

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет инфокоммуникационных технологий

Образовательная программа Интеллектуальные системы в гуманитарной сфере
(Академический бакалавр, Очная ф/о)

Направление подготовки (специальность) 45.03.04 Интеллектуальные системы в гуманитарной сфере

О Т Ч Е Т

о курсовой работе по дисциплине «Основы Web-программирования»

Тема задания: программная система, позволяющая отслеживать распределение по почтовым отделениям газет, печатающихся в типографиях города

Обучающийся Литвинова Елизавета Вячеславовна, группа К3342

Руководитель курсовой работы: Говоров А.И., ассистент факультета инфокоммуникационных технологий Университета ИТМО

Оценка курсовой работы _____

Подписи членов комиссии:

(подпись)

(подпись)

(подпись)

Дата 04.07.2020

Санкт-Петербург
2020

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	2
ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ	3
ВВЕДЕНИЕ.....	5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ	7
1.1. Описание системы	7
1.2. Выводы.....	8
2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ.....	9
2.1. Выбор технологий и инструментов для реализации системы.....	9
2.2. Разработка модели данных	9
2.3. Проектирование архитектуры системы	11
2.4. Выводы.....	12
3. ПРОГРАММИРОВАНИЕ СИСТЕМЫ.....	13
3.1. Серверная часть	13
3.2. Клиентская часть.....	14
3.3. Пользовательские интерфейсы.....	14
3.4. Контейнеризация и оркестрация	19
3.5 Выводы	19
ЗАКЛЮЧЕНИЕ	20
СПИСОК ЛИТЕРАТУРЫ.....	21
Приложение 1. Использованные модули и программы	22

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

AJAX (Asynchronous Javascript And Xml) – технология обращения к серверу без перезагрузки страницы.

API (программный интерфейс приложения, интерфейс прикладного программирования) (англ. application programming interface) – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

Backend – программно-аппаратная часть сервиса.

DELETE – HTTP метод, удаляющий указанный ресурс.

Django – свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC (Model-View-Controller, «Модель-Представление-Контроллер»).

Django REST Framework – это библиотека, которая работает со стандартными моделями Django для создания гибкого и мощного API для проекта.

Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.

Frontend – клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

GET – HTTP метод, позволяющий запрашивать представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript.

Muse-UI – фреймворк для создания интерфейсов, библиотека компонентов, основанная на Vue.js.

POST – HTTP метод, использующий для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

PostgreSQL – свободная объектно-реляционная система управления базами данных.

PUT – HTTP метод, заменяющий все текущие представления ресурса данными запроса.

URL (Uniform Resource Locator) – унифицированный указатель ресурса.

Vue.js – это прогрессивный фреймворк для создания пользовательских интерфейсов.

Атрибут – это информационное отображение свойств сущности.

Контейнеризация – метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного.

Оркестрация – это координация взаимодействия нескольких контейнеров.

Связь один-ко-многим – в типе связей один ко многим одной записи первой таблицы соответствует несколько записей в другой таблице.

Сериализация – это процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл.

Сущность – это любой объект в базе данных, который можно выделить исходя из сути предметной области, для которой разрабатывается эта база данных.

Хук жизненного цикла – функции, с помощью которых можно выполнять код на определённых этапах.

Эндпоинт – (endpoint – конечная точка) – обращение к маршруту отдельным HTTP методом. Эндпоинт выполняет конкретную задачу, принимает параметры и возвращает данные клиенту.

ВВЕДЕНИЕ

Цель работы – создание программной системы, позволяющей эффективно работать гостиничному бизнесу. Созданный продукт должен быть прост в использовании, отличаться удобным и понятным для любого человека интерфейсом. Проект также должен способствовать решению задач, с которыми сталкиваются сотрудники гостиниц в своей работе. От системы требуется работать быстро и не использовать большой объем вычислительных ресурсов, поскольку она взаимодействует с пользователем, который ожидает мгновенного результата.

