

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, Muse-UI

Обучающийся Димова Алина Евгеньевна, 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	6
2.3. Модель данных	7
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	11
3.5. Настройка маршрутизации	12
3.6. Выводы	13
4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ	14
4.1. Описание средств разработки клиентской части	14
4.2. Разработанные интерфейсы	14
4.2.1. Вход в систему	14
4.2.2. Интерфейсы для администратора	15
4.2.3. Интерфейсы для уборщика	20
4.3. Выводы	21
5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ	22
5.1. Описание средства контейнеризации и оркестрации	22
5.2. Контейнеризация проекта	22
5.3. Оркестрация проекта	23
5.4. Выводы	24
ЗАКЛЮЧЕНИЕ	25
СПИСОК ЛИТЕРАТУРЫ	26

ВВЕДЕНИЕ

Интернет является важной составляющей современного общества. Невозможно переоценить влияние всемирной паутины на жизнь человека в наши дни. Поэтому специалист в области информационных технологий должен не только знать теоретические основы организации сети Интернет, но и иметь практические навыки реализации веб-сервисов.

Целью данной курсовой работы является разработка веб-сервиса согласно выбранному варианту. В рамках работы нужно было изучить и научиться применять средства создания веб-сервисов, которые используют в современных системах.

В ходе 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 проведен 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. Пятая глава посвящена контейнеризации и оркестрации проекта.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1.1. Описание предметной области

В качестве варианта был выбран вариант 1 – создание программной системы для администратора гостиницы. Предметной областью является гостиница. Основными пользователями являются администратор и служащие работники гостиницы.

Информационная система для данной области представляет собой сервис, в котором реализованы стандартные процедуры для данной области: заселение и выселение клиентов в номера гостиницы, прием на работу и увольнение сотрудников. Такая система должна обеспечивать хранение сведений об имеющихся номерах, о проживающих в гостинице клиентах и о служащих, убирающихся в номерах. Номера в гостинице имеются трех типов: одноместный, двухместный и трехместный, они отличаются стоимостью проживания в сутки. В каждом номере есть телефон. О каждом клиенте имеется следующая информация: номер паспорта, фамилия, имя, отчество, город, из которого прибыл, дата поселения в гостинице, дата выезда, выделенный гостиничный номер. О работниках гостиницы имеется следующая информация: фамилия, имя, отчество, где (этаж) и когда (день недели) он убирает. Служащий гостиницы убирает все номера на одном этаже в определенные дни недели.

1.2. Описание функциональных требований

На основе анализа предметной области выявлены функциональные требования в разрабатываемой системе. В системе должна храниться вся имеющаяся информация о номерах, клиентах и работниках гостиницы.

Работа с системой предполагает получение информации о заселениях клиентов в определенные номера, поиск клиентов по городу, из которого они прибыли, просмотр расписаний уборок работников с указанием этажей и дней недели.

Администратор гостиницы должен иметь возможность выполнить следующие действия:

- Принять на работу или уволить служащего работника.
- Изменить расписание уборок работника.
- Поселить клиента в выбранный номер.

Служащий работник должен иметь возможность:

- Зарегистрироваться в системе.
- Просмотреть личное расписание уборок в системе.

- Добавить отчет о проделанной уборке.

1.3. Выводы

Проведен анализ предметной области и функциональных требований системы. В результате было улучшено понимание данной области и того, как должна работать разрабатываемая система.

2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА

2.1. Описание архитектуры сервиса

Для веб-сервиса, разрабатываемого в рамках курсовой работы, была выбрана архитектура «клиент-сервер», представленная на рис. 1. В данной архитектуре сетевая нагрузка распределена между поставщиками услуг, серверами, и заказчиками услуг, клиентами.

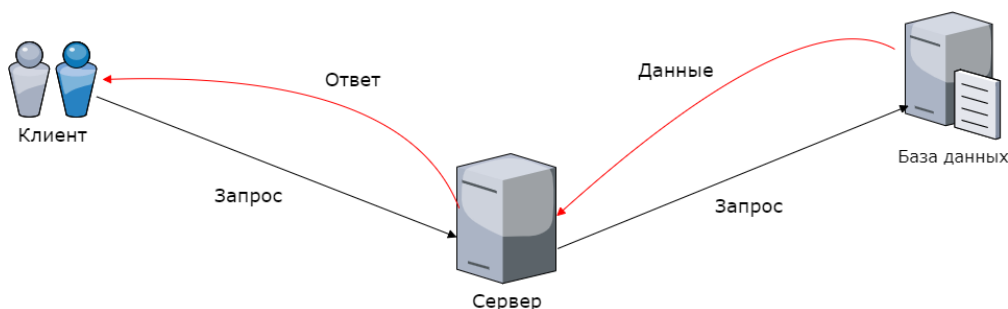


Рисунок 1 – Архитектура «Клиент-Сервер»

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.

