



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

Магистерская программа Современные интеллектуальные программно-аппаратные комплексы

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Система управления умным домом

Студент ИУ6-32М
(Группа)

(Подпись, дата)

Д.А. Троицкий
(И.О. Фамилия)

Руководитель курсового проекта

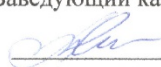
(Подпись, дата)

М.В. Фетисов
(И.О. Фамилия)

2020 г.

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ6
(Индекс)
 А.В. Пролетарский
(И.О.Фамилия)
« 15 » 09 2020 г.

ЗАДАНИЕ на выполнение курсового проекта

по дисциплине **Современные технологии разработки программного обеспечения**

Студент группы ИУ6-32М

Троицкий Дмитрий Александрович
(Фамилия, имя, отчество)

Тема курсовой работы Система управления умным домом

Магистерская программа **Современные интеллектуальные программно-аппаратные комплексы**

Направленность КП (учебная, исследовательская, практическая, производственная, др.)

учебная

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения КР: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Основные требования к курсовой работе:

1. Интегрировать устройства умного дома в концентратор Raspberry Pi
2. Разработать портал для управления устройствами


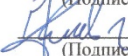
Оформление курсовой работы:

1. Расчетно-пояснительная записка (РПЗ) на 20-30 листах формата А4.
2. Все материалы и исходный текст программы записать в вузовский репозиторий исходного кода, документы – на страницу дисциплины на сайте кафедры.

Дата выдачи задания «01» сентября 2020 г.

Студент

Руководитель курсового проекта

 14.09.2020 Д.А. Троицкий
(Подпись, дата) (И.О.Фамилия)
 М.В. Фетисов
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

РЕФЕРАТ

Записка 34 с., 0 таб., 17 рис., 7 источников, 0 прил.

УМНЫЙ ДОМ, RASPBERRY PI, CI, ZIGBEE, HOMEASSISTANT, DJANGO, API

Объектом разработки является «Система управления умным домом», предназначенная для получения информации с датчиков или изменения состояния устройств «умного дома».

Целью работы является анализ предметной области, разработка бизнес-процессов разрабатываемой системы, проектирование схемы реляционной базы данных, создание форм интерфейса, а также развертывание на Raspberry Pi сервиса по интеграции данных от устройств «умного дома», который будет передавать эти данные для обработки разрабатываемому сервису.

В результате был спроектирован прототип информационной системы управления «умным домом»

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	5
ВВЕДЕНИЕ.....	6
1 Анализ требований и уточнение спецификаций.....	7
1.1 Анализ предметной области и технического задания	7
1.2 Выбор модели жизненного цикла, методов и средств разработки.....	10
1.3 Разработка бизнес-процессов	11
1.4 Проектирование базы данных	19
1.4.1 Выделение основных сущностей.....	19
1.4.2 Разработка даталогической модели.....	19
2 Проектирование структуры и компонентов программного продукта.....	24
2.1 Настройка Raspberry Pi	24
2.2 Разработка интерфейса пользователя.....	24
2.2.1 Выбор средств реализации интерфейса	26
2.2.2 Разработка форм интерфейса	27
2.3 Реализация программного продукта.....	29
3 Выбор стратегии тестирования и разработка тестов.....	32
ЗАКЛЮЧЕНИЕ	33
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	34

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

API – (*Application Programming Interface*) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.

Django – фреймворк для веб-приложений на языке Python.

JS (*JavaScript*) – язык программирования, применяемый к веб-страницам.

jQuery – это быстрая, небольшая и многофункциональная библиотека JavaScript.

HTML (*Hypertext Markup Language*) – стандартизированный язык разметки документов во Всемирной паутине (web).

Home Assistant – приложение с открытым исходным кодом для организации умного дома.

MVC (*Model – View - Controller*) – схема разделения данных приложения, интерфейса и управляющей логики на три отдельных компонента для независимой разработки компонентов.

Python – язык программирования.

SQL (*Structured Query Language*) – язык программирования, применяемый для управления данными в реляционных СУБД.

WEB (*World Wide Web*) – система, предоставляющая доступ к компьютерам, подключённым к сети Интернет.

Zigbee – спецификация сетевых протоколов.

БД – база данных.

СУБД – система управления базами данных.

Фреймворк – заготовки или шаблоны для разработки программного обеспечения, определяющие архитектуру программной системы.

ВВЕДЕНИЕ

Данная работа посвящена разработке информационной системы управления умным домом.

В настоящее время развивается тренд на автоматизацию всех процессов. Одно из направлений в автоматизации – это умный дом. Несколько лет назад создание системы управления было дорогостоящим процессом и позволить его могли не все. С развитием технологий на рынке появилось множество устройств (лампочки, розетки, датчики и т. п.), выпускаемых разными производителями.

Из-за разнообразия стандартов обмена данными и разных производителей устройств возникает проблема: для управления нужно использовать множество приложений, сайтов и несколько концентраторов (хабов). На рынке уже есть решения, которые можно использовать для объединения устройств разных компаний, но данные системы либо слишком дороги, либо их необходимо дорабатывать по старым инструкциям.

Разрабатываемая информационная система учитывает эти недостатки и предлагает решение в виде двух компонентов: Raspberry Pi с настроенным приложением Home Assistant и информационный портал, обрабатывающий данные от Raspberry Pi.

Информационная система разрабатывается на основе учебного плана кафедры ИУ6 МГТУ им. Н. Э. Баумана.

1 Анализ требований и уточнение спецификаций

1.1 Анализ предметной области и технического задания

На данный момент самые популярные устройства умного дома используют технологии Wi-Fi, Bluetooth, ZigBee и Z-Wave. У каждой из технологий есть свои плюсы и минусы, их можно использовать вместе, компенсируя недостатки каждой. Но для разных задач и разных типов умных устройств используются разные технологии.

Wi-Fi используется для управления IP-камерами, телевизорами, аудио/медиа-плеерами и другой техникой для передачи видеосигнала. Конечно, Wi-Fi может использоваться и в выключателях света, датчиках, термостатах, но отсутствие ретрансляции сигнала и высокое энергопотребление не позволяют делать на нем датчики, работающие годами. Каждый производитель для своего Wi-Fi – устройства выпускает свое собственное приложение, и нет единого стандарта, чтобы управлять всей техникой из одного приложения. Это не позволяет сделать умный дом только на Wi-Fi по-настоящему удобным.

Bluetooth имеет малое энергопотребление, поэтому его можно использовать в датчиках. Но отсутствие общего стандарта управления вынуждают каждого производителя делать свое собственное приложение, что неудобно для пользователя.

Беспроводной протокол *ZigBee* специально разрабатывался для применения в сети датчиков. Он основан на технологии mesh, то есть все датчики видят друг друга и могут передавать информацию на хаб через соседей. Это повышает надёжность управления устройствами, так как датчик, установленный на большом расстоянии от концентратора, передаст и получит данные от других датчиков (в отличие от Bluetooth, дальность действия которого ограничена 10 метрами без препятствий). С ZigBee выпускают почти все устройства для создания домашней автоматизации: реле, диммеры, лампы, термостаты, замки, датчики. По сравнению с другими протоколами для умного дома у ZigBee-устройств самые привлекательные цены, однако отсутствие 100%

совместимости между устройствами и хабами разных производителей не позволяет собрать умный дом только на ZigBee.

Беспроводной протокол *Z-Wave*, разрабатывался с 2001 года специально для домашней автоматизации. Главным его преимуществом является полная совместимость между устройствами разных производителей. В *Z-Wave*, так же, как и в ZigBee, используется топология mesh с поддержкой ретрансляции сигнала и автоматическим нахождением лучшего маршрута. Главный минус – цена. В среднем, стоимость устройства составляет 60-80 евро, что примерно вдвое выше, чем у аналогов с ZigBee.

Так как протоколы управления устройствами создавались для разных целей и каждому нужно своё приложение, то необходимо устройство, которое имеет поддержку этих протоколов. Для этого будет использоваться Raspberry Pi – одноплатный компьютер с малыми размерами. Из коробки он имеет поддержку Wi-Fi, Bluetooth, Ethernet и несколько разъёмов USB. По сравнению с профессиональными системами управления умным домом его стоимость небольшая – не более 5000 рублей. Для управления ZigBee устройствами существует USB адаптер CC2531, который настраивается для интеграции с системами управления умным домом.

Теперь нужно выбрать систему управления умным домом. На рынке есть несколько систем с открытым исходным кодом. Рассмотрим их преимущества и недостатки.

Domoticz – мультиплатформенное ПО с открытым кодом, ориентированное на создание системы управления умным домом. Поддерживает большое количество различных устройств разных вендоров, в том числе работает с устройствами Xiaomi. Domoticz не требует большой производительности устройств, поэтому можно взять старую версию Raspberry Pi. Для установки нужно ввести одну команду в терминал, далее система произведёт все необходимые настройки. Настройка системы и управление устройствами происходит через web интерфейс. Недостатками системы можно считать

устаревший дизайн системы и отсутствие ролевой модели (можно создать только одного пользователя).

Majordomo – белорусско-российская система управления умным домом, написанная на PHP. Интерфейс приложения на русском языке, поддерживает установку на Raspberry Pi. Недостатками системы можно считать непонятные ошибки при установке, некорректно написанные модули управления устройствами, а также отсутствие удаленного доступа к системе через интернет.