Целевой аудиторией реализованного проекта являются как потенциальные клиенты отеля, так и сотрудники, а также люди, которые хотят получить работу в данной гостинице.

В процессе реализации проекта необходимо было изучить различные инструменты, используемые при создании веб-сервисов, а также научиться применять их на практике.

Задачи, требующие решения:

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. Описание включает в себя задачи, которые необходимо выполнять сотруднику, информацию, которая может понадобиться работнику, а также, полное описание компонентов системы

Во второй главе описан процесс проектирования и разработки системы. В первом пункте содержится обоснование выбора технологий и инструментов для реализации системы. В следующем пункте приведена разработка модели данных, а также построенная на этой модели схема, были описаны сущности, их атрибуты и связи между ними. В последнем пункте непосредственно описана разработка архитектуры системы с описанием взаимодействия выбранных технологий.

В третьей главе представлено описание процесса программирования системы. Рассмотрен подход к разработке серверной и клиентской частей системы. Приведено

краткое описание сериализаторов, представлений, URL- адресов проекта, объяснена структура Vue.js компонентов, контейнеризация и оркестрация проекта.

В заключении приводится краткая характеристика проделанной работы, описание достигнутых целей, обозначение дальнейших перспектив развития проекта.

В приложении указаны версии использованных модулей и программ.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1.1. Описание системы

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 варианта размещения: номер на одного человека, номер для двух человек, номер для трех человек. Бронируя номер, клиент указывает свою личную информацию - свое имя, фамилию, номер паспорта, дату заезда, город, из которого он приехал, гостиничный номер, который он планирует забронировать. 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.

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 оте

1.2. Выводы

Таким образом, были поставлены задачи, которые должна помогать решать система. Также описаны компоненты системы, которые послужат основой для разработки базы и модели данных. Были обозначены запросы, ответы на которые дают информацию, являющейся важной для сотрудников отдела.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1. Выбор технологий и инструментов для реализации системы

На основе анализа компонентов и функционала системы были обозначены технологии и инструменты для реализации системы:

- Django [1] для написания веб-приложения;
- Django REST Framework [3] для создания API;
- Vue.js [7] и Muse-UI [4] для создания для пользовательских интерфейсов;
- PostgreSQL [5] для управления базой данных;
- Docker [2] для контейнеризации и оркестрации.

2.2. Разработка модели данных

Модель была разработана согласно описанию системы в главе 1 (рис. 1).