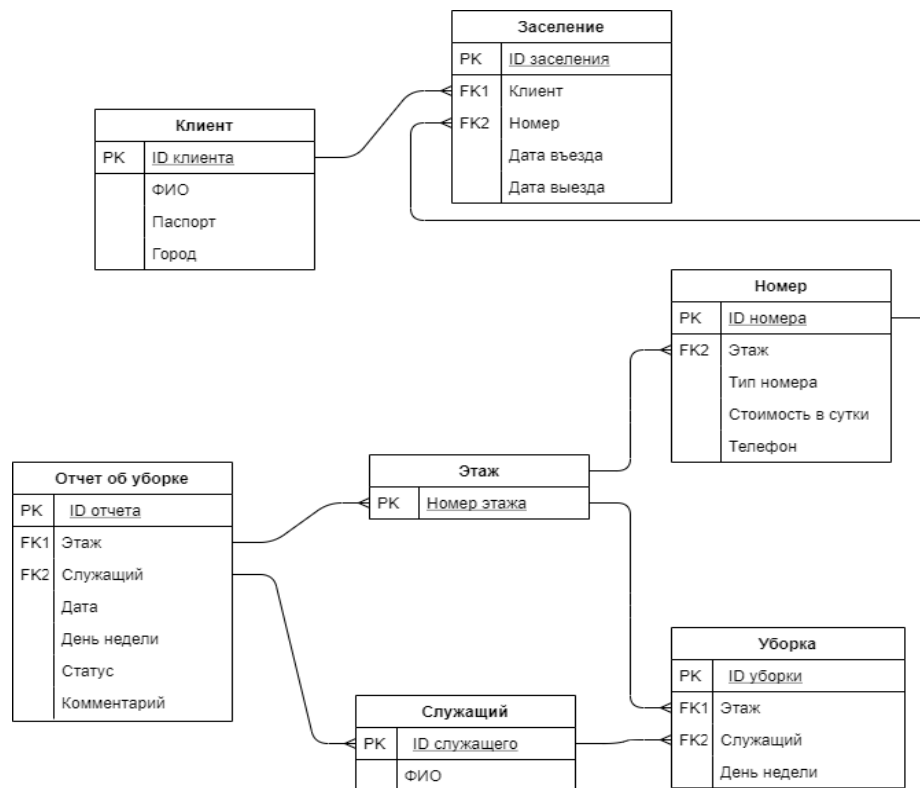


Рисунок 3 – Модель данных

Модель содержит 7 сущностей:

- Клиент.
- Номер.
- Заселение.
- Этаж.
- Уборка.
- Служащий.
- Отчет об уборке.

Сущности соединены между собой связями Один-ко-многим и Многие-ко-многим. Основными из них являются: Клиент, Номер, Этаж, Работник и Отчет. Сущности Заселение, Уборка были созданы вспомогательно для реализации связи Многие-ко-многим между основными сущностями.

2.4. Выводы

В ходе проектирования архитектуры 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

Для реализации серверной части был использован фреймворк Django Rest, который является удобным инструментом, основанным на идеологии Django, для работы с rest.

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

REST (сокр. 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:

- Client (клиент) – таблица содержит информацию о клиентах гостиницы.
- Room (номер) – таблица содержит информацию о номерах в гостинице.
- Checkin (заселение) – таблица содержит информацию о заселениях клиентов в номера гостиницы.
- Floor (этаж) – таблица содержит информацию об этажах в гостинице.
- Worker (работник) – таблица содержит информацию о служащих работниках гостиницы.
- Cleaning (уборка) – таблица содержит расписание уборок служащего по дням недели.
- Otchet (отчет об уборке) – таблица содержит информацию об уборках, проведенных работниками.

3.3. Сериализация

В программировании сериализация представляет собой процесс перевода какой-либо структуры данных в последовательность битов. Другими словами, 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:

- UserSerializer – используется для получения данных о пользователе.
- AddUserSerializer – используется для добавления нового пользователя.
- ClientSerializer – используется для получения данных о клиенте.
- AddClientSerializer – используется для добавления нового клиента.
- RoomSerializer – используется для получения данных о номере.
- WorkerSerializer – используется для получения данных о работнике.
- AddWorkerSerializer – используется для добавления нового работника.
- CheckinSerializer – используется для получения данных о заселении клиента.
- AddCheckinSerializer – используется для добавления нового заселения.
- FloorSerializer – используется для получения данных об этаже.
- CleaningSerializer – используется для получения данных об уборке.
- AddCleaningSerializer – используется для добавления новой уборки в расписание.
- CleaningListSerializer – используется для получения списка уборок.
- RoomDetailSerializer – используется для получения детализированных данных о номере.
- ClientDetailSerializer – используется для получения детализированных данных о клиенте.
- OtchetSerializer – используется для получения данных об отчетах.
- AddOtchetSerializer – используется для добавления нового отчета об уборке.