Home Assistant – система написана на Python, существуют docker образы и образ для Raspberry Pi. Система просто устанавливается, имеет поддержку большого количества устройств (список постоянно обновляется), есть REST API [2], доступен удалённый доступ. Для настройки устройств не нужно быть программистом, так как большинство настроек производится через WEB интерфейс. Также можно создавать сценарии – реакция устройств на возникающее действие (например, включение лампочек в комнате при срабатывании датчика движения). Если необходимого модуля нет среди начальной конфигурации Home Assistant, то его можно написать самому и разместить в магазине модулей. Недостатками системы можно считать документацию на английском языке (хотя есть русскоязычные сообщества с подробной информацией) и неполная настройка устройств через Web – интерфейс [1].

Среди рассмотренных систем управления нет ни одной системы, которая позволяет разграничивать доступ по ролям, а также если пользователь имеет несколько систем умного дома, то невозможно их объединение в одной Web – интерфейсе.

Исходя из анализа предметной области было принято решение использовать устройства умного дома разных технологий и разных производителей, установка системы Home Assistant на устройство Raspberry Pi и разработка приложения для управления несколькими домами с добавлением ролевой модели и дополнительной сетевой защитой концентратора (Raspberry Pi).

1.2 Выбор модели жизненного цикла, методов и средств разработки

Существуют несколько моделей жизненного цикла программного обеспечения: каскадная модель, итерационная модель, спиральная модель.

Проанализировав преимущества и недостатки каждой из них, была выбрана спиральная модель жизненного цикла. Её преимущества:

- сокращение число итераций и, следовательно, число ошибок и несоответствий, которые необходимо исправлять,
- сокращение сроков проектирования,
- упрощение создания проектной документации.

Спиральная модель лежит в основе технологии быстрой разработки продуктов или RAD-технологии (*rapid application development*), которая предполагает активное участие конечных пользователей будущей системы в процессе ее создания.

В ходе курсовой работы разрабатывается прототип информационной системы, и не ставится вопрос масштабирования продукта, поэтому для выполнения поставленной задачи можно выбрать следующую архитектуру: «клиент – один сервер – одна база данных».

Для серверной части используется фреймворк Django, так как он использует язык программирования Python, а на данном языке написано множество пакетов (расширений) для получения и обработки данных с вебсайтов. На языке Python написан Home Assistant, что уменьшает количество ошибок при обмене данными.

Для получения информации о состоянии устройств нужен круглосуточный мониторинг. Данная функция выполняется долго и с периодичностью, что будет мешать работе приложению с веб-приложением. Для выполнения таких задач существует пакет Celery, совместимый с Django [3]. При работе данной связки пишется функция на языке Python, которую можно совместно использовать и Django-приложению, и Celery [6]. С помощью Django можно следить за выполнением задач Celery, менять график запуска задач или отключать их

выполнение при проведении технических работ на серверах компаний, предоставляющих данные об отправлениях.

Для удобства управления и просмотра результата выполнения задач используются фреймворки Django-celery-results и Django-celery-beat. Первый отвечает за создание интерфейса для просмотра результата выполнения задач, а второй создаёт интерфейс и таблицы в БД для задания периодических задач.

Для работы Celery необходима база данных. Можно использовать БД веб-приложения, но выполнение задач будет мешать работе веб-сервиса. Поэтому разработчики расширения рекомендуют использовать RabbitMQ – программный брокер сообщений на основе стандарта AMQP [7]. Он также совместим и с Django.

Теперь следует выбрать СУБД для веб-приложения. Самыми распространёнными СУБД являются MySQL и PostgreSQL. Но для хранения данных был выбран PostgreSQL, так как он быстрее MySQL и все таблицы в PostgreSQL представлены в виде объектов, которые могут наследоваться, а все действия с таблицами выполняются с помощью объективно ориентированных функций, что является подходящим для данной разработки. Сильными сторонами PostgreSQL так же считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликации,
- наследование,
- легкая расширяемость.

1.3 Разработка бизнес-процессов

Проанализировав техническое задание и существующие аналоги, можно сформулировать сценарии взаимодействия пользователя с системой, которые будут использоваться при построении бизнес-процессов.

В системе две роли: владелец и гость умного дома. Пользователь не может одновременно иметь две роли на одном умном доме, но может быть владельцем одного умного дома и гостем в другом умном доме.

Сначала рассмотрим бизнес-процессы, доступные пользователям с любой ролью.

Для авторизации в системе пользователь заходит на страницу авторизации и вводит учётные данные в форму. Если пользователь с такими учётными данными существует в системе, то веб-приложение авторизует пользователя. Если данные неверные, то программный продукт выдаёт сообщение об ошибке. Полученная схема бизнес-процесса изображена на рисунке 1.

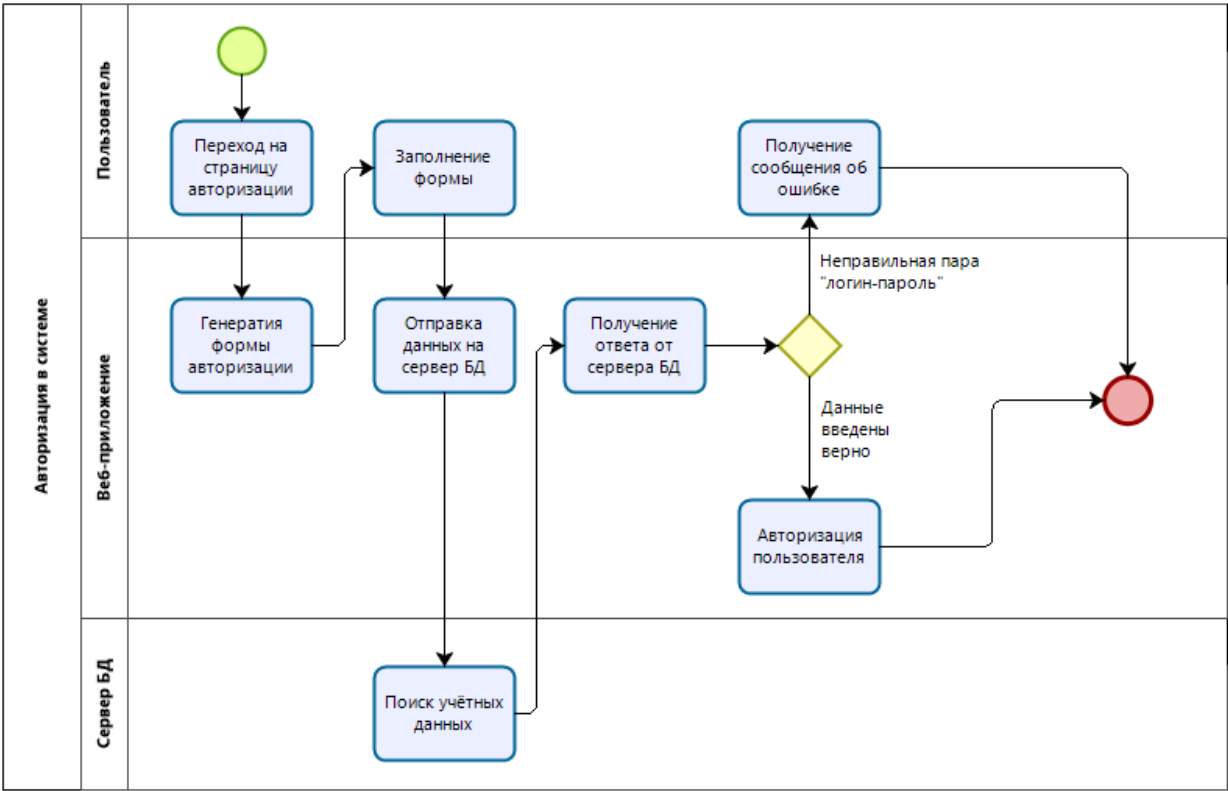


Рисунок 1 – Схема бизнес-процесса авторизации в системе

Для регистрации в системе пользователь заходит на страницу регистрации и вводит свои учётные данные: логин, email, пароль, подтверждение пароля. Веб-приложение производит валидацию введённых данных (проверяет введённые данные на соответствие шаблону данного поля и ищет незаполненные поля) и, если возникают ошибки, то выводит сообщение об ошибке. Если данные верны, то программная система ищет введённые учётные данные среди зарегистрированных пользователей. Если пользователь с введённым логином или email уже существует, то выдаётся сообщение об ошибке. Если пользователей с такими учётными записями не зарегистрировано, то сервер БД

сохраняет полученные данные. Затем программный продукт отправляет по указанному email письмо, содержащее ссылку для активации аккаунта и подтверждения адреса электронной почты. Если пользователь переходит по данной ссылке, то аккаунт активируется и можно использовать все возможности портала. Полученная схема бизнес-процесса изображена на рисунке 2.