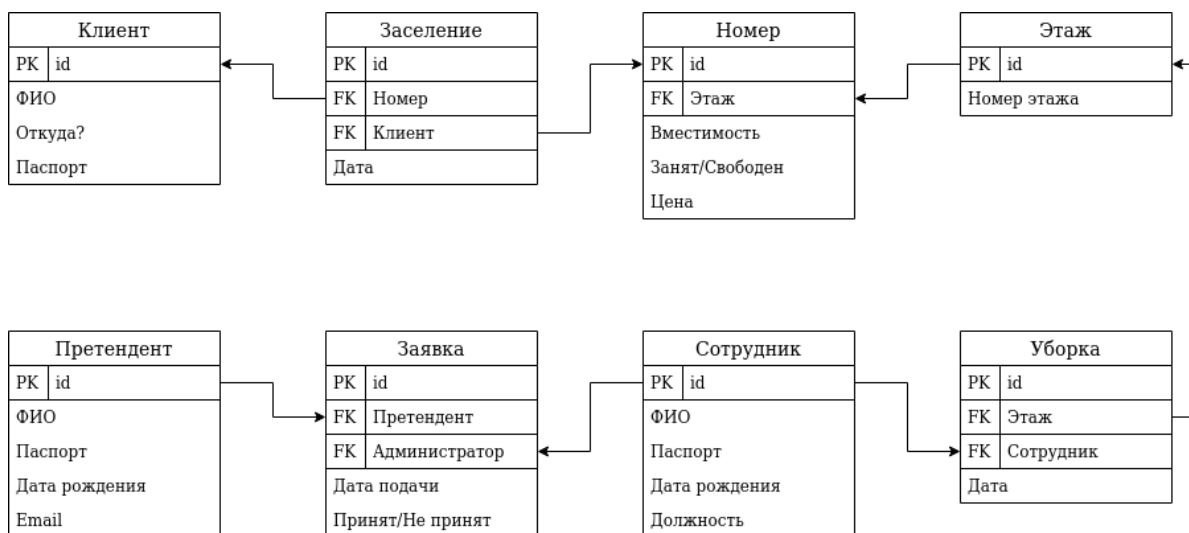


Рисунок 1 – Модель базы данных системы газет

Таким образом, модель состоит из следующих сущностей: Модель содержит 8 сущностей:

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

Атрибуты *Клиент*:

- ФИО;
- откуда;
- паспорт.

Атрибуты *Заселение*:

- ID (первичный ключ);
- номер;
- клиент;
- дата.

Атрибуты *Номер*:

- ID (первичный ключ);
- этаж;
- вместимость;
- статус (занято или свободно).

Атрибуты *Этаж*:

- ID (первичный ключ);
- этаж.

Атрибуты *Заявка*:

- id;
- дата подачи;
- принят\не принят.

Атрибуты *Сотрудник*:

- ФИО;
- дата рождения;
- паспорт.

Атрибуты *Уборка*:

- ID;
- этаж;
- сотрудник;
- дата.

2.3. Проектирование архитектуры системы

Для реализации системы была выбрана клиент-серверная архитектура. Сервером в данном случае считается абстрактная машина в сети, способную получить HTTP-запрос, обработать его и вернуть корректный ответ. Под клиентом понимается программная оболочка, с которой взаимодействует пользователь.

Для разработки приложения используются средства Django, написанном на Python. Этот инструмент удобен для разработки сайтов, работающих с базами данных. Непосредственно в качестве базы данных используется PostgreSQL, в основе которого лежит свободная объектно-реляционная система управления базами данных. Система создает и управляет моделями и запросами базы данных.

Для создания веб-API интегрируется инструмент Django REST Framework, с помощью которого осуществляется сериализация данных из PostgreSQL базы в JSON формат, позволяющий обмениваться данными между клиентской и серверной частями. REST API [6] интерфейс очень удобен для межпрограммного взаимодействия. REST API подразумевает под собой следующие правила: каждый URL является ресурсом; при обращении к ресурсу методом GET возвращается описание этого ресурса; метод POST добавляет новый ресурс; метод PUT изменяет ресурс; метод DELETE удаляет ресурс. Эти правила предоставляют простой CRUD (Create, Read, Update, Delete) интерфейс для других приложений, взаимодействие с которым происходит через протокол HTTP. В данном проекте GET запрос позволяет получить информацию из таблицы базы данных. В зависимости от целей запроса, результатом могут являться как все данные, так и только отдельный аспект данных, например, названия газет. POST запрос позволяет добавить новую запись в определенную таблицу базы данных. DELETE запрос позволяет удалять из таблицы определенную запись базы данных. PUT запрос позволяет изменить конкретную информацию уже существующей записи таблицы. Пользователь может изменить одно выбранное поле записи или несколько полей сразу.

Клиентская часть, разработанная средствами Vue.js и библиотеки Muse-UI для создания интерфейсов, передает данные в теле интересующего GET, POST, DELETE или PUT запроса через REST API. В результате запроса происходит изменение или извлечение данных из базы. Если данные введены не верно, или запрос выполнен некорректно, то появится сообщения об ошибке, вызванное соответствующим ошибке кодом состояния. Если запрос выполнен успешно, то программа продолжает свою работу.