На рис.4 приведен сериализатор для модели номера гостиницы.

```
class RoomSer(serializers.ModelSerializer):
    floor = serializers.SlugRelatedField(slug_field='floor_num', read_only=True)

    class Meta:
        model = Room
        fields = "__all__"
```

Рисунок 4 – Сериализатор номера 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:

- Rooms – представление для получения данных о всех номерах.
- Clients – представление для получения данных о всех клиентах и добавление клиентов (через метод POST).
- ClientsList – получение списка клиентов и применение к нему фильтра.
- RoomDetailsView – получение данных об определенном номере гостиницы.
- ClientDetailsView – получение данных об определенном клиенте гостиницы.
- WorkersView – вывод всех работников гостиницы и добавление нового работника (через метод POST).
- DeleteWorker – удаление определенного работника.
- UserWorkerView – получение всех пользователей, которые являются служащими работниками (не администраторы), и добавление данных к новому пользователю (через метод POST).
- CleaningTable – получение расписания уборок одного работника и добавление либо изменение данных в нем (через метод POST).
- AddCheckinView – получение всех заселений в гостинице и добавление нового заселения (через метод POST).
- FloorsView – получение данных обо всех этажах.
- CheckinList – получение списка заселений по номеру в гостинице.
- CleaningsView – получение списка всех уборок по дню недели.
- OtchetView – просмотр всех отчетов одного работника и добавление нового отчета от работника (через метод POST).

Во всех отображениях кроме UserWorkerView, который используется при авторизации, разрешен доступ только зарегистрированным пользователям, вошедшим в систему.

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 WorkersView(APIView):
    """Вывод всех работников"""
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request):
        workers = Worker.objects.all()
        serializer = WorkerSer(workers, many=True)
        return Response(serializer.data)

    def post(self, request):
        worker = AddWorkerSer(data=request.data)
        if worker.is_valid():
            worker.save()
            return Response({'status': 'Добавлено'})
        else:
            return Response({'status': 'Ошибка'})

```

Рисунок 5 – Представление для вывода и добавления работников

3.5. Настройка маршрутизации

Когда разработаны представления, нужно создать URL-адреса для того, чтобы система начала работать. 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('clients/', views.Clients.as_view()),
    path('client_detail/<int:pk>', views.ClientDetailsView.as_view()),
    path('clients_filter/', views.ClientsList.as_view()),
    path('rooms/', views.Rooms.as_view()),
    path('workers/', views.WorkersView.as_view()),
    path('del_worker/<int:pk>', views.DelWorker.as_view()),
    path('room_detail/<int:pk>', views.RoomDetailsView.as_view()),
    path('cleanings/<int:pk>', views.CleaningTable.as_view()),
    path('cleanings_view/<int:pk>', views.CleaningsView.as_view()),
    path('checkin/', views.AddCheckin.as_view()),
    path('checkin_filter/', views.CheckinsList.as_view()),
    path('floors/', views.FloorsView.as_view()),
    path('users/<int:pk>', views.UserWorkerView.as_view()),
    path('otchets/<int:pk>', views.OtchetView.as_view()),
]

```

Рисунок 6 – Список url-адресов приложения

3.6. Выводы

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

4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ

4.1. Описание средств разработки клиентской части

Для разработки клиентской части системы был использован фреймворк Vue.js и библиотека Muse UI.

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 Muse UI представляет собой набор компонентов для Vue, которые используют Material Design [4]. Это фреймворк для быстрого создания и запуска пользовательского интерфейса с приятным и удобным дизайном.

4.2. Разработанные интерфейсы

4.2.1. Вход в систему

При запуске системы открывается стартовая страница (рис.7). На ней пользователю предлагается войти в систему либо зарегистрироваться. При этом войти в систему можно как администратор или как работник (уборщик). Разные страницы входа направляют на разные сценарии работы с системой.

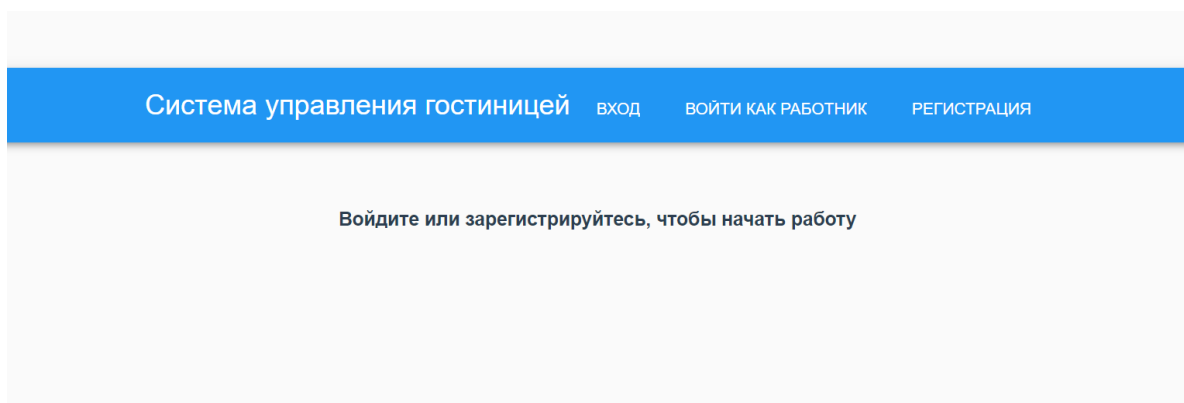


Рисунок 7 – Стартовая страница

Если пользователь еще не имеет своего аккаунта в системе, ему необходимо зарегистрироваться, нажав на кнопку регистрация.

