**Санкт-Петербургский государственный электротехнический университет**

**“ЛЭТИ” им. В. И. Ульянова (Ленина)**

**(СПбГЭТУ “ЛЭТИ”)**

|  |
| --- |
|  |

**Направление подготовки:** 09.03.01 “Информатика и вычислительная техника”

**Профиль:** “Организация и программирование вычислительных и информационных систем”

**Факультет компьютерных технологий и информатики**

**Кафедра вычислительной техники**

*К защите допустить:*

**Заведующий кафедрой**

д. т. н., профессор\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. С.Куприянов

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

**Тема: “Система сбора данных о загрузке апартаментов в жилых объектах”**

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Змеу

Руководитель

к. т. н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Родионов

Консультант по экономическому

обоснованию к. э. н. \_\_\_\_\_\_\_\_\_\_\_\_\_ М. А. Косухина

Консультант от кафедры

к. т. н., доцент, с. н. с. \_\_\_\_\_\_\_\_\_\_\_\_\_ И. С. Зуев

Санкт-Петербург

2023 г.

**Санкт-Петербургский государственный электротехнический университет**

**“ЛЭТИ” им. В. И. Ульянова (Ленина)**

**(СПбГЭТУ “ЛЭТИ”)**

|  |  |
| --- | --- |
| Направление Информатика и вычислительная техника  Профиль Организация и программирование вычислительных и информационных систем  Факультет компьютерных технологий  и информатики  Кафедра вычислительной техники | **УТВЕРЖДАЮ**  Заведующий кафедрой ВТ  д. т. н., профессор  (М. С. Куприянов)  “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Змеу Станислав Владимирович |  | Группа № | 9306 |

|  |  |
| --- | --- |
| **1. Тема** | Система сбора данных о загрузке апартаментов в жилых объектах |
| *(утверждена приказом № \_\_\_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_)* | |

Место выполнения ВКР: Кафедра ВТ

**2. Объект и предмет исследования**

Объектом исследования являются системы сбора и хранения данных.

Предметом исследования являются система сбора данных в отеле.

**3. Цель**

Создание системы сбора данных о загрузке апартаментов в жилых объектах, которая позволит следить за загруженностью апартаментов в календарные дни. Это поможет сделать необходимые выводы при распределении нагрузке на сотрудников, изменении цен или проведении различного рода ремонтных работ.

Разрабатываемый программный продукт предназначен для использования отелями в качестве основного инструмента сбора данных

**4. Исходные данные**

Исходными данными для разработки являются русскоязычные и

англоязычные статьи и видео в сети Интернет, документация к средствам разработки, форумы разработчиков, книги.

**5. Содержание**

* Обзор и постановка задачи работы.
* Проектирование системы сбора данных.
* Разработка системы сбора данных.
* Результаты работы.

**6. Технические требования**

Разрабатываемая система должна позволять:

* Дистанционный сбор данных срабатывания ДР в текущие моменты времени.
* Хранение данных в базе данных (БД) фактов срабатывания ДР в заданные периоды
* Представление данных из БД по запросам с АРМ администратора.
* Хранение данных в БД в течение 3-х месяцев.
* Выдачу сигнала "движение" при заселении апартамента в канал передачи данных в локальной сети на роутер сети интернет в IР-адрес сервера.

**7. Дополнительные разделы**

Разработка и стандартизация программных средств.

**8. Результаты**

Пояснительная записка, реферат, аннотация, презентация. Программная документация содержит руководство пользователя и примеры работы и обзор системы.

|  |  |  |
| --- | --- | --- |
| Дата выдачи задания |  | Дата представления ВКР к защите |
| «28» февраля 2023 г. |  | «14» июня 2023 г. |

|  |  |  |
| --- | --- | --- |
| Руководитель  к. т. н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | С. В. Родионов |
| Студент | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | С. В. Змеу |

**Санкт-Петербургский государственный электротехнический университет**

**“ЛЭТИ” им. В. И. Ульянова (Ленина)**

**(СПбГЭТУ “ЛЭТИ”)**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Направление (09.03.01 «Информатика и вычислительная техника»)  Профиль («Организация и программирование вычислительных и информационных систем»)  Факультет компьютерных технологий  и информатики  Кафедра вычислительной техники | **УТВЕРЖДАЮ**  Заведующий кафедрой ВТ  д. т. н., профессор  (М. С. Куприянов)  “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 201\_\_г. |

**КАЛЕНДАРНЫЙ ПЛАН**

**выполнения выпускной квалификационной работы**

|  |  |
| --- | --- |
| Тема | **Система сбора данных о загрузке апартаментов в жилых объектах** |
|  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | С. В. Змеу |  | Группа № | **9306** |

|  |  |  |
| --- | --- | --- |
| № этапа | Наименование работ | Срок  выполнения |
| 1 | Подробное изучение выбранной темы работы | 05.04–10.04 |
| 2 | Составление плана проведения работы | 10.04–23.04 |
| 3 | Написание теоретической части работы | 23.04–03.05 |
| 4 | Реализация практической части | 03.05–20.05 |
| 5 | Создание подробного описания практической части | 22.05–28.05 |
| 6 | Формирование пояснительной записки | 28.05–02.06 |
| 7 | Проведение предварительного обзора работы | 04.06–12.06 |
| 8 | Представление работы к защите | 14.06.2023 |

|  |  |  |
| --- | --- | --- |
| Руководитель  к. т. н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | С. В. Родионов |
| Студент | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | С. В. Змеу |

РЕФЕРАТ

Пояснительная записка содержит: 71 стр., 32 рис., 21 ист., 2 прил.

Цель работы: Создание системы сбора данных о загрузке апартаментов в жилых объектах.

Данная выпускная квалификационная работа направлена на разработку системы сбора данных о загрузке апартаментов в жилых объектах. Были рассмотрены и выбраны соответствующие инструменты для создания такой системы. Выделены ключевые особенности будущей платформы, которые учитывают специфику целевой аудитории и особенности жилого объекта. С использованием выбранных инструментов был создан пример системы сбора данных, который демонстрирует её функциональность в реальном времени.

В результате работы была создана система, которая может без труда интегрироваться и использоваться как рабочим персоналом, так и гостями

ABSTRACT

This final qualification work is aimed at developing a system for collecting data on the loading of apartments in residential facilities. The appropriate tools for creating such a system were considered and selected. The key features of the future platform are highlighted, which take into account the specifics of the target audience and the features of the residential facility. Using the selected tools, an example of a data collection system was created that demonstrates its functionality in real time.

As a result of the work, a system was created that can be easily integrated and used by both working staff and guests.

СОДЕРЖАНИЕ

[ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 9](#_Toc136875660)

[ВВЕДЕНИЕ 10](#_Toc136875661)

[1 Обзор задачи создания системы сбора данных 11](#_Toc136875662)

[1.1 Постановка задачи и обоснование актуальности темы работы 11](#_Toc136875663)

[1.2 Обзор и выбор методов сбора данных 12](#_Toc136875664)

[1.3 Сравнение с аналогичными система сбора данных 13](#_Toc136875665)

[2 Проектирование системы сбора данных 15](#_Toc136875666)

[2.1 Проектирование базы данных и структуры сайта 15](#_Toc136875667)

[2.2 Обзор и выбор оборудования для сбора данных 20](#_Toc136875668)

[2.3 Описание выбранного оборудования 23](#_Toc136875669)

[3 Разработка системы сбора данных 26](#_Toc136875670)

[3.1 Разработка серверной части сайта отеля 26](#_Toc136875671)

[3.2 Разработка выбранного метода сбора данных 33](#_Toc136875672)

[4. Результаты реализации системы 43](#_Toc136875673)

[4.1 Демонстрация серверной части сайта отеля 43](#_Toc136875674)

[4.2 Демонстрация системы сбора данных 50](#_Toc136875675)

[4.3 Описание планов по дальнейшей разработке и внедрению реализованной системы 53](#_Toc136875676)

[5. Разработка и стандартизация программных средств 54](#_Toc136875677)

[5.1 Планирование работ проекта ПС 55](#_Toc136875678)

[5.2 Определение затрат на выполнение и внедрение проекта ПС 56](#_Toc136875679)

[5.2.1 Расчет себестоимости продукта 56](#_Toc136875680)

[5.2.2 Расчет цены предлагаемого продукта 60](#_Toc136875681)

[5.3 Определение классификационного кода разрабатываемого ПС 60](#_Toc136875682)

[5.4 Определение списка стандартов для ВКР 60](#_Toc136875683)

[ЗАКЛЮЧЕНИЕ 62](#_Toc136875684)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 63](#_Toc136875685)

[ПРИЛОЖЕНИЕ А Спецификация программных модулей 64](#_Toc136875686)

[ПРИЛОЖЕНИЕ Б Исходный код программы 65](#_Toc136875687)

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

DC - Docker compose. Иструмент для определения и запуска составных приложений Docker, которые могут состоять из нескольких контейнеров.

ПО – программное обеспечение.

ORM - Object–relational mapping. Технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных»

БД – база данных

URL - Uniform Resource Locator. Ссылка

АРМ – автоматизированное рабочее место администратора

ВВЕДЕНИЕ

Современный рынок недвижимости заставляет управляющие компании постоянно улучшать и оптимизировать свою деятельность. Таким образом, системы сбора данных становятся одним из наиболее эффективных инструментов для управления жилыми объектами.

В данном проекте мы рассмотрим основные элементы системы сбора данных, от отслеживания количества жителей до мониторинга использования коммунальных услуг. Мы также рассмотрим интеграцию системы с другими технологиями и как это может улучшить эффективность управления жилыми объектами.

Данный дипломный проект направлен на разработку системы сбора данных о загрузке апартаментов жилых объектов. Она позволит управляющим компаниям получить более точную информацию о количестве жителей и их потребностях

Цель работы: создание системы сбора данных о загрузке апартаментов жилых объектов.

Объект исследования: Системы сбора и хранения данных.

Предмет исследования: Система сбора данных в отеле.

Для достижения основной цели требуется выявить ключевые особенности системы, выбрать программное обеспечение для разработки, с помощью выбранных средств разработать систему, соответствующую выявленным требованиям.

В первом разделе проведена постановка задачи и обоснована актуальность работы.

Во втором разделе описано проектирование базы данных для системы и проектирование архитектуры серверной части сайта, с помощью которого системы будет тестироваться.

В третьем разделе представлена реализация системы.

В четвертом разделе представлен результат реализации системы.

# Обзор задачи создания системы сбора данных

В этом разделе производится подробный обзор задачи и обоснование важности данной работы.

Рассматриваются различные существующие методы сбора данных о загрузке апартаментов, анализируются их особенности, преимущества и недостатки и приводятся реальные примеры существующих систем.

Так же проводится описание инструментов для создания самой системы сбора данных и, так же, серверной части сайта жилого комплекса, в которую и будет интегрироваться система.

## Постановка задачи и обоснование актуальности темы работы

Главной задача системы сбора данных заключается в создании эффективного инструмента для сбора и хранения информации о нагрузке на жилые апартаменты в определенные календарные дни. Это необходимо для того, чтобы отслеживать спрос на определенные апартаменты в целом, отслеживать более популярные апартаменты в праздники или выходные, эффективно распределять нагрузку на персонал.

Такая система необходима в целом для оптимизации управления имуществом и повышения качества обслуживания жильцов, например, для планирования ремонтных работ в каких-либо комнатах, или изменения ценовой политики комнаты в связи с её либо повышенным, либо пониженным спросом и т.д.

Актуальность данной работы объясняется растущим спросом на жилье, особенно в крупных городах, что создает потребность в более эффективном использовании доступного жилого пространства и оптимизации загрузки апартаментов. В современном мире системы сбора данных и аналитики становятся все более популярными среди компаний, занимающихся строительством и управлением недвижимостью, так как они стремятся повысить эффективность своих процессов управления.

## Обзор и выбор методов сбора данных

В настоящее время существует множество различных способов сбора и хранения данных о загрузке апартаментов. Ниже приведены описания нескольких таких методов [13].

1. Метод ручного сбора данных [13]. В данном подходе информация о загрузке апартаментов собирается путем ручной записи количества проживающих в каждом помещении, либо посредством опроса жильцов. Этот способ является простым и экономичным, однако, он обладает недостаточной точностью и неэффективен при работе с большими объемами данных.
2. Метод датчиков веса [14]. Они позволяют точно отслеживать количество людей, находящихся в каждом помещении. Этот метод обладает повышенной точностью, однако использование таких датчиков стоит достаточно дорого. Сами датчики стоят не мало, так же для внедрения этих датчиков требуется сложная и дорогостоящая установка. Так же существенный минус этих датчиков ощущается при возникновении каких-либо ремонтных работ пола. Нужно более профессиональные и дорогостоящие специалисты, которые не повредят датчики при ремонте.
3. Метод электронных ключей доступа [15]. В данном подходе каждому жильцу предоставляется индивидуальный электронный ключ доступа, который регистрирует их приход и уход из апартаментов. Это позволяет собирать информацию о времени пребывания жильцов в помещениях и оценивать уровень загрузки. У данного метода уже не такие существенные минусы. В первую очередь это установка специальных электронных замков, которая хоть и стоят дешевле датчиков веса, но все же обойдутся не дешево. Та же необходимо качественное ПО для правильной обработки сигналов.
4. Метод датчиков движения [16]. Для отслеживания количества людей внутри апартаментов можно установить датчики движения. Обычно такие датчики применяются в системах умного дома или системах безопасности, однако они также могут использоваться для сбора данных о загрузке апартаментов. В зависимости от оборудования цена установки такой системы может быть различной. Стоит учитывать такую же необходимость хорошего ПО, как и в случае с методом электронных ключей доступа. Еще один существенный минус данного метода – ложные срабатывания.

В нашей работе будет спроектирован и реализован метод сбора данных с помощью датчиков движения.

## Сравнение с аналогичными система сбора данных

В настоящее время уже есть проекты, в которых применяются системы, основанные на использовании датчиков движения для сбора данных. Ниже приведены примеры таких проектов:

1. People Counter [17] – это открытая система, которая использует датчики движения для регистрации количества людей, проходящих через определенную зону. Она находит применение в различных общественных местах, включая магазины, с целью контроля потока посетителей.
2. Smart Building [18] – система основана на использовании датчиков движения для сбора информации о том, как различные помещения в здании используются. Ее применение позволяет оптимизировать энергопотребление, управлять распределением ресурсов и может быть полезной в коммерческих зданиях или офисах.
3. PointGrab [19] – система, которая использует интеллектуальные датчики для отслеживания количества присутствующих людей в квартире, а также для оптимизации энергопотребления и снижения затрат на электроэнергию.

Можно заметить, что датчики движения для сбора данных – очень популярная и широко используемая технология. Она помогает людям в контроле за помещениями и оптимизации расхода используемых ресурсов.

# Проектирование системы сбора данных

В разделе 1 была подробно поставлена задача и описана её важность. В этом разделе будет полностью спроектирована система сбора данных. При проектировке будут сделаны следующие шаги:

1. Выбор лучшей БД для системы и её проектирование
2. Выбор инструментов для создания серверной части сайта отеля
3. Создание архитектуры сайта
4. Выбор оборудования для сбора данных

Все вышеперечисленное играет существенную роль в обеспечении функциональности, эффективности и применимости разрабатываемой системы. Они не только удовлетворяют текущие потребности и требования, но и учитывают возможности ее расширения и улучшения в будущем, чтобы система могла приспосабливаться к изменяющимся условиям и пользовательским потребностям. Это гарантирует долговечность, гибкость и потенциал роста и развития системы.

## Проектирование базы данных и структуры сайта

Для более реалистичной проверки системы сбора данных был выбран отель в качестве жилого объекта. Однако, для начала необходимо выбрать инструменты для создания серверной части сайта.

