Министерство науки высшего образования 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: Разработка программной системы, предназначенной для администрации аэропорта

Обучающийся Рейбандт Александр Александрович, К3342

Руководитель курсовой работы: Говоров Антон Игоревич, ассистент

Оценка по курсовой работе _____

Дата _____

Санкт-Петербург
2020

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ 4
1.1. Описания проекта 4
2. ОСНОВНАЯ ЧАСТЬ 5
2.1. Анализ требований и выбор средств разработки 5
2.2. Описание задач 5
2.2.1. Настройка рабочего окружения 5
2.2.2. Разработка макета и дизайна сайта 5
2.2.3. Реализация вкладки с выводом рейсов 5
2.2.4. Реализация системы фильтров 6
2.2.5. Реализация системы регистрации и авторизации 6
2.2.6. Создание сценария с потенциальной подачей заявления на работу пользователем и его дальнейшим рассмотрением администрацией аэропорта 6
2.2.7. Реализация возможности записаться на конкретные рейсы 7
2.2.8. Контейнеризация итогового проекта для удобства его развертывания в дальнейшем 7
2.3. Этапы работы по выполнению задания 7
2.3.1. Настройка рабочего окружения 7
2.3.2. Создание макета и дизайна сайта 7
2.3.3. Реализация вкладки с выводом рейсов 8
2.3.4. Реализация системы фильтров 10
2.3.5. Реализация системы регистрации и авторизации 12
2.3.6. Создание сценария с потенциальной подачей заявления на работу пользователем и его дальнейшим рассмотрением администрацией аэропорта 14
2.3.7. Реализация возможности записаться на конкретные рейсы 15
2.3.8. Контейнеризация итогового проекта для удобства его развертывания в дальнейшем 17

3. ВЫВОДЫ 19
3.1. Результаты 19
3.2. Список использованной литературы 20

1. ВВЕДЕНИЕ

1.1. Описания проекта

Данная работа посвящена изучению способов создания веб-приложений, их архитектуры, разбору хороших и плохих практик в этой области, и применению первых во время реального производственного процесса. Актуальность темы заключается в том, что современное общество невозможно представить без сферы веб-разработки. Она развивается семимильными шагами, и чтобы быть в курсе дел, нужно постоянно совершенствовать свои знания и изучать актуальные фреймворки и средства разработки веб-сайтов.

Основной целью курсовой работы по данной дисциплине было создание программной системы, предназначенной для администрации аэропорта некоторой компании-авиаперевозчика.

Конечная система должна включать в себя следующие интерфейсы:

- рейсы, включающие в себя следующую информацию: пункт вылета и назначения, дистанцию, авиакомпанию-перевозчика, дату и время вылета и прибытия, общее количество мест и количество проданных билетов, пересадки, тип самолета и состав летного экипажа;
- система фильтров для удобного представления рейсов;
- структура регистрации и авторизации для расширения возможностей взаимодействия с сайтом;
- возможность подачи заявления на работу пользователем и его дальнейшего рассмотрения администрацией аэропорта;
- сценарий с потенциальной записью на конкретные рейсы;
- гибкую систему навигации для переключения между интерфейсами.

2. ОСНОВНАЯ ЧАСТЬ

2.1. Анализ требований и выбор средств разработки

В первую очередь было проведено подробное изучение поставленной задачи для осознания необходимых программ и компонентов для ее выполнения. В итоге, описанная выше система реализовывалась с использованием следующих основных средств разработки:

- PostgreSQL (для работы с данными).
- Django + Django Rest Framework (DRF) (для реализации серверной части – бэк-энда).
- Vue.js (для реализации клиентской части – фронт-энда).
- Docker (в качестве средства контейнеризации).

Также использовалась среда разработки PyCharm.

2.2. Описание задач

2.2.1. Настройка рабочего окружения

В данном пункте необходимо определиться с программным обеспечением, которое будет использоваться в процессе реализации проекта. Сюда можно отнести язык программирования, его фреймворки и библиотеки, а также текстовый редактор для комфортной разработки.

2.2.2. Разработка макета и дизайна сайта

Здесь должна быть проработана структура страниц сайта с учетом принципов будущей верстки. Также должен быть оформлен их визуальный образ, включающий в себя технические и дизайнерские решения.