Регистрация нового пользователя [НАЗАД](#)

Ваше ФИО
Иванова Мария Сергеевна

Вы являетесь уборщиком?
Да

Имя пользователя
maria

Ваш email
maria@gmail.com

Пароль

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 8 – Пример заполненной формы регистрации

На рис. 8 представлен пример заполненной формы регистрации в системе. Основные поля формы включают ФИО, имя пользователя, email, пароль. Также пользователю необходимо указать, является ли он уборщиком или нет. После заполнения формы нужно нажать на кнопку Зарегистрироваться и войти в систему как работник либо как администратор через интерфейс входа (рис. 9).

Войдите в свой аккаунт

Логин
admin

Пароль

Войти

Рисунок 9 – Интерфейс входа в систему

4.2.2. Интерфейсы для администратора

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

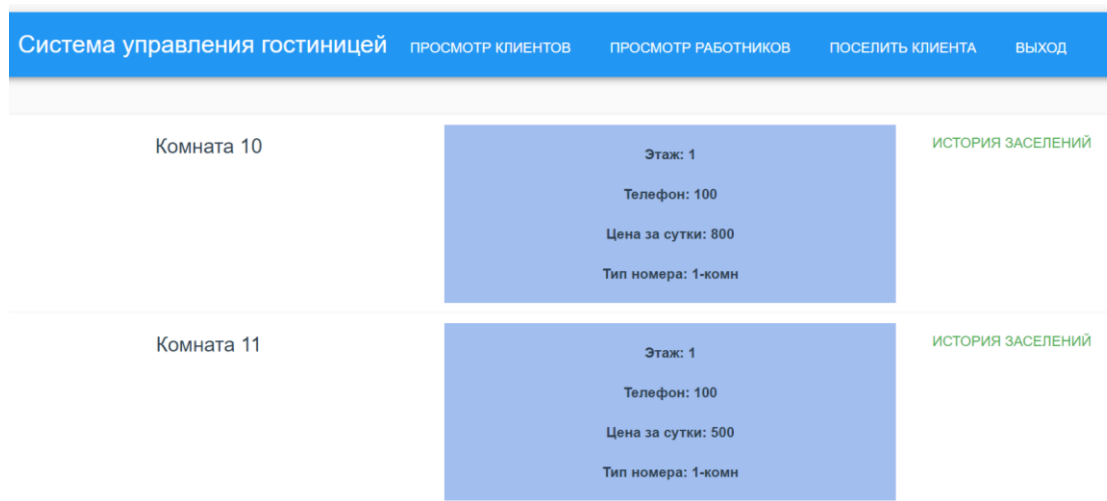


Рисунок 10 – Главная страница системы

На главной странице выведен список всех имеющихся номеров в гостинице с указанием подробной информации о номерах (этаж, на котором находится номер, телефонный номер, цена за сутки и тип номера). Рядом с каждым номером есть кнопка История заселений, которая открывает список заселений (рис.11) с указанием клиентов и дат в данный номер гостиницы.

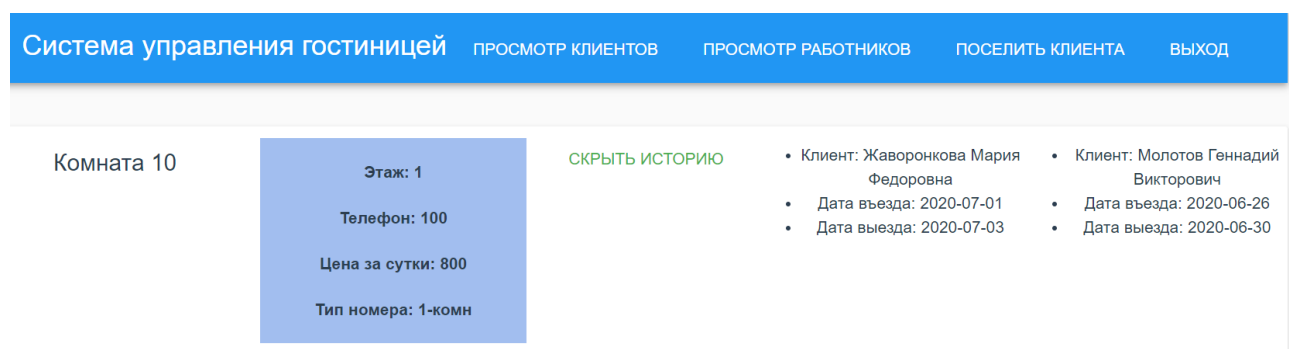


Рисунок 11 – Открытый список заселений в номер

На этой же странице на верхней панели имеются кнопки Просмотр клиентов, Просмотр работников, Поселить клиента и Выход. Кнопка Выход обеспечивает выход пользователя из системы и возвращение на стартовую страницу.

