

**Министерство образования и науки Российской Федерации**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  
**“УНИВЕРСИТЕТ ИТМО”**

Факультет      ИКТ

Образовательная программа 45.03.04 – Интеллектуальные системы в гуманитарной сфере

Направление подготовки (специальность) 45.03.04 – Интеллектуальные системы в гуманитарной сфере

## О Т Ч Е Т

по курсовой работе

Тема задания: Реализация web-сервисов средствами Django REST framework, Vue.js

Обучающийся Ларионова Мария Владиславовна, гр. К3343

Руководитель: Говоров А. И., ассистент факультета ИКТ Университета ИТМО

Оценка за курсовую работу \_\_\_\_

Подписи членов комиссии:

\_\_\_\_\_ (            )  
(подпись)

\_\_\_\_\_ (            )  
(подпись)

\_\_\_\_\_ (            )  
(подпись)

Дата \_\_\_\_

Санкт-Петербург  
20 20

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ .....	4
1.1. Описание предметной области .....	4
1.2. Описание функциональных требований .....	4
1.3. Выводы .....	5
2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА .....	6
2.1. Описание архитектуры сервиса .....	6
2.2. Архитектура веб-приложения .....	7
2.3. Модель данных .....	8
2.4. Выводы .....	8
3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ .....	9
3.1. Описание средств разработки серверной части .....	9
3.2. Реализация базы данных .....	9
3.3. Сериализация .....	10
3.4. Создание представлений .....	10
3.5. Настройка маршрутизации .....	11
3.6. Выводы .....	12
4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ .....	13
4.1. Описание средств разработки клиентской части .....	13
4.2. Разработанные интерфейсы .....	13
4.2.1. Вход в систему .....	13
4.2.2. Интерфейсы для администратора .....	14
4.2.3. Интерфейсы для служащего персонала (уборщика) .....	16
4.2.4. Интерфейсы для потенциального посетителя отеля .....	17
4.2.5. Интерфейсы для потенциального соискателя работы в отеле .....	19
4.3. Выводы .....	20
5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ .....	21
5.1. Описание средства контейнеризации и оркестрации .....	21
5.2. Контейнеризация проекта .....	21
5.3. Оркестрация проекта .....	22
5.4. Выводы .....	23
ЗАКЛЮЧЕНИЕ .....	24
СПИСОК ЛИТЕРАТУРЫ .....	25

## **ВВЕДЕНИЕ**

В настоящее время компьютерные технологии развиваются особенно активно. Применяются новые техники, совершенствуются старые. Специалист по компьютерным технологиям должен быть человеком, который обладает всеми необходимыми знаниями для реализации того или иного проекта и который может найти решение для любой технической задачи. Реализация веб-сервисов требует как теоретических, так и практических знаний, а также современных навыков и умений, поскольку данная сфера компьютерных технологий развивается очень активно и требует от специалиста большого опыта.

Целью данной курсовой работы является разработка веб-сервиса. В процессе реализации проекта необходимо было изучить различные инструменты, используемые при создании веб-сервисов, а также научиться применять их на практике.

В процессе работы должны быть выполнены следующие задачи:

1. Изучение предметной области.
2. Анализ функциональных требований.
3. Проектирование архитектуры веб-сервиса.
4. Разработка серверной части системы.
5. Разработка клиентской части системы.
6. Контейнеризация проекта.

В первой главе данного отчета представлен анализ предметной области и функциональных требований. Вторая глава посвящена проектированию архитектуры веб-сервиса. В третьей и четвертой главах рассмотрен подход к разработке серверной и клиентской частей системы соответственно.

# **1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

## **1.1. Описание предметной области**

В данной работе был реализован вариант, посвященный созданию программной системы для администратора отеля или гостиницы. Веб-сервис предназначен для служащих гостиницы (рабочий персонал), а также администратора гостиницы.

Сервис представлен следующими необходимыми для отеля или гостиницы процедурами: оформление новые посетителей гостиницы, рассмотрение резюме потенциальных соискателей работы в отеле, а также прием или отклонение заявок на работу в гостинице. В данной системе содержится информация о клиентах, занимающих те или иные номера, о пустых номерах, не занятых постояльцами, о служащих, осуществляющих уборку номеров, а также их расписании работы. Отель представляет на выбор три типа различных номеров: номер на одного человека, номер для двух человек, номер для трех человек. Бронируя номер, клиент сообщает свое имя, фамилию, номер паспорта, дату заезда, город, из которого он приехал, гостиничный номер, который он забронировал. О рабочем персонале отеля известна следующая информация: фамилия служащего, имя, отчество, этаж, на котором он(а) должен произвести работу и день, в который он должен произвести работу.

## **1.2. Описание функциональных требований**

Проанализировав предметную область данного проекта, можно сделать вывод о функциональных требованиях, которые должны быть реализованы в разрабатываемой системе. В системе должна храниться вся необходимая информация о номерах, клиентах и служащем персонале отеля или гостиницы.

Реализуемая система должна предоставлять пользователю информацию о клиентах, заселенных в те или иные номера, о прибыли, которую на данный момент имеет отель с заселенных номеров, просмотр графика работы служащего персонала, осуществляющего уборку того или иного этажа в тот или иной день, а также просмотр заявлений на работу от потенциальных соискателей.

Администратор отеля имеет возможность осуществить следующие действия:

- Посмотреть информацию о занятых номерах и посетителях отеля
- Посмотреть информацию о графике работы служащего персонала
- Принять или отклонить заявку потенциального соискателя на должность в отеле

Работник служащего персонала должен иметь возможность:

- Просмотреть график уборки, по которому он(а) должен работать
- Отправить заявку на рассмотрение о принятии его на должность администратора

Потенциальный посетитель отеля должен иметь возможность:

- Оставить заявку о бронировании номера в отеле

### **1.3. Выводы**

Был проведен тщательный анализ предметной области реализуемого веб-сервиса и функциональных требований системы. В ходе анализа были учтены все нюансы, необходимые для реализации веб-сервиса и подготовлен алгоритм выполнения работы.