2.4. Выводы

На основе анализа компонентов и функционала системы был выбран стек технологий, позволяющий эффективно и удобно создать системы, удовлетворяющую предъявляемым требованиям.

Спроектирована модель данных, которая в дальнейшем послужит основой для программирования модели с помощью обозначенных технологий. В модели обозначены сущности, их атрибуты и связи между ними.

Разработка архитектуры системы основывалась на выбранных инструментах, поставленных задачах и определенных интерфейсах.

Таким образом, этап разработки и проектирования системы достиг целей проекта в вопросах удобства система, ее эффективности и быстродействия.

3. ПРОГРАММИРОВАНИЕ СИСТЕМЫ

3.1. Серверная часть

Первым шагом в создании сайта является инициализация Django проекта. Также был подключен PostgreSQL, что было отражено в настройках.

Следующим действием выступает создание моделей. Модель является единственным источником информации о данных. Она содержит основные поля данных, которые хранятся. Каждая модель отображается в одной таблице базы данных. Каждая модель представляет собой класс Python, который является подклассом `django.db.models.Model`. Каждый атрибут модели представляет поле базы данных. Модели и их поля соответствуют сущностям и атрибутам базы данных (см. 2.3. Разработка модели данных).

Также были созданы сериализаторы. 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.

Вслед за этим были созданы представления, основанные на функциях. Каждое представление принимает объект `HttpRequest` в качестве первого позиционного аргумента и возвращает объект `HttpResponse` или вызывает исключение. Представления было создано для каждой модели, кроме Пользователя, данных с функциями `get` (GET-запрос), `get_object` (поиск объекта по ID), `delete` (DELETE-запрос), `put` (PUT-запрос), `post` (POST-запрос). Были созданы представления под каждый запрос информации, справки и отчета.

Таким образом, для взаимодействия API с сервером были созданы эндпоинты, некоторые из которых представлены ниже. Эндпоинт GET-запроса с URL-адресом сервера и /editorslist/, который возвращает полный список имен редакторов для представления в ниспадающем списке.

3.2. Клиентская часть

Для каждого интерфейса был разработан свой Vue.js компонент, состоящий из трех частей *template*, *script*, *style*. В части компонента *template* использовались Muse-UI элементы: панели, кнопки, таблицы, текстовые поля, вкладки, выпадающие списки. В части *script* содержится информация о данных компонента, хук создания (хук жизненного цикла) *created()*, а также функции компонента, описанные в методах системы. Некоторые функции выполняются автоматически при загрузке страницы, другие функции выполняются после нажатия на элемент интерфейса. Функции, выполняющиеся при загрузке страницы, позволяют отображать данные незамедлительно. Функции, выполняющиеся после нажатия на элемент интерфейса, подставлены в двух типах. Первые относятся к изменению какого-либо свойства элемента или совершению над ним действия. Примером может являться раскрытие панелей. Вторые содержат *Ajax* запросы.

3.3. Пользовательские интерфейсы

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.

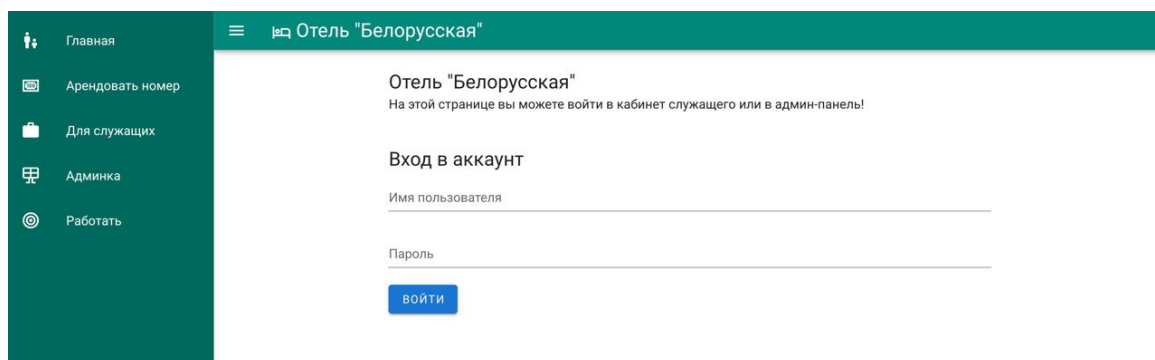


Рисунок 2 – Страница авторизации

Войдя в систему как администратор гостинцы, пользователь попадает на главную страницу системы (рис. 3).