Кнопка Просмотр клиентов открывает страницу со списком всех клиентов гостиницы (рис. 12). Для каждого клиента указаны ФИО, номер паспорта и город, из которого клиент прибыл.

Просмотр клиентов гостиницы					ПОИСК КЛИЕНТОВ ПО ГОРОДУ	НА ГЛАВНУЮ СТРАНИЦУ
Клиент - Жаворонкова Мария Федоровна	Клиент - Молотов Геннадий Викторович	Клиент - Сергиенко Максим Евгеньевич	Клиент - Носов Павел Петрович	Клиент - Любимов Игорь Дмитриевич		
Паспорт: 111222	Паспорт: 222333	Паспорт: 333444	Паспорт: 444555	Паспорт: 555666		
Город: Санкт-Петербург	Город: Выборг	Город: Санкт-Петербург	Город: Москва	Город: Москва		
ДОБАВИТЬ КЛИЕНТА						

Рисунок 12 – Страница просмотра клиентов

На странице просмотра клиентов реализована функция добавления нового клиента (рис. 13). Для открытия формы необходимо нажать на кнопку Добавить клиента.

Добавление клиента	НА ГЛАВНУЮ СТРАНИЦУ
ФИО клиента	
Номер паспорта	
Город	
ПОТВЕРДИТЬ ДОБАВЛЕНИЕ	

Рисунок 13 – Форма добавления клиента

Также на странице просмотра клиентов (рис. 12) возможен поиск клиентов по городу, из которого они прибыли. Нужно нажать на кнопку Поиск клиентов по городу и откроется страница поиска. На рисунке 14 представлен пример поиска клиентов.

Поиск клиентов по городу	НА ГЛАВНУЮ СТРАНИЦУ
Город	Москва
НАЙТИ	
Клиент Носов Павел Петрович	
Паспорт 444555	
Город Москва	
Клиент Любимов Игорь Дмитриевич	
Паспорт 555666	
Город Москва	

Рисунок 14 – Пример поиска клиентов по городу

На этом описание функций страницы просмотра клиентов окончены. Перейдем к странице заселения клиентов (рис. 15). Для ее открытия необходимо нажать на кнопку Поселить клиента на главной странице системы (рис. 10).

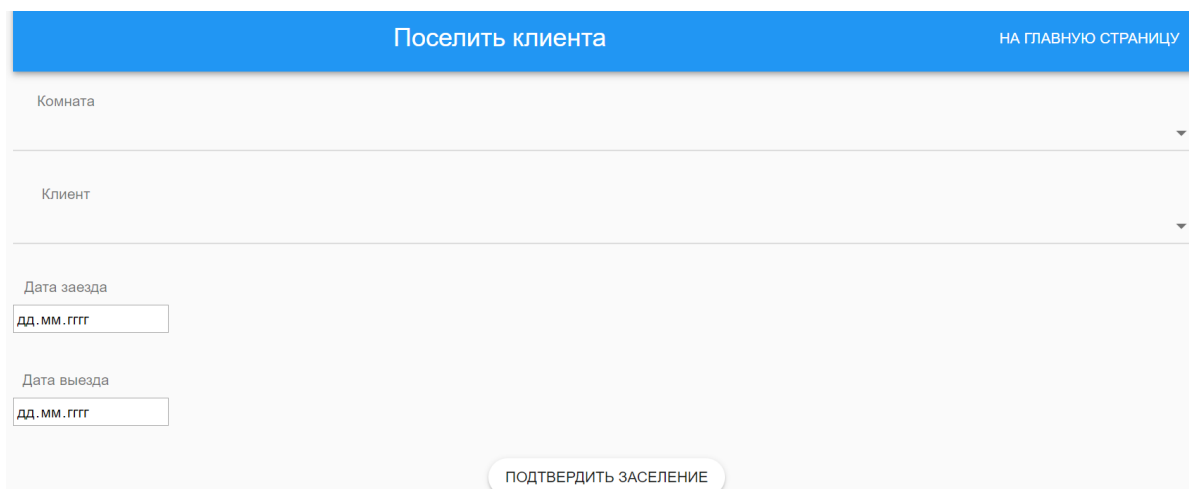


Рисунок 15 – Страница для заселения клиентов

На странице предлагается заполнить форму. Поля Комната и Клиент предлагают выбор из имеющихся комнат и клиентов соответственно. Следует отметить, что перед заселением нового клиента в номер этот клиент должен быть добавлен в базу клиентов системы. Иначе его просто не будет в выпадающем списке поля Клиент. Также предлагается выбрать дату заезда и выезда клиента. После добавления нового заселения его можно будет увидеть на главной странице в истории заселений выбранного номера гостиницы.

Перейдем к странице просмотра работников (рис. 16). Для этого нужно нажать на соответствующую кнопку на главной странице системы (рис.10).