Было принято решение использовать Python [5] в качестве языка программирования, поскольку он является универсальным и мощным инструментом программирования. Django [6], в свою очередь, был выбран в качестве фреймворка для разработки всей функциональной части сайта, потому что он обладает широкими возможностями и позволяет ускорить процесс разработки. Наконец, Docker[1, 2] будет использован для упрощения процесса развертывания, грамотного объединения всех частей приложения и управления приложением.

Docker - это современный инструмент, который позволяет легко разворачивать и управлять приложениями в разных средах. Он позволяет упаковывать приложение со всеми зависимостями в один образ, который быстро запускается на любом хосте, где поддерживается Docker. Это существенно экономит время и упрощает настройку и установку приложений в новой среде. Таким образом, разработчики могут сосредоточиться на написании кода, не тратя время на ручное управление инфраструктурой.

Docker также позволяет создавать контейнеры с различными инструментами и приложениями, которые масштабируются и быстро обслуживаются с помощью Docker Compose [20]. Это позволяет создавать отдельные контейнеры, каждый из которых содержит свои инструменты и приложения в рамках одного приложения.

Благодаря Docker, разработчики могут быть уверены в совместимости их приложения на всех этапах разработки и производства. Он также упрощает процесс тестирования и отладки приложений.

При разработке веб-приложений так же важно учесть, что хранение данных играет решающую роль в их функциональности и эффективности. Использование базы данных позволяет веб-приложению эффективно управлять пользователями, хранить информацию и осуществлять операции над данными, включая фильтрацию, сортировку и аналитику. При правильном проектировании и оптимизации базы данных возможно достичь высокой реактивности и быстрого времени отклика веб-приложения, что имеет существенное значение для удовлетворения потребностей пользователей. Поэтому важно правильно выбрать и спроектировать БД.

Существуют следующие типы баз данных [3]:

1. Реляционные базы данных [3]: Такие СУБД представляют собой тип баз данных, использующий таблицы с установленными между ними связями для хранения и управления структурированными данными. Такой вид баз данных нашел широкое применение в управлении данных большого объема, таких как информация о товарах, клиентах или финансовых транзакциях. Например, PostgreSQL, MySQL и Oracle.
2. Нереляционные базы данных [3]: В отличие от реляционных не используют таблицы и связи между ними, вместо этого они оперируют документами, графами или ключ-значение парами. В качестве примеров нереляционных баз данных можно привести такие системы, как MongoDB, Redis и Cassandra.
3. Иерархические базы данных [3]: Иерархические базы данных организуют данные в виде древовидной структуры, состоящей из родительских и дочерних элементов. Данный вид баз данных нашел применение в организации и хранении большого объема иерархически связанных данных в информационных системах.
4. Сетевые базы данных [3]: Сетевые базы данных, похожие на иерархические, организуют данные в иерархическую структуру, но в отличие от последних, объекты могут иметь несколько родителей в связях "многие-ко-многим". Такой тип баз данных широко использовался в системах управления производством или медицинских системах.

Это только небольшой список типов баз данных, которые могут использоваться в различных проектах, в зависимости от задач и требований.

Для создания системы сбора данных о загрузке апартаментов очень хорошо подойдет реляционная база данных, такая как Postgresql [4]. Она обладает широкими возможностями и подходит для сложных проектов, а также обеспечивает надежное и эффективное хранение большого количества данных, что может быть особенно важно для системы сбора данных.

Как только все необходимые инструменты были выбраны и готовы к использованию, можно приступить к проектированию структуры сайта. Так как проект будет создаваться на Django, то необходимо сразу разделить сайт на отдельные Django-приложения. Разделение на приложения необходимо для правильной и удобной разработки сайта. В каждом таком приложении должен быть реализован функционал, соответствующий приложению. Главное, чтобы в одном приложении было не слишком много такого функционала, иначе такое приложение придется делить на несколько более мелких. Теперь необходимо определиться с общим функционалом. Сайт отеля должен поддерживать следующие функции:

1. Регистрация на сайте
2. Вход в аккаунт
3. Выход из аккаунта
4. Профиль пользователя на сайте
   * Просмотр и отмена текущих бронирований
   * Просмотр и изменение данных
5. Бронирование номера
6. Просмотр всех возможных категорий номеров
7. Просмотр детальной информации о каждой категории комнаты

Пункты 1-4 можно отнести к приложению аккаунта пользователя, назовем это приложение «userlogin». Это достаточно универсальное название для подобного приложения.

Пункты 5-7 можно отнести к приложению самого отеля, назовем это приложение «hotel». Такого простого названия будет достаточно, так как при дальнейшем добавлении на сайт таких разделов как например «Вакансии», «О нас» и т.д. это можно и нужно будет отнести к другому приложению. Название у такого приложения может быть, например, «info\_hotel».

Когда структура сайта готова, необходимо создать структуры базы данных, которая отвечала бы за эффективное хранение и управление всей информацией. Для этого необходимо создать несколько таблиц с определенными колонками:

1. Комната
   * Категорий комнаты
   * Номер комнаты
   * Количество людей на комнату
   * Количество кроватей на комнату
2. Бронирование
   * Имя
   * Фамилия
   * Почта
   * Номер телефона
   * Комната (ссылка на запись таблицы 1)
   * Дата заезда
   * Дата выезда
3. Пользователь
   * Имя
   * Фамилия
   * Почта
   * Номер телефона
   * Паспорт
   * Является ли работником
   * Является ли разработчиком

На рисунке 1 можно увидеть диаграмму базы данных, на которой показана сохраненная структура таблиц, описанных выше

Информация о существующих комнатах, клиентах и забронированных номерах будет храниться в таблицах, согласно созданной структуре. Система оптимизирована таким образом, чтобы не содержать лишних столбцов, обеспечивая легкий и быстрый доступ к данным. С помощью данной структуры будет значительно упрощен процесс хранения и отправки данных, что увеличит эффективность работы.

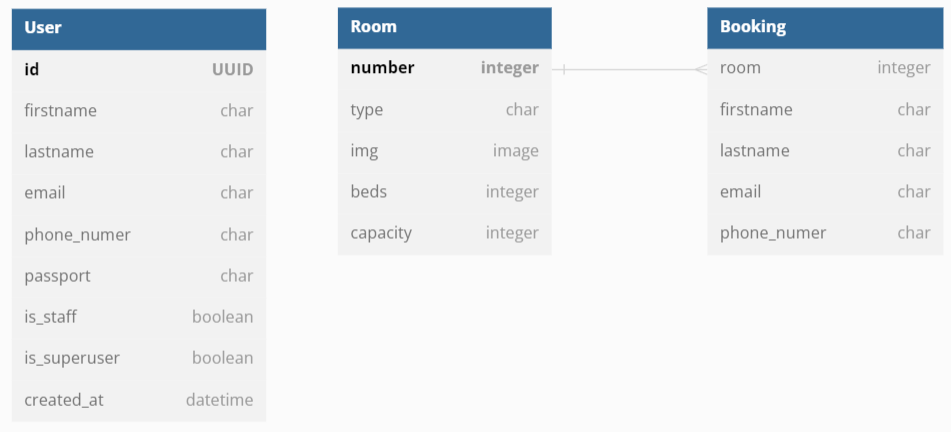


Рисунок 1 – Структура базы данных

## Обзор и выбор оборудования для сбора данных

В данном подразделе будет произведен обзор и осуществлен выбор оборудования для сбора данных о загрузке жилых апартаментов.

Качественное и точное сбор данных является важным аспектом для эффективного управления жилыми объектами и оптимизации их использования. В связи с этим, выбор подходящего оборудования становится неотъемлемой частью разработки системы сбора данных. Будут рассмотрены различные виды оборудования, его характеристики и преимущества, а также приведены рекомендации по выбору наиболее подходящих решений для достижения поставленных целей.

Выбор датчика движения - это первоочередная задача. От выбора датчика зависит качество сбора информации, что напрямую влияет на результаты анализа загрузки жилых помещений. Учитывая это, выбор датчика должен быть обоснован и основываться на анализе его характеристик и возможностей. Существуют следующие разновидности датчиков движения [9]:

1. Инфракрасные датчики [16]: Реагируют на изменение инфракрасного излучения, которое испускается объектами, включая людей, при их движении.
2. Микроволновые датчики [16]: Они используют микроволновые радиоволны для обнаружения движущихся объектов. Эти датчики могут проникать через стены и другие непрозрачные материалы и обеспечивают широкий угол обнаружения. Однако они могут быть более чувствительны к воздействию окружающей среды, такой как ветер или движение растений.
3. Ультразвуковые датчики [16]: Эти датчики излучают ультразвуковые звуковые волны и измеряют время, которое требуется для отражения волны от движущихся объектов. Они обладают высокой точностью и могут работать в широком диапазоне температур. Однако они могут быть более чувствительны к шумам и эхо в помещении.
4. Лазерные датчики [9]: Эти датчики используют лазерные лучи и фотодиоды для обнаружения движения объектов. Они обладают высокой точностью и могут обнаруживать движение на больших расстояниях. Однако они могут быть более дорогими и сложными в установке и настройке.
5. Капаситивные датчики [9]: Эти датчики обнаруживают изменение емкости вокруг себя при приближении объектов. Они широко применяются в сенсорных системах и обладают высокой чувствительностью. Однако они могут быть менее эффективными в обнаружении движения на больших расстояниях.

При выборе датчика движения для сбора данных в отеле, мы приняли решение остановиться на инфракрасном датчике, а именно на пироэлектрическом ИК-датчике движения. Этот выбор был обусловлен рядом преимуществ, которые предлагает данный тип датчика:

1. Дальность обнаружения [21]: Инфракрасные датчики обладают достаточной дальностью обнаружения, что позволяет охватывать большую площадь внутри помещения. Они могут обнаруживать движение на расстоянии от нескольких метров до десятков метров, в зависимости от конкретной модели датчика.
2. Устойчивость к помехам [21]: Пироэлектрические ИК-датчики имеют хорошую устойчивость к помехам, таким как изменения освещенности и шумы, что позволяет им надежно функционировать в различных условиях окружающей среды в отеле.
3. Распространенность и доступность [21]: Пироэлектрические ИК-датчики являются одним из наиболее распространенных типов датчиков движения, и их можно легко найти и приобрести на рынке. Это обеспечивает удобство и доступность для интеграции в системы сбора данных в отелях.
4. Простота установки и использования [21]: Пироэлектрические ИК-датчики обычно имеют простую структуру и легко устанавливаются. Они не требуют сложной настройки и обслуживания, что делает их удобными для использования в отеле.
5. Стоимость [21]: Пироэлектрические ИК-датчики предлагают хорошее соотношение цены и качества. Они являются более доступными и экономически эффективными по сравнению с некоторыми другими типами датчиков движения.

Датчик обнаружения движения предоставляет сигнал о возникновении движения в окружающей среде, однако этот сигнал требует дальнейшей обработки и передачи для целей сбора данных. Для эффективного сбора и передачи сигналов с датчика нам необходима специальная электронная плата.

Эта электронная плата, называемая платой сбора данных, выполняет несколько важных функций. Прежде всего, она обеспечивает соединение между датчиком и другими компонентами системы. Она принимает сигналы, поступающие от датчика, и проводит их дальнейшую обработку. Существует несколько плат, которые можно использовать для сбора данных с датчиков движения. Вот некоторые из них [8]:

1. Arduino [8]: Arduino - это платформа для создания простых электронных устройств и прототипирования. Она поддерживает широкий спектр датчиков, включая датчики движения, и имеет множество вариантов плат, таких как Arduino Uno, Arduino Nano, Arduino Mega и другие.
2. Raspberry Pi [8]: Raspberry Pi - это одноплатный компьютер, который также можно использовать для сбора данных с датчиков движения. Он обладает большей вычислительной мощностью и может быть полезен, если вам нужно выполнять более сложные операции обработки данных.
3. Particle Photon [8]: Particle Photon - это плата разработки, которая поддерживает облачные сервисы и Wi-Fi-соединение. Она может использоваться для сбора данных с датчиков движения и передачи их в облако для дальнейшего анализа.

Критерии выбора могут включать функциональность, доступность, стоимость, экосистему поддержки, требования к энергопотреблению и т. д.. Например, Arduino и Raspberry Pi широко используются и обладают большим сообществом разработчиков и обширной базой знаний.

Для нашего проекта будет выбрана плата Arduino, а точнее Arduino Uno R3.

## Описание выбранного оборудования

В данном подразделе мы представим подробный обзор и описание выбранного оборудования, необходимого для реализации нашей системы сбора данных.

Arduino [7] - это плата для создания программно-аппаратных проектов. Она состоит из микроконтроллера, программного обеспечения и набора электронных компонентов, которые можно подключать к плате.

Плата Arduino имеет малый размер, проста в использовании и доступна по цене. Это делает ее популярной среди людей, которые хотят создавать свои проекты, но не обладают большими знаниями в области электроники и программирования.

Arduino можно использовать для создания различных устройств, например:

* + Датчики и системы сбора данных
  + Роботы и автоматические устройства
  + Умный дом и системы автоматизации
  + Системы управления освещением и т.д.

Программное обеспечение для Arduino, известное как Arduino IDE, позволяет создавать и загружать программный код на плату. Язык программирования Arduino основан на языке C/C++, но имеет упрощенный синтаксис и библиотеки функций, что делает его более доступным для начинающих.

На плате Arduino есть различные порты ввода-вывода, такие как аналоговые и цифровые порты, которые можно использовать для подключения датчиков, светодиодов, моторов и других устройств. Кроме того, существует множество различных модулей и дополнительных устройств, которые можно подключить к плате для расширения ее возможностей.

Главным преимуществом Arduino является огромное сообщество пользователей и разработчиков, которые активно обмениваются опытом и создают новые проекты.

Одной из самых популярных моделей Arduino является Arduino Uno. Arduino Uno основана на микроконтроллере ATmega328P и предлагает широкие возможности для создания различных проектов.

Пироэлектрический ИК-датчик [10] (инфракрасный датчик движения) - это электронное устройство, способное обнаруживать изменения инфракрасного излучения в окружающей среде и преобразовывать их в электрический сигнал. Он широко применяется в системах безопасности, автоматическом освещении, системах умного дома и других приложениях, где требуется обнаружение движения.

Основная принципиальная часть пироэлектрического ИК-датчика - это пироэлектрический сенсор, который состоит из материала с пироэлектрическим эффектом. Когда объекты или живые существа движутся в поле зрения датчика, они излучают инфракрасное излучение, которое может быть обнаружено и преобразовано в электрический сигнал с помощью пироэлектрического материала.

Пироэлектрический материал в датчике обладает способностью изменять свой электрический заряд при изменении температуры. При движении объектов, излучающих инфракрасное излучение, пироэлектрический материал претерпевает изменение температуры, что приводит к генерации заряда в сенсоре. Этот заряд затем обрабатывается электронной схемой датчика и преобразуется в цифровой или аналоговый сигнал, который может быть использован для определения наличия движения.

Одной из ключевых особенностей пироэлектрического ИК-датчика является его высокая чувствительность к изменениям температуры и инфракрасного излучения. Он способен обнаруживать движение объектов в определенном радиусе и с заданной чувствительностью. Некоторые модели датчиков имеют возможность настройки чувствительности и времени задержки, что позволяет адаптировать их под конкретные требования и условия использования.

# Разработка системы сбора данных

В этом разделе будут представлены подробности о разработке метода сбора данных с датчиков движения, передаче на веб-сервер и дальнейшей их обработке.

Так же в этом разделе будет рассмотрена реализация серверной части сайта отеля, то есть программную составляющую, которая запущена на сервере и осуществляет обработку запросов от пользователей. В коде серверной части будут описаны все основные функции и методы, которые осуществляют работу всей бизнес-логики отеля.