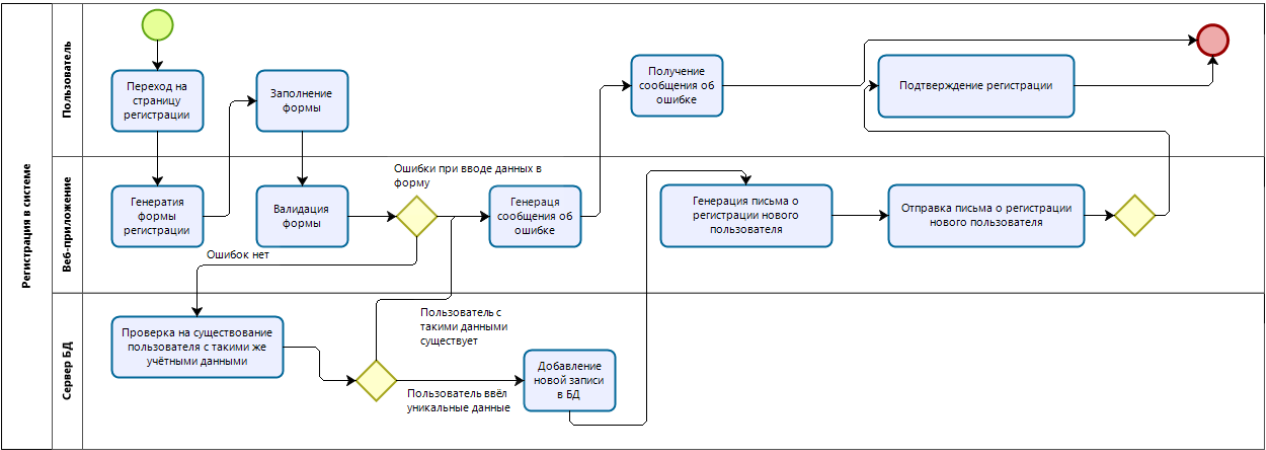


Рисунок 2 – Схема бизнес-процесса регистрации в системе

Для изменения информации учётной записи пользователь должен перейти на страницу профиля и нажать кнопку редактирования. Пользователь может изменять адрес электронной почты и пароль. После изменения программный продукт производит валидацию введённых данных (проверяет введённые данные на соответствие шаблону данного поля и ищет незаполненные поля) и, если возникают ошибки, то выводит сообщение об ошибке. Если пользователь с новым email уже существует в системе, то веб-приложение выдаёт сообщение об ошибке и восстанавливает старый адрес электронной почты. Если пользователей с новым адресом электронной почты нет в системе, то данные успешно сохраняются. Полученная схема бизнес-процесса изображена на рисунке 3.

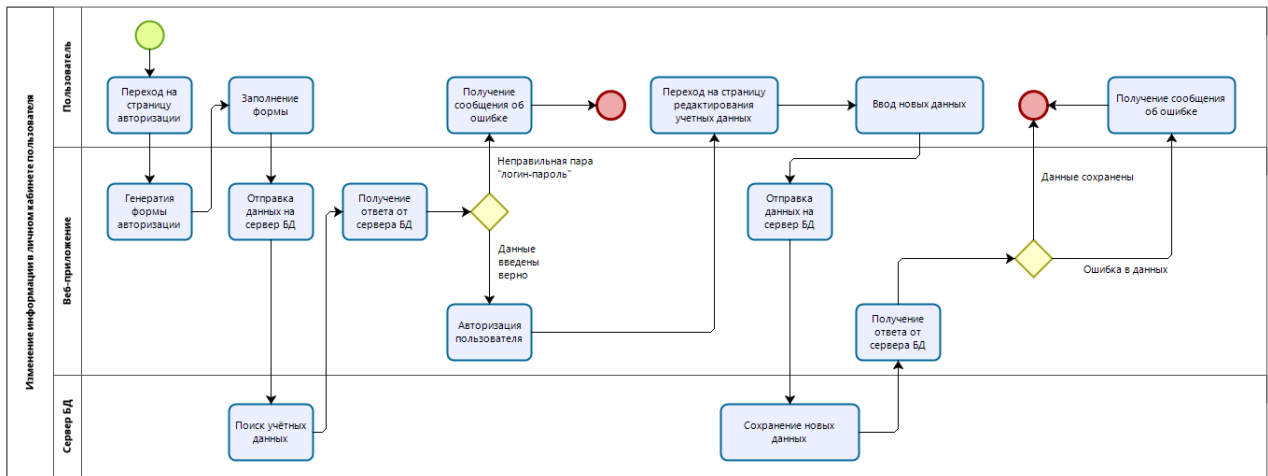


Рисунок 3 – Схема бизнес-процесса изменения информации в профиле пользователя

Для добавления умного дома необходимо получить токен умного дома из Home Assistant. Это делается по ссылке http://IP_ADDRESS:8123/api/. Затем в интерфейсе разрабатываемого программного обеспечения нужно ввести URL умного дома, полученный токен и описание умного дома (необязательное поле). Система делает запрос к умному дому и, если все данные верны, добавляет умный дом с единственным владельцем и включает мониторинг состояния устройств. Если данные введены неверно, то выдаётся ошибка. Полученная схема бизнес-процесса изображена на рисунке 4.

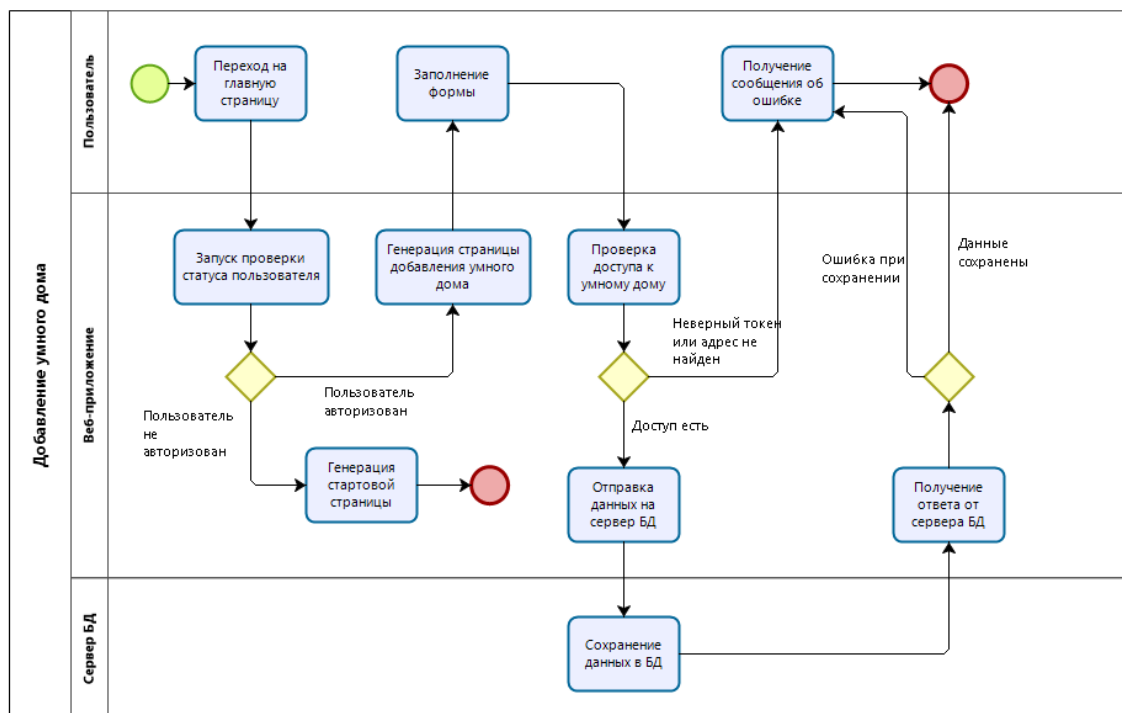


Рисунок 4 – Схема бизнес-процесса добавления умного дома в систему

Для удаления умного дома из системы нужно выбрать умный дом и нажать кнопку удаления. При этом отключается мониторинг состояний устройств и удаляется связь пользователей с этим умным домом. Полученная схема бизнес-процесса изображена на рисунке 5.

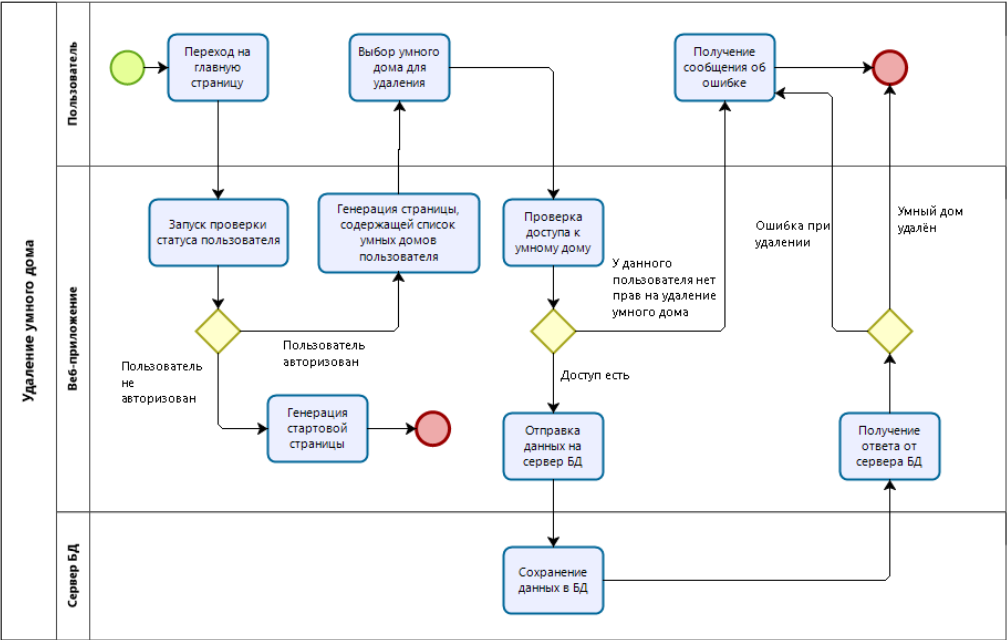


Рисунок 5 – Схема бизнес-процесса удаления умного дома из системы

Для просмотра списка доступных умных домов пользователь делает запрос из Web интерфейса. Система производит поиск по базе данных и, если есть доступные умные дома, выводит их список вместе со списком пользователей, которые тоже имеют доступ к этому умному дому. Полученная схема бизнес-процесса изображена на рисунке 6.