## 2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА

### 2.1. Описание архитектуры сервиса

Для разработки данного веб-сервиса, описываемого в данной курсовой работе, использовалась архитектура «клиент-сервер». Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций).

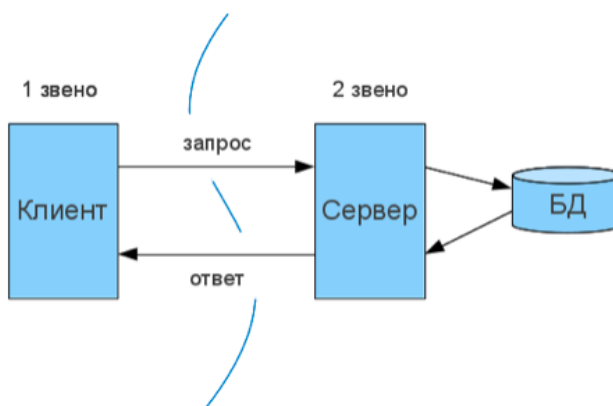


Рисунок 1 – Архитектура «Клиент-Сервер»

Сервером в данном случае считается абстрактная машина в сети, которая способна получить HTTP-запрос, обработать его и вернуть корректный ответ. Клиентом может считаться все, что способно сформировать и отправить HTTP-запрос. Сервер ожидает от клиента запрос и предоставляет свои ресурсы в виде данных или в виде сервисных функций. [6]

База данных представляет собой третье звено архитектуры. Она нужна для того, чтобы информация могла сохраняться даже при падении и рестарте системы. Наличие базы данных гарантирует облегченный поиск по данным и их сохранность.

Преимуществом использования данной архитектуры является отсутствие дублирования кода, так как сервер и база данных вынесены отдельно и, следовательно, нет необходимости в хранении одинакового кода по обработке логики системы на клиентских машинах. Еще одним преимуществом клиент-серверной архитектуры является повышенная безопасность системы, потому что клиент может видеть только доступную ему информацию.

## 2.2. Архитектура веб-приложения

В соответствии с функциональными требованиями к веб-сервису была разработана архитектура веб-приложения, представленная на рис. 2.

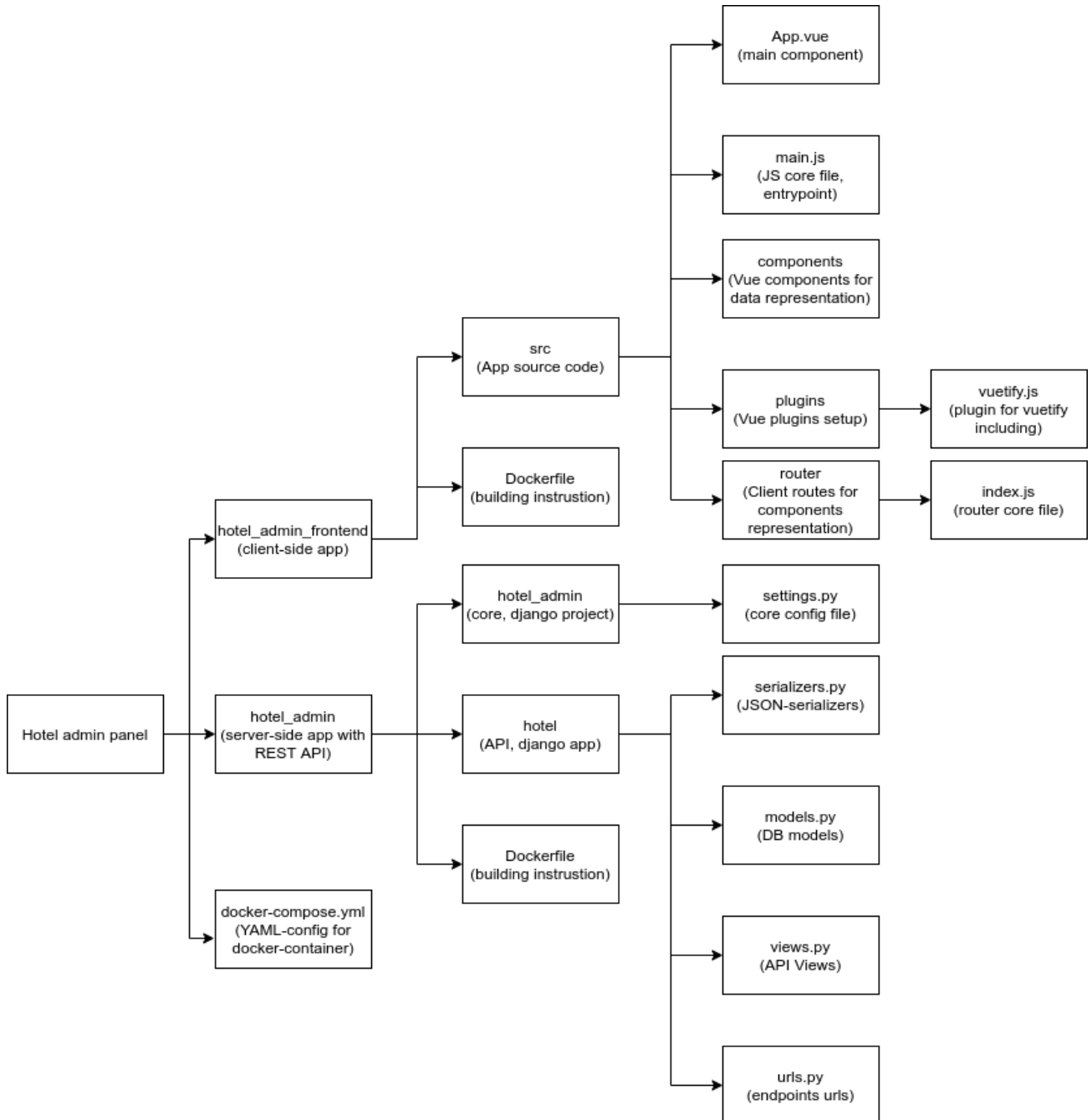


Рисунок 2 – Архитектура веб-приложения

### 2.3. Модель данных

В соответствии с вариантом задания и функциональными требованиями была создана модель данных, представленная на рис. 3.