## Разработка серверной части сайта отеля

Перед началом разработки необходимо задать настройки проекта через Docker. Для этого в одну директорию с корневой папкой проекта будут помещены два файла ‘Dockerfile’ и ‘docker-compose’. Dockerfile - это текстовый файл, используемый в технологии Docker для автоматизированной сборки контейнера. Он содержит инструкции, которые описывают шаги по созданию образа контейнера, включая настройку окружения, установку зависимостей, копирование файлов и запуск команд. Docker-compose - это инструмент, который позволяет определить и запускать множество связанных контейнеров Docker как одно приложение. Он используется для управления многоконтейнерными приложениями и облегчает развертывание, масштабирование и управление контейнерами. В нашем случае будет два контейнера, которые благодаря docker-compose можно будет без проблем запустить как единое приложение: сайта отеля и база данных.

В Dockerfile были выполнены следующие основные настройки: определение базового образа языка Python, копирование файлов и директорий проекта внутрь контейнера Docker, установка рабочего каталога внутри контейнера. Также были определены команды, которые выполняются на этапе сборки контейнера. Одной из основных команд является создание виртуального окружения Python и удаление предыдущих зависимостей и временных файлов.

В файле Docker-compose были описаны настройки версии синтаксиса для текущего файла, определены сервисы app (приложение отеля) и db (база данных).

В сервисе "app" определены следующие основные настройки: определение порта, на котором будет работать сервер, привязка каталога "hotel\_app" к рабочему каталогу "/hotel\_app" внутри контейнера, определение команд, которые будут выполняться при запуске контейнера, включая ожидание запуска базы данных и запуск сервера. Также указаны данные для подключения к базе данных.

В сервисе "db" определены следующие основные настройки: загрузка образа базы данных PostgreSQL, определение раздела с постоянным хранилищем данных, а также определение данных, которые позволяют сторонним сервисам подключаться к базе данных.

После задание основных настроек Docker было создано основное корневое Django приложение hotel\_app, так же были созданы дополнительные приложения, которые будут отвечать за определенную логику сайта: userlogin и hotel.

В итоге серверная часть сайта отеля будет иметь структуру, представленную рисунке 2.

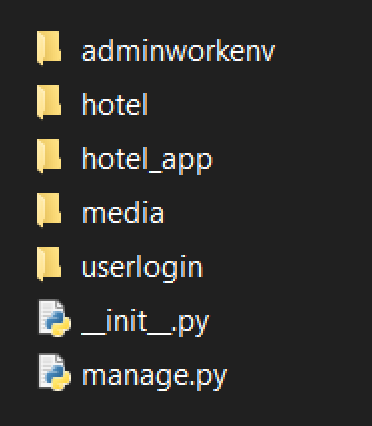


Рисунок 2 – Структура серверной части

Корневая папка приложения ‘hotel\_app’ содержит структуру, представленную на рисунке 3.

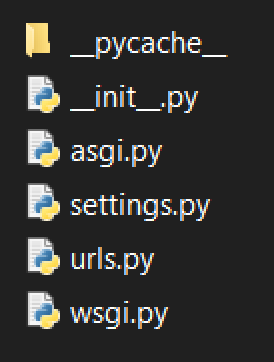


Рисунок 3 – Структура корневой папки

Из всех файлов взаимодействие будет происходить только с файлом settings.py и urls.py.

Файл settings.py содержит все настройки проекта.

Все входящие запросы изначально поступают в корневую папку и в файл urls.py, пытаясь найти обработчик запроса. Поэтому в данном файле мы будет перенаправлять запросы на другие приложения, в файле urls.py которых эти запросы уже смогут найти нужный обработчик. Код представлен ниже.

urlpatterns = [

path('admin/', admin.site.urls),

path('', include(('userlogin.urls', 'userlogin')), name='userlogin'),

path('', include(('hotel.urls', 'hotel')), name='hotel'),

]

Здесь объявлена ссылка ‘admin/’ на панель администратора Django, так же подключаются файлы ссылок из приложений userlogin и hotel.

Приложение userlogin имеет структуру, представленную на рисунке 4.

В файле models.py определена модель пользователя, а также класс, который содержит функции для создания пользователей. При выполнении этих функций создается запись о пользователе в базе данных. Кроме того, используя встроенную функцию set\_password, устанавливается пароль, введенный пользователем, который будет использоваться при аутентификации в системе

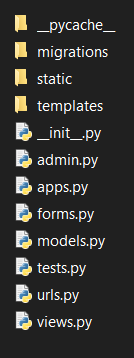


Рисунок 4 – Структура приложения userlogin

Файл forms.py содержит формы, которые используются для регистрации, входа и обновления данных пользователей. Форма для регистрации включает обязательные поля, которые пользователь должен заполнить при регистрации: имя, фамилию, почту и номер телефона. Поле для пароля не включается в эту форму, оно указывается только в HTML-файле и передается в функцию создания пользователя при выполнении регистрации. Форма для входа содержит только поля для ввода почты и пароля. Форма для изменения данных содержит те же поля, что и форма для регистрации. Так же в этой форме содержится метод save, которая подтверждает обновление данных и, при необходимости, обновляет пароль.

Файл admin.py содержит строку, которая регистрирует модель пользователя. Зарегистрированная модель будет отображаться в панели администратора Django.

Файл urls.py содержит следующие ссылки:

1. 'register/' – страница регистрации
2. 'login/' – страница входа
3. 'logout/' – ссылка, которая производит выход из аккаунта
4. ‘myprofile/' – страница профиля
5. 'update\_profile/' – страница обновления данных
6. 'error\_login/' – страница ошибки входа
7. 'error\_reg/' – страница ошибки регистрации

В файле views.py содержаться обработчики ссылок, перечисленных выше.

Для страницы регистрации установлен обработчик, который просто отображает страницу "register.html". Если форма заполнена правильно, то в обработчике вызывается функция "form\_valid", которая создает нового пользователя с введенными данными. При помощи встроенной функции "login" в Django автоматически выполняется вход в аккаунт.

При открытии страницы для входа в систему, обработчик также отображает соответствующий HTML-файл (login.html). При правильном заполнении формы, встроенная функция аутентификации Django - "authenticate" вызывается с передачей почты и пароля в качестве параметров. Если аутентификация прошла успешно, функция "login" также используется для выполнения входа в учетную запись.

Переход по ссылке ‘logout/’ заставит обработчик вызвать встроенную функцию logout, которая выполняет выход из аккаунта.

Обработчик ссылки 'myprofile/' осуществляет запрос к базе данных для получения пользователя по его идентификатору. Полученный пользователь передается в HTML-файл, где выводится вся его информация. Необходимое значение идентификатора извлекается через 'request.user.id'. Часть команды 'request.user' автоматически определяет учетную запись пользователя, с которой происходит переход по ссылке.

При переходе на ‘update\_profile/’ обработчик отображает форму изменения данных, в которую автоматически вводятся текущие данные пользователя. При правильном заполнении формы вызывается метод save, который описан в forms.py в форме обновления данных.

Обработчики страницы ошибок только отображают на экран нужные html страницы.

Переходы по ссылкам ‘myprofile/', ‘update\_profile/’ и ‘logout/’ доступны только если пользователь вошел в свой аккаунт на сайте.

Приложение hotel имеет такую же структуру за исключением двух папок: management и booking.

В папке management хранится файл wait\_for\_db.py, который выполняется при запуске контейнера. Он отвечает за проверку запуска БД, т.е. пока БД не запустится файл не закончит выполнение.

В папке booking находится файл booking\_availability.py, который по переданным в него данным проверяет доступность бронирования.

В файле models.py определена модель комнаты, а также модель бронирования, которая связана с моделью комнаты по ее первичному ключу (номер комнаты).

Файл forms.py содержит только форму бронирования с полями выбора типа комнаты выпадающим списком, фамилии, имени, почты, номера телефона, датами заезда и выезда

Файл urls.py содержит следующие ссылки:

1. '' – главная страница
2. 'booking/' – страница бронирования номера
3. 'room/<type>' – страница подробного описание определенной категории комнаты
4. ‘booking\_list/' – страница «Мои бронирования»
5. 'error\_booking/' – страница ошибки броинрования
6. 'error\_date/' – страница ошибки ввода дат

Обработчик главной страницы выполняет следующие действия: он создает список, в который добавляются первые комнаты каждой категории из базы данных, а также создает URL для просмотра подробной информации о каждой категории. Это достигается путем добавления категории комнаты в качестве аргумента в ссылку. При нажатии на категорию пользователь будет перенаправлен на страницу с подробным описанием этой категории, используя переданный аргумент в ссылке. Затем выполняется отображение файла room\_list.html, в который передается созданный и заполненный список комнат.

При переходе на страницу 'booking/', отображается форма для заполнения информации о бронировании. Если форма заполнена правильно, происходит проверка правильности введенных дат. Если даты введены неверно, выводится ошибка. Если ошибок нет, происходит проверка доступности бронирования на запрошенные даты. Для этого все комнаты выбранной категории помещаются в список. Затем производится проверка каждой комнаты на доступность путем вызова функции check\_booking из файла booking\_availability.py. В эту функцию передается комната и даты бронирования. Затем даты всех существующих бронирований для данной комнаты сравниваются с переданными датами. Если найдутся пересечения с запрошенными датами, функция возвращает отрицательный результат, и происходит проверка следующей комнаты. Если комната доступна на запрошенные даты, функция возвращает положительный результат, проверка доступности прекращается, и создается новое бронирование. Если после проверки не было найдено доступной комнаты, пользователю будет показана страница ошибки с просьбой выбрать другие даты или другую категорию комнаты.

При нажатии на главной странице на определенную категорию комнаты происходит переход по ссылке 'room/<type>'. Обработчик этой ссылки получает аргумент категории комнаты, который был передан при формировании списка категорий на главной странице. Затем происходит отображение страницы 'room\_detail\_view.html' и передача полученной категории в качестве данных. Это позволяет отобразить соответствующее описание и фотографии в зависимости от выбранной категории комнаты.

При переходе на страницу "Мои бронирования" обработчик выполняет запрос к базе данных и из всех бронирований отбирает бронирования пользователя, который просматривает страницу. Для отбора используются данные пользователя, полученные через 'request.user', включая адрес электронной почты, номер телефона и дату заезда. Дата заезда должна быть позднее или равной текущей дате, поскольку отображение бронирований с прошедшей датой заезда не имеет смысла. Ссылка на страницу доступна только для пользователей, которые вошли в свою учетную запись на сайте.

Функции показа ошибок имеют только одну строчку – вывод html страницы ошибки.

Ниже представлена схема взаимодействия пользователя с приложением (рисунок 5).

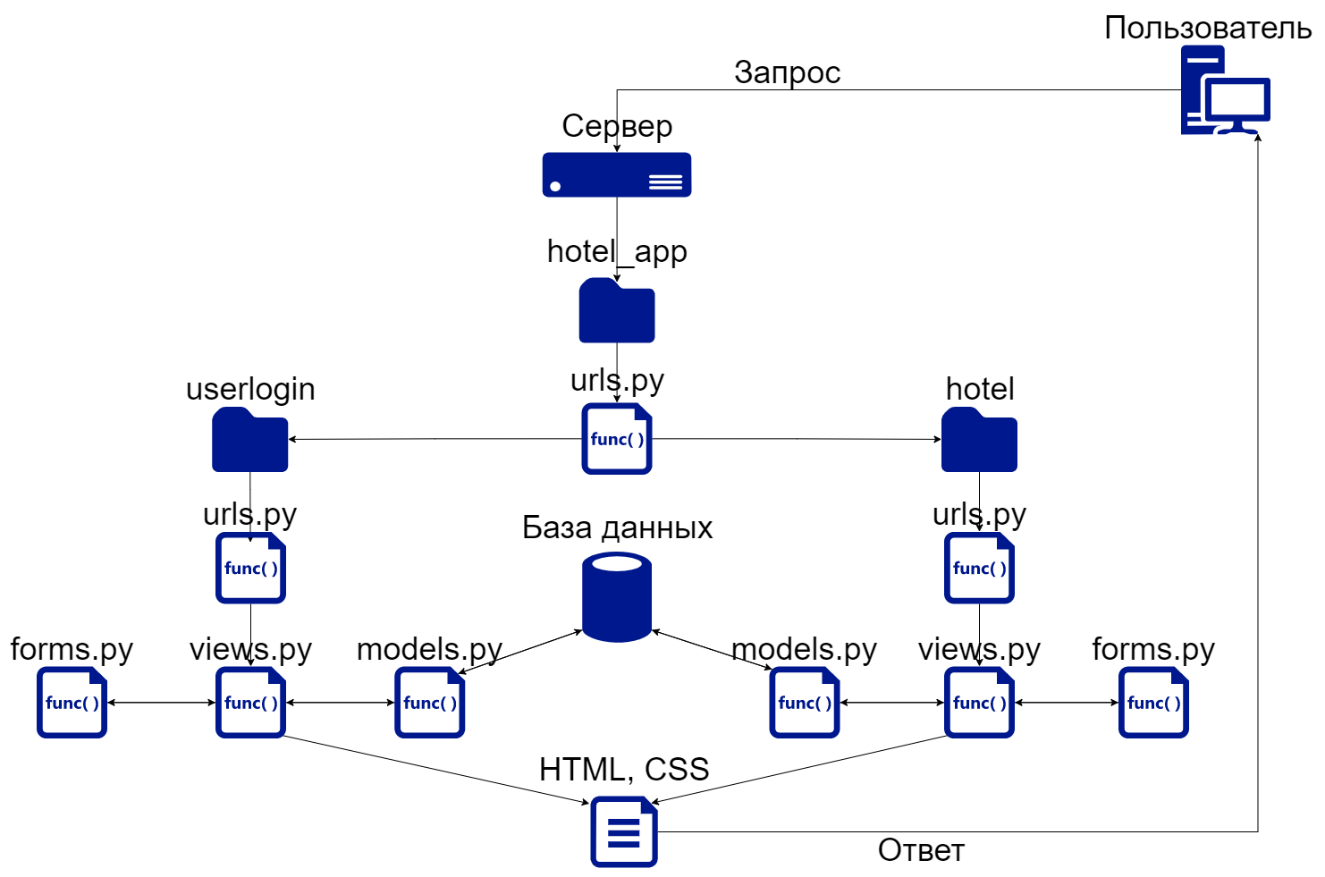


Рисунок 5 – Взаимодействие пользователя с приложением

В результате выполнения была создана функциональная и надежная серверная часть сайта отеля, которая обеспечивает удобный интерфейс для регистрации, входа, бронирования комнат и просмотра информации о бронированиях. Реализованный функционал позволяет пользователям легко взаимодействовать с системой и получать актуальную информацию о доступных комнатах и их бронировании

## Разработка выбранного метода сбора данных

Перед началом разработки будет продемонстрирована общая схема системы сбора данных. Схема представлена на рисунке 6.

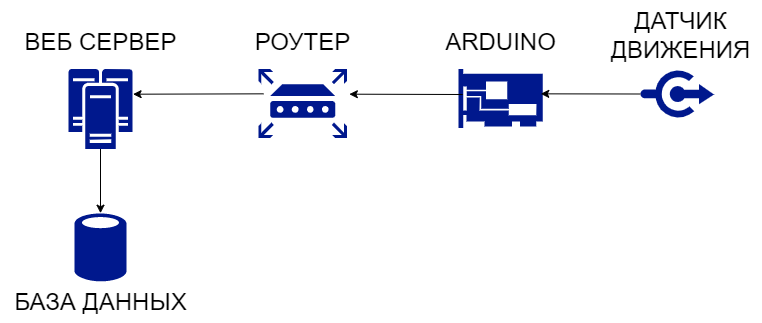


Рисунок 6 – Общая схема системы сбора данных

Датчик движения будет генерировать сигнал об отсутствии или наличие движения. Плата Arduino будет принимать, обрабатывать и передавать сигнал дальше на веб-сервер через Wi-Fi роутер. После его принятия будут срабатывать обработчики сигнала на веб-сервере, которые, в зависимости от полученных данных, будут вносить определенные изменения в базу данных.