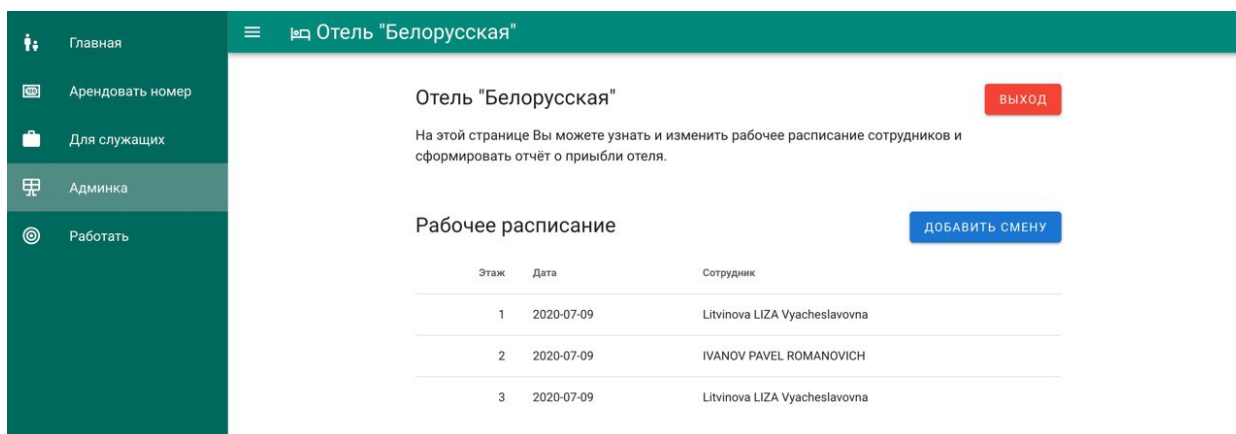


Рисунок 3 – Интерфейс кабинета администратора

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 расписании смен служащих отеля (рис. 3), о количестве свободных номеров, разделенных по их вместительности - на 1 персону, на 2 персоны, на 3 персоны, информация о доходе, полученном с занятых номеров (рис. 4). 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.

3	2020-07-09	Litvinova LIZA Vyacheslavovna
Отчёт		
Актуальная статистика		
Количество занятых номеров: 3		
Количество занятых номеров на одного человека: 1		
Количество занятых номеров на двух человек: 2		
Количество занятых номеров на трёх человек: 0		
Количество клиентов: 5		
Прибыль: 6000		

Рисунок 4 – статистика номеров

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 (рис. 5).

Отель "Белорусская"

Арендуя номер сегодня - вы получаете гарантированно счастливый отпуск и пару улыбок в подарок!

Аренда номера

Фамилия

Имя

Отчество

Город, из которого приезжаете

Выберите номер

с
ДД.ММ.ГГГГ

по
ДД.ММ.ГГГГ

АРЕНДОВАТЬ СБРОСИТЬ

Рисунок 5 – Интерфейс бронирования 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. Также перейдя в Django Administration, администратор увидит поданную заявку и свяжется с клиентом (рис. 6).