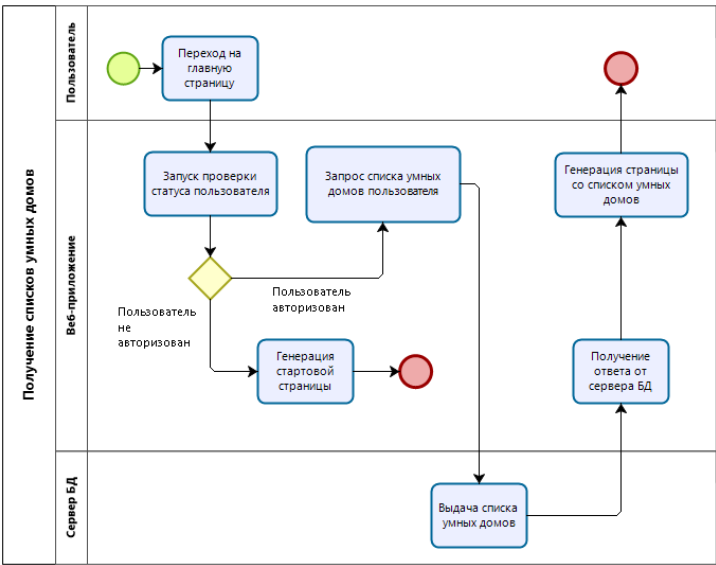


Рисунок 6 – Схема бизнес-процесса получения списка умных домов

Для переключения состояния устройства пользователь нажимает на переключатель около устройства. Если умный дом недоступен, то система выдаст ошибку. Полученная схема бизнес-процесса изображена на рисунке 7.

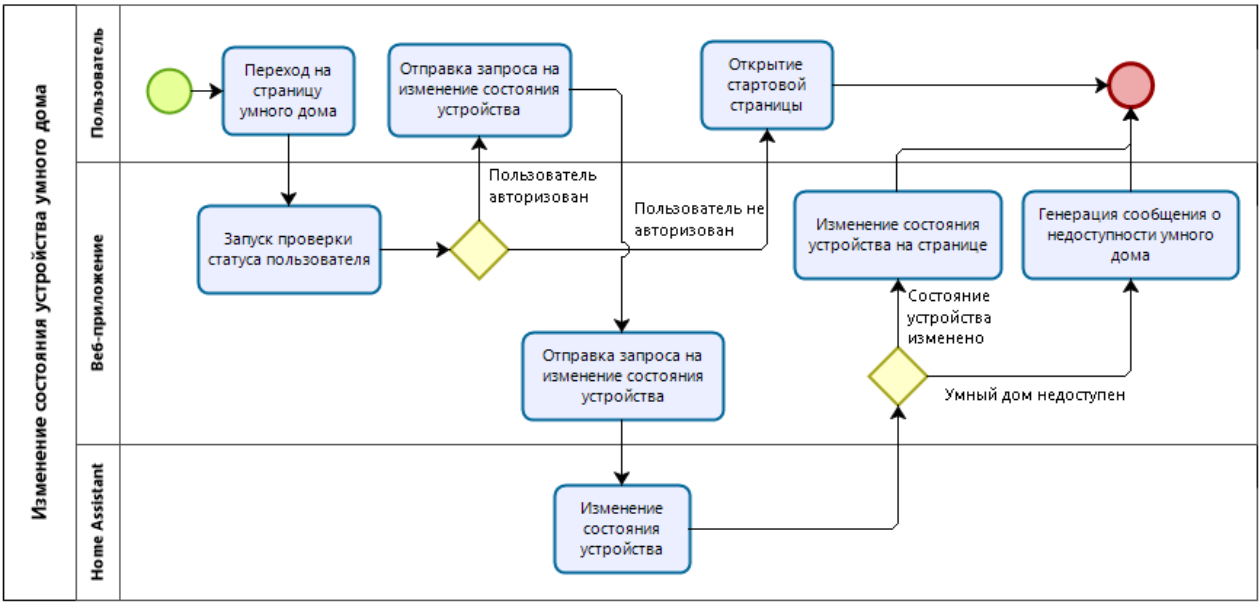


Рисунок 7 – Схема бизнес-процесса изменения состояния устройства умного дома

Далее рассмотрим бизнес-процессы владельца умного дома: добавление / удаление владельца или гостя для умного дома, редактирование описания умного дома, изменение названия устройств и просмотр истории состояний устройств.

Для того, чтобы добавить владельца или гостя для умного дома, владелец должен выбрать умный дом, ввести логин добавляемого пользователя и выбрать его роль. Если пользователь не найден, то система должна сообщить об этом и предложить исправить логин. На рисунке 8 представлен описанный бизнес-процесс.

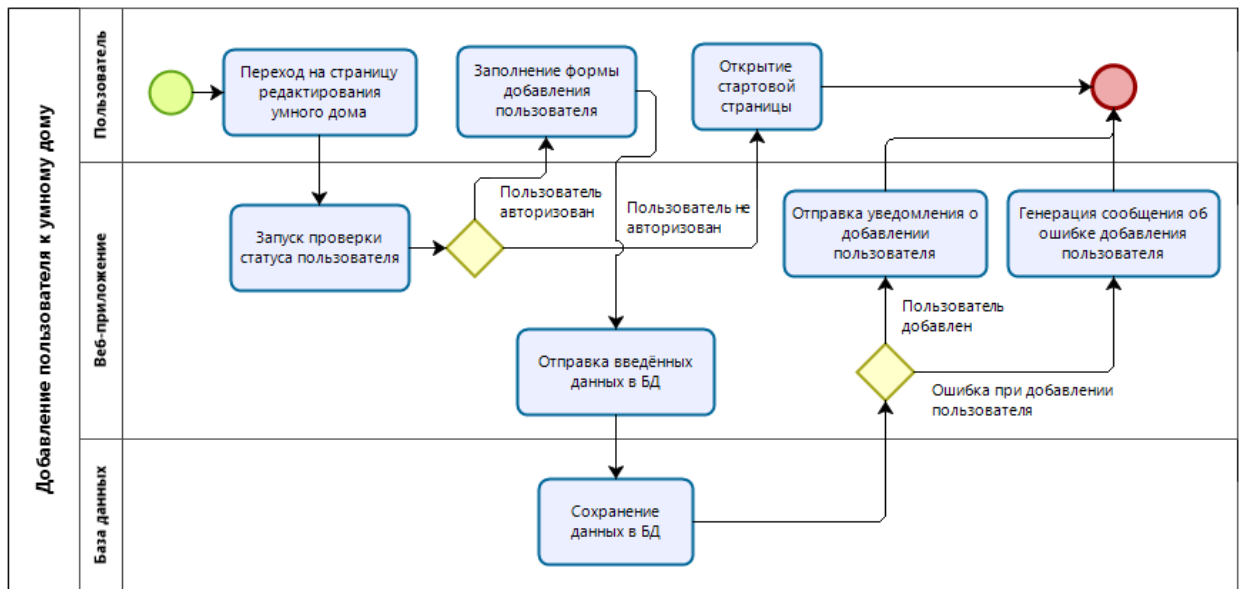


Рисунок 8 – Схема бизнес-процесса добавления пользователя к умному дому

Для удаления владельцев или гостей умного дома пользователь выбирает нужного пользователя из списка. Сам себя пользователь удалить не может. На рисунке 9 представлен описанный бизнес-процесс.