Ниже представлено подробное описание проделанной работы для каждого из компонентов системы.

**Датчик движения.** Когда в зоне действия датчика обнаруживается движение, сложная система срабатывает и генерирует небольшой электрический сигнал, который затем передается на плату Arduino для дальнейшей обработки. Датчик движения представлен на рисунке 7.



Рисунок 7 – Пироэлектрический ИК-датчик движения

**Плата Arduino**. Для того, чтобы Arduino [11] могла принимать и передавать сигналы дальше, необходимо разработать соответствующий код программы. Этот код будет определять логику работы Arduino и управлять ее функциональностью.

Для Arduino можно определить различные функции и инструкции, которые позволят принимать сигналы, выполнять определенные действия и взаимодействовать с другими компонентами системы. Arduino Uno R3 представлена на рисунке 8.

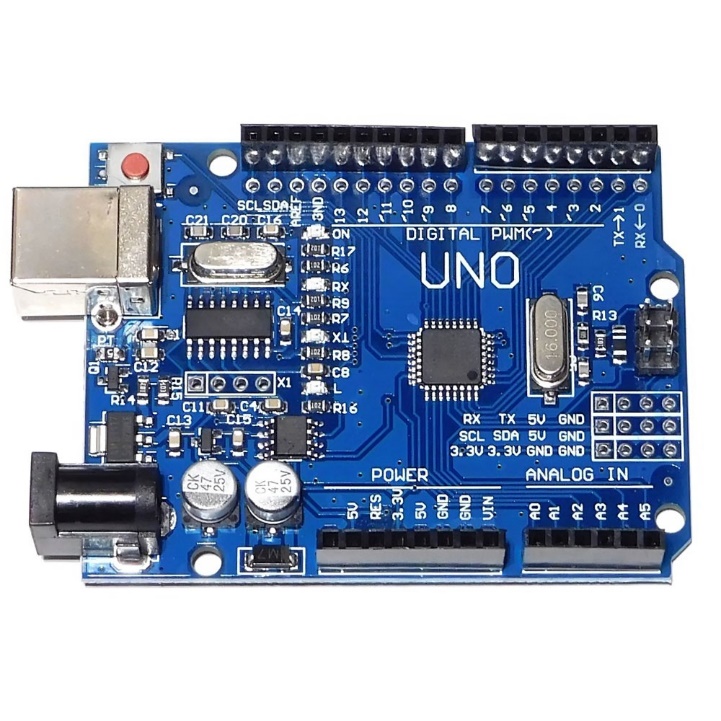


Рисунок 8 – Arduino Uno R3

Для передачи данных с Arduino на веб-сервер через Wi-Fi можно использовать модуль Wi-Fi, такой как ESP8266 или ESP32, в сочетании с протоколом передачи данных HTTP. Модуль ESP8266 представлен на рисунке 9.



Рисунок 9 – Модуль Wi-Fi ESP8266

Схема соединения Arduino и ESP8266 представлена на рисунке 10.

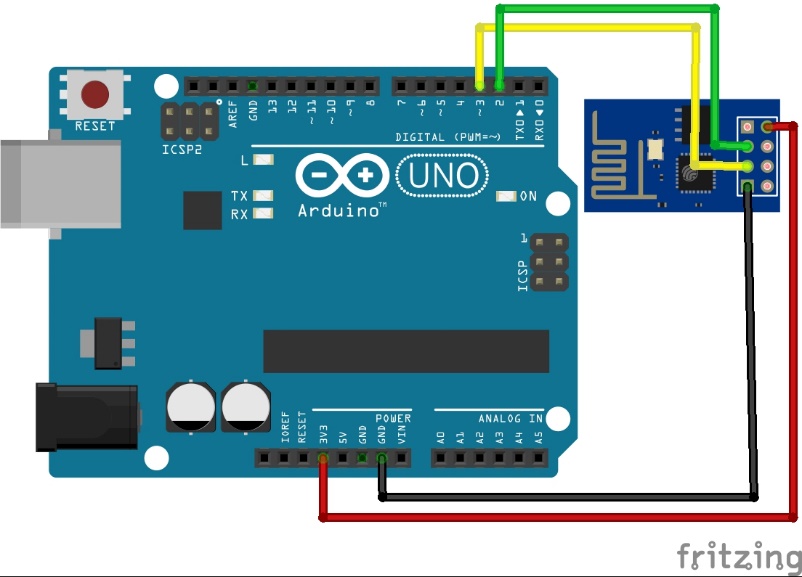


Рисунок 10 – Схема соединения Arduino и ESP8266

Ниже приведен последовательный порядок действий, которые должны быть выполнены для удачной передачи данных с Arduino на веб-сервер:

1. Подключение Wi-Fi модуля: установка Wi-Fi модуль на плату Arduino и подключение его соответствующим образом. Необходимо убедится, что Wi-Fi модуль правильно настроен для подключения к Wi-Fi сети.
2. Код Arduino: написание кода для Arduino, который будет устанавливать с Wi-Fi сетью и считывать данные с веб-сервера или другого источника.
   * Подключение всех необходимых библиотек. Для успешного подключения ESP8266 к Wi-Fi сети и выполнения POST-запросов, мы воспользуемся следующими библиотеками: ESP8266WiFi.h для установления соединения с Wi-Fi и ESP8266HTTPClient.h для доступа к методам, необходимым для отправки POST-запросов. Подключение библиотек представлено в коде ниже.

#include <ESP8266HTTPClient.h>

#include <ESP8266WiFi.h>

* + Установка соединения ESP8266 с Wi-Fi-сетью, чтобы обеспечить возможность отправки HTTP-запроса. Для этого достаточно вызвать метод begin для объекта WiFi, указав в качестве первого аргумента имя сети (SSID), а вторым параметром - пароль. соединения ESP8266 с Wi-Fi представлено в коде ниже.

WiFi.begin(ssid, password);

* + Далее необходимо создать экземпляр класса HTTPClient, через который мы будем вызывать различные методы для подготовки заголовков и содержимого запроса, отправки запроса и проверки результата. Для удобства будем использовать переменную с именем http, которая будет ссылаться на этот объект. Создание экземпляра класса представлено в коде ниже.

HTTPClient http;

* + После этого можно вызвать метод begin объекта http и в качестве параметра передать URL-адрес, к которому необходимо выполнить подключение. Код представлен ниже.

// Формирование строки URL с данными

String url = "http://" + String(server) + "/data/";

// Установка URL

http.begin(url);

* + Чтобы определить заголовок для запроса, нужно использовать метод addHeader. Тип содержимого в данном случае указывается как "application/x-www-form-urlencoded", так как будет отправляться строка запроса в формате "имя=значение". Код представлен ниже.

http.addHeader("Content-Type","application/x-www-form-urlencoded");

* + Тело запроса указывается в качестве параметра при вызове метода POST для объекта HTTPClient. Возвращаемое значение этого метода соответствует коду ответа HTTP. Код представлен ниже.

// Формирование тела запроса с данными датчика

String requestBody = "sensor1=" + String(sensorValue1);

// Отправка POST запроса с данными датчиков

int httpResponseCode = http.POST(requestBody);

* + Наконец, необходимо вызвать метод end объекта http, чтобы гарантировать закрытие TCP-соединения и освобождение ресурсов. Особенно важно не забывать об этом при работе с большим количеством запросов. Код представлен ниже.

http.end();

**Роутер.** Процесс передачи данных данного вида предусматривает участие Wi-Fi роутера, что обеспечивает эффективную передачу данных и их принятие на определенном веб-сервере, где они будут обработаны. Первоначально, данные отправленные с источника, проходят через Wi-Fi роутер, который обеспечивает пересылку информации.

**Веб-сервер.** После того, как Arduino отправит данные на сервер, процесс обработки этих данных является не менее важным этапом. Необходимо разработать функционал, который будет принимать переданные данные и выполнять набор действий в зависимости от их значений.

При отправке POST-запроса идет переход по ссылке ‘data/', значит первым делом необходимо реализовать ссылку ‘data/' в файле urls.py.

Далее в файле views.py требуется добавить обработчик ссылки ‘data\_handler’. В начале обработчика происходит проверка метода запроса к ссылке. Дальнейшие действия выполняются только в случае, если метод запроса является POST. После проверки метода мы извлекаем значение датчика движения из данных, отправленных в запросе. Это значение представляет собой номер комнаты, в которую произошло заселение. Затем происходит проверка времени. Для выполнения дальнейших действий необходимо, чтобы время было позднее 13:00 (так как заселение происходит после 13:00). Если время не соответствует этому условию, то дополнительные действия не требуются, поскольку датчик движения может быть активирован персоналом, например, уборщицей. Затем происходит извлечение записи из базы данных таблицы заселений, где номер комнаты соответствует значению, полученному от датчика. Кроме того, дата заселения в записи должна соответствовать текущей дате. Если такая запись найдена, происходит проверка времени заселения. Если время заселения равно полуночи (00:00), то оно заменяется на текущее время, чтобы соответствовать точному времени срабатывания датчика движения. В противном случае время не изменяется, поскольку оно уже было изменено ранее и повторное изменение не требуется. В этом обработчике также происходит изменение времени выезда. Поскольку неизвестно, в какое конкретное время гость планирует выехать из номера, при каждом срабатывании датчика время этого срабатывания записывается в базу данных как время выезда. Таким образом, после окончательного выезда гостя из номера, в базе данных будет отображаться точное время его последнего пребывания там. Однако для изменения данных, время срабатывания должно быть либо до 12 часов дня, либо после 8 часов вечера.

**База данных.** При регистрации информации о заселении в базу данных будут указываться дата въезда и выезда, но время в обоих случаях будет установлено на 00:00. Однако, точное время появления сигнала датчика движения будет использоваться для обновления записи в базе данных.

База данных должна обеспечивать автоматическое удаление данных спустя 3 месяца после их создания. Для этой задачи необходимо внедрить Celery и Redis в проект. Celery позволяет выполнять определенные задачи с заданной периодичностью. Redis является распространенным выбором для использования вместе с Celery в качестве брокера сообщений. Многие задачи, которые обрабатываются Celery, необходимы для выполнения повторно, если они не выполняются в указанный срок времени. Redis, в свою очередь, предлагает быструю и надежную работу с очередями сообщений, что идеально совпадает с задачами обработки в Celery. С помощью Redis в качестве брокера сообщений можно быстро и надежно передавать задачи в Celery и упростить организацию процесса работы с очередями.

Для внедрения Celery необходимо создать файл celery.py в корневой папке проекта (рисунок 11).

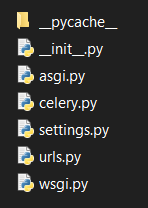


Рисунок 11 – Создание celery.py в корневой папке проекта

В файле celery.py первым делом необходимо указать путь к файлу с корневыми настройками всего проекта. Код представлен ниже.

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "hotel\_app.settings")

Затем следует создать экземпляр приложения Celery, который представляет собой объект, содержащий всю необходимую конфигурацию и определения для запуска воркеров Celery. Далее мы вызываем метод config\_from\_object объекта приложения, который загружает настройки Celery из указанного модуля или файла Python. После этого мы вызываем метод autodiscover\_tasks, который автоматически находит все модули с определенными задачами в указанных директориях и загружает их в приложение Celery. Это позволяет автоматически обнаруживать и регистрировать задачи, определенные в приложении без необходимости явно указывать их. Код представлен ниже.

app = Celery("hotel\_app", include=['hotel.tasks'])

app.config\_from\_object("django.conf:settings", namespace="CELERY")

app.autodiscover\_tasks()

Следующим шагом будет определение списка очереди задач. Для этого необходимо вызвать атрибут объекта app, который определяет расписание периодических задач beat\_schedule. Это словарь, в котором ключом является название периодической задачи, а значением – словарь, содержащий определенные свойства для задачи. Код вызова атрибута представлен ниже.

app.conf.beat\_schedule = {

'delete': {

'task': 'hotel.tasks.delete\_data',

'schedule': crontab(hour='\*/24'),

},

}

В этом словаре: delete - название периодической задачи, task - указывает на имя задачи (таска), schedule - время, когда должна выполняться задача. В нашем случае задача будет запускаться раз в 24 часа, т.е. один раз каждые сутки будет происходить проверка на то, появились ли какие-либо данные для удаления.

В корневом файле настроек необходимо задать переменную, которая определяет URL брокера сообщений для Celery и переменную, которая определяет бэкэнд результатов для Celery. Бэкэнд результатов используется для хранения и получения результатов выполнения асинхронных задач, которые были отправлены в Celery. Код представлен ниже.

CELERY\_BROKER\_URL = "redis://redis:6379/0"

CELERY\_RESULT\_BACKEND = "redis://redis:6379/0"

Следующим шагом после внедрения в проект Celery и Redis и полной их настройки станет написание самой задачи по автоматическому удалению данных. Для этого необходимо создать файл с задачами tasks.py и поместить его в папку проекта hotel (рисунок 12).

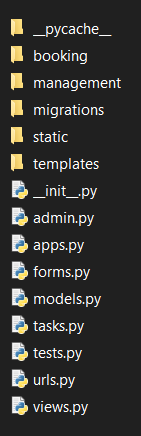


Рисунок 12 – Создание tasks.py в папке проекта hotel

В этом файле была создана функция delete\_data, на которую был навешен декоратор @app\_task, который превращает обычную функцию в асинхронную задачу Celery. В теле функция происходит обращение к таблице заселений CheckIn в БД. В обращении идет отбор тех записей, которые были созданы ровно 91 день назад. Все отобранные записи удаляются из БД.

В результате разработки была реализована полноценная система сбора данных с датчиков движения. Были установлены и настроены необходимые компоненты для обработки полученных данных и их передачи. В частности, была разработана интеграция с веб-сервером, позволяющая принимать данные от датчиков и обеспечивающая их правильную обработку. Одним из ключевых достижений является внедрение системы периодического выполнения задач, которая демонстрирует гибкость и расширяемость разработанной системы. Это позволяет выполнять задачи по расписанию или с определенными интервалами времени. Такой подход может быть весьма полезным для будущих потребностей и возможностей системы, позволяя автоматизировать повторяющиеся операции и обеспечивать более эффективное использование ресурсов.

# Результаты реализации системы

В данном разделе представлены результаты создания системы сбора данных и серверной части сайта отеля. Будет продемонстрирована работа каждого модуля и функции программы.

## Демонстрация серверной части сайта отеля

Первым делом попробуем зарегистрировать аккаунт. Введем данные представленные на рисунке 13.