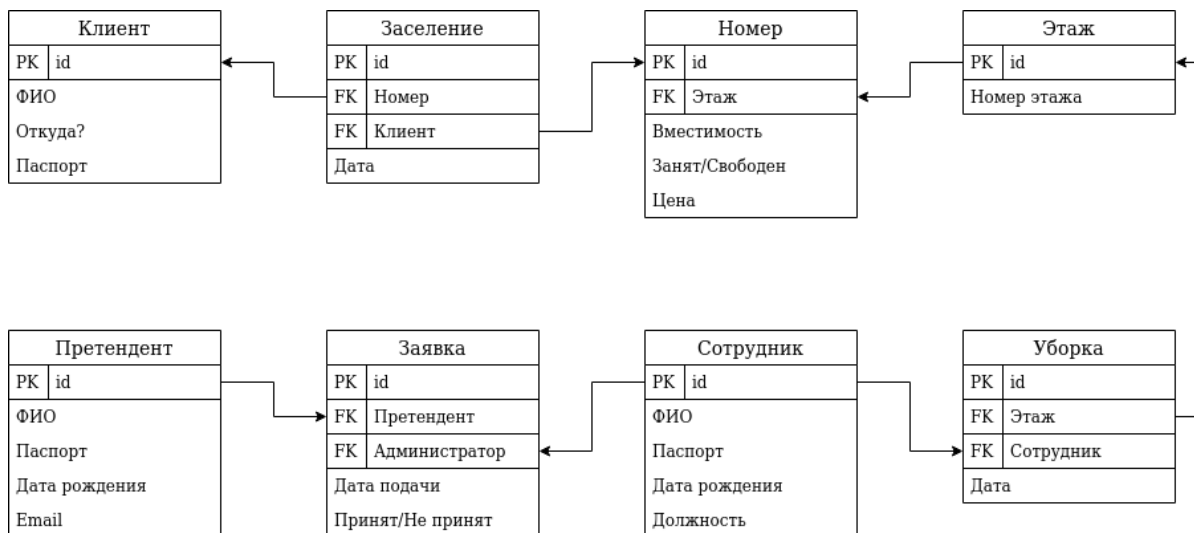


Рисунок 3 – Модель данных

Модель содержит 8 сущностей:

- Клиент
- Номер
- Заселение
- Этаж
- Уборка
- Сотрудник
- Заявка
- Претендент

Присутствуют связывающие сущности: заселение, заявка. Сущность «заселение» хранит информацию о клиенте и занимаемом им гостиничном номере, а сущность заявка – о претендентах, находящихся среди сотрудников отеля, которые бы желали занять позицию администратора.

### 2.4. Выводы

В процессе проектирования архитектуры разрабатываемого веб-сервиса был выбран тип архитектуры «клиент-сервер». На основе функциональных требований к системе была создана архитектура веб-приложения и модель данных.



### **3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ**

#### **3.1. Описание средств разработки серверной части**

В ходе выполнения проекта для реализации его серверной части был задействован фреймворк Django Rest, который является инструментом, основанным на Django, для работы с rest. Этот инструмент широко известен в виду его удобств.

Django – это высокоуровневая веб-инфраструктура языка Python, которая позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

REST (англ. Representational State Transfer, «передача состояния представления») — стиль построения архитектуры распределенного приложения. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол, как и HTTP, должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ». Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями. Кроме того, преимуществами использования REST являются надежность, производительность, прозрачность системы взаимодействия, простота интерфейсов и способность приспосабливаться к новым требованиям. [1]

#### **3.2. Реализация базы данных**

Представленная на рис. 3 модель данных была реализована с помощью СУБД PostgreSQL. PostgreSQL – это свободная объектно-реляционная система управления базами данных (СУБД). Преимуществами данной СУБД являются высокопроизводительные и надежные механизмы транзакций, расширенная система встроенных языков программирования, наследование и расширяемость [2].

Разработанная по представленной модели база данных содержит следующие таблицы:

- Клиент – информация о посетителях гостиницы.
- Номер – информация о номерах в гостинице.
- Заселение – информация о постояльцах отеля, заселившихся в тот или иной номер.
- Этаж – информация об этажах в отеле.
- Сотрудник – информация о служащем персонале отеля.
- Уборка – информация о графике уборок служащего персонала.
- Претендент – информация о соискателе на должность администратора в отеле
- Заявка – информация о заявке на работу, поданной соискателем.

### 3.3. Сериализация

Сериализация представляет собой процесс перевода какой-либо структуры данных в последовательность битов. Иначе говоря, это процесс создания потокового представления данных, которые можно передавать по сети. Обратным процессом является десериализация.