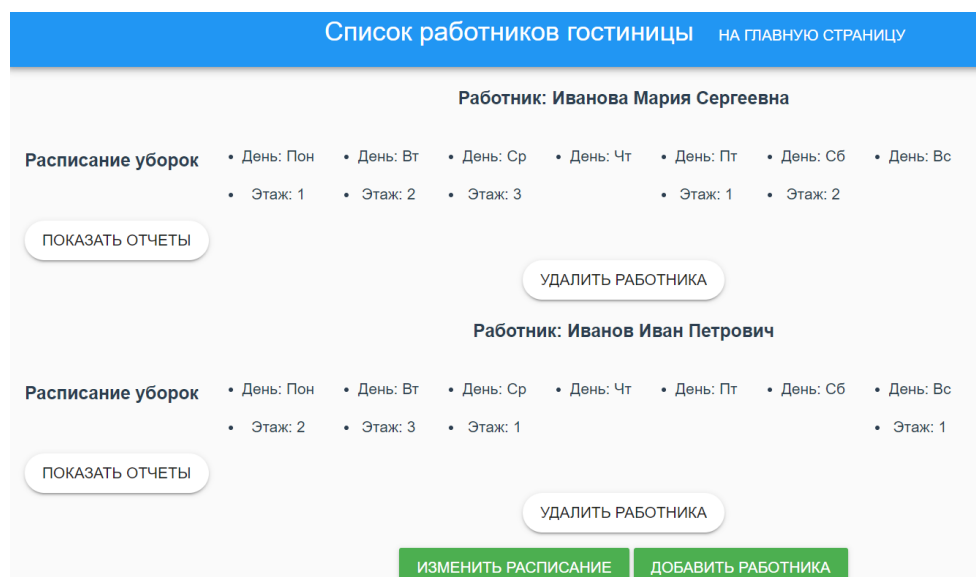
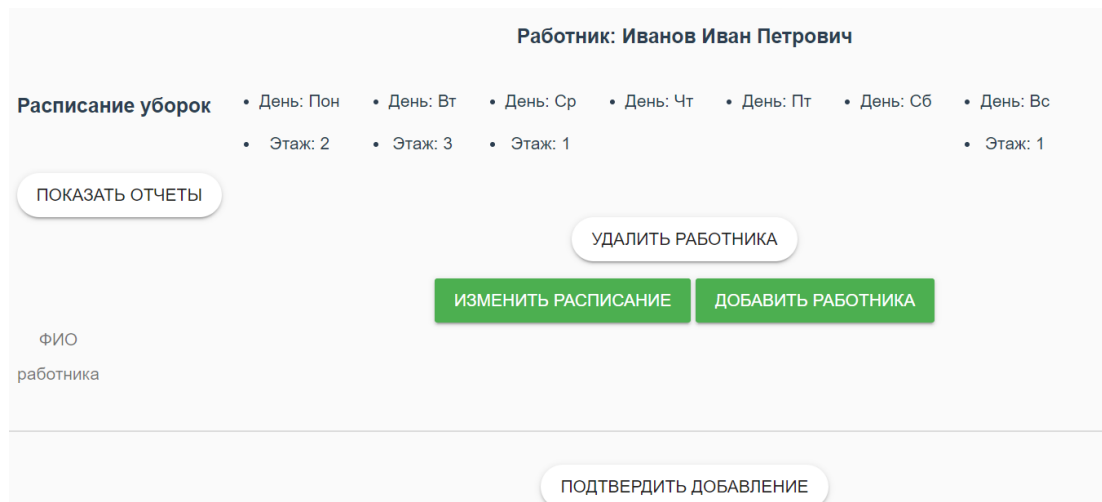


Рисунок 16 – Страница просмотра работников

На данной странице имеется список работников гостиницы с их расписание уборок. Расписание уборок работника представляет собой список дней недели с указанием этажей гостиницы. В некоторые дни этаж может отсутствовать, это означает, что расписание не заполнено до конца либо этот день выходной у работника.

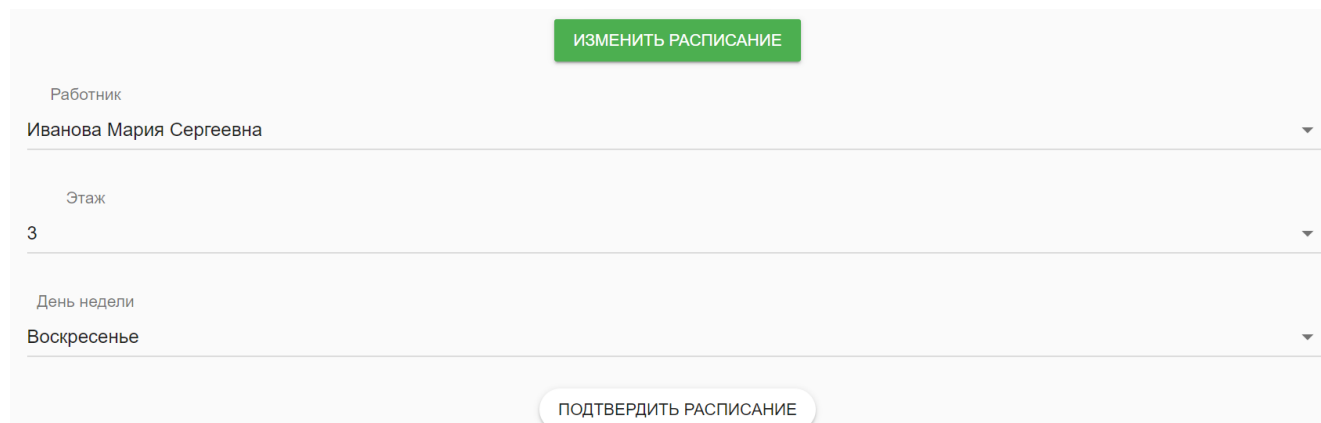
Каждого работника можно уволить, нажав на кнопку Удалить работника. Также можно добавить нового работника, тогда откроется форма добавления работника (рис. 17).