2.2.3. Реализация вкладки с выводом рейсов

В рамках этой задачи необходимо было продумать, в каком виде будут выводиться рейсы, чтобы не было их нагромождения, и информация о них была читабельна и интуитивна понятна. Также нужно было решить, какие данные стоит включить в общее описание для каждого вылета, а какие в детальное.

2.2.4. Реализация системы фильтров

В данном пункте нужно было решить, по каким критериям будет осуществляться фильтрация на сайте, а также как эти фильтры будут интегрированы в общую структуру проекта с точки зрения дизайна.

2.2.5. Реализация системы регистрации и авторизации

В рамках этой задачи необходимо было создать возможность регистрации с последующей авторизацией на сайте. Здесь важно было также учесть, какие поля нужно включить в необходимые для заполнения во время регистрации.

2.2.6. Создание сценария с потенциальной подачей заявления на работу пользователем и его дальнейшим рассмотрением администрацией аэропорта

Эта задача заключалась, в основном, в том, что нужно было разработать форму для подачи заявления, выбрать место ее расположение в общей эко-структуре проекта, а также реализовать вывод всех поданных заявок, когда вход на сайт осуществляется админом.

2.2.7 Реализация возможности записаться на конкретные рейсы

В этом пункте нужно было создать возможность для пользователя записаться на нужный ему рейс.

2.2.8 Контейнеризация итогового проекта для удобства его развертывания в дальнейшем

Финальной целью было «упаковать» реализованное приложение со всем его окружением и зависимостями в контейнер для автоматизации его развёртывания и управления в будущем.

2.3. Этапы работы по выполнению задания

2.3.1. Настройка рабочего окружения

Первым этапом стала установка и настройка необходимого софта, то есть *PostgreSQL*, *DRF*, *Vue.js* и *Docker*.

PostgreSQL – свободная объектно-реляционная система управления базами данных (СУБД).

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

Vue.js – это *JavaScript*-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других *JavaScript*-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле. Установить его можно с официального сайта.

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

2.3.2. Создание макета и дизайна сайта

Следующий этап относился уже к конкретной разработке и заключался в создании структуры и дизайна сайта, которые должны были стать основой дальнейшей работы. Для разработки приятного дизайна (красивых шрифтов, списков и карточек) использовался *Bootstrap 4*. Его кэшированная версия была

подключена к проекту через *CDN*. Было принято решение создать навигационную панель, из которой можно будет удобно переключаться между нужными вкладками. Также в оформлении веб-сайта использовались *CSS* стили.

К завышающей стадии данного этапа реализуемый сайт выглядел следующим образом:

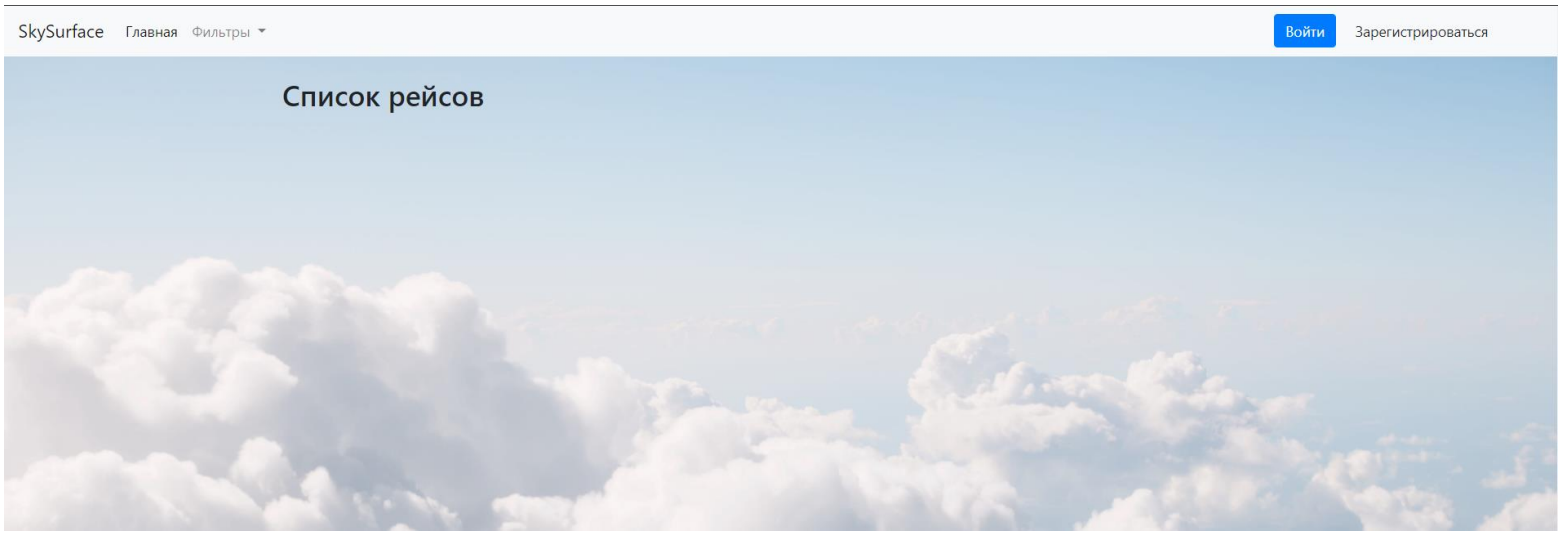


Рисунок 1 – Дизайн и структура сайта на начальном этапе

2.3.3. Реализация вкладки с выводом рейсов

Вывод всех рейсов был осуществлен на вкладке «Главная». Для этого на серверной стороне к модели данных были подключены сериализаторы (гибкий и мощный компонент DRF). С их помощью информация, хранящаяся в базе данных, была легко преобразована и эффективно передана через API. В видах (*ViewSet*) были определены функции для вывода рейсов (использовался наиболее распространённый *ViewSet* – *ModelViewSet*). Затем в файле urls.py были определены URL-адреса, которые предоставляют доступ как к общему просмотру всех рейсов, так и к конкретному рейсу в отдельности. Связать бэк-энд с фронт-эндом получилось благодаря отправке GET-запросов к серверу (использовался современный метод *fetch()*).

Теперь список рейсов имеет такой вид:

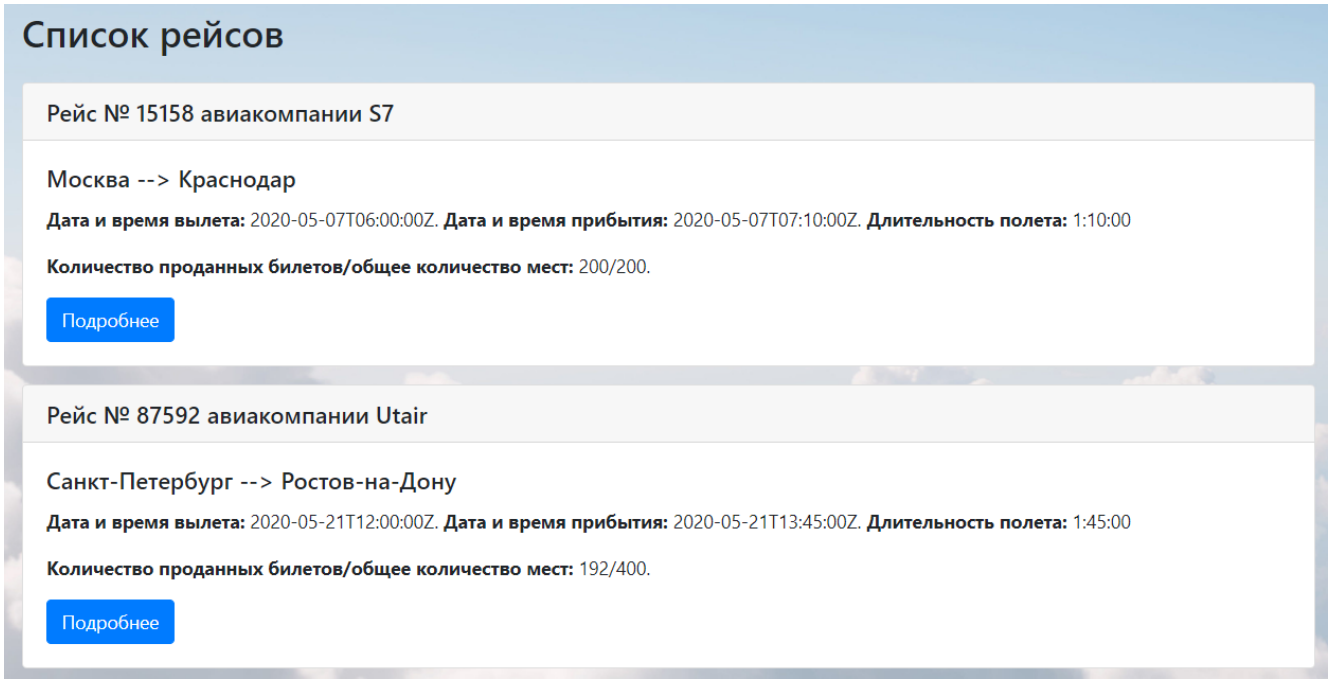


Рисунок 2 – Отображение всех рейсов

А информация о конкретном вылете представляется после нажатия на кнопку «Подробнее»:

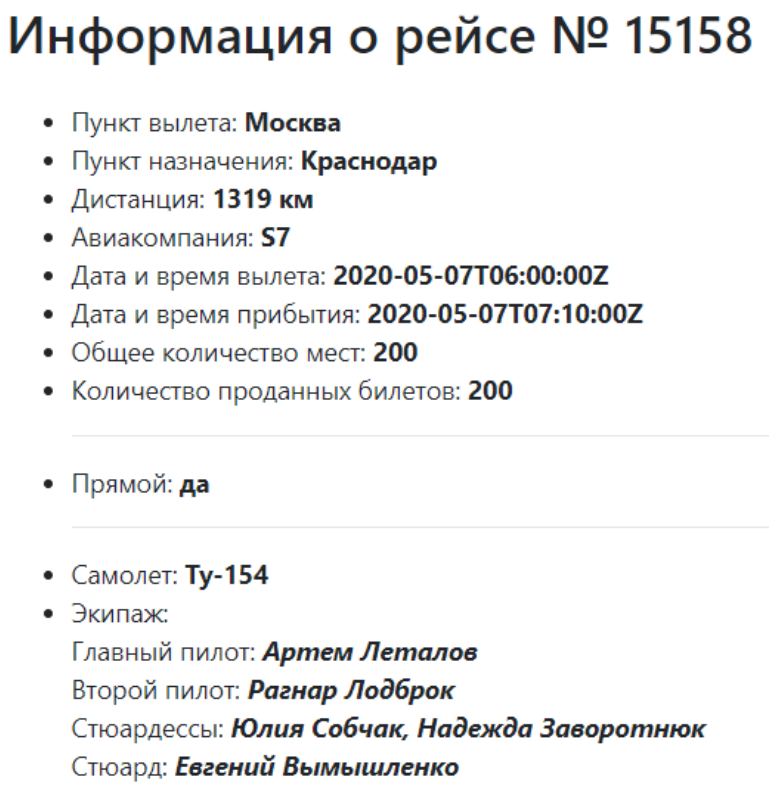


Рисунок 3 – Детальное отображение рейса

2.3.4. Реализация системы фильтров

На этом этапе был реализован ряд фильтров, позволяющих улучшить удобство использования сайта. Среди доступных фильтров:

- Рейсы только со свободными местами.
- Рейсы без пересадок.
- Возможность отфильтровать вылеты по авиакомпаниям.