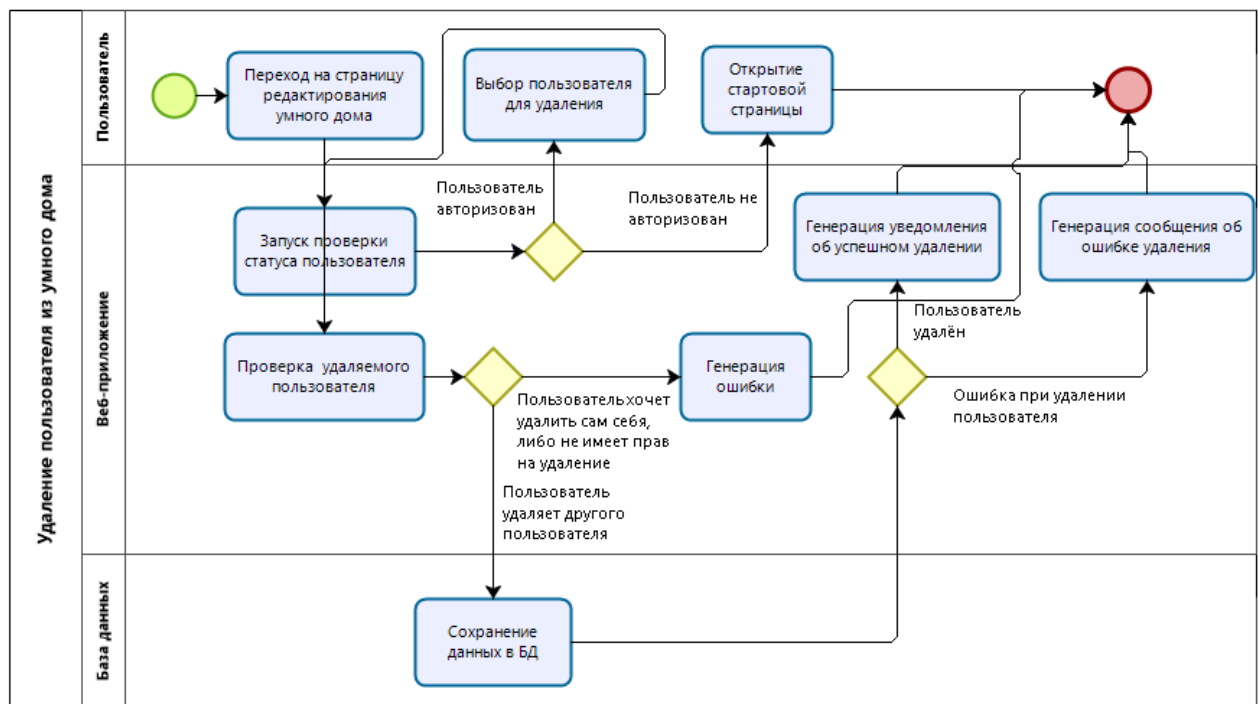


Рисунок 9 – Схема бизнес-процесса удаления пользователя из умного дома

Для редактирования описания умного дома пользователь вводит новое описание в соответствующее поле и нажимает кнопку сохранения. Пользователь с ролью гостя получит ошибку обновления описания. На рисунке 10 представлен описанный бизнес-процесс.

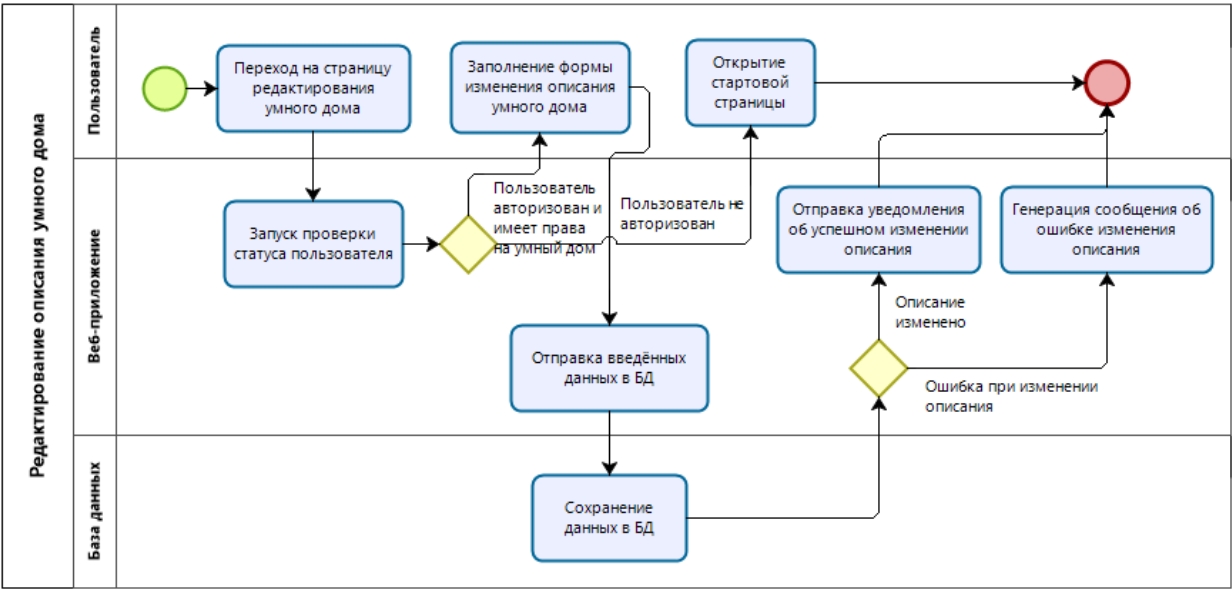


Рисунок 10 – Схема бизнес-процесса редактирования описания умного дома

Для просмотра истории состояний устройства пользователь нажимает на название устройства и система строит график с историей состояний выбранного устройства. Пользователь с ролью «гость» получит ошибку при нажатии на просмотр истории. На рисунке 11 представлен описанный бизнес-процесс.

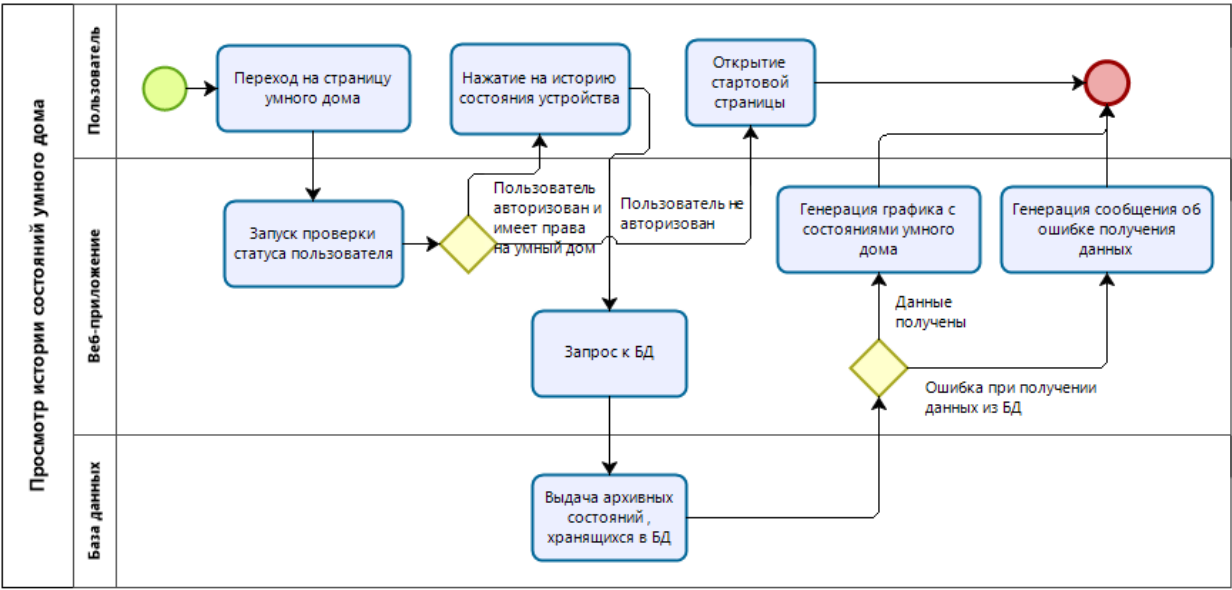


Рисунок 11 – Схема бизнес-процесса просмотра истории состояний устройства

Есть ещё один процесс, который запускается без участия пользователя – это мониторинг состояния устройств умного дома. В настройках задаётся интервал опроса умных домов, и система мониторинга с этим интервалом запускает функцию проверки новых состояний устройств.

1.4 Проектирование базы данных

1.4.1 Выделение основных сущностей

Необходимо проанализировать предметную область и выделить основные сущности, обладающие определенным набором свойств и аспектов поведения.

Анализ предметной области позволил выделить следующие сущности:

- 1) Пользователь (содержит учетные данные: логин, пароль, адрес электронной почты и т. п.),
- 2) Умный дом (содержит адрес и токен умного дома),
- 3) Устройства умного дома – отвечает за хранение данных устройства. Данная сущность должна быть универсальной для всех типов устройств, поэтому большинство полей не являются обязательными для заполнения,
- 4) Состояния устройств (ID устройства, единица измерения, состояние),
- 5) Умные дома пользователей – связь умных домов и пользователей. Также должно быть поле с ролью пользователя в текущем умном доме (владелец или гость),
- 6) Периодические задачи – содержит название задачи, код задачи в приложении, флаг – работоспособность задачи, интервал работы задачи. Данная сущность отвечает за работу периодических задач, она добавляется автоматически при использовании пакетов Celery,
- 7) Интервал периодических задач (единица измерения: секунды, минуты, часы, дни, недели, месяцы; количество) – отвечает за интервал запуска периодических задач. Добавляется автоматически при использовании пакетов Celery.

1.4.2 Разработка даталогической модели

Даталогическое конструирование заключается в переносе инфологической модели в схему базы данных в терминах выбранной системы управления базами данных (СУБД), то есть с помощью даталогической модели представляют сами данные (то, что будет сохранено на сервере в виде записей в таблицах) и связи

между данными. СУБД базируется на конкретной модели данных. Поскольку на этапе анализа была выбрана СУБД PostgreSQL, то проводится описание реляционной модели.

Реляционная модель данных – логическая модель данных. На текущий момент именно эта модель является стандартом почти всех современных коммерческих СУБД.

Реляционная модель данных может иметь следующую структуру:

- структурная,
- манипуляционная,
- целостная.

Структурная часть модели определяет, то, что единственной структурой данных является нормализованное n -арное отношение. Отношения удобно представлять при помощи таблиц, где каждая строка – это кортеж, а каждый столбец – это атрибут, определенный на некотором домене. Реляционная база данных представляет собой конечный набор таблиц.