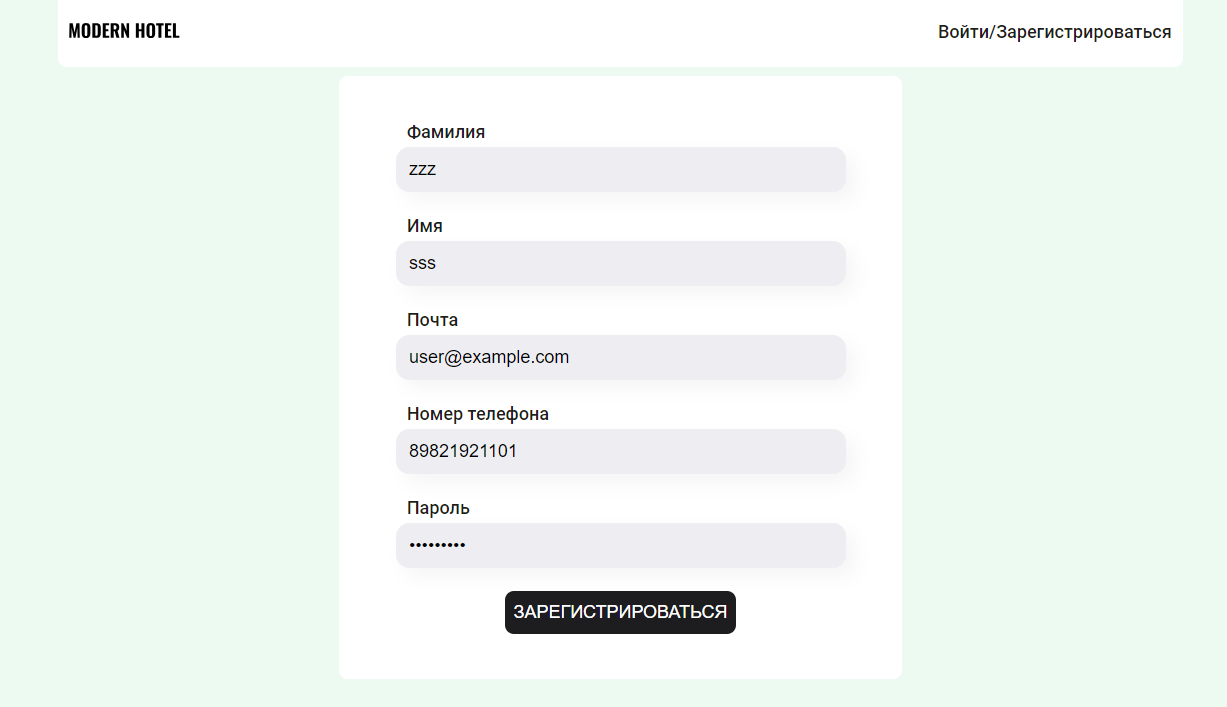


Рисунок 13 – Регистрации аккаунта

После регистрации сразу произошел переход на главную страницу. Необходимо проверить на странице профиля все данные аккаунта (рисунок 14).

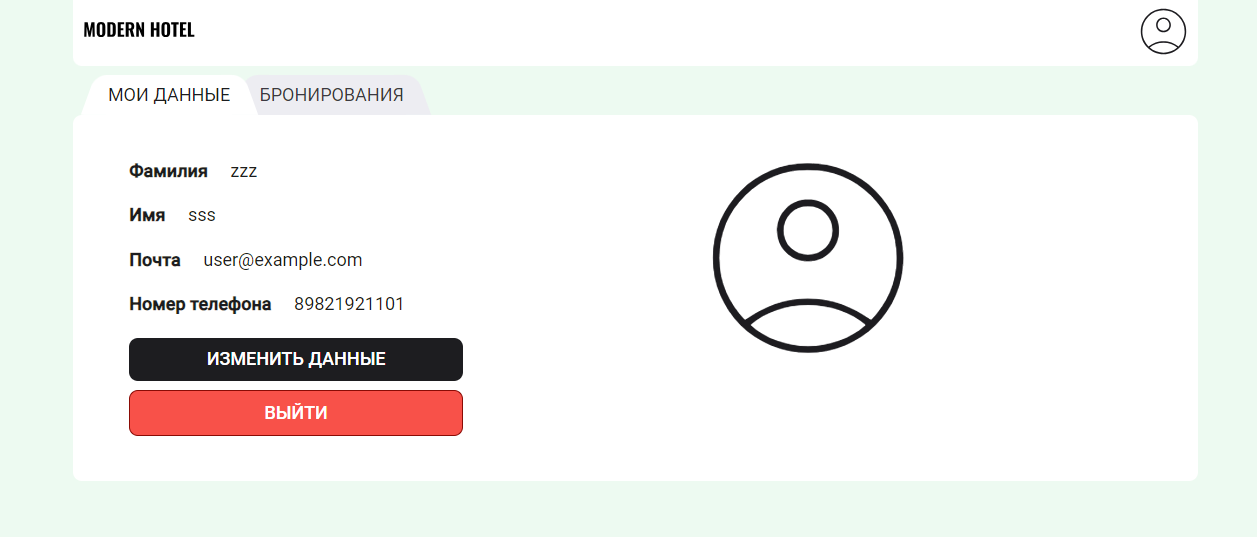


Рисунок 14 – Профиль

Далее следует убедиться, что на аккаунте еще нету бронирований (рисунок 15).

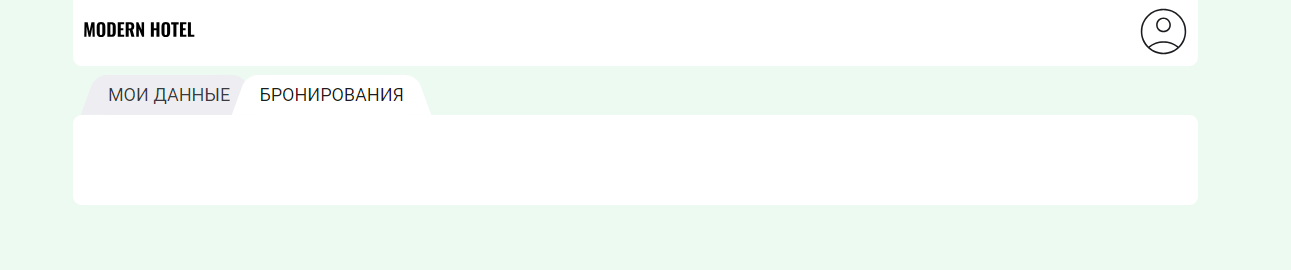


Рисунок 15 – Текущие бронирования

Чтобы совершить бронирование требуется перейти на главную страницу (рисунок 16).

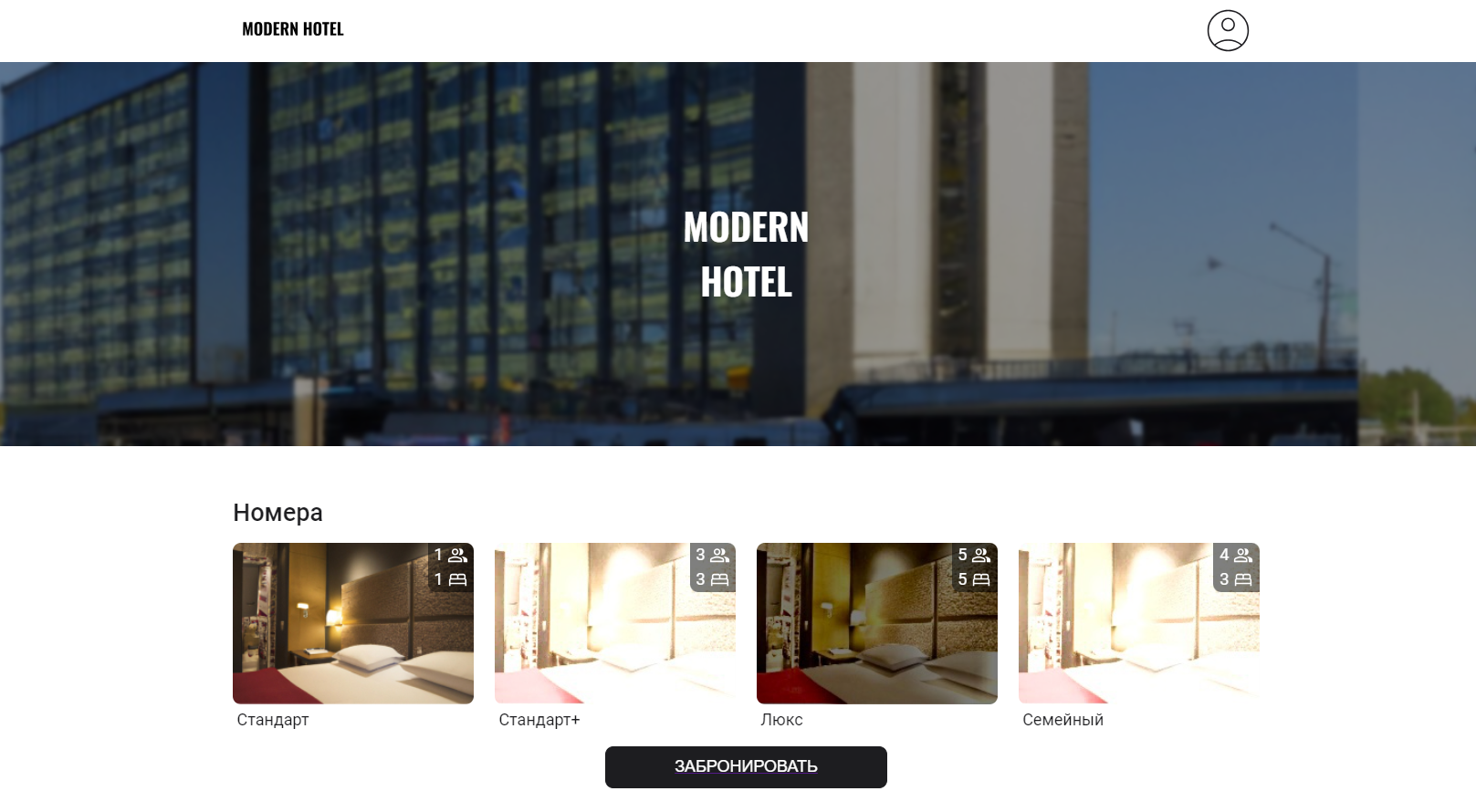


Рисунок 16 – Главная страница

Перед совершением бронирования следует просмотреть подробную информацию о категориях комнаты (рисунок 17, рисунок 18).



Рисунок 17 – Подробнее о категории «Стандарт+»

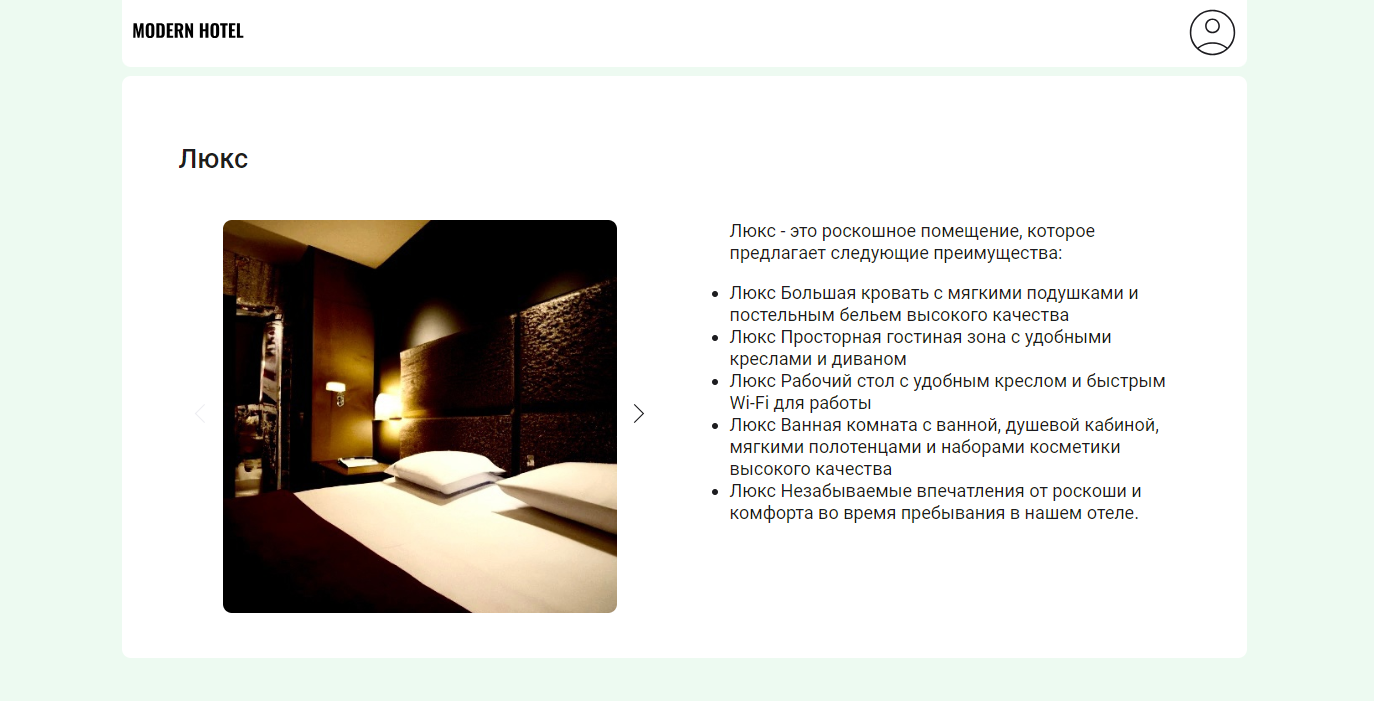


Рисунок 18 – Подробнее о категории «Люкс»

При нажатии на кнопку «Забронировать» на главной странице будет произведен переход на страницу бронирования. Поля имени, фамилии, номера телефона и почты заполнятся автоматически (рисунок 19).

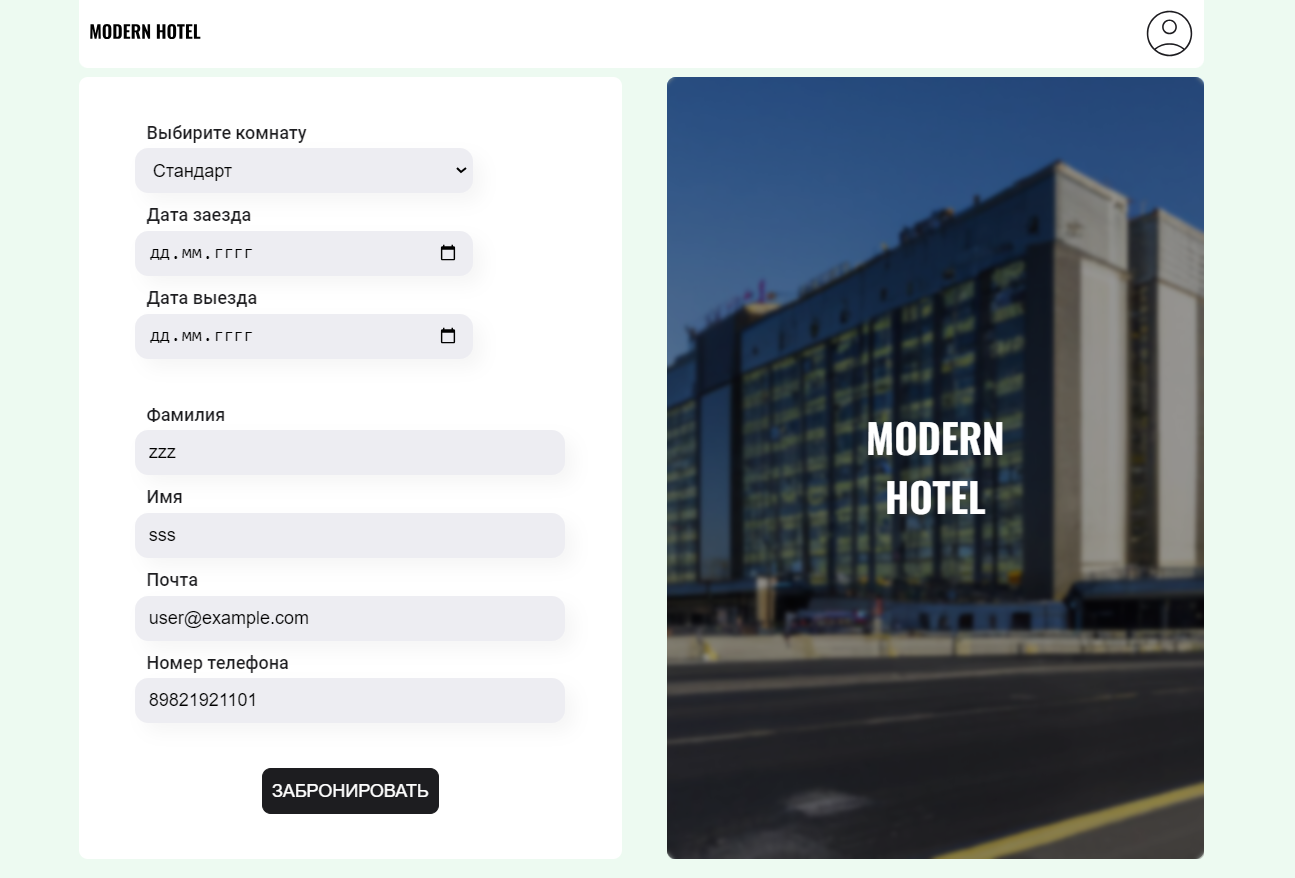


Рисунок 19 – Страница бронирования

После успешного бронирования будет совершен автоматический переход на страницу текущих бронирований, на которой уже будет отображено только что созданное бронирование (рисунок 20).