Form for adding a worker. At the top, it says "Работник: Иванов Иван Петрович". Below this is a section titled "Расписание уборок" with a grid of buttons for days of the week (Пон, Вт, Ср, Чт, Пт, Сб, Вс) and floors (Этаж: 2, 3, 1, 1). There are three buttons: "ПОКАЗАТЬ ОТЧЕТЫ", "УДАЛИТЬ РАБОТНИКА", and "ДОБАВИТЬ РАБОТНИКА". A green button "ИЗМЕНИТЬ РАСПИСАНИЕ" is also present. At the bottom, there is a "ПОДТВЕРДИТЬ ДОБАВЛЕНИЕ" button.

Рисунок 17 – Форма для добавления работника

Чтобы изменить расписание уборок, нужно нажать на кнопку изменить расписание и выбрать нужные данные в открывшейся форме (рис. 18).



Form for changing a worker's schedule. It has a green button "ИЗМЕНИТЬ РАСПИСАНИЕ" at the top. Below it are three dropdown menus: "Работник" (Iванова Мария Сергеевна), "Этаж" (3), and "День недели" (Воскресенье). At the bottom is a "ПОДТВЕРДИТЬ РАСПИСАНИЕ" button.

Рисунок 18 – Форма для изменения расписания

При этом если ранее в расписании на выбранный день уже был назначен этаж, то он изменится. Если же в расписание не было записи в этот день, то добавится новая запись с выбранным этажом.

Также у администратора есть возможность просмотреть отчеты об уборках, добавленные работниками. Для этого нужно нажать на кнопку Показать отчеты рядом с

выбранным работником. Отчеты выводятся по дням недели с указанием даты уборки, номера этажа, статуса уборки и комментария к ней. На рисунке 19 представлен пример списка отчетов работника.

Работник: Иванов Иван Петрович

Расписание уборок

• День: Пон

• День: Вт

• День: Ср

• День: Чт

• День: Пт

• День: Сб

• День: Вс

• Этаж: 2

• Этаж: 3

• Этаж: 1

• Этаж: 1

СКРЫТЬ ОТЧЕТЫ

Отчеты об уборках

<div>День: Пон</div> <div>Дата: 2020-06-29</div> <div>Этаж: 2</div> <div>Статус: Ок</div> <div>Комментарий</div> <div>Уборка проведена</div>	<div>День: Вт</div> <div>Дата: 2020-06-30</div> <div>Этаж: 3</div> <div>Статус: Ок</div> <div>Комментарий</div> <div>Уборка проведена во всех номерах этажа</div>	<div>День: Ср</div> <div>Дата: 2020-07-01</div> <div>Этаж: 2</div> <div>Статус: Ок</div> <div>Комментарий</div> <div>Все прошло в штатном режиме</div>	<div>День: Чт</div>	<div>День: Пт</div> <div>Дата: 2020-07-03</div> <div>Этаж: 3</div> <div>Статус: Ок</div> <div>Комментарий</div> <div>Уборка проведена успешно</div>	<div>День: Сб</div>
<div>День: Вс</div>					

Рисунок 19 – Пример вывода отчетов работника

На этом функционал системы для администратора окончен. Перейдем к функционалу для служащего работника гостиницы.

4.2.3. Интерфейсы для уборщика

Уборщик может войти в систему после нажатия на кнопку Войти как работник на стартовой странице. Если работник не зарегистрирован в системе, то ему нужно пройти регистрацию и указать свое ФИО и то, что он является уборщиком. После входа в систему пользователь попадает на страницу со своим расписанием (рис. 20).

Кабинет работника Иванова Мария Сергеевна

выход

Расписание уборок

• День: Пон

• День: Вт

• День: Ср

• День: Чт

• День: Пт

• День: Сб

• День: Вс

• Этаж: 1

• Этаж: 2

• Этаж: 3

• Этаж: 1

• Этаж: 2

• Этаж: 3

ПОКАЗАТЬ ОТЧЕТЫ

ДОБАВЛЕНИЕ ОТЧЕТА

Рисунок 20 – Страница работника

Здесь он может увидеть расписание своих уборок и просмотреть свои отчеты, нажав на кнопку Показать отчеты. На рисунке 21 представлен пример просмотра отчетов работника.

20

Расписание уборок

- День: Пон
- День: Вт
- День: Ср
- День: Чт
- День: Пт
- День: Сб
- День: Вс

- Этаж: 1
- Этаж: 2
- Этаж: 3
- Этаж: 1
- Этаж: 2
- Этаж: 3

СКРЫТЬ ОТЧЕТЫ

Отчеты об уборках

День: Пон	День: Вт	День: Ср	День: Чт	День: Пт	День: Сб
	Дата: 2020-06-09 Этаж: 2 Статус: Ок Комментарий Все номера убраны без проблем	Дата: 2020-07-08 Этаж: 2 Статус: Пр Комментарий Уборка проведена, но не было доступа в 33 комнату			
	Дата: 2020-07-07 Этаж: 1 Статус: Пр Комментарий Уборка проведена во всех комнатах кроме 10				

Рисунок 21 – Просмотр отчетов об уборках

Основной функцией страницы работника является добавление отчета об уборке. При нажатии на кнопку **Добавить отчет** на странице работника открывается форма добавления отчета (рис. 22).

Дата уборки

День недели

▼

Этаж

▼

Проведена без проблем?

▼

Комментарий

ПОДТВЕРДИТЬ ДОБАВЛЕНИЕ

Рисунок 22 – Форма добавления отчета

4.3. Выводы

Была разработана клиентская часть веб-сервиса. Разработаны интерфейсы для входа в систему, для работы администратора и для уборщика. Фреймворк Vue.js позволил сделать разработку быстрой и удобной. Благодаря использованию библиотеки Muse UI были получены приятные и стильные интерфейсы.

5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ

5.1. Описание средства контейнеризации и оркестрации

Под контейнеризацией понимается 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.

```
1 FROM python:3.8.2
2
3 RUN mkdir /server
4
5 WORKDIR /server
6
7 ADD . /server
8
9 RUN pip install -r req.txt
10
```

Рисунок 23 – Пример файла Dockerfile для серверной части

На рис. 24 представлен пример файла Dockerfile для создания контейнера для клиентской части системы. Он включает версию Node.js, указание рабочей директории и установку библиотек для запуска проекта.

```
FROM node:12

WORKDIR /hotel-vue

COPY package*.json ./

RUN npm install --silent

COPY . .

RUN npm run build
```

Рисунок 24 – Пример файла Dockerfile для клиентской части

База данных в данном случае не хранится в папке проекта, поэтому для нее не нужен отдельный файл Dockerfile. Однако контейнер для нее создается и прописывается в файле для оркестрации.

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:
      - "5432:5432"
    environment:
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=password
      - POSTGRES_DB=hotel-db
  backend:
    container_name: hotel_backend
    build: ./server
    command: bash -c "python3 manage.py makemigrations && python3 manage.py migrate && python3 manage.py runserver 0.0.0.0:8000";
    volumes:
      - ./server:/server
    ports:
      - "8000:8000"
    depends_on:
      - db
```

Рисунок 25 – Часть файла docker-compose.yml

Первым этапом запуска проекта является сборка всех контейнеров с помощью команды *docker-compose build*. Затем необходимо выполнить команду *docker-compose up*, чтобы запустить проект.

5.4. Выводы

Проведена контейнеризация проекта для удобства запуска его в различных средах. Созданы необходимые файлы для оркестрации и запуска проекта в Docker.

ЗАКЛЮЧЕНИЕ

По итогам данной работы были выполнены следующие задачи: проанализирована предметная область и функциональные требования, создана архитектура проекта, разработана серверная и клиентская части системы, выполнена контейнеризация проекта. В результате реализована система для управления гостиницей, которая соответствует требованиям и обладает необходимым функционалом. В системе реализовано:

- Вход и регистрация.
- Просмотр имеющихся номеров гостиницы и истории заселений в них.
- Просмотр и добавление клиентов.
- Заселение клиентов в номера.
- Просмотр, добавление и удаление работников.
- Просмотр и изменение расписаний работников.
- Просмотр отчетов об уборках (для администратора).
- Просмотр личного расписания (для уборщика).
- Добавление отчетов об уборках (для уборщика).

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 Muse UI.

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: 20.05.2020.
2. Документация PostgreSQL [Электронный ресурс] — <https://www.postgresql.org/docs/>. Дата обращения: 23.05.2020.
3. Документация Vue.js [Электронный ресурс] — <https://ru.vuejs.org/v2/guide/>. Дата обращения: 20.05.2020.
4. Документация Muse UI [Электронный ресурс] — <https://muse-ui.org/#/en-US>. Дата обращения: 30.05.2020.
5. Документация Docker [Электронный ресурс] — <https://docs.docker.com/engine/install/>. Дата обращения: 20.06.2020.
6. «Клиент-серверная архитектура в картинках» [Электронный ресурс] — <https://habr.com/ru/post/495698/>. Дата обращения: 20.05.2020.