Манипуляционная часть модели определяет два основных механизма манипулирования данными – реляционную алгебру и реляционное исчисление. Основная функция манипуляционной части реляционной модели состоит в том, чтобы обеспечить меру реляционности любого выбранного языка реляционных баз данных.

Язык можно назвать реляционным, если он имеет не меньшую выразительность и мощь, чем реляционное исчисление и реляционная алгебра.

Целостная часть модели определяет требования к целостности сущностей и целостности ссылок. Главное требование заключается в том, что любое отношение должно иметь первичный ключ. Требование ссылочной целостности или требование внешнего ключа заключается в том, что для каждого значения внешнего ключа, появляющегося в ссылающемся отношении (в отношении, на которое ведет ссылка) должен существовать кортеж с тем же значением первичного ключа, либо значение внешнего ключа должно быть

неопределенным (т.е. ни на что не указывать). На рисунке 12 представлена даталогическая модель разрабатываемой системы.

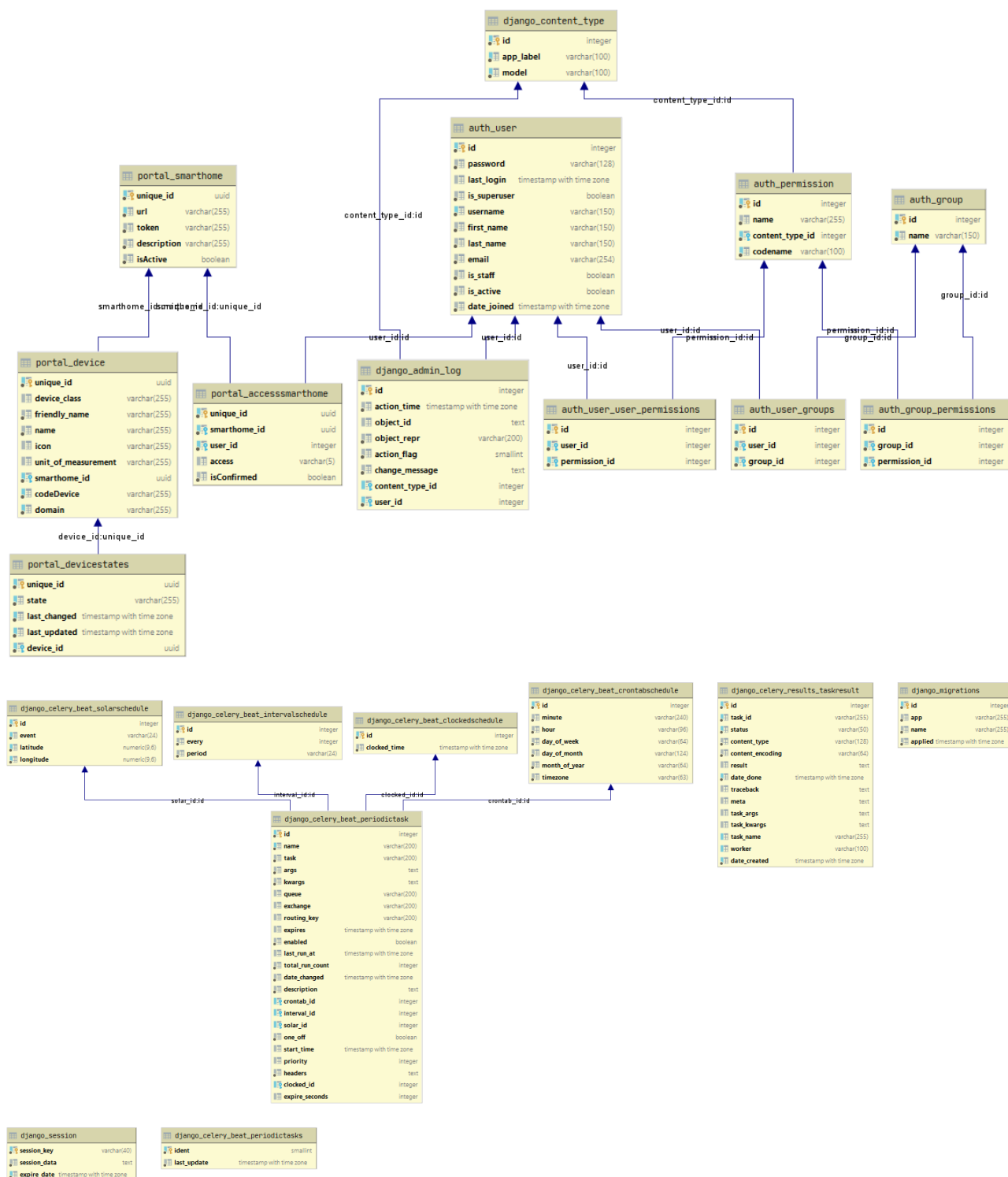


Рисунок 12 – Даталогическая модель разрабатываемой системы

Рассмотрим основные таблицы подробнее (остальные создаются автоматически или являются вспомогательными):

1) Таблица «Пользователи» (*auth_user*):

- id (первичный ключ)
- логин и пароль,
- фамилия и имя (*first_name* и *last_name*),
- email,
- дата последнего входа (поле заполняется автоматически при каждом входе в аккаунт),
- администратор / *is_superuser* – флаг, с помощью которого приложение проверяет принадлежность пользователя к группе администраторов (True, если является администратором; False – если не является администратором),
- аккаунт подтверждён (*is_active*): так как после регистрации пользователь должен подтвердить email, то при создании автоматически этот флаг принимает значение False. После подтверждения значение поля меняется на True и с помощью данного аккаунта можно работать с данными на сайте,

2) Таблица «Умные дома» (*smarthome*):

- *unique_id* (первичный ключ),
- url (адрес умного дома),
- token (токен доступа к умному дому),
- description (описание умного дома). Поле не является обязательным
- *is_active* (активный / неактивный умный дом),

3) Таблица «Умный дома пользователей» (*accesssmarthome*):

- *unique_id* (первичный ключ),
- *smarthome_id* (ID умного дома),
- *user_id* (ID пользователя),
- access (уровень доступа к умному дому). Имеет два значения: владелец (*owner*) и гость (*guest*),

4) Таблица «Устройства» (*device*):

- `unique_id` (первичный ключ),
- `smarthome_id` (ID умного дома),
- `device_class` (класс устройства). Поле не является обязательным,
- `friendly_name` (название устройства, заданное в HomeAssistant),
- `name` (название устройства, заданное пользователем),
- `icon` (иконка устройства). Поле не является обязательным,
- `unit_of_measurement` (мера измерения). Поле не является обязательным,
- `codeDevice` (код устройства из HomeAssistant)
- `domain` (тип устройства из HomeAssistant),

5) Таблицы `Django_celery_beat*` создаются автоматически при подключении данного пакета.

2 Проектирование структуры и компонентов программного продукта

2.1 Настройка Raspberry Pi

Для управления умным домом необходим умный дом, который развернут на Raspberry Pi. Рассмотрим, как происходит данная настройка.

Home Assistant можно установить несколькими способами: через развертывание docker образа, через установку специальной сборки для Raspberry Pi и через установку пакета в venv. Если Raspberry Pi будет использоваться только для умного дома, то самый оптимальный вариант – это установка специальной ОС, которая содержит Home Assistant. Если Raspberry Pi будет выполнять несколько функций, то самый оптимальный вариант – это установить Raspbian (система на основе Debian), который официально поддерживается разработчиками устройства, а на ней развернуть docker образ.

Если установка Home Assistant завершилась успешно, то по ссылке <http://IP-адрес:8123> будет доступен интерфейс Home Assistant.

Для использования умного дома в разрабатываемой системе нужно получить токен. Для этого нужно перейти по ссылке <http://IP-адрес:8123/api/> и скопировать оттуда токен доступа к умному дому.

Для удобства использования умного дома можно установить домен для умного дома, чтобы не использовать IP адрес. Но для этого нужно иметь «белый» IP или использовать сторонние средства (ngrok, сервисы роутера и т. п.).

Для дополнительной защиты в конфигурации Home Assistant можно настроить фильтр доступности умного дома, например, добавить IP адрес разрабатываемой системы в умный дом и тогда он не будет доступен нигде, кроме разрабатываемого приложения.

2.2 Разработка интерфейса пользователя

Интерфейс (англ. interface) – это определенная стандартами граница между взаимодействующими независимыми объектами.

Интерфейс задает как, будут взаимодействовать процедуры и характеристики объектов.

Пользовательский интерфейс (интерфейс пользователя) (англ. user interface) в информационных технологиях — это компоненты и элементы программы, влияющие на взаимодействие пользователя с программным продуктом; это совокупность методов, правил и программно-аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером.

Все интерфейсы условно можно разделить на три группы (расположены в историческом порядке):

- командные (текст-ориентированные) интерфейсы (используются в тех случаях, когда управление программой происходит при помощи определённого списка команд),
- смешанные (псевдографические) интерфейсы (расширение командного интерфейса, но начинают появляться графические кнопки),
- графические интерфейсы.

Первостепенной задачей проектирования интерфейса пользователя является не то, чтобы человек привыкал к разработанному интерфейсу, а в том, чтобы разработать систему взаимодействия двух партнеров: человека и аппаратно-программного комплекса, рационально использующих объект управления.