Django administration

Home > Hotel_App > Clients

Select client to change

Action: Go 0 of 3 selected

<input type="checkbox"/>	CLIENT
<input type="checkbox"/>	Komarov Igor Ilich
<input type="checkbox"/>	Rymancev Igor Ivanovich
<input type="checkbox"/>	Karapetov Ivan Ilich

3 clients

Рисунок 6 – Просмотр заявки на бронирование в Django Administration

Помимо бронирования номеров и личного кабинета для служащих существует возможность подать на работу в данную гостиницу. Это можно осуществить с главной страницы сайта, как и бронь номера (рис. 7).

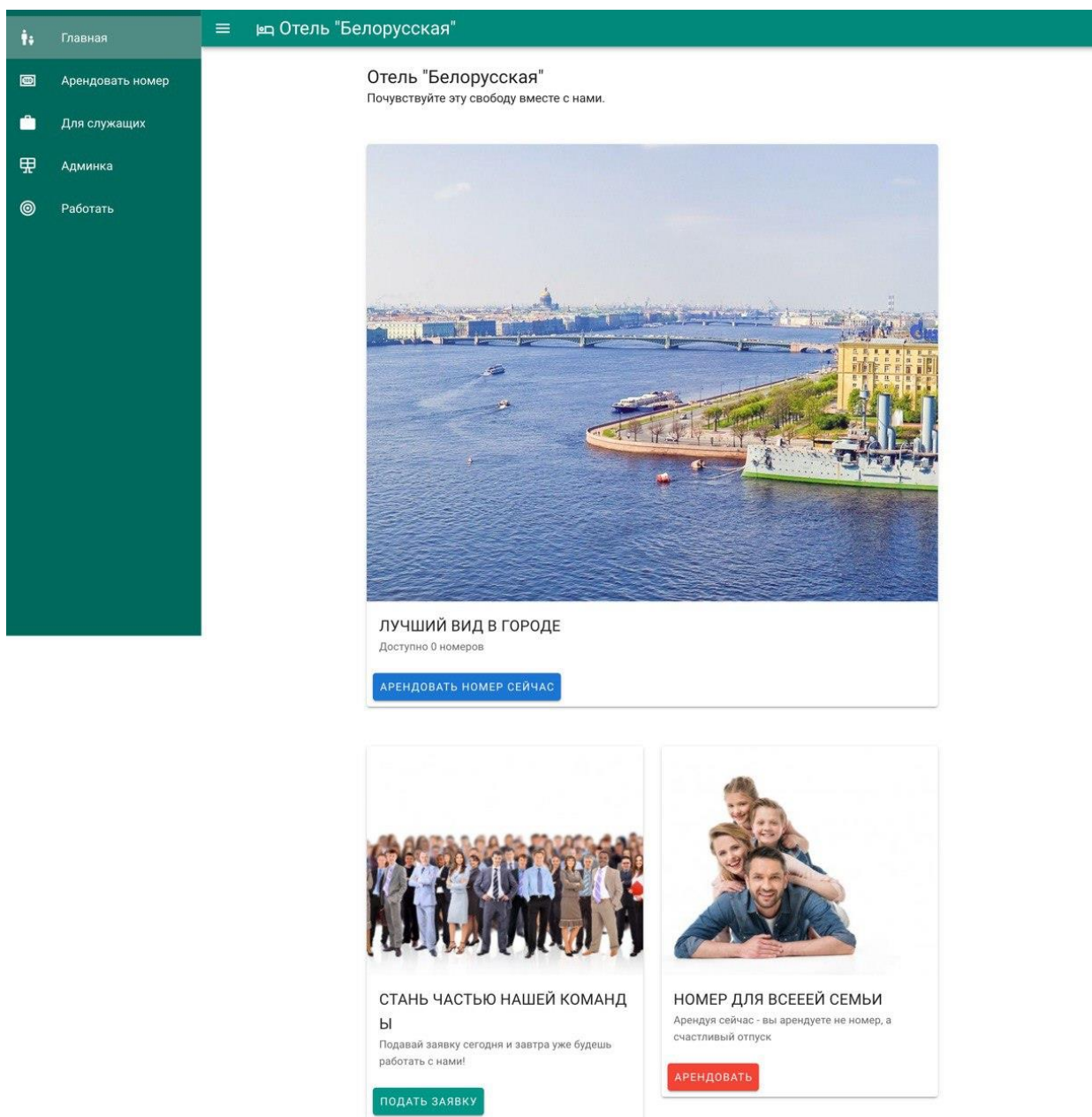


Рисунок 7 – главная страница сайта

Перейдя по кнопке «Подать заявку» пользователь попадает в анкету, заполнив которую личными данными (рис. 8) администратор получит его\её заявку на работу.