Для разработки веб-приложения с Django Rest были созданы следующие сериализаторы:

- ClientSerializer – используется для получения данных о клиенте.
- CreateClientSerializer – используется для добавления нового клиента.
- RoomSerializer – используется для получения данных о номере.
- EmployeeSerializer – используется для получения данных о работнике.
- ChallengerSerializer – используется для получения данных о соискателе работы.
- FloorSerializer – используется для получения данных об этаже.
- CleaningSerializer – используется для получения данных об уборке.
- RequestSerializer – используется для получения данных о заявках на работу в отеле.
- На рис.4 приведен сериализатор для модели этажа гостиницы.

```
class FloorSerializer(serializers.ModelSerializer):  
    class Meta:  
        model = Floor  
        fields = '__all__'
```

Рисунок 4 – Сериализатор этажа гостиницы

### 3.4. Создание представлений

Представление (view) – это функция обработчик запросов, которая получает HTTP-запросы и возвращает ответы. View имеет доступ к данным через модели, которые определяют структуру данных приложения и предоставляют механизмы для управления базой данных.

В рамках работы были созданы следующие представления:

- CreateFloorView – создание информации об этаже
- RetrieveFloorView – получение данных об определенном этаже
- ListFloorView – получение данных обо всех этажах в отеле.
- CreateRoomView – создание информации о номере в отеле
- ListRoomView – получение информации обо всех номерах в отеле
- RetrieveRoomView – получение данных об определенном номере в отеле
- CreateClientView – создание информации о постояльце отеля
- RetrieveClientView – получение информации о постояльце отеля
- ListClientView – получение данных обо всех посетителях отеля

- CreateClientRoomView – создание информации о заселении посетителя в тот или иной номер
- RetrieveClientRoomView – получение информации о заселении посетителя в тот или иной номер.
- ListClientRoomView – получении информации обо всех посетителях, заселившихся в какой-либо номер
- CreateCleaningView – создание информации об уборке
- RetrieveCleaningView – получение информации об уборке
- ListCleaningView – получение информации о всех уборочных работах
- CreateChallengerView – создание информации о соискателе на работу
- RetrieveChallengerView – получение информации о соискателе на работу
- ListChallengerView – получение информации обо всех соискателях на работу

Для реализации представлений был использован простой класс APIView и более расширенный класс generics. ListAPIView для вывода списков. На рис. 5 представлен пример готового View.

```
class ListCleaningView(generics.ListAPIView):
    serializer_class = CleaningSerializer

    def get_queryset(self):
        queryset = Cleaning.objects.all()

        worker = self.request.query_params.get('worker', None)

        if worker:
            queryset = queryset.filter(employee__user__username=worker)

        return queryset
```

Рисунок 5 – Представление для вывода всех уборочных работ

### 3.5. Настройка маршрутизации

После разработки представлений было необходимо создать URL-адреса для того, чтобы система могла работать должным образом. В системе имеется список основных адресов, который включает адреса приложения и адреса для авторизации. URL-адреса приложения основаны на разработанных ранее представлениях. На рис. 6 представлен список URL-адресов веб-приложения.

```
urlpatterns = [
    path('floor/new', CreateFloorView.as_view()),
    path('floor/<int:pk>', RetrieveFloorView.as_view()),
    path('floor/all', ListFloorView.as_view()),
    path('room/new', CreateRoomView.as_view()),
    path('room/<int:pk>', RetrieveRoomView.as_view()),
    path('room/all', ListRoomView.as_view()),
    path('client/new', CreateClientView.as_view()),
    path('client/<int:pk>', RetrieveClientView.as_view()),
    path('client/all', ListClientView.as_view()),
    path('client/room/new', CreateClientRoomView.as_view()),
    path('client/room/<int:pk>', RetrieveClientRoomView.as_view()),
    path('client/room/all', ListClientRoomView.as_view()),
    path('cleaning/new', CreateCleaningView.as_view()),
    path('cleaning/<int:pk>', RetrieveCleaningView.as_view()),
    path('cleaning/all', ListCleaningView.as_view()),
    path('challenger/new', CreateChallengerView.as_view()),
    path('request/new', CreateRequestView.as_view()),
    path('request/update/<int:pk>', UpdateRequestView.as_view()),
    path('request/all', RetrieveRequestView.as_view()),
    path('auth/', include('djoser.urls')),
    path('auth/token', obtain_auth_token, name='token')
]
```

Рисунок 6 – Список url-адресов приложения

### 3.6. Выводы

Благодаря инструментам фреймворка Django REST был разработан бэкенд системы для управления отелем. Были созданы и описаны сериализаторы, представления и url-адреса веб-приложения.

## 4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ

### 4.1. Описание средств разработки клиентской части

В разработке клиентской части веб-сервиса был задействован фронтенд-фреймворк Vue.js и Vuetify UI фреймворк.

Vue.js — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками. [3]

Библиотека Vuetify UI является одной из самых популярных библиотек для Vue.js и активно используется для создания разных проектов. Лаконичный дизайн и простота использования являются главными преимуществами.

### 4.2. Разработанные интерфейсы

#### 4.2.1. Вход в систему

При запуске системы пользователю предлагается войти в систему. Для этого ему необходимо ввести свой логин и пароль. Войти в систему может администратор или, например, сотрудник рабочего персонала (уборщик) или потенциальный соискатель вакансии.

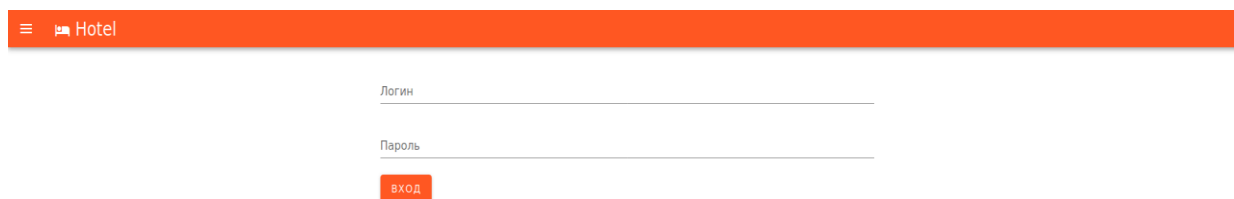


Рисунок 7 – Страница авторизации

#### 4.2.2. Интерфейсы для администратора

Войдя в систему как администратор гостиницы, пользователь попадает на главную страницу системы (рис. 10).