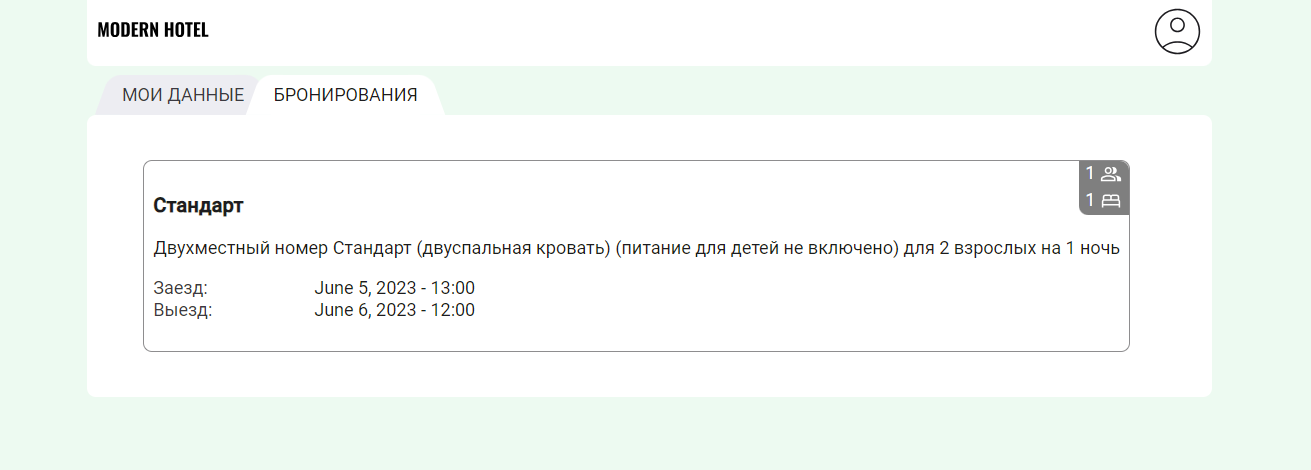


Рисунок 20 – Текущие бронирования

При переходе на страницу изменения информации произойдёт автоматическое заполнение формы, которую можно редактировать и вписывать новую информацию (рисунок 21).

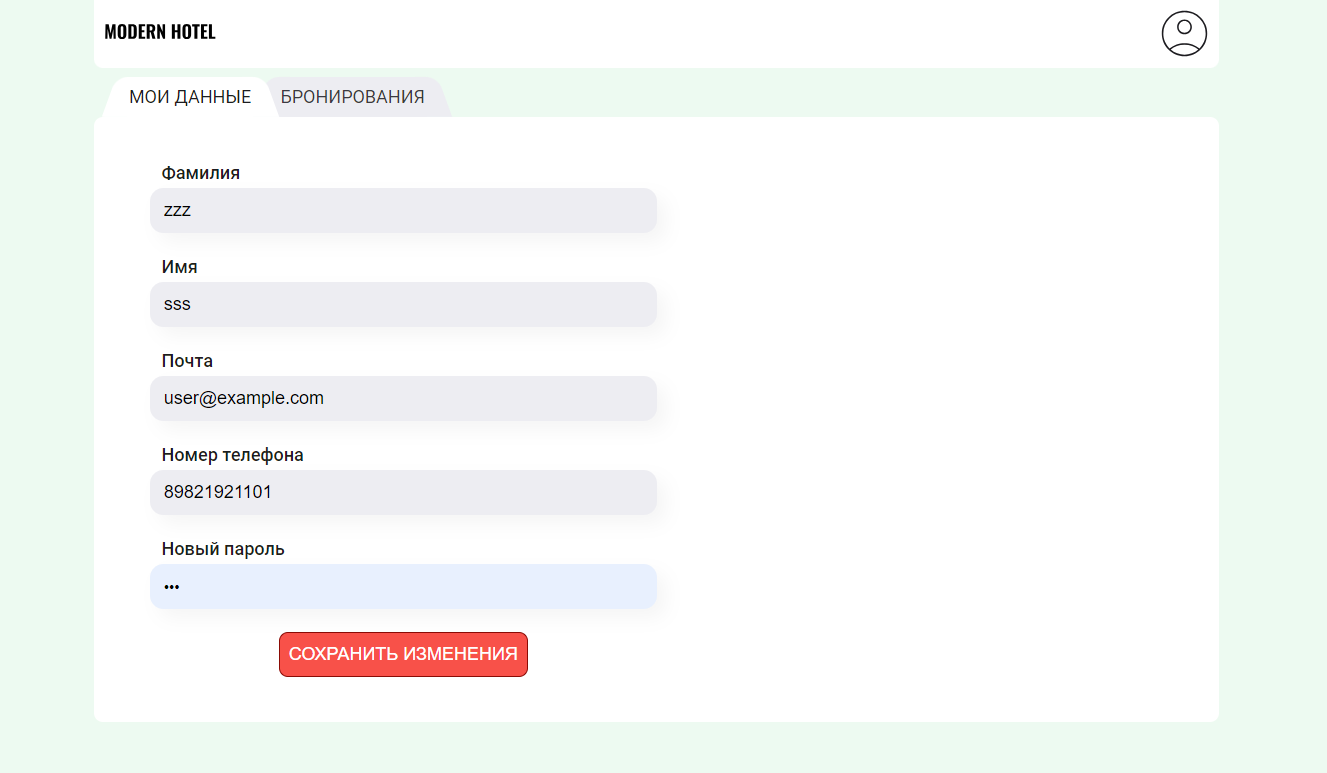


Рисунок 21 – Страница изменения данных

После изменения данных и сохранения изменений они сразу же отобразятся на странице (рисунок 22).

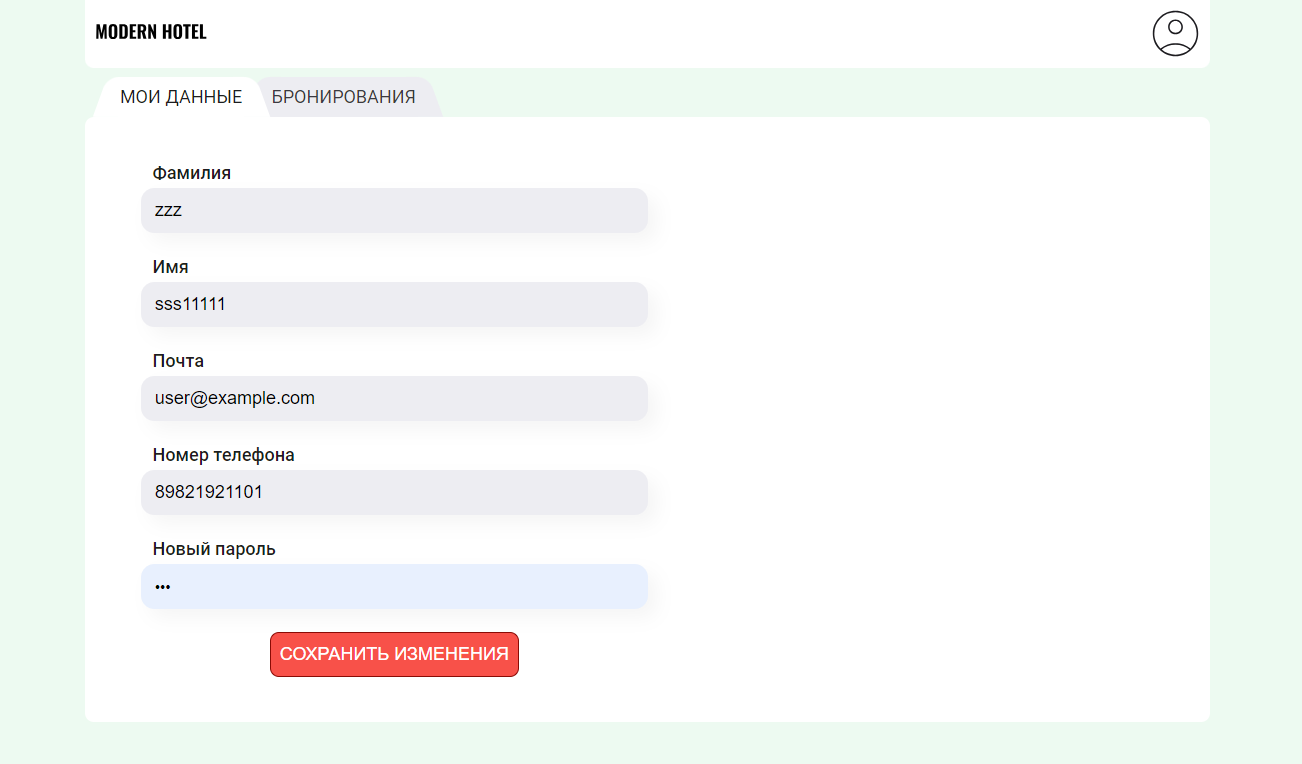


Рисунок 22 – Измененные данные

При попытке поставить дату заезда позже даты выезда или забронировать номер ранее чем на сегодня будет показана страница ошибки (рисунок 23).

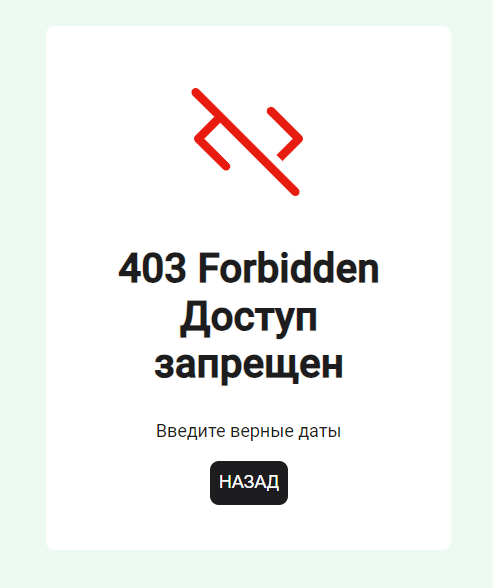


Рисунок 23 – Ошибка ввода дат

При недоступности номеров на выбранные даты будет показана страница ошибки (рисунок 24).

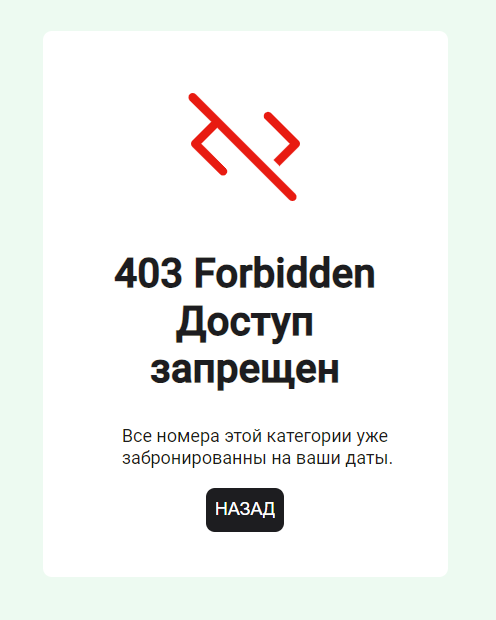


Рисунок 24 – Ошибка бронирования

При неверном вводе данных во время входа или попытке зарегистрировать аккаунт на почту или номер телефона, на которые уже зарегистрированы аккаунты, будут показаны страницы ошибки (рисунок 25, рисунок 26).

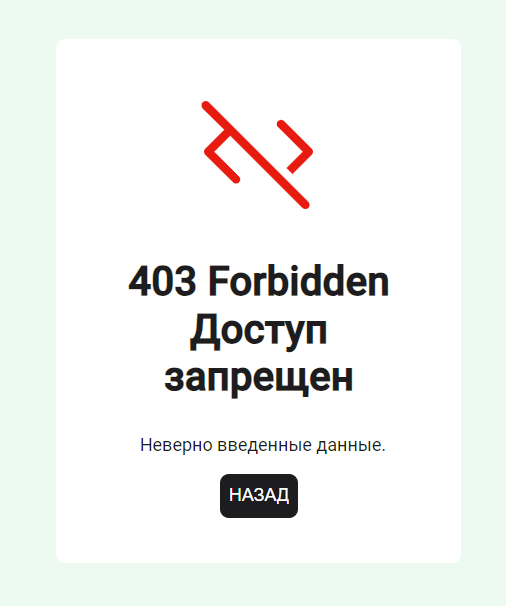


Рисунок 25 – Ошибка входа в аккаунт

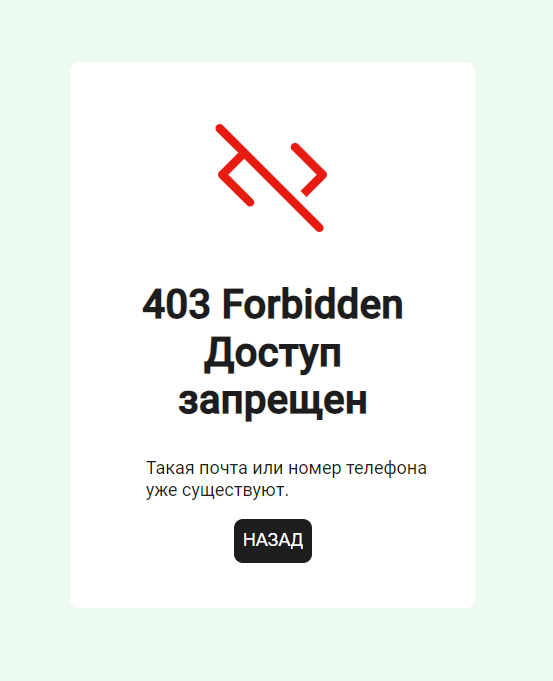


Рисунок 26 – Ошибка регистрации аккаунта

## 4.2 Демонстрация системы сбора данных

При соединении пироэлектрического датчика движения, платы Arduino и Wi-Fi модуля ESP8266, получится схема, представленная на рисунке 27

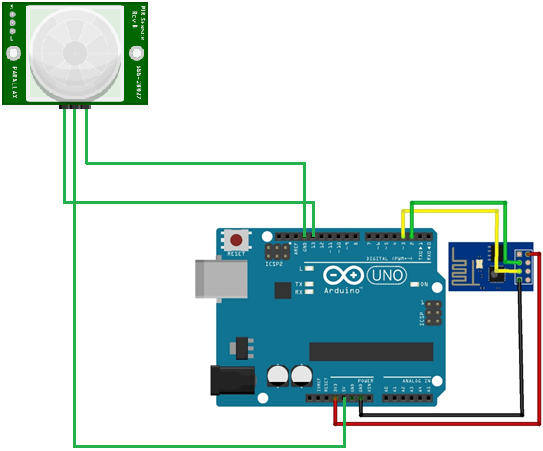


Рисунок 27 – Схема сбора данных через датчик движения

При обнаружении движения пироэлектрическим датчиком значение сигнала становится равным номеру комнаты, в которой произошло движение. Изменение сигнала необходимо для того, чтобы при его передаче на веб-сервер не пришлось каким-либо образом расшифровывать из какой комнаты пришел сигнал.

После изменения сигнала создается тело POST-запроса, в который помещается значение датчика движения в формате ключ=значение (датчик=сигнал).

После отправки сигнала происходит его обработка на веб-сервере. Для демонстрации обработки сигнала была создана запись о заселении (рисунок 28).

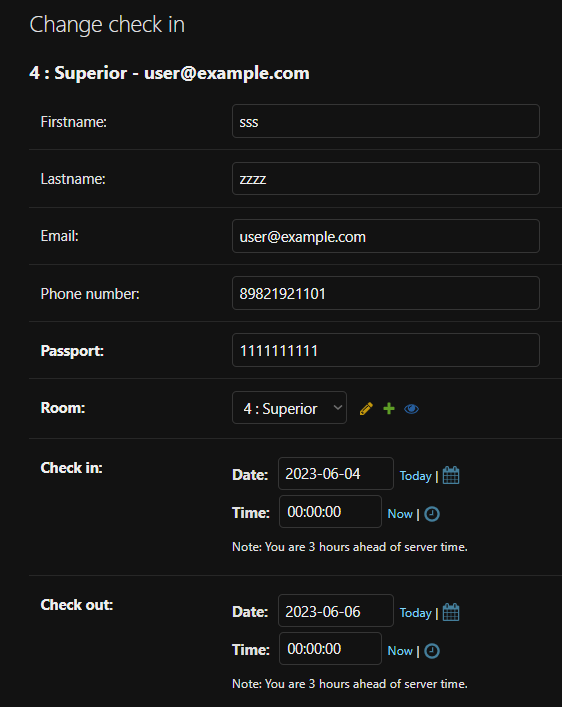


Рисунок 28 – Запись о заселении

Время установлено в полночь, и после срабатывания обработчика сигнала оно должно измениться на текущее.

Симуляция POST-запроса будет создана с помощью Postman. Отправим запрос, в котором имитируем передачу сигнала с датчика движения. Значение датчика – условный номер комнаты, в которой произошло движение (рисунок 29).

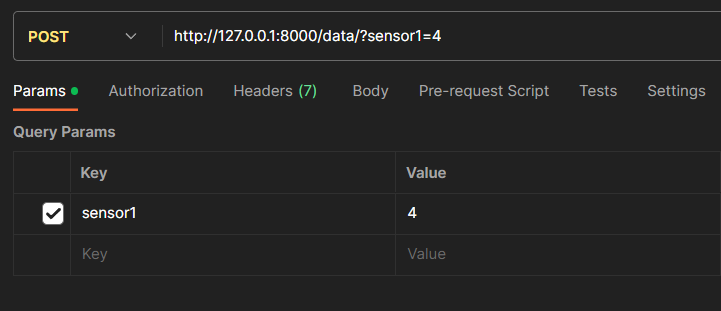


Рисунок 29 – Симуляция POST-запроса

Запрос был отправлен, успешно отработал и вернул ответ в виде кода 200. Время на записи о заселении изменилось на текущее с точностью до секунды (рисунок 30, рисунок 31).

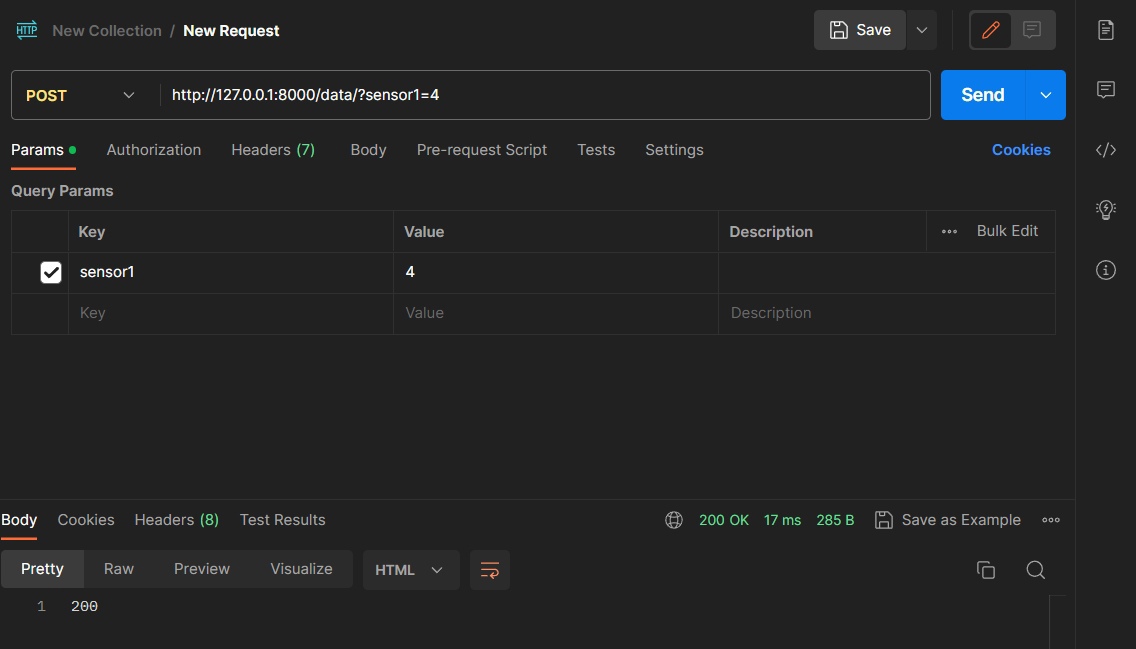


Рисунок 30 – Код ответа

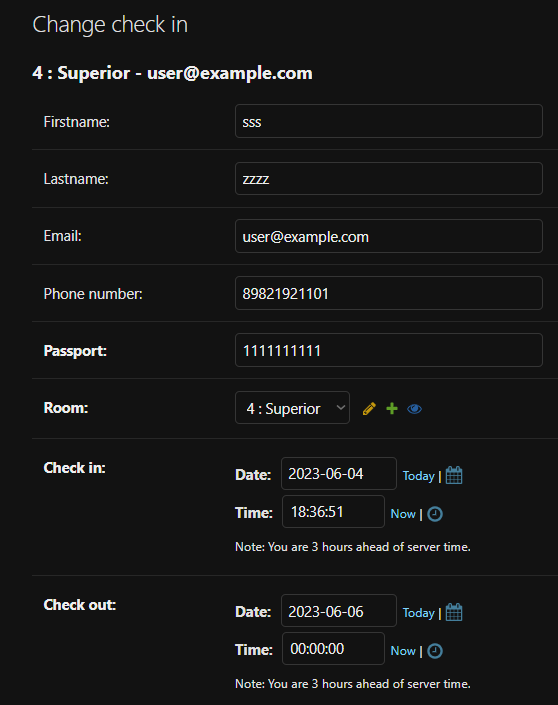


Рисунок 31 – Результат запроса

Для демонстрации автоматического удаления данных создадим запись о заселении, дату создании которой выставим на 91 день назад (рисунок 32, рисунок 33).

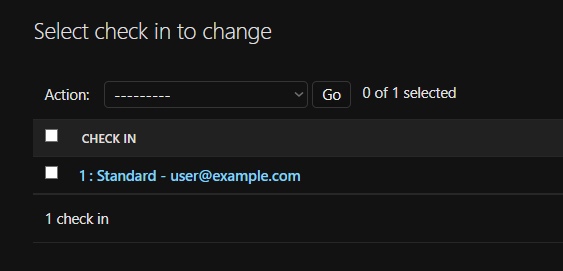


Рисунок 32 – Запись о заселении

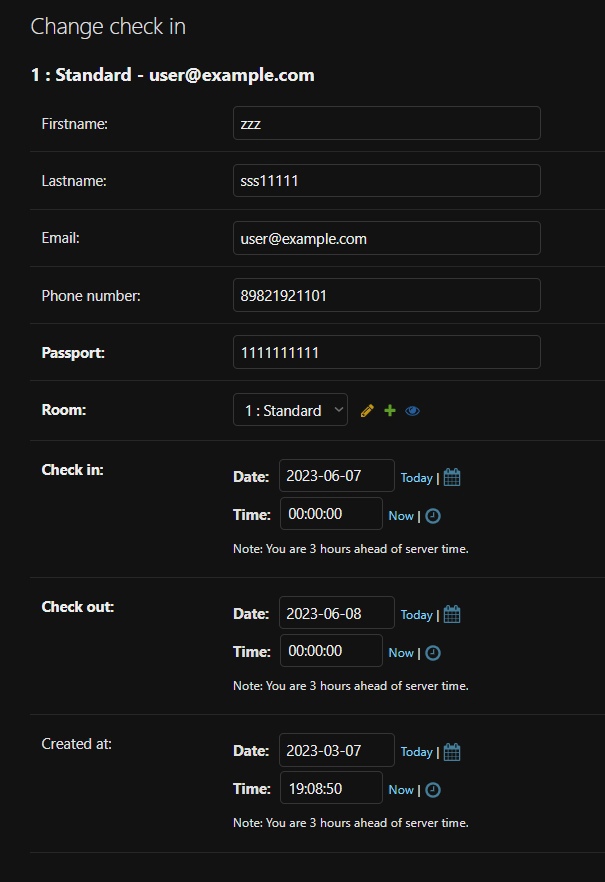


Рисунок 33 – Изменение даты создания

При запуске сервера произойдет проверка и поиск записей которые необходимо удалить. На рисунке 34 и рисунке 35 приведен результат работы периодической задачи по удалению данных.

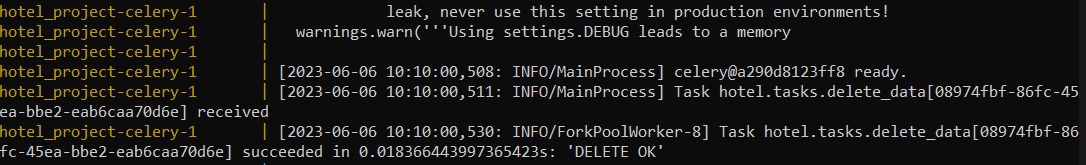


Рисунок 34 – Результат работы периодической функции

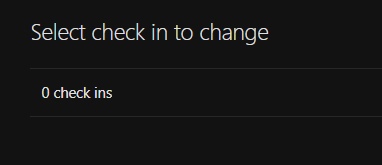


Рисунок 35 – Отсутствие записи о заселении

При отсутствии записей о заселении, которые необходимо удалить, периодическая функция будет выдавать результат, представленный на рисунке 36.

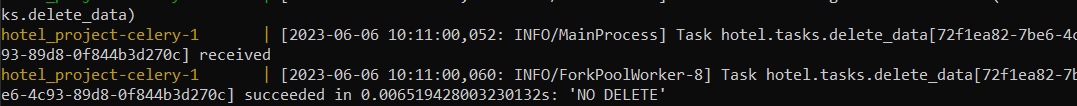


Рисунок 36 – Результат работы периодической функции

## 4.3 Описание планов по дальнейшей разработке системы

В данном разделе представлены планы и стратегия дальнейшей разработки системы сбора данных о загрузке жилых апартаментов. Развитие и усовершенствование этой системы играют важную роль в оптимизации работы отельных комплексов. Развитие сосредоточенно на различных аспектах, включая улучшение существующей функциональности, добавление новой, повышение эффективности и оптимизация. В рамках текущей системы сбора данных о загрузке жилых апартаментов планируем внести следующие улучшения:

1. Большая автоматизация сбора данных: интеграция с другими автоматическими системами считывания данных, чтобы собирать информацию о загрузке апартаментов в режиме реального времени без участия персонала.
2. Улучшение точности данных: разработка алгоритмов или методов проверки и фильтрации данных, чтобы устранить ошибки или неполные записи, обеспечивая точность и надежность получаемой информации.
3. Оптимизация: Оптимизация обработки сигналов со стороны веб-сервера позволит снизить нагрузку на сервер и базу данных. Это может включать в себя оптимизацию алгоритмов обработки запросов, кэширование данных, параллельную обработку или использование асинхронных операций. Изменение принципов срабатывания датчиков так же позволит улучшить эффективность системы сбора данных. Например, можно настроить датчики на работу с определенным интервалом времени или изменить пороговые значения, при которых датчик активируется. Это позволит снизить количество отправляемых запросов на веб-сервер и уменьшить нагрузку на систему.

На текущем этапе система сбора данных представляет собой удобную и простую систему для хранения и сбора информации. Реализованный функционал обеспечивает точность данных, простое и удобное их хранение, а также осуществляет автоматический сбор этих данных. Автоматическое изменение времени заселения, недорогое и простое в настройке оборудование, а также удобное хранение и передача информации - все это вместе и составляет разработанную систему сбора данных.

# Разработка и стандартизация программных средств

В данный раздел входят решения и результаты, полученные по следующим вопросам:

● планирование работ проекта, в рамках техно-рабочего проектирования с использованием диаграмм Ганта (ленточных диаграмм);

● определение кода разрабатываемого программного изделия в соответствии с общероссийскими классификаторами продукции;

● расчет затрат на выполнение и внедрение проекта, расчет цены проекта и цены предлагаемого программного продукта.

* 1. **Планирование работ проекта ПС**

Описываемый проект можно разделить на следующие части работы:

1. Поиск литературы и подробное изучение выбранной темы работы;
2. Составление плана проведения работы;
3. Реализация практической части;
4. Создание подробного описания практической части;
5. Формирование пояснительной записки;
6. Проверка на антиплагиат;

В соответствии со списком работ и планируемыми сроками их выполне- ния была составлена диаграмма Ганта, представленная на рисунке 32.



Рисунок 32 – Диаграмма Ганта выполнения проекта

* 1. **Определение затрат на выполнение и внедрение проекта ПС**

Перед началом разработки проекта важно определить его себестоимость и цену. Однако на этапе заключения договора сложно провести детальный расчёт себестоимости, который будет убедительным для заказчика. В таких случаях можно использовать методику укрупнённого расчёта базирующаяся на расчете затрат и цены одного рабочего дня проектировщика.

* + 1. **Расчет себестоимости продукта**

Расчет начинается с определения месячной зарплаты разработчика.

За основу взяты данные career.habr.com (<https://career.habr.com/salaries>). По этим данным средняя по стране в сфере IT зарплата составляет 180 000 руб. в месяц.

Для начинающего разработчика, выполняющего ВКР, зарплата составлять 60 000 руб.

Следовательно, значение среднемесячной зарплаты разработчика 𝑍зп

примем за 60 000 руб./мес.

𝑍зп = 60 000 руб./мес.

Среднемесячное количество рабочих дней = 20,58 (https://www.consultant.ru/law/ref/calendar/proizvodstvennye/2023/)

Тарифная дневная ставка разработчика 𝑍д:

𝑍 = 𝑍зп = 60 000 руб./мес. = 2915,45 руб./день

д 𝑇ср

20,58 дней/мес.

Процент страховых взносов, исчисляемых от фонда заработной платы

Ф, определим согласно действующему законодательству:

* Обязательное пенсионное страхование (ОПС) – 22%;
* Обязательное медицинское страхование (ОМС) – 5,1%;
* Обязательное социальное страхование на случай временной нетрудо- способности и в связи с материнством (ВНиМ) – 2,9%;
* Обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний (страхование от несчастных случаев) – 0,2%.

Итоговый процент страховых взносов является суммой вышеперечисленных процентов страховых взносов:

Ф = 22% + 5,1% + 2,9% + 0,2% = 30,2%

Величина страховых взносов на работника в день 𝐶сд:

𝐶сд = 𝑍д ∗ Ф = 2915,45 руб./день ∗ 30,2% = 880,47 руб./день

Дневная оплата труда работника с учетом страховых взносов 𝑍дс:

𝑍дс = 𝑍д + 𝐶сд = 2915,45 руб./день + 880,47 руб./день = 3 795,92 руб./день

Процент накладных расходов Н для проектных организаций в сфере ин- форматики колеблется, как правило, от 40% до 80%. Для условий ЛЭТИ может быть принят уровень накладных расходов 42%:

𝐻 = 42%

Накладные расходы на одного работника в день 𝐶нр:

𝐶нр = 𝑍д ∗ 𝐻 = 2 915,45 руб./день ∗ 42% = 1 224,49 руб./день

Стоимость человеко/дня 𝐶ч/д:

𝐶ч/д = 𝑍дс + 𝐶нр = 3 795,92 руб./день + 1 224,49 руб./день

= 5 020,41 руб./день

Средняя прибыль исполнителя П при выполнении проектов в сфере ИТ находится, как правило, в диапазоне от 15% до 20% (в отдельных случаях 25%). Для примера возьмем 16%:

П = 16%

Дневная прибыль на одного работника 𝐶прд:

𝐶прд = 𝐶ч/д ∗ П = 5 020,41 руб./день ∗ 16% = 803,27 руб./день

Дневная ставка специалиста (без учета НДС) Сдсс:

𝐶дсс = 𝐶ч/д + Спрд = 5 020,41 руб./день + 803,27 руб./день

= 5 823,67 руб./день

Ставка налога на добавленную стоимость НДС (по состоянию на 2023 год) равняется 20%:

НДС = 20%

Величина налога на добавленную стоимость в расчете на день 𝐶ндс:

𝐶ндс = 𝐶дсс ∗ НДС = 5 823,67 руб./день ∗ 20% = 1 164,73 руб./день

Полная дневная стоимость работы специалиста 𝐶полн:

𝐶полн = 𝐶дсс + 𝐶ндс = 5 823,67 руб./день + 1 164,73 руб./день

= 6 988,41 руб./день

Сведем рассчитанные значения в таблицу 4.1. Также добавим в таблицу данные для остальных участников проекта – руководителя ВКР и консультанта по доп. Разделу (https://opensalary.ru/reports/regions/universities/).

Таблица 5.1 – Расчет себестоимости рабочего дня

|  |  |  |  |
| --- | --- | --- | --- |
| Статья расходов | Затраты на разработчика | Затраты на  консультанта по доп. разделу | Затраты на  научного руко- водителя ВКР |
| Среднемесячная зарплата разработ- чика, руб./месяц | 60 000 | 67 200 | 67 200 |
| Среднемесячное  число рабочих дней, дней/мес. | 20,58 | 20,58 | 20,58 |
| Расчет стоимости дневной работы специалиста | | | |
| Тарифная дневная ставка разработ- чика, руб./день | 2 915,45 | 3 265,31 | 3 265,31 |
| % страховых взносов | 30,2 | 30,2 | 30,2 |
| Величина  страховых взносов, руб./день | 880,47 | 986,12 | 986,12 |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| Дневная оплата работника со стра- ховыми взносами,  руб./день | 3 795,92 | 4 251,43 | 4 251,43 |
| % накладных расходов | 42 | 42 | 42 |
| Накладные  расходы, руб./день | 1 224,49 | 1 371,43 | 1 371,43 |
| Стоимость человеко/дня,  руб./день | 5 020,41 | 5 622,86 | 5 622,86 |
| % прибыли | 16 | 16 | 16 |
| Дневная прибыль, руб./день | 803,27 | 899,66 | 899,66 |
| Дневная ставка спе- циалиста (без НДС),  руб./день | 5 823,67 | 6 522,51 | 6 522,51 |
| Ставка НДС, % | 20 | 20 | 20 |
| Величина НДС, руб./день | 1 164,73 | 1 304,50 | 1 304,50 |
| Полная стоимость работы специалиста, руб./день | 6 988,41 | 7 827,02 | 7 827,02 |

Рассчитав полную дневную стоимость работы участвующих специалистов, рассчитаем общую себестоимость разработки проекта 𝐶пр:

𝑛

𝐶пр = ∑ 𝐶полн 𝑖 ∗ 𝑇𝑖 ∗ 𝐾загр 𝑖

𝑖=1

Где 𝐶полн 𝑖 - полная дневная стоимость работы *i*-го специалиста (руб./день)

𝑇𝑖 – время участия *i*-го специалиста в работе над проектом (дней)

𝐾загр 𝑖 *–* коэффициент загрузки *i*-го специалиста работами в проекте

𝑛 *–* число специалистов, занятых в проекте.

Итого, общая себестоимость разработки проекта:

𝐶пр = (6 988,41 руб./день ∗ 30 дней ∗ 1)

+ (7 827,02 руб./день ∗ 20,58 дней ∗ 0,03)

+ (7 827,02 руб./день ∗ 20,58 дней ∗ 0,05) = 222 838,70 руб.

* + 1. **Расчет цены предлагаемого продукта**

Данный проект подразумевает вывод на рынок нового программного средства. Цена в таком случае может быть определена из соотношения:

𝐶прогр = 𝐶пр + 𝐶изгот(1 + П)(1 + НДС)

Где

𝐶прогр – цена программы, созданной в рамках проекта;

𝐶пр – цена проекта, рассчитанная в разделе 4.2.1;

𝐶изгот – затраты на изготовление копий;

П – прибыль, заложенная разработчиком в цену; НДС – налог на добавленную стоимость.

𝐶прогр = 𝐶пр + (1 + (17% ∗ 𝐶пр))(1 + 20%)

= 222 838,70 руб. +(1 + (0,17 ∗ 222 838,70 руб. )) ∗ 1,2

= 268 298,99 руб.

* 1. **Определение классификационного кода разрабатываемого ПС**

Согласно классификации ОКПД 2, проект, разрабатываемый в рамках данной ВКР, можно отнести к следующему коду:

62.01.11.000 «Услуги по проектированию и разработке информацион- ных технологий для прикладных задач и тестированию программного обеспе- чения».

* 1. **Определение списка стандартов для ВКР**

В данном подразделе определен список стандартов, которые были использованы в рамках выполнения ВКР. Определение списка стандартов является важным шагом в процессе подготовки ВКР и позволяет убедиться в его правильности и полноте. К таким стандартам относятся прежде всего:

- стандарты, регулирующие порядок и требования к содержанию проектных работ в предметной области ВКР;

- стандарты, регулирующие оформление проектной документации.

Перечень стандартов, который был отобран из первой группы, включает следующие пункты:

* + 1. ГОСТ 34.601-90. «Информационная технология. Автоматизированные системы. Стадии создания».
    2. ГОСТ 34.602-89. «Информационная технология. Техническое задание на создание автоматизированной системы».
    3. ГОСТ 19.201-78 «Единая система программной документации. Техническое задание. Требования к содержанию и оформлению».
    4. РД 50-34.698-90. «Методические указания. Информационная технология. Автоматизированные системы. Требования к содержанию документов».

Так же был осуществлен выбор стандартов, которые входят во вторую группу, и они включают в себя следующие пункты:

1. ГОСТ 7.32 — 2017 «Отчет о научно-исследовательской работе. Структура и правила оформления»
2. ГОСТ 7.11—78 «Сокращение слов и словосочетаний на иностранных языках в библиографическом описании»

ЗАКЛЮЧЕНИЕ

Основная цель данной дипломной работы заключалась в разработке системы для сбора и хранения данных о загрузке жилых апартаментов в отеле. В процессе выполнения проекта был проведен анализ различных инструментов разработки и создан прототип такой системы. Эта система предоставляет удобство в сборе данных, снижает нагрузку на персонал, обеспечивает простую настройку, эффективное хранение и передачу данных.

Для разработки системы было использовано следующее оборудование: плата Arduino, Wi-Fi модуль ESP8266 и пироэлектрический датчик движения. База данных PostgreSQL была выбрана в качестве хранилища данных. Использование этих инструментов позволило создать простую, но эффективную автоматическую систему сбора и хранения информации. Следует отметить, что все выбранные компоненты оборудования являются доступными и недорогими.

Выполнение данной работы привело к успешной реализации поставленных целей и созданию системы сбора данных о загрузке жилых апартаментов в отеле. Разработанная система значительно упрощает и автоматизирует некоторые задачи, обеспечивая грамотный сбор, обработку и хранение данных.

В дальнейшем планируется расширение функционала системы путем улучшения ее автоматизации и оптимизации текущих процессов сбора данных. Развитие системы в этих направлениях способствует ее усовершенствованию и повышению эффективности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Полное практическое руководство по Docker: с нуля до кластера на AWS. URL: https://habr.com/ru/articles/310460/.
2. Docker Docs: How to build, share, and run applications. URL: <https://docs.docker.com/>.
3. База данных: что такое БД, их типы, свойства, структура. URL: https://practicum.yandex.ru/blog/chto-takoe-bazy-dannyh/.
4. Руководство по работе с базами данных PostgreSQL. URL: https://metanit.com/sql/postgresql/.
5. Руководство по языку программирования Python. URL: https://metanit.com/python/tutorial/.
6. Django documentation. URL: <https://docs.djangoproject.com/en/4.2/>.
7. Что такое Arduino? URL: https://alexgyver.ru/lessons/about-arduino/.
8. Лучшие аналоги Arduino. URL: <https://losst.pro/luchshie-analogi-arduino/>.
9. Виды датчиков движения и их особенности. URL: https://www.fedomo.ru/article/vidy-datchikov-dvizheniya-i-ih-osobennosti/.
10. Пироэлектрический инфракрасный (PIR) датчик движения и Arduino. URL: https://arduino-diy.com/arduino-piroelektricheskiy-infrakrasnyy-PIR-datchik-dvizheniya/.
11. Arduino Documentation. URL: <https://docs.arduino.cc/>.
12. Пироэлектрический инфракрасный (PIR) датчик движения и Arduino. URL: <https://arduino-diy.com/arduino-piroelektricheskiy-infrakrasnyy-PIR-datchik-dvizheniya/>.
13. Методы сбора данных: Источники и примеры. URL: <https://hr-portal.ru/story/metody-sbora-dannyh-istochniki-i-primery/>.
14. Точные измерения аналоговых сигналов в системах сбора данных. URL: <https://russianelectronics.ru/tochnye-izmereniya-analogovyh-signalov-v-sistemah-sbora-dannyh//>.
15. Электронные ключи и карты. URL:  [https://blog.regolit.com/2023/03/03/key-cards-for-access-control-systems/](https://arduino-diy.com/arduino-piroelektricheskiy-infrakrasnyy-PIR-datchik-dvizheniya/).
16. Как работают датчики движения. URL:  [https://anlan.ru/articles/detektory-dvizheniya-vse-chto-vam-nuzhno-znat/](https://arduino-diy.com/arduino-piroelektricheskiy-infrakrasnyy-PIR-datchik-dvizheniya/).
17. People Counter. URL: https://en.wikipedia.org/wiki/People\_counter/.
18. Smart Building. URL: https://evvr.io/blogs/newsroom-2/smart-building-technology-solutions-and-trends/.
19. PointGrab. URL: <https://www.pointgrab.com/>.
20. Docker Compose overview. URL: https://docs.docker.com/compose/.
21. Подробный обзор PIR датчика движения. URL: https://vidsyst.ru/datchik/datchik-dvizheniya/pir.html/.

ПРИЛОЖЕНИЕ А

**Спецификация программных модулей**

В таблице 1 представлена спецификация программных модулей системы сбора данных.

Таблица 1 – Спецификация программных модулей

|  |  |
| --- | --- |
| Название модуля | Назначение |
| Urls | Маршруты и связи между адресами URL и обработчиками |
| Views | Обработчики запросов, поступающих на сервер |
| Celery | Настройки периодически выполняемых задач |
| Tasks | Периодические задачи |
| Arduino | Обработчик и отправитель сигналов датчиков движения |

ПРИЛОЖЕНИЕ Б

**Исходный код программы**

Arduino

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>  
#include <WiFiClient.h>

const char\* ssid = "Your WiFi SSID"; // Имя вашей Wi-Fi сети

const char\* password = "Your WiFi Password"; // Пароль вашей Wi-Fi сети

const char\* server = "Your Server IP Address"; // IP-адрес вашего веб-сервера Django

const sensorPin = 13;

void setup() {

Serial.begin(115200); // Инициализация последовательного порта для вывода отладочной информации

WiFi.begin(ssid, password); // Подключение к Wi-Fi сети

while (WiFi.status() != WL\_CONNECTED) { // Ожидание установления соединения с Wi-Fi

delay(1000);

Serial.println("Connecting to WiFi...");

}

WiFiClient client;

Serial.println("Connected to WiFi");

}

void loop(){

int motionValue = digitalRead(sensor1Pin);

if (motionValue == 1){

motionValue = 4;

post\_data();

}

delay(1000);

}

void post\_data() {

// Формирование строки URL с данными

String url = "http://" + String(server) + "/data/";

// Создание объекта HTTPClient

HTTPClient http;

// Установка URL и типа контента запроса

http.begin(client, url);

http.addHeader("Content-Type", "application/x-www-form-urlencoded");

// Формирование тела запроса с данными датчиков

String requestBody = "sensor1=" + String(motionValue);

// Отправка POST запроса с данными датчиков

int httpResponseCode = http.POST(requestBody);

if (httpResponseCode > 0) {

String response = http.getString(); // Получение ответа от сервера

Serial.println(httpResponseCode);

Serial.println(response);

} else {

Serial.print("Error code: ");

Serial.println(httpResponseCode);

}

http.end(); // Завершение HTTP-соединения

}

Urls

urlpatterns = [

path('data/', views.data\_handler, name='data-handler'), #Ссылка для принятия данных с Arduino

]

Views

def data\_handler(request):

if request.method == 'POST': # Проверка метода

sensor1 = request.POST.get('sensor1') # Считывание значения датчика

now = datetime.datetime.now()

if now.time() >= datetime.time(hour=13, minute=0):

checkin = CheckIn.objects.get( # Получение нужной записи заселения

room=sensor1,

check\_in\_\_contains=datetime.date.today(),

)

if checkin.check\_in.time() == datetime.time(0, 0, 0): # Проверка необходимости менять время

checkin.check\_in = now # Установка нужного времени заезда

checkin.save()

if now.time() <= datetime.time(hour=12, minute=0):

checkout = CheckIn.objects.get(

room=sensor1,

check\_out\_\_contains=datetime.date.today(),

)

checkout.check\_out = now # Установка нужного времени выезда

checkout.save()

return HttpResponse(200)

Celery

import os

from celery import Celery

from celery.schedules import crontab

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "hotel\_app.settings") # Указание файла с настройками

app = Celery("hotel\_app", include=['hotel.tasks']) # Создание экземпляра celery

app.config\_from\_object("django.conf:settings", namespace="CELERY") # Откуда брать настройки

app.autodiscover\_tasks() # Автоматическое обнаружение тасков

app.conf.beat\_schedule = { # Настройки каждого таска

'delete': {

'task': 'hotel.tasks.delete\_data',

'schedule': crontab(hour='\*/24'),

},

}

Tasks

import datetime

from hotel\_app.celery import app

from adminworkenv.models import CheckIn

@app.task(ignore\_result=False) # Пометка функции как таск

def delete\_data():

now = datetime.datetime.now()

checkin = CheckIn.objects.filter( # Получение записей о заселении с датой создания 91 день назад

created\_at\_\_contains=datetime.date.today() - datetime.timedelta(days=91),

).delete() # Удаление данных

if checkin[0] == 0:

return 'NO DELETE'

else:

return 'DELETE OK' # Возврат результата