The screenshot shows the 'Отель "Белорусская"' application interface. On the left is a dark green sidebar with navigation links: Главная, Арендовать номер, Для служащих, Админка, and Работать. The main content area has a header with the hotel name and a welcome message. Below it is the 'Анкета для служащего' (Employee Application Form) with the subtitle 'Данные для входа' (Login Data). The form includes input fields for: Имя пользователя (Username), Пароль (Password), Ваши данные для анкеты (Your application data), Фамилия (Surname), Имя (Name), Отчество (Patronymic), Паспорт (Passport), Дата рождения (Date of birth) in DD.MM.YYYY format, and Стаж работы (Work experience). At the bottom are two buttons: 'ОТПРАВИТЬ ЗАЯВКУ' (Submit) and 'СБРОСИТЬ' (Reset).

Рисунок 8 – анкета для соискателей работы

Анкета, поступив администратору, может быть одобрена или нет. Также администратор имеет возможность не только нанять сотрудника из своего личного кабинета, но и уволить (рис.9).

The screenshot shows the 'Отель "Белорусская"' application interface for employee management. It features a header with the hotel name. The main content area includes a section for calculating profit and client count for a period, with input fields for 'с' (from) and 'по' (to) in DD.MM.YYYY format, and buttons 'ПЕРЕСЧИТАТЬ' (Calculate) and 'СБРОСИТЬ' (Reset). Below this is a section for calculating profit and client count for a specific city, with an input field for 'Название города' (City name) and buttons 'ВЫВЕСТИ' (Output) and 'СБРОСИТЬ' (Reset). The bottom section is titled 'Управление служащими' (Employee Management) and contains a table with columns: Сотрудник (Employee), Статус (Status), and Действие (Action).

Сотрудник	Статус	Действие
Ivanova Arina Ivanovna	Претендент	ПРИНЯТЬ
IVANOV PAVEL ROMANOVICH	Служащий	УВОЛИТЬ
Golybcov Anton Ivanovich	Служащий	УВОЛИТЬ

Рисунок 9 – интерфейс управления служащими

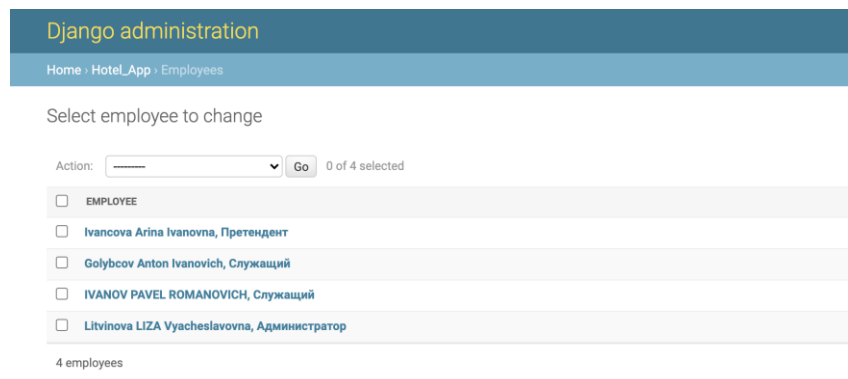


Рисунок 10 – Просмотр работников в Django Administration

3.4. Контейнеризация и оркестрация

Средствами docker и docker-compose проект контейнеризован и оркестрирован.