Было принято решение создать систему фильтрации с использованием хранилища *store* (инструмент *Vuex*) для большей универсальности и соблюдения принципа *DRY (Don't repeat yourself)*. Фрагмент кода, воплощающий описанные фильтры в жизнь представлен ниже.

```
const store = new Vuex.Store({
    state: {
        backendUrl: "http://127.0.0.1:8000/api/v1",
        filters: {},
        flights: [],
        airline_companies: [],
        profile: {},
    },
    mutations: {
        setFilters(state, filters) {
            state.filters = {
                ...state.filters,
                ...filters,
            };
        },
        setFlights(state, flights) {
            state.flights = flights;
```

```
        },
        setAirlineCompanies(state, airline_companies) {
          state.airline_companies = airline_companies;
        },
        setProfile(state, profile) {
          state.profile = profile;
        }
      },
      actions: {
        async getAirlineCompanies({commit, state}) {
          const airline_companies = await fetch(`${state.backendUrl}/flight/airline_company`);
          commit('setAirlineCompanies', await airline_companies.json());
        },
        async getFlights({commit, state}, filters) {
          commit('setFilters', filters);

          const formData = new FormData();
          Object
            .keys(filters || {})
            .forEach(filter => formData.append(mapFilterToQueryParam(filter), filters[filter]));

          const queryString = new URLSearchParams(formData).toString();

          const flights = await fetch(`${state.backendUrl}/flight/?${queryString}`);
          commit('setFlights', await flights.json());
        }
      }
```

Визуально вкладка с фильтрами теперь выглядит так:

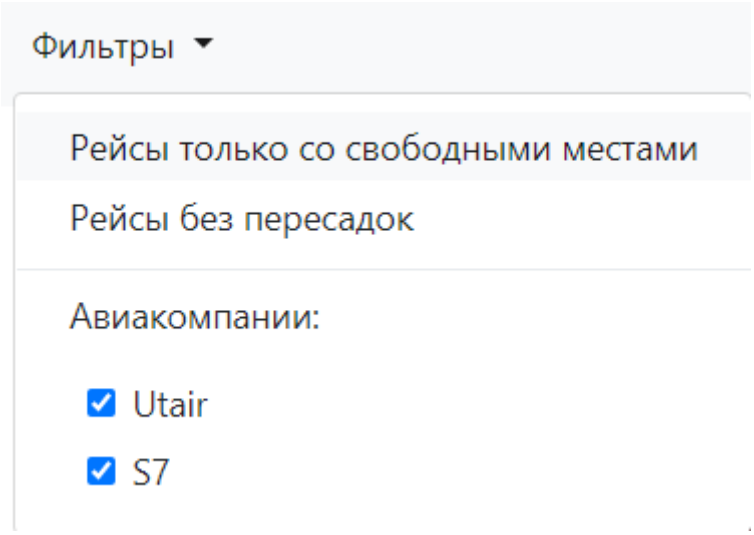


Рисунок 3 – Вкладка с фильтрами

2.3.5. Реализация системы регистрации и авторизации

Данный этап разработки включал в себя создание расширенной регистрации с последующей авторизацией на сайте на основе выдаваемого токена. В процессе реализации последней использовался инструмент *AuthToken*, предоставляемый *DRF*. Так как профиль пользователя — это нечто глобальное, и к нему должны иметь доступ практически все компоненты и виды, то опять же было решено вынести в клиентской части процесс его загрузки в хранилище *store*. Это продемонстрировано в следующем куске кода:

```
loadProfile({commit, state}, {token, id}) {
        return new Promise((resolve, reject) => {
            $.ajax({
                url: `${state.backendUrl}/flight/profile/${id}/`,
                type: "GET",
                headers: {'Authorization': "Token " + token},
                success: (response) => {
                    commit('setProfile', response);
                    resolve();
                },
                error: (error) => {
                    reject(error);
                }
            })
        });
```

Как было упомянуто выше, количество полей, необходимых для регистрации было увеличено для упрощения взаимодействия с сайтом. Сюда были включена такие поля, как логин, имя, фамилия, возраст, паспортные данные и пароль. На стороне клиента регистрация и авторизация теперь выглядит следующим образом:

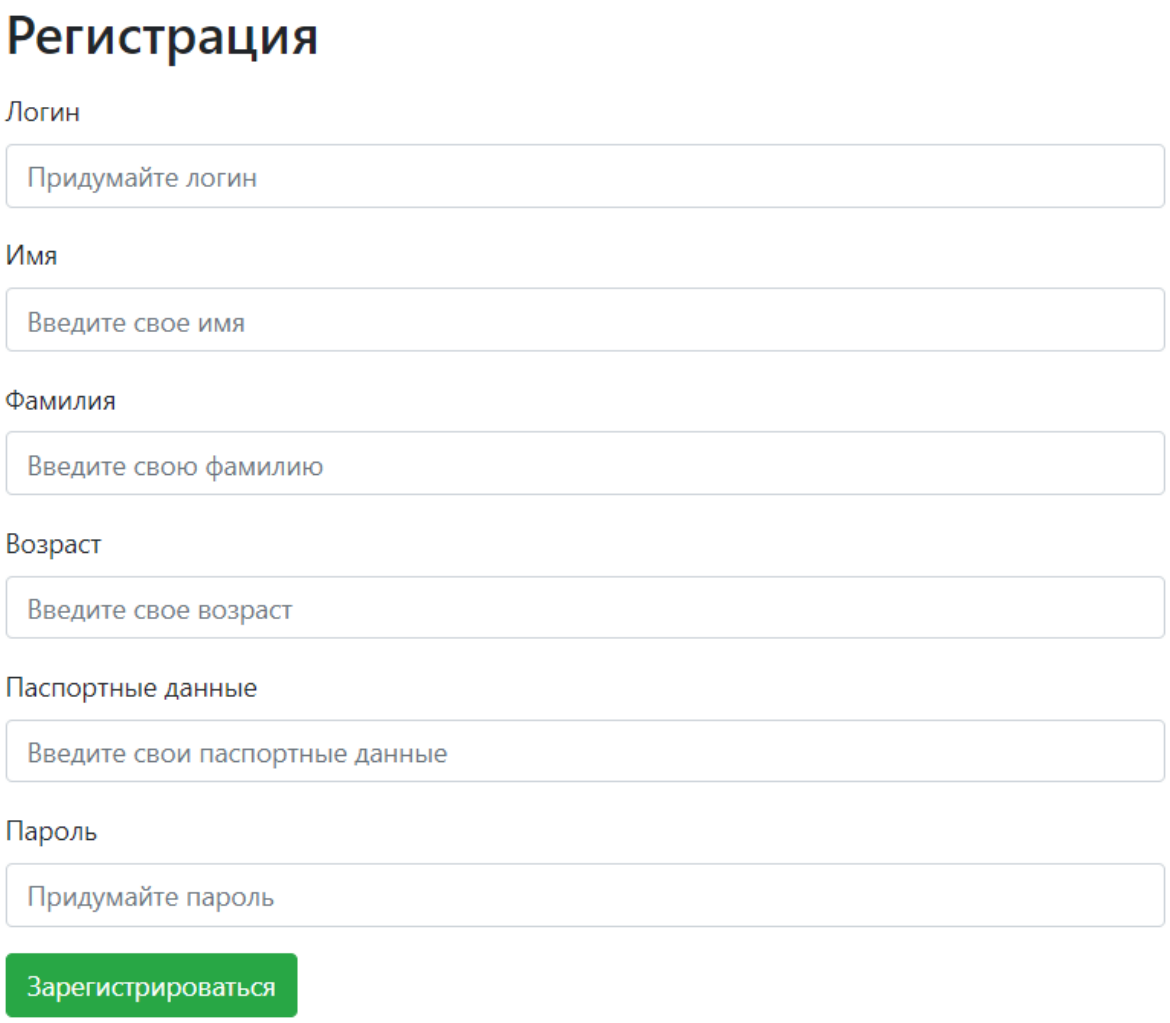


Рисунок 4 – Регистрация пользователя

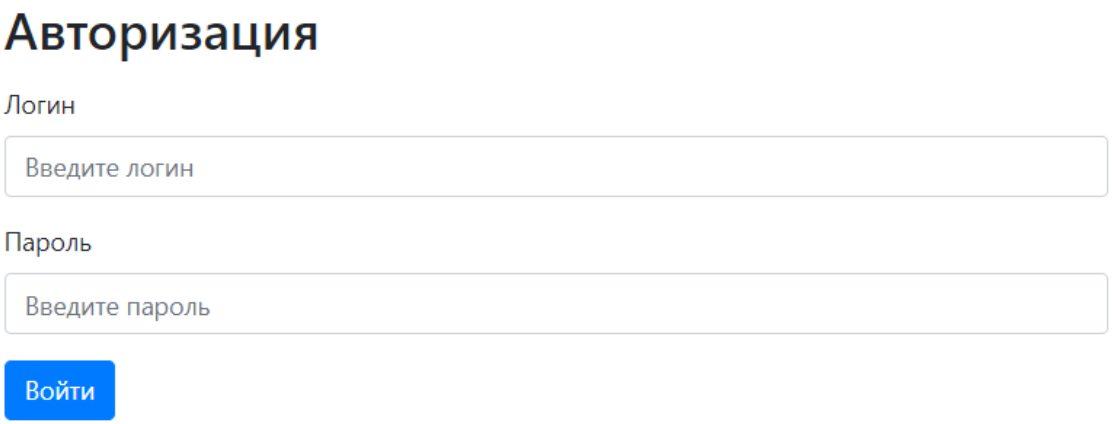


Рисунок 5 – Авторизация пользователя

2.3.6. Создание сценария с потенциальной подачей заявления на работу пользователем и его дальнейшим рассмотрением администрацией аэропорта

С целью создания возможности подачи заявления на работу был создан личный кабинет пользователя, содержащий актуальную информацию о нем, а также была предусмотрена возможность заполнить поля формы и отправить заявку. Здесь стоит