

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 Зимарин Андрей Дмитриевич, гр. К3342

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

Оценка за курсовую работу _____

Подписи членов комиссии:

_____ (Говоров А.И.)
(подпись)

_____ (Чунаев А.В.)
(подпись)

_____ (Антонов М.Б.)
(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
4.3. Выводы	21
ЗАКЛЮЧЕНИЕ	22
СПИСОК ЛИТЕРАТУРЫ.....	23

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

Выбранной предметной областью является гостиница, для которой требовалось реализовать создание системы администрирования. Основными пользователями являются администратор и служащие работники гостиницы.

В данной области информационная системы — это сервис, в котором реализованы стандартные процедуры: заселение гостей в номера, прием на работу и увольнение сотрудников, составление расписание работы сотрудников, ведение базы клиентов и сотрудников, а также поиск по ней. Номера делятся по своему типу на одноместные, двухместные и трехместные. От типа номера зависит и стоимость проживания в нем. Для каждого номера указывается контактный номер, этаж и ведется история заселений. Каждый гость оставляет свои личные данные: номер паспорта, фамилия, имя, отчество, город, дата поселения в гостинице, дата выезда и закреплённый за ним гостиничный номер. Каждый работник, помимо личной информации, обладает графиком работы.

1.2. Описание функциональных требований

На этапе анализа предметной области были выявлены функциональные требования к разрабатываемой системе. Системе должна иметь всю необходимую информацию о номерах, клиентах и работниках гостиницы.

Функциональные требования также включают в себя получение информации о заселениях клиентов, о номерах, поиск клиентов по городу, просмотр и изменение графика работы работников с указанием этажей и дней недели.

Администратор гостиницы должен обладать возможностью выполнить следующие действия:

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

Служащий 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.

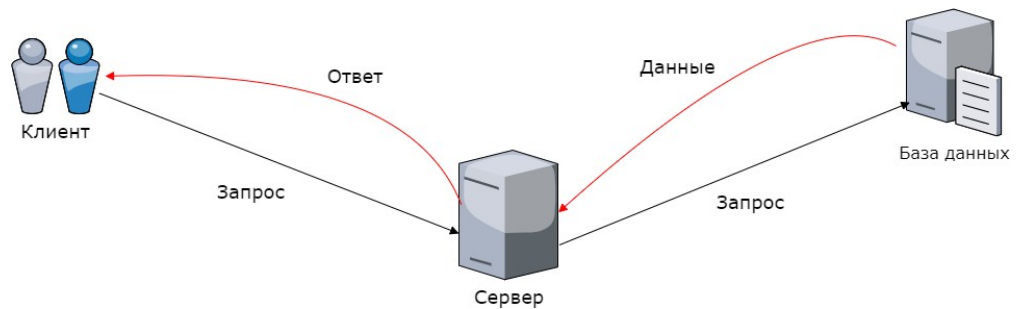


Рисунок 1 – Архитектура «Клиент-Сервер»

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. [1]

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.

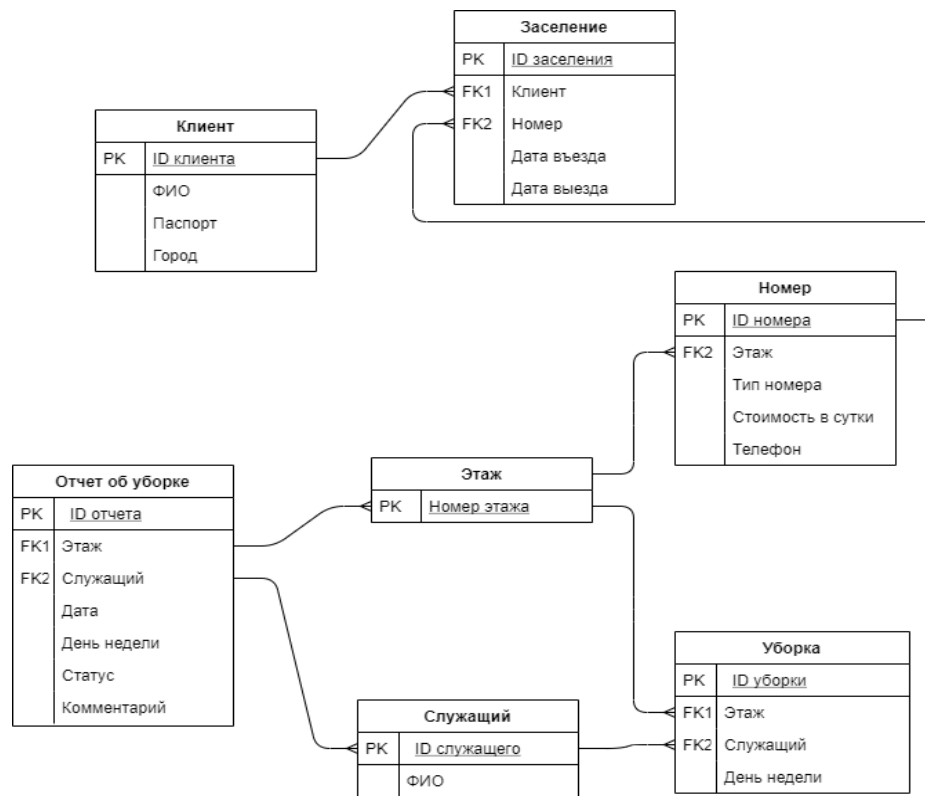


Рисунок 3 – Модель данных

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 как вспомогательные 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. [2]

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 [3].

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.

```
25 class RoomSer(serializers.ModelSerializer):
26     floor = serializers.SlugRelatedField(slug_field='floor_num', read_only=True)
27
28     class Meta:
29         model = Room
30         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.

```

65 class WorkersView(APIView):
66     permission_classes = [permissions.IsAuthenticated, ]
67
68     def get(self, request):
69         workers = Worker.objects.all()
70         serializer = WorkerSer(workers, many=True)
71         return Response(serializer.data)
72
73     def post(self, request):
74         worker = AddWorkerSer(data=request.data)
75         if worker.is_valid():
76             worker.save()
77             return Response({'status': 'Добавлено'})
78         else:
79             return Response({'status': 'error'})

```

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.

```

5 urlpatterns = [
6     path('clients/', views.Clients.as_view()),
7     path('client_detail/<int:pk>', views.ClientDetailsView.as_view()),
8     path('clients_filter/', views.ClientsList.as_view()),
9     path('rooms/', views.Rooms.as_view()),
10    path('workers/', views.WorkersView.as_view()),
11    path('del_worker/<int:pk>', views.DelWorker.as_view()),
12    path('room_detail/<int:pk>', views.RoomDetailsView.as_view()),
13    path('cleanings/<int:pk>', views.CleaningTable.as_view()),
14    path('cleanings_view/<int:pk>', views.CleaningsView.as_view()),
15    path('checkin/', views.AddCheckin.as_view()),
16    path('checkin_filter/', views.CheckinsList.as_view()),
17    path('floors/', views.FloorsView.as_view()),
18 ]

```

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 КЛИЕНТСТКОЙ ЧАСТИ

4.1. Описание средств разработки клиентской части

В качестве инструмента для разработки фронтенда был использован фреймворк Vue.js и интерфейс 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. [4]

Библиотека Muse UI представляет собой набор компонентов для Vue, которые используют Material Design [5]. Это фреймворк для быстрого создания и запуска пользовательского интерфейса с приятным и удобным дизайном.

4.2. Разработанные интерфейсы

4.2.1. Вход в систему

Стартовая страница является начальной точкой входа. (рис.7). На ней пользователю предлагается войти в систему либо зарегистрироваться.

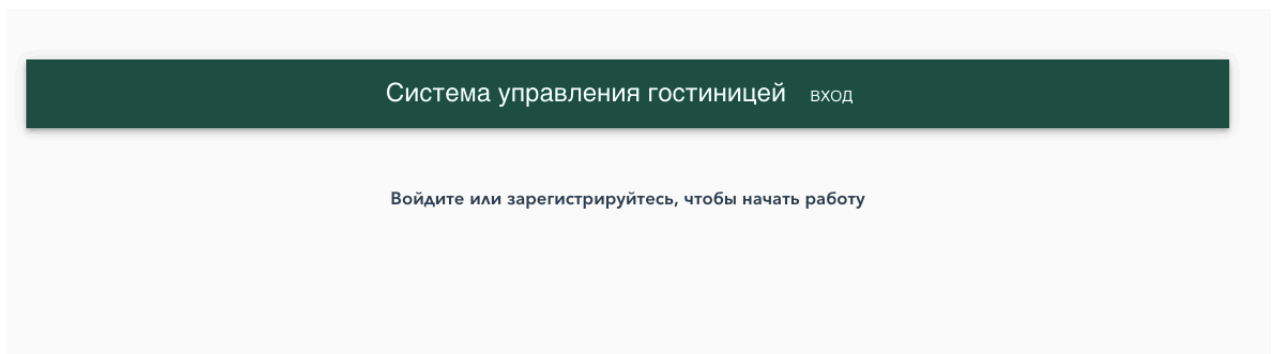
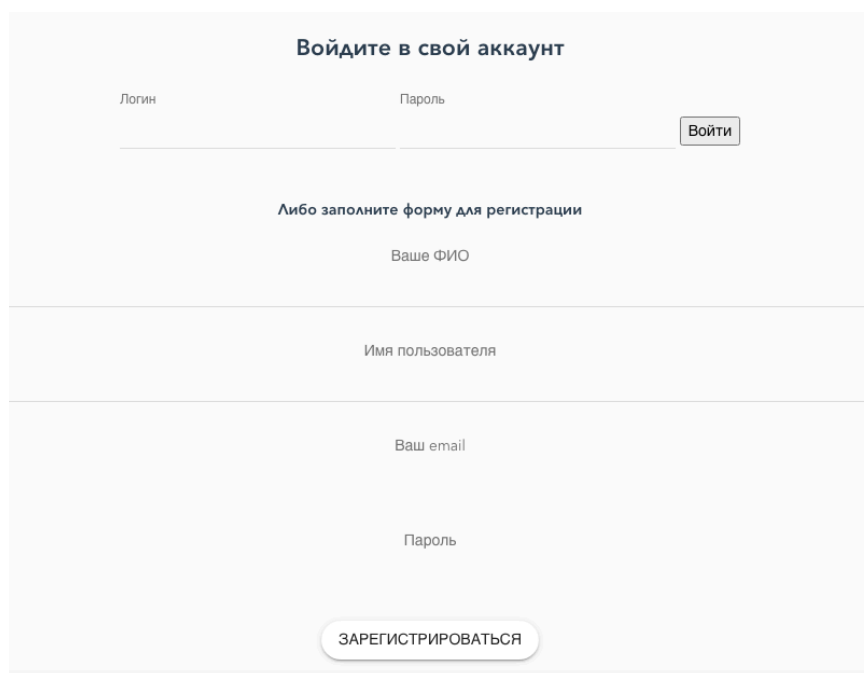


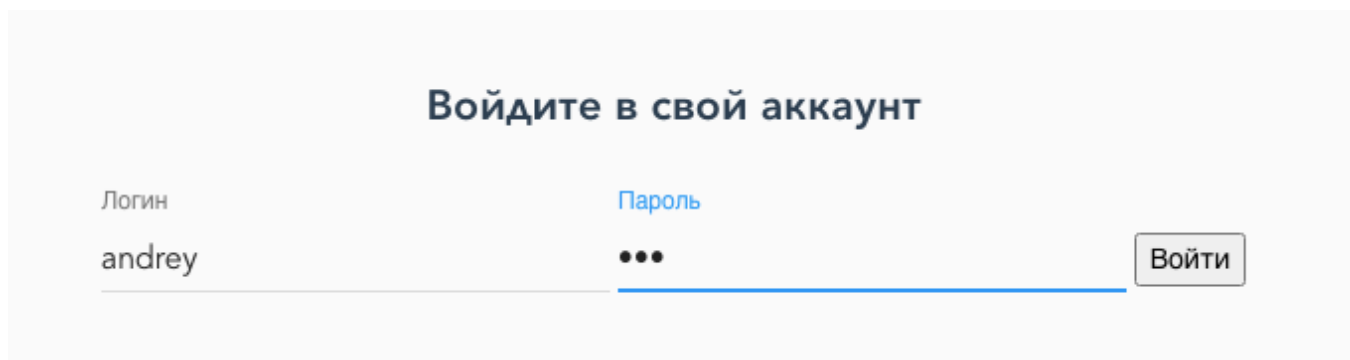
Рисунок 7 – Стартовая страница



The image shows a registration form titled "Войдите в свой аккаунт" (Log in to your account). It is divided into two sections. The top section is for login, with fields for "Логин" (Login) and "Пароль" (Password), and a "Войти" (Log in) button. The bottom section is for registration, with the heading "Либо заполните форму для регистрации" (Or fill out the registration form). It contains fields for "Ваше ФИО" (Your full name), "Имя пользователя" (Username), "Ваш email" (Your email), and "Пароль" (Password), followed by a "ЗАРЕГИСТРИРОВАТЬСЯ" (REGISTER) button.

Рисунок 8 – Пример заполненной формы регистрации

На рис. 8 представлена страница логина на сайт администрации гостиницы, где существующему пользователю предлагается ввести свои данные для входа, либо указать полную информацию для регистрации нового пользователя.