Существует два типа интерфейса: процедурно-ориентированный и объектно-ориентированный. Для разработки был выбран объектно-ориентированный, т. к. ранее был выбран объектно-ориентированный подход к разработке web-приложения.

Существует два вида интерфейса: графический и командный. Был выбран графический, т. к. этот вид наиболее современный и сайт не сможет работать в виде командной строки.

2.2.1 Выбор средств реализации интерфейса

Любое веб-приложение использует HTML для показа страниц в браузере. HTML является языком разметки для веб-страниц и дает браузеру необходимые инструкции о том, как отображать текст и другие элементы страницы на мониторе. Важно отметить, что на основе одного и того же HTML кода страница может выглядеть по-разному. Это происходит из-за различий в движках браузеров, а также в зависимости от размера. Поэтому, при кодировании это нужно помнить и использовать различные инструменты совместимости, например, это нужно использовать для старых версий Internet Explorer.

Язык HTML предоставляет только разметку, но необходим ещё и дизайн, поэтому дополнительно используется CSS – язык, описывающий внешний вид HTML страницы.

Для удобства функционирования было решено использовать динамический интерфейс с применением JS и jQuery. Это наиболее популярные и универсальные средства для реализации логики работы приложения со стороны пользователя.

JavaScript (JS) — это легкий, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык с функциями первого класса. В последнее время JS используется во множестве фреймворков, но изначально задумывался как язык сценариев веб-страниц.

jQuery — библиотека JavaScript, расширяющая возможности использования JS. С помощью jQuery легче описывается доступ к объектам DOM (для доступа используется простой синтаксис `$ («идентификатор или тег»)`), легче получать различные параметры элемента. Библиотека jQuery также предоставляет удобный API для работы с AJAX [4].

Кроме того, было решено максимально использовать AJAX (другое название — «асинхронный» JS) — подход, заключающийся в совместном использовании ряда существующих технологий, включая HTML или XHTML, каскадные таблицы стилей, JavaScript, DOM. Когда эти технологии объединены в модели AJAX, веб-приложения могут выполнять быстрые, постепенные

обновления пользовательского интерфейса без перезагрузки всей страницы браузера. Это делает приложение более быстрым и более отзывчивым на действия пользователя.

Конечно, можно ограничиться данными технологиями и самому создавать дизайн, адаптивность страниц к различным дисплеям и устройствам, но на это может уйти значительное время. Но существуют фреймворки, с помощью которых на дизайн уходит меньше времени. Самым популярным является Bootstrap. Он включает в себя HTML и CSS шаблоны оформления для веб-страниц, а также JS расширения [5].

2.2.2 Разработка форм интерфейса

Следующим этапом проектирования интерфейса является организация формы обмена информацией. Эта форма разрабатывается с учетом удобства её использования. В данном случае она должна быть динамической и адаптированной под различные разрешения экрана. Как известно, обмен информации в веб-приложениях выполняется в форме страниц.

Далее продемонстрирован процесс разработки главной страницы.

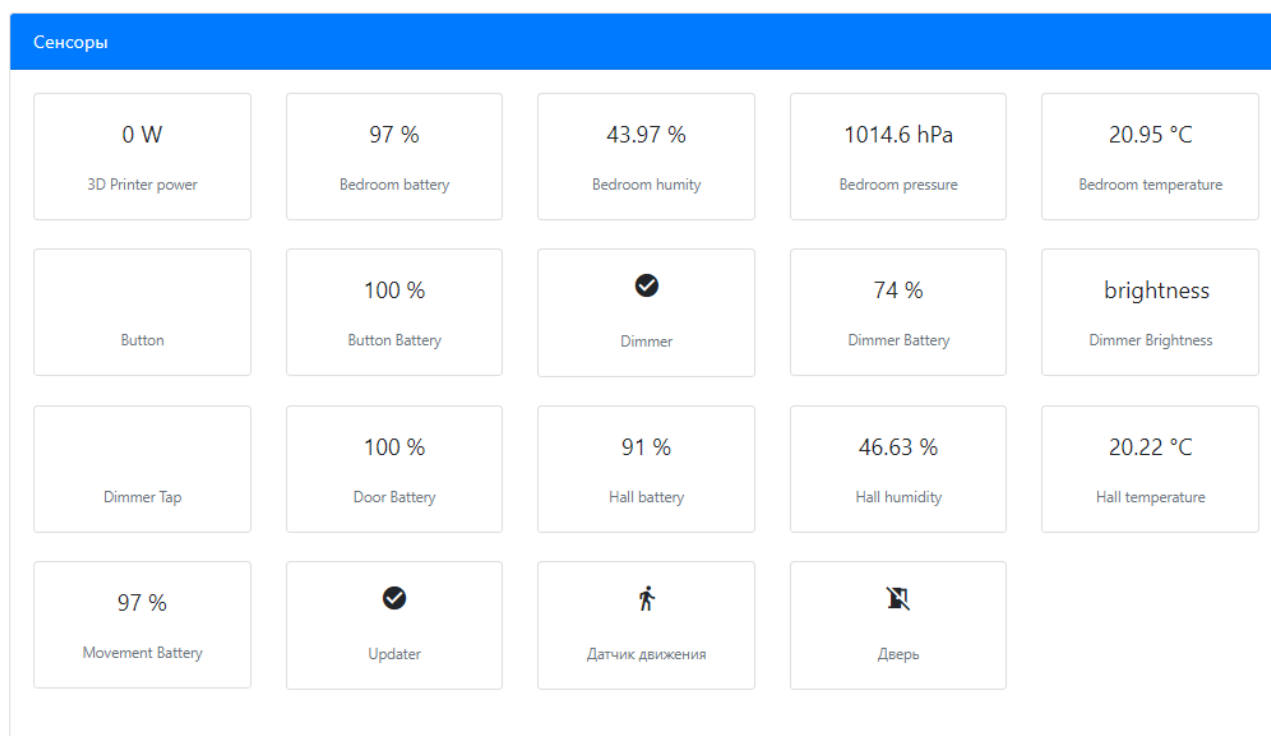
Для удобства использования портала все функции управления состоянием устройств умного дома были расположены на одной странице. Доступные типы устройств были разделены на три группы: датчики, освещение и переключатели.

Для переключения состояний переключателей и освещения нужно нажать на радиокнопку. Если умный дом будет доступен, то состояние устройства изменится без перезагрузки страницы.

Для просмотра истории состояний устройства нужно нажать на название устройства или нажать на блок датчика, после чего откроется модальное окно с графиком, содержащим архив состояний данного устройства.

В модальном окне можно изменить название устройства, для этого необходимо нажать на «Изменить», ввести новое название и нажать Enter.

Скриншот главной страницы представлен на рисунке 13, скриншот модального окна с состоянием устройства представлен на рисунке 14.



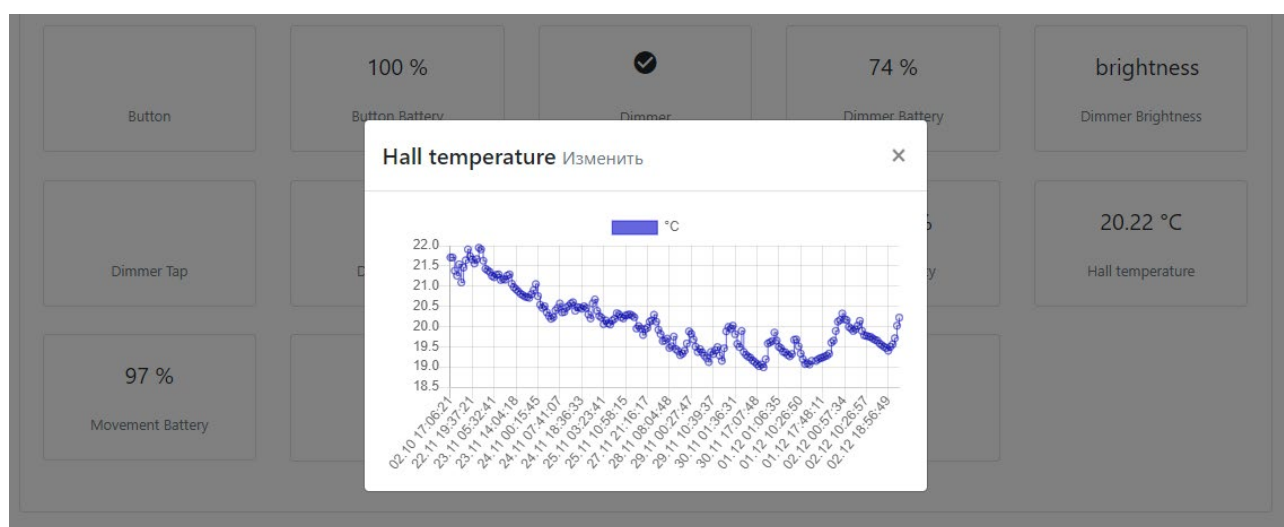
Освещение

☀	IKEA Light 1	☐
☀	IKEA Light 2	☐
☀	IKEA Light 3	☐
☀	IKEA Light 4	☐
☀	IKEA Light 5	☐

Выключатели

⚡	3D Printer	☐
⚡	PC	☐

Рисунок 13 – Скриншот разработанной страницы



2.3 Реализация программного продукта

В данном разделе будет описан процесс создания Django проекта.