обратить внимание, что изначально человек, создавший аккаунт, получает статус «Пользователь сайта». После отправки формы же его статус меняется на «На рассмотрении». Также возможны такие статусы, как «Является сотрудником» и «Уволен». На данном этапе разработке личный кабинет выглядит следующим образом:

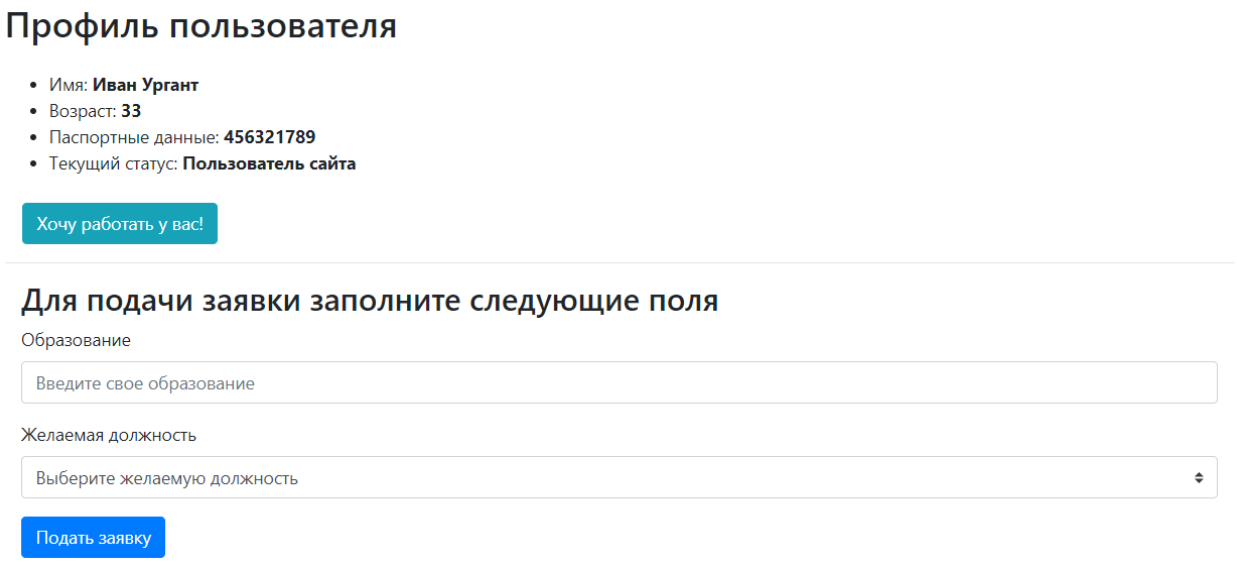


Рисунок 6 – Личный кабинет пользователя на этапе подачи заявления на работу

Список всех поданных заявок доступен администратору аэропорта в его личном кабинете. Он выглядит так:

