое задание**

Техническое задание доступно в репозитории по адресу <https://github.com/inmovery/>TechnologicalPractice.