Проект Django создаётся командой *django-admin startproject project*, где *project* - название проекта. После выполнения команды создаётся структура, изображённая на рисунке 15.

```
project/  
  manage.py  
  project/  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py
```

Рисунок 15 – Структура нового Django проекта

Рассмотрим, за что отвечает каждый файл:

- */project* – это внешняя папка, в которой хранится проект, её можно переименовать,
- *manage.py* – утилита для командной строки, с помощью которой возможно взаимодействие с проектом различными способами,
- внутренний каталог *project/* – это главное приложение данного проекта. Его имя – это имя пакета Python, которое используется для импорта чего-либо внутри всего проекта (например, *project.urls* для импорта путей),
- *project/__init__.py* – пустой файл, который сообщает Python, что этот каталог следует считать пакетом,
- *project/settings.py* – настройки для этого проекта Django,
- *project/urls.py* – в данном файле объявляются URL для всего проекта Django,
- *project/wsgi.py* – точка входа для WSGI-совместимых веб-серверов для обслуживания проекта.

В терминологии Django приложение – это web-приложение, которое отвечает за определённую часть сайта; проект представляет собой набор приложений для конкретного проекта; который обычно включает в себя

несколько приложений; приложение может использоваться в нескольких проектах.

Для создания основы проекта используется команда *startproject*, а для создания приложений в проекте используется команда *python manage.py startapp appl*, где *appl* – название приложения.

Разработанное приложение имеет структуру, показанную на рисунке 16.

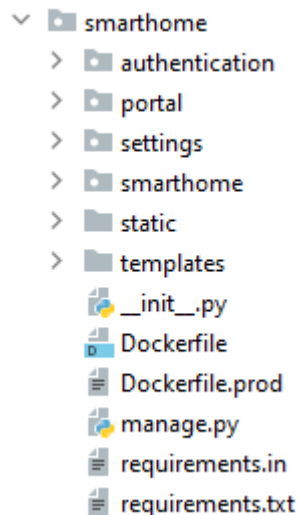


Рисунок 16 – Структура Django приложения

Рассмотрим, за что отвечает каждый проект в данной структуре:

- *authentication* – проект для авторизации, регистрации и изменения учётных данных,
- *portal* – проект, отвечающий за работу с умными домами: получения состояний устройств, изменение состояния устройств, просмотр историй состояний устройств и т. п.,
- *settings* – данный проект отвечает за настройку умных домов: добавление / удаление умных домов, добавление / удаление пользователей умных домов и т. п.,
- *smarthome* – основной проект приложения, в котором происходит объединение всех написанных проектов и настройка приложения,
- *static* – папка с css и js файлами,
- *templates* – в данной папке хранятся шаблоны html страниц,

Созданное приложение необходимо разместить на сервере. Для его запуска необходимо выполнить команду *python manage.py runserver* в командной строке,

но использовать такой запуск в production неверно, так как необходимо держать запущенным это окно.

Для полноценного запуска проекта используется gunicorn – промежуточный веб-сервер, который запускает Django проект и получает запросы от основного сервера – Nginx. Для запуска gunicorn был написан файл конфигурации, в котором описано местонахождение проекта, местонахождение Python интерпретатора.

Также на сервере была развёрнута СУБД PostgreSQL, в которой хранятся данные, которые использует Django проект.

Для работы периодических задач также было настроено расширение Celery и установлен RabbitMQ, который отвечает за обработку сообщений от Celery.

Для упрощения разработки и развёртывания приложения на сервере был использован Docker, в который были включены Django, RabbitMQ, Celery и Nginx.

На стороне пользователей должен быть развернут сервер с Home Assistant с «белым» IP или доменом.

На основе этих данных была разработана диаграмма размещения, изображённая на рисунке 17.

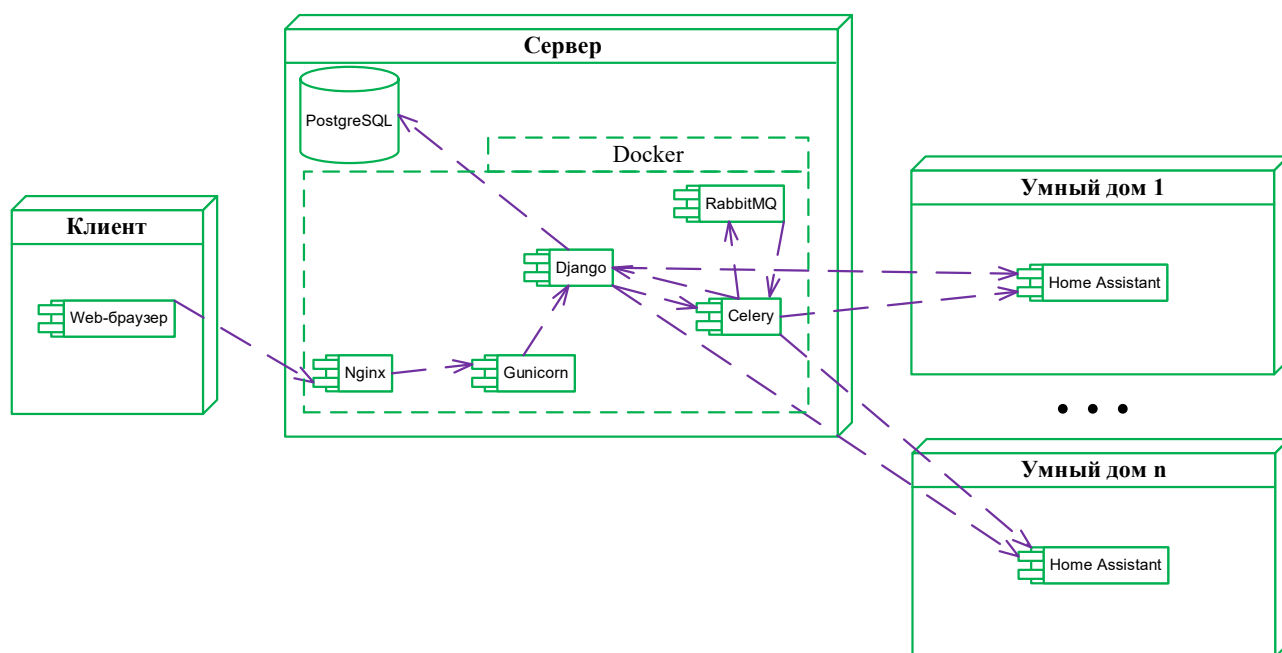


Рисунок 17 – Диаграмма размещения проекта

3 Выбор стратегии тестирования и разработка тестов

Так как в проекте есть интеграция с внешними подсистемами, то автотесты не совсем подходят для полного покрытия тестами. Если рассматривать разработанное приложение как API сервис, то для тестирования лучше использовать специализированное ПО, например, Postman. Для проведения тестирования в данном приложении создаётся коллекция запросов, а в каждом запросе пишутся тесты, который проверяют код ответа, пришедшие данные на соответствие шаблону и т. п. Данное тестирование проводилось перед каждым коммитом в репозиторий и, если возникали ошибки, то они оперативно исправлялись и тестирование запускалось вновь. То есть, при помощи Postman проводилось регрессионное тестирование и тестирование изменений и исправлений.

Для проведения UI тестирования можно использовать средство автоматизированного тестирования Selenium или Katalon Studio. Для проведения тестирования с помощью этих инструментов сначала записываются действия пользователей через специальные программы вышеописанных средств, а затем полученные скрипты дорабатываются для универсальности (например, замена явно описанного токена авторизации на автоматически получаемый при каждом запуске теста).

Если разрабатываемое приложение планируется выводить в промышленную эксплуатацию с большим числом пользователей, то необходимо нагрузочное тестирование. Но так как таких планов нет, то нагрузочное тестирование не проводилось.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы была создана информационная система управления умным домом.

При выполнении курсовой работы был проведён анализ предметной области, на основании которого были выделены основные сущности, на базе которых была получена схема базы данных. Также был проведён выбор средств разработки программной системы.

При помощи выбранных инструментов была успешно реализована информационная система.

Полученную систему можно доработать. Вот лишь несколько идей:

- 1) добавить гибкую систему доступа к устройствам различным пользователям (например, уменьшить количество доступных устройств или ограничить изменение состояния устройства),
- 2) добавить функционал из Home Assistant, который доступен для настроек только на Raspberry Pi (например, добавление устройств через редактирование конфигурации или добавление сценариев автоматизации),
- 3) улучшить интерфейс (добавить интерфейс уведомлений, изменить иконки, вид графиков и т. п.),
- 4) добавить управление умным домом через чат-бот Telegram.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Home Assistant [Электронный ресурс] – URL: <https://www.home-assistant.io/> (дата обращения 10.10.2020)
- 2 REST API | Home Assistant Developer Docs [Электронный ресурс] – URL: <https://developers.home-assistant.io/docs/api/rest> (дата обращения 10.10.2020)
- 3 Django [Электронный ресурс] – URL: <https://www.djangoproject.com/> (дата обращения 10.10.2020).
- 4 JQuery [Электронный ресурс] – URL: <https://jquery.com/> (дата обращения 10.10.2020).
- 5 Bootstrap [Электронный ресурс] – URL: <https://getbootstrap.com/> (дата обращения 10.10.2020).
- 6 Celery [Электронный ресурс] – URL: <http://www.celeryproject.org/> (дата обращения 10.10.2020).
- 7 RabbitMQ [Электронный ресурс] – URL: <https://www.rabbitmq.com/> (дата обращения 10.10.2020).