☰ 🏠 Hotel		
<b>Кабинет администратора</b>		
<b>Статистика</b>		
Свободных номеров на 1 персону: 2		
Свободных номеров на 2 персоны: 2		
Свободных номеров на 3 персоны: 2		
Актуальный доход с занятых номеров: 600		
<b>Расписание смен служащих</b>		
Дата	Этаж	Служащий
2020-06-06	1	Popov Oleg Petrovich

Рисунок 8 – Интерфейс кабинета администратора

Интерфейс «Кабинет администратора» доступен администратору отеля, который будет получать всю необходимую информацию о постояльцах отеля, номерах и сотрудниках отеля. В данном интерфейсе представлена информация о количестве свободных номеров, разделенных по их вместительности (на 1 персону, на 2 персоны, на 3 персоны), информация о доходе, полученном с занятых номеров, а также информация о расписании смен служащих отеля, осуществляющих уборку номеров. В расписании смен служащих администратор отеля может увидеть фамилию, имя, отчество служащего персонала, а также этаж, на котором он будет работать и дату, в которую он будет работать.

Также администратор может просмотреть все поданные заявления о принятии на работу в Django Administration. Будет отображена вся оставленная потенциальным сотрудником информация, а также будет отображен статус заявки: принят ли человек на работу (hired) или не принят (not hired).

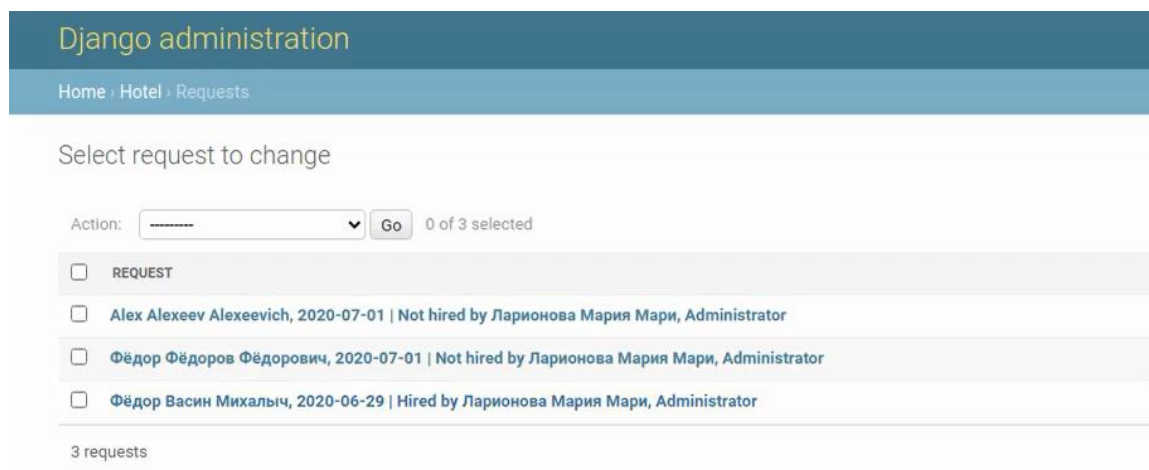


Рисунок 9 – Просмотр оставленных заявок в Django Administration

Администратор сможет просмотреть каждую заявку в отдельности, а также внести необходимые коррективы, например, назначить заявку на другого администратора или поменять статус заявку (принять человека на работу или, наоборот, отклонить его заявление).

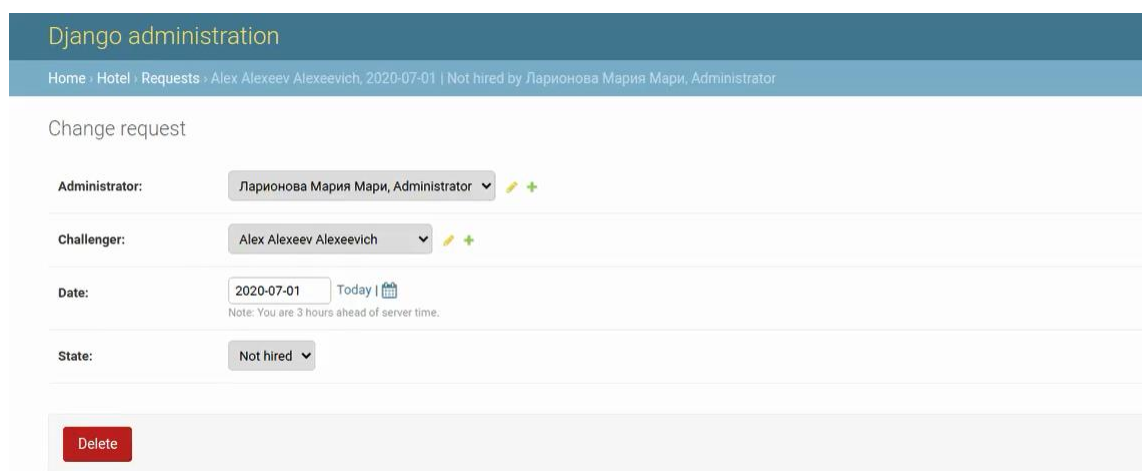


Рисунок 10 – Просмотр отдельно выбранной заявки в Django Administration

Примером запроса, осуществляемым от имени администратора может быть доход отеля, который отображается в кабинете администратора



Рисунок 11 - Реализация запроса «доход с занятых номеров»

### 4.2.3. Интерфейсы для служащего персонала (уборщика)

Интерфейс «Кабинет служащего» предназначен для рабочего персонала отеля, занимающегося уборкой номеров. Используя данный интерфейс, служащий увидит информацию об этаже, который закреплен к нему в тот или иной день для проведения там работ, а также дату, когда служащий должен произвести уборку этого этажа. На рисунке 6 представлена реализация интерфейса «Кабинет служащего».