Все компоненты, необходимые для запуска приложения, упаковываются как один образ и могут быть использованы повторно. Приложение в контейнере работает в изолированной среде и не использует память, процессор или диск хостовой операционной системы. Это гарантирует изолированность процессов внутри контейнера.

1.. 3.5 Выводы

В результате шага программирования системы был создан каркас проекта, который был позднее доработан. По части backend были обновлены настройки и URL-адреса, добавлены представления, модели, сериализаторы. По части frontend были созданы Vue.js компоненты, которые послужили основой для клиентской части проекта. Ресурсы системы были виртуализированы и стали изолированы на уровне операционной системы.

ЗАКЛЮЧЕНИЕ

Рассмотрим достижение цели данного проекта.

Его непосредственной целью является создание программной системы, позволяющей отслеживать эффективную работу определенной гостиницы. Данная цель была достигнута в полной мере.

Созданный продукт прост в использовании, отличается удобным и понятным для любого человека интерфейсом. Проект также способствует решению задач, с которыми сталкиваются сотрудники гостиниц в своей работе. Система работает быстро и не использует большой объем вычислительных ресурсов, выполняет указанные действия мгновенно.

Каждый требуемый компонент рекомендательной системы был реализован.

Система была протестирована на предмет выявления ее недостатков. Задача включила в себя тестирование различных компонентов системы, тестирование работы системы при различных данных.

Данные, полученные в результате использования системы, являются адекватными и точными, так как система была разработана на принципе максимальной практичности для пользователя.

Дальнейшим шагом развития системы может являться расширение функционала на основе увеличения задач, требуемых к выполнению от сотрудников почты, доработка интерфейса относительно его комфорта в использовании, расширение базы данных за счет увеличения сущностей и атрибутов, которые может контролировать работник.

Таким образом, данный продукт отвечает поставленным на стадии его проектирования задачам и целям. Он имеет возможность быть востребованным на рынке, а также значительное время оставаться актуальным благодаря тому, что имеет множество перспектив развития.

СПИСОК ЛИТЕРАТУРЫ

1. Django documentation [Электронный ресурс] // Django documentation | Django documentation | Django [сайт]. 2005 – 2020. URL: <https://docs.djangoproject.com/en/3.0/> (дата обращения: 25.05.2020).
2. Get Started with Docker [Электронный ресурс] // Empowering App Development for Developers | Docker [сайт]. 2020. URL: <https://www.docker.com/> (дата обращения: 21.06.2020).
3. Home - Django REST framework [сайт]. 2011 – present. URL: <https://www.django-rest-framework.org/> (дата обращения 26.05.2020).
4. Muse-UI [Электронный ресурс] // Muse-UI[сайт]. 2016. URL: <https://muse-ui.org/#/en-US> (дата обращения 28.05.2020).
5. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс] // PostgreSQL: The world's most advanced open source database [сайт]. 1996-2020. URL: <https://www.postgresql.org/> (дата обращения 26.05.2020).
6. REST API [Электронный ресурс] // REST API — Основы Веб-программирования [сайт] (дата обращения 27.05.2020).
7. The Progressive JavaScript Framework [Электронный ресурс] // Vue.js [сайт]. 2014-2020. <https://vuejs.org/> (дата обращения 26.05.2020).

Приложение 1. Используемые модули и программы

Версии использованных модулей и программ:

- Django 3.0.4;
- django-allauth 0.41.0;
- django-cors-headers 3.2.1;
- django-rest-swagger 2.2.0;
- django-templated-mail 1.1.1;
- djangorestframework 3.11.0;
- djangorestframework-jsonapi 3.1.0;
- djangoframework-jwt 1.11.0;
- djoser 2.0.3;
- Docker Toolbox for Windows 19.03.1
- Muse-UI 3.0.2;
- PostgreSQL 11.8;
- Python 3.7.7;
- Vue.js 2.6.11.