The image shows a login form titled "Войдите в свой аккаунт" (Log in to your account). It has two input fields: "Логин" (Login) with the text "andrey" and "Пароль" (Password) with masked characters "•••". A "Войти" (Log in) button is located to the right of the password field.

Рисунок 9 – Интерфейс входа в систему

4.2.2. Интерфейсы для администратора

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

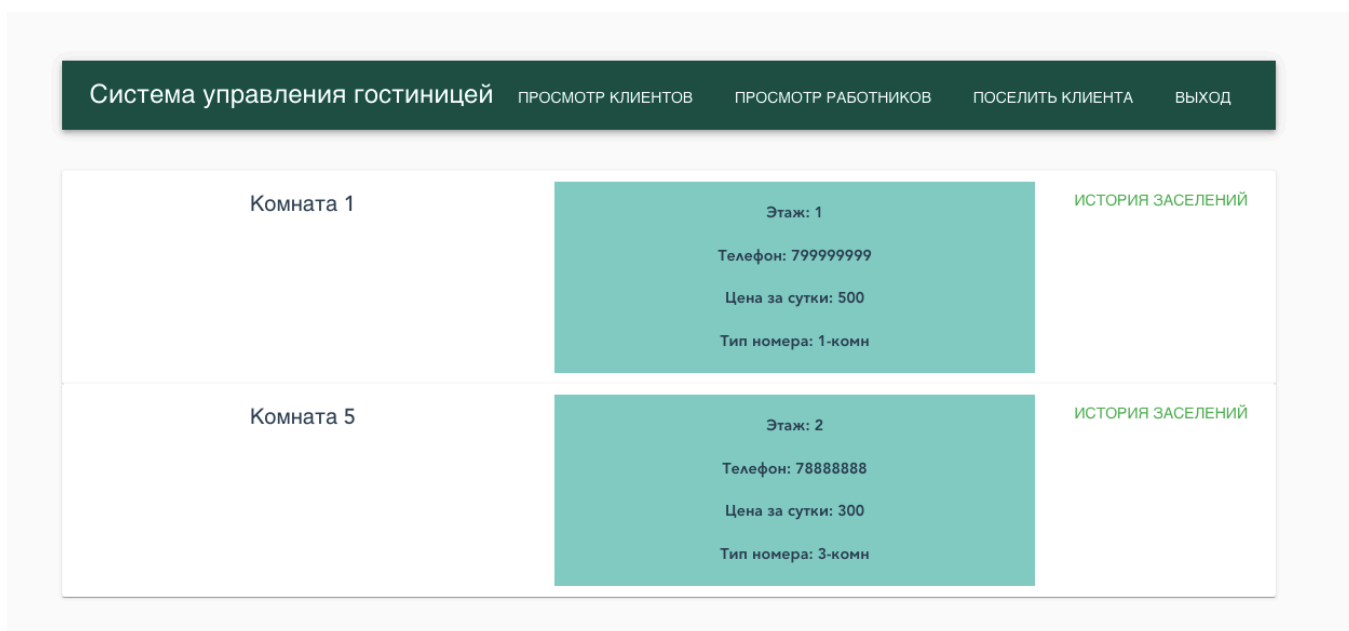


Рисунок 10 – Главная страница системы

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.

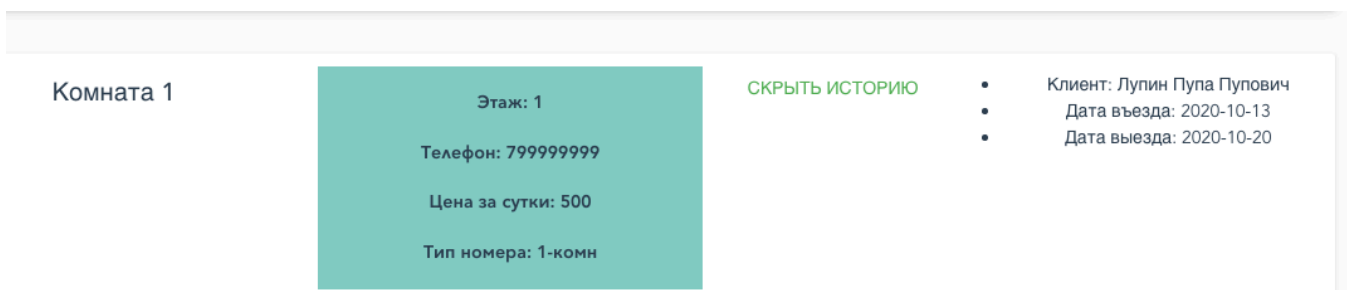


Рисунок 11 – Открытый список заселений в номер

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 - Лупин Пупа Пупович	Клиент - Иванов Иван Иванович	Клиент - Сергеев Петр Абрамович
Паспорт: 3333333234	Паспорт: 3333333333	Паспорт: 5554445555
Город: Санкт-Петербург	Город: Москва	Город: Стамбул

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

ПОДТВЕРДИТЬ ДОБАВЛЕНИЕ

14).

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

НАЙТИ

Клиент Лупин Пупа
Пупович

Паспорт 3333333234

Город Санкт-Петербург

Клиент Иванов Иван
Иванович

Паспорт 3333333333

Город Москва

Клиент Сергеев
Петр Абрамовович

Паспорт 555444555

Город Стамбул

Рисунок 14 – Пример поиска клиентов по городу

В отдельной странице можно заселить существующих клиентов (рис. 15).

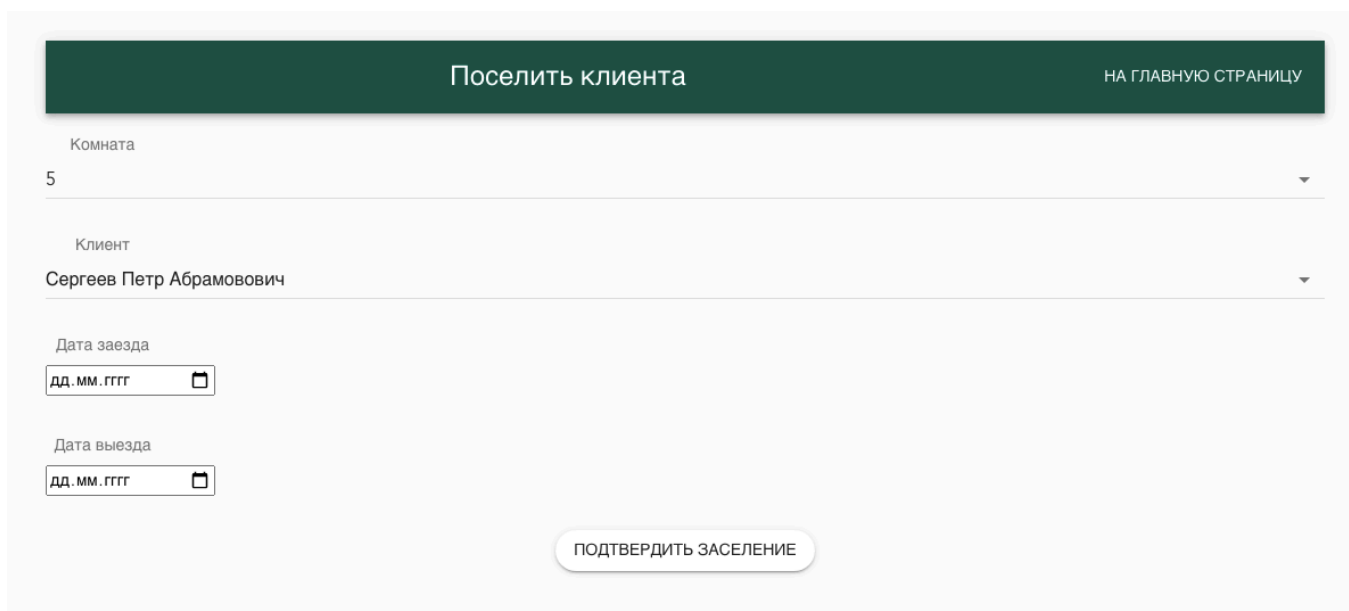


Рисунок 15 – Страница для заселения клиентов

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).

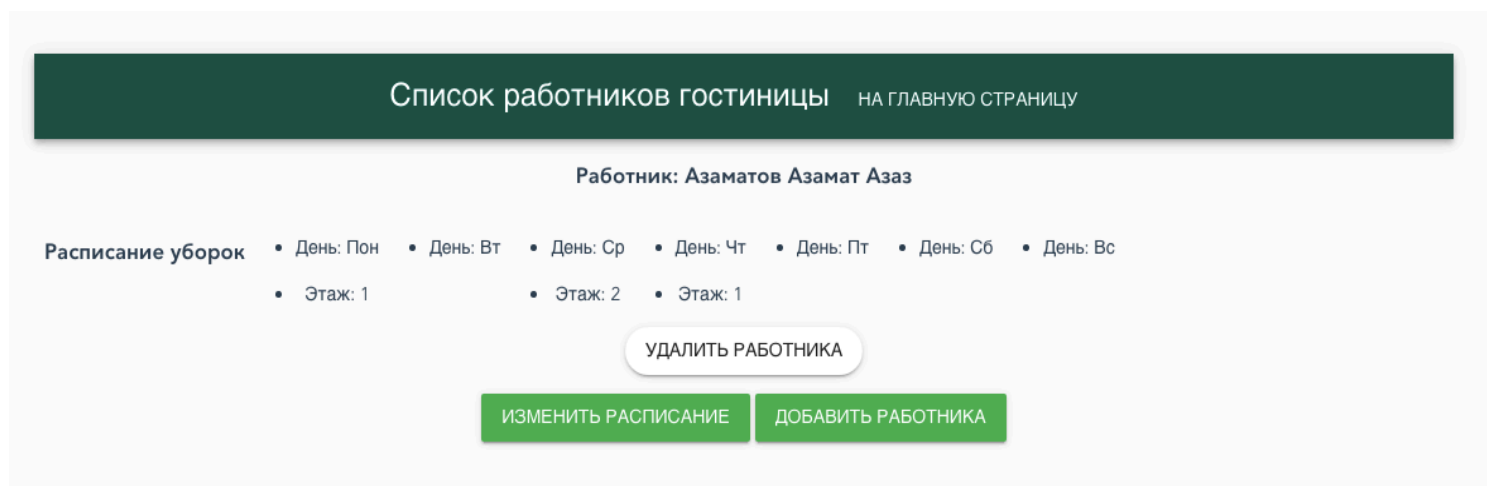
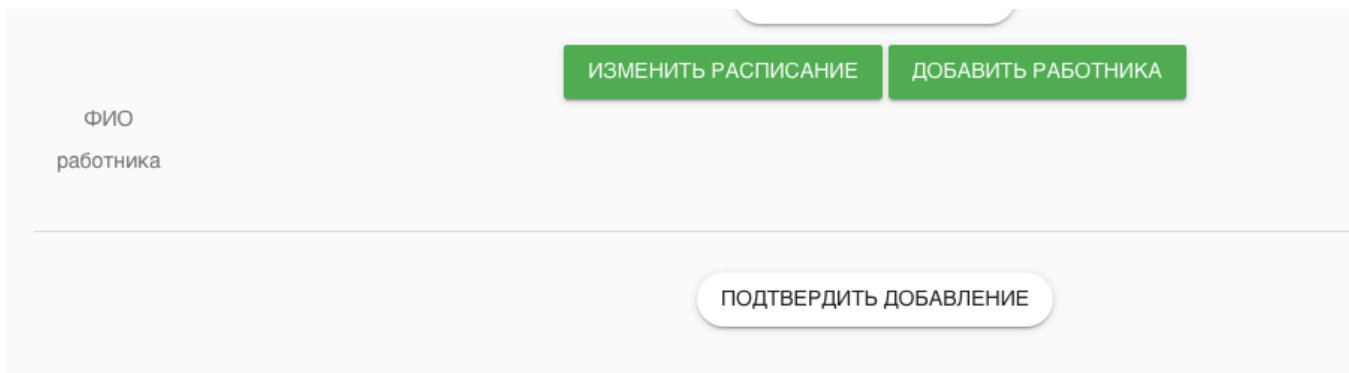


Рисунок 16 – Страница просмотра работников

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.

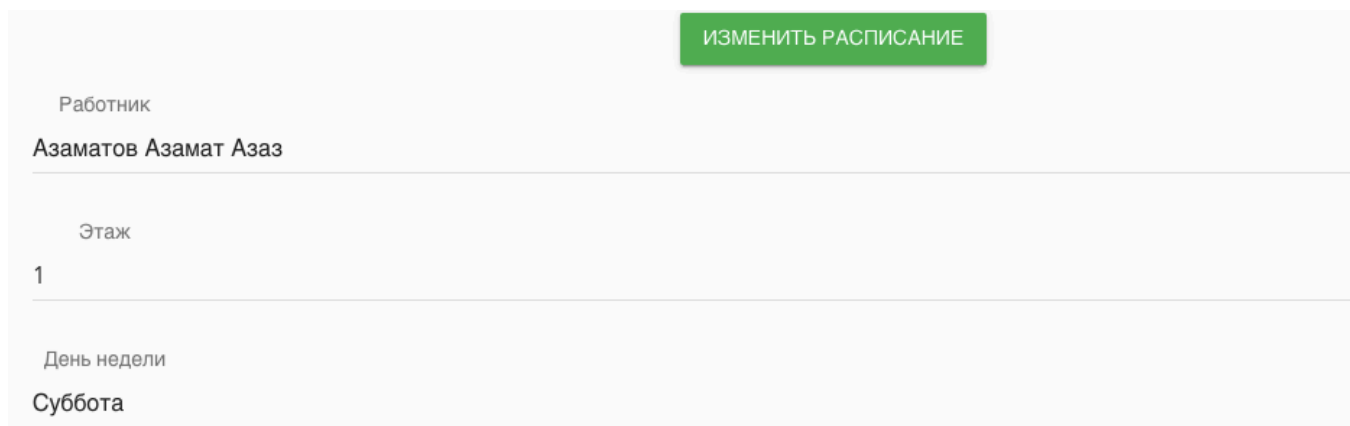
Каждого работника можно уволить, нажав на кнопку Удалить работника. Также можно добавить нового работника, тогда откроется форма добавления работника (рис. 17).



The form for adding a worker consists of a light gray background. At the top right, there are two green buttons with white text: 'ИЗМЕНИТЬ РАСПИСАНИЕ' and 'ДОБАВИТЬ РАБОТНИКА'. Below these buttons, on the left side, is a text input field with the placeholder 'ФИО работника'. At the bottom center, there is a white button with a green border and green text that says 'ПОДТВЕРДИТЬ ДОБАВЛЕНИЕ'.

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).



The form for changing a worker's schedule has a light gray background. At the top right, there is a green button with white text that says 'ИЗМЕНИТЬ РАСПИСАНИЕ'. Below this button, the form is divided into three sections by horizontal lines. The first section is labeled 'Работник' and contains the text 'Азаматов Азамат Азат'. The second section is labeled 'Этаж' and contains the number '1'. The third section is labeled 'День недели' and contains the word 'Суббота'.

Рисунок 18 – Форма для изменения расписания

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

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. «Клиент-серверная архитектура в картинках» [Электронный ресурс] — <https://habr.com/ru/post/495698/>. Дата обращения: 20.05.2020.
2. Документация Django Rest Framework [Электронный ресурс] — <https://www.django-rest-framework.org/topics/documenting-your-api/>. Дата обращения: 20.05.2020.
3. Документация PostgreSQL [Электронный ресурс] — <https://www.postgresql.org/docs/>. Дата обращения: 23.05.2020.
4. Документация Vue.js [Электронный ресурс] — <https://ru.vuejs.org/v2/guide/>. Дата обращения: 20.05.2020.
5. Документация Muse UI [Электронный ресурс] — <https://muse-ui.org/#/en-US>. Дата обращения: 30.05.2020.