The screenshot shows the 'Кабинет служащего' (Staff Office) interface. It has an orange header bar with a menu icon and the text 'Hotel'. Below the header, the title 'Кабинет служащего' is displayed. A table with two columns, 'Дата' (Date) and 'Этаж' (Floor), is shown. The table contains one row of data.

Дата	Этаж
2020-06-06	1

Рисунок 12 - Реализация интерфейса “Кабинет служащего”



#### 4.2.4. Интерфейсы для потенциального посетителя отеля

Посетителю веб-сервиса будет представлен интерфейс «Главная страница», который отображает информацию о названии отеля, свободных номерах, имеющихся на данный момент, а также ценах на номера. Здесь же представлены фото-превью каждого номера, чтобы потенциальный путешественник смог сориентироваться, что из себя представляет тот или иной номер.

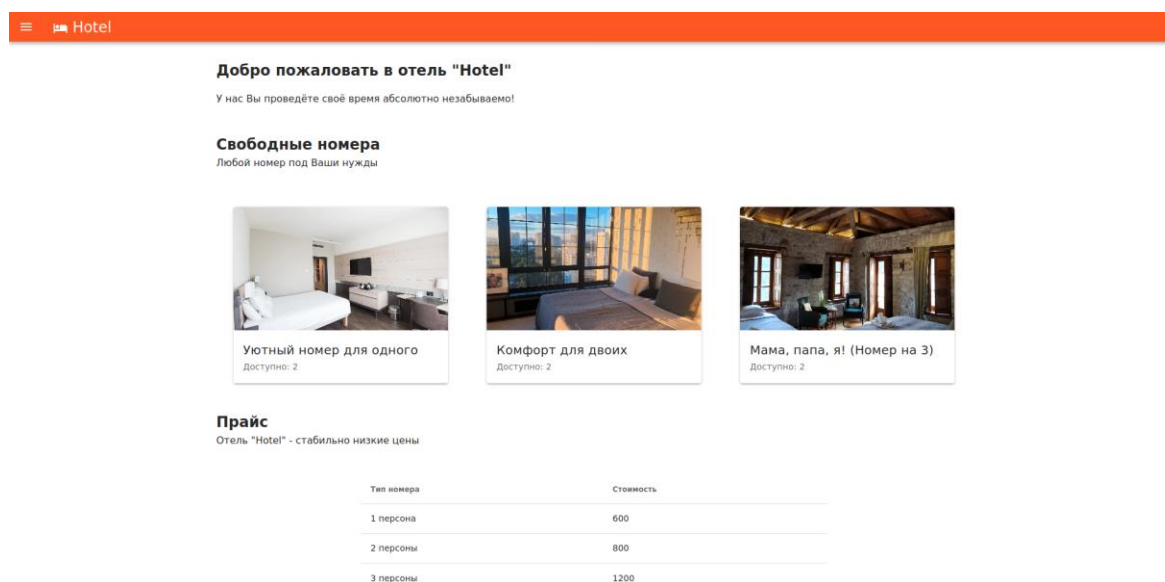
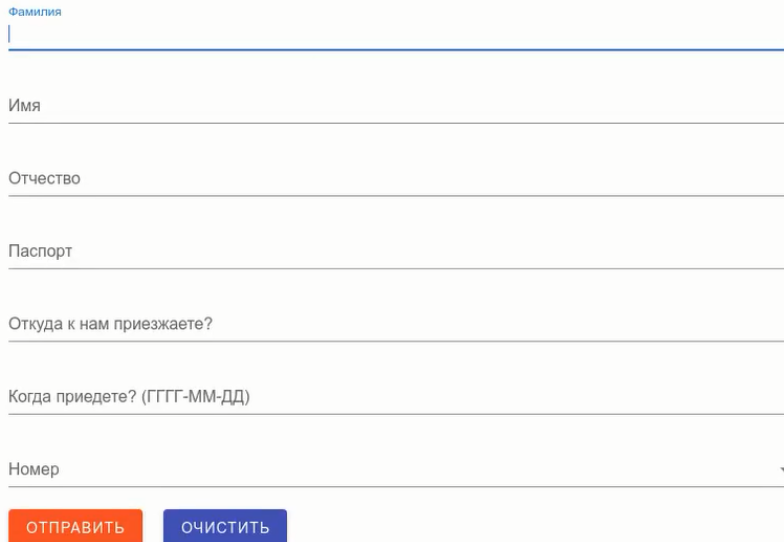


Рисунок 13 – Интерфейс главной страницы

Для того чтобы забронировать номер, путешественнику нужно заполнить форму, состоящую из следующих полей: фамилия, имя, отчество, паспорт, город, из которого путешественник приезжает, дату приезда. В последнем поле «номер» путешественник сможет выбрать понравившийся ему номер из предложенных на сайте. Ему будет отображен список номеров, которые свободны на данный момент, и он сможет выбрать один из них, после чего уже отправить свое бронирование. На рисунке 8 представлена реализация интерфейса «бронирование номера».

## Арендовать номер

Просто заполните форму и мы с Вами свяжемся



Фамилия

Имя

Отчество

Паспорт

Откуда к нам приезжаете?

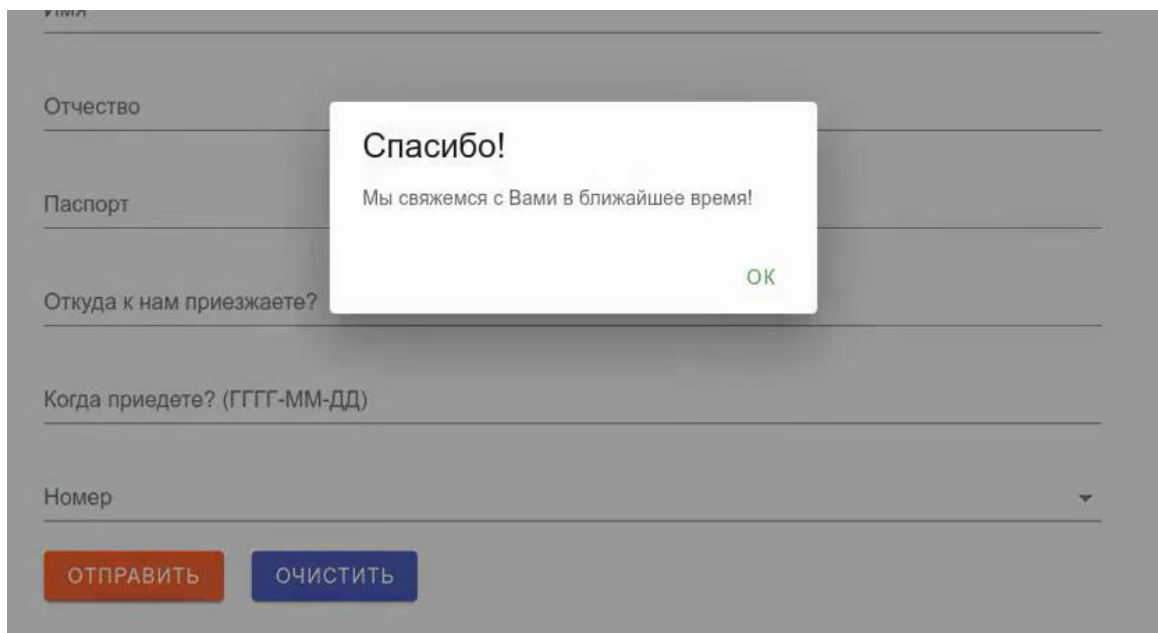
Когда приедете? (ГГГГ-ММ-ДД)

Номер

ОТПРАВИТЬ ОЧИСТИТЬ

Рисунок 14 - Реализация интерфейса “Бронирование номера”

После отправки формы бронирования номера пользователь увидит модальное окно, на котором будет написано, что его заявка на бронирование номера получена и что с ним свяжутся.



Отчество

Паспорт

Откуда к нам приезжаете?

Когда приедете? (ГГГГ-ММ-ДД)

Номер

ОТПРАВИТЬ ОЧИСТИТЬ

**Спасибо!**  
Мы свяжемся с Вами в ближайшее время!  
ОК

Рисунок 15 – Модальное окно, всплывающее после отправки формы бронирования

#### 4.2.5. Интерфейсы для потенциального соискателя работы в отеле

Пользователь, заинтересованный в работе в данном отеле, может заполнить форму подачи заявки на работу. Для этого ему нужно будет заполнить поля: фамилия, имя, отчество, номер паспорта, email, дата рождения, имя пользователя и пароль.

Как устроиться служащим?  
Всё просто! Заполните анкету:

Фамилия

Имя

Отчество

Паспорт

Email

Дата рождения (ГГГГ-ММ-ДД)

Имя пользователя

Пароль

ОТПРАВИТЬ ОЧИСТИТЬ

Рисунок 16 – Реализация интерфейса «Подача заявления на работу»

После заполнения формы пользователь увидит модальное окно, на котором будет написано, что с ним свяжутся в ближайшее время.

Спасибо!

Мы свяжемся с Вами в ближайшее время!

ОК

Рисунок 17 – Модальное окно, всплывающее после отправки формы заявления на работу

Отправив заявление о принятии на работу, соискатель сможет просматривать статус отправленной им заявки в «Кабинете претендента». Зайдя в личный кабинет, он увидит оставленную им информацию о заявке, а именно: его фамилию, имя, отчество, дату рождения, номер паспорта, должность, на которую он подал заявку. Также он увидит информацию о статусе его заявки (принят ли он или нет) и дату изменения статуса заявки.

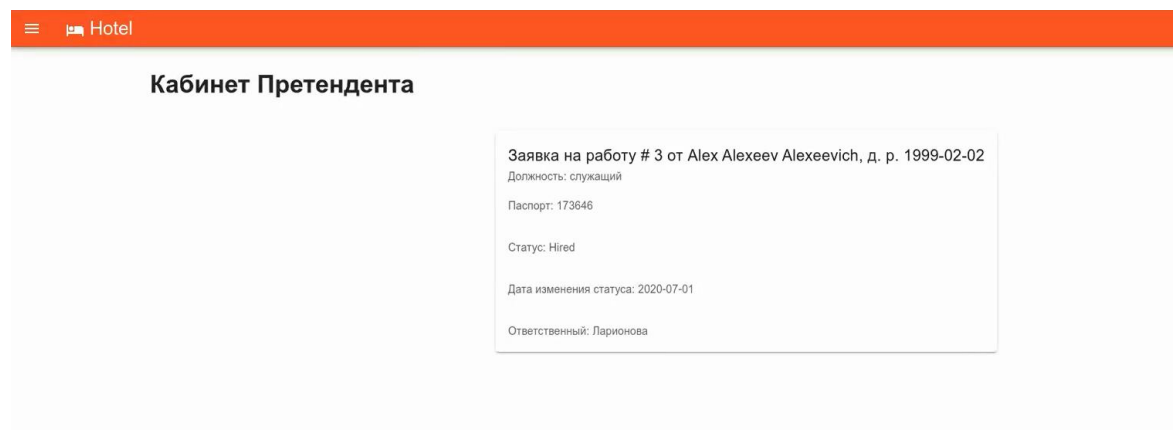


Рисунок 18 - Реализация интерфейса “Кабинет претендента”

### 4.3. Выводы

В ходе работы была разработана клиентская часть веб-сервиса. Разработаны интерфейсы для авторизации пользователя, для работы администратора и для служащего персонала, была реализована возможность для потенциального посетителя отеля забронировать номер, для потенциального соискателя работы – отправить заявку о принятии на работу. Благодаря фреймворку Vue.js разработка реализация происходила легко и удобно.. Благодаря использованию библиотеки Vuetify UI были разработаны лаконичные и приятные глазу интерфейсы.

## **5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ**

### **5.1. Описание средства контейнеризации и оркестрации**

Контейнеризацией называется такой подход к разработке программного обеспечения, при котором все части приложения упаковываются вместе в образ контейнера. Преимущество такого подхода в том, что он обеспечивает работоспособность приложения в различных средах.

Оркестрация представляет собой метод координации нескольких контейнеров. Оркестрацию проекта можно опустить, но если ее использовать, то приложение получает гибкость, масштабируемость и взаимосвязанность процессов в контейнерах. Использование оркестрации позволяет создавать системы из множества контейнеров, где каждый контейнер отвечает только за свою задачу. Кроме того, каждый их контейнеров можно заменить другим, не перезапуская весь проект.

В качестве средства контейнеризации и оркестрации проекта используется Docker.

Docker – это открытая платформа для разработки, доставки и эксплуатации приложений. С помощью технологии Docker (контейнеризации) можно разделить исходное приложение на несколько компонентов, которые взаимодействуют между которыми возможно реализовать. Подобный подход может привести разные методы реализации компонентов, тем самым предоставляя разработчику широкий спектр возможностей. Благодаря этому, разработчику предоставляется возможность создания микро-сервисных архитектур. [5]

### **5.2. Контейнеризация проекта**

Для контейнеризации проекта были созданы файлы Dockerfile в каждой из директорий для будущих контейнеров. Так как проект состоит из трех основных частей (база данных, фронтенд и бэкенд), то и контейнеров будет в итоге три.

На рисунке 23 представлен пример Dockerfile для серверной части проекта. Файл серверной части включает в версию Python, указание рабочей директории и установку библиотек, необходимых для запуска проекта.

```

FROM python:3.6.9

ENV PYTHONUNBUFFERED 1

RUN mkdir /hotel_admin

WORKDIR /hotel_admin

COPY . /hotel_admin

RUN pip3 install -r requirements.txt

```

Рисунок 19 – Пример файла Dockerfile для серверной части

### 5.3. Оркестрация проекта

Для оркестрации проекта в корневой папке необходимо создать файл `docker-compose.yml`, который отвечает за оркестрацию. В нем описаны детали для запуска каждого контейнера, указаны порты и необходимые команды. Пример файла оркестрации представлен на рис. 25.

```

version: '3'

services:
  db:
    image: postgres
    ports:
      - "5436:5432"
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=hotel_db
    volumes:
      - ./dbs/postgres-data:/var/lib/postgresql

  backend:
    container_name: hotel_backend
    build: ./hotel-admin/hotel_admin
    command: bash -c "
      sleep 3 &&
      python3 manage.py makemigrations && python3 manage.py migrate &&
      python3 manage.py runserver --insecure 0.0.0.0:8000";
    volumes:
      - ./hotel-admin/hotel_admin:/hotel_admin
    ports:
      - "8000:8000"
    depends_on:
      - db

```

Рисунок 20 – Часть файла `docker-compose.yml`

#### **5.4. Выводы**

В ходе работы была проведена контейнеризация проекта для удобства его запуска. Созданы необходимые файлы для оркестрации и запуска проекта в Docker.

## ЗАКЛЮЧЕНИЕ

Можно сделать вывод, что в ходе реализации проекта были выполнены следующие задачи: была проанализирована предметная область и функциональные требования, была создана подходящая архитектура проекта, были разработаны серверная и клиентская части системы, была выполнена контейнеризация проекта. В итоге была реализована система для администрирования отеля. В системе реализовано:

- Вход в систему для администратора, служащего, кандидата на работу
- Бронирование номера для клиента
- Подача заявки на работу в отеле
- Просмотр статуса заявки на работу в отеле
- Просмотр расписания служащего персонала (для администратора и для служащего персонала)
- Принятие/отклонение заявки на работу (для администратора)
- Просмотр доступных на данный момент свободных номеров
- Просмотр текущей прибыли

В рамках реализации задачи по созданию веб-сервиса были получены практические навыки работы с современными средствами разработки такими, как фреймворк Django REST, фреймворк Vue.js и библиотека

Также в ходе работы для удобства запуска разработанного веб-сервиса в различных средах была использована платформа Docker и выполнена контейнеризация проекта.



## СПИСОК ЛИТЕРАТУРЫ

1. Документация Django Rest Framework [Электронный ресурс] — <https://www.django-rest-framework.org/topics/documenting-your-api/>. Дата обращения: 02.07.2020.
2. Документация PostgreSQL [Электронный ресурс] — <https://www.postgresql.org/docs/>. Дата обращения: 02.07.2020.
3. Документация Vue.js [Электронный ресурс] — <https://ru.vuejs.org/v2/guide/>. Дата обращения: 02,07.2020.
4. Документация Vuetify. Документация Vuetify [Электронный ресурс]. URL: <https://vuetifyjs.com/ru/> (дата обращения: 01.07.2020).
5. Документация Docker [Электронный ресурс] — <https://docs.docker.com/engine/install/>. Дата обращения: 01.07.2020.
6. «Клиент-серверное взаимодействие и роли серверов» [Электронный ресурс] — <http://www.4stud.info/networking/lecture5.html>. Дата обращения: 02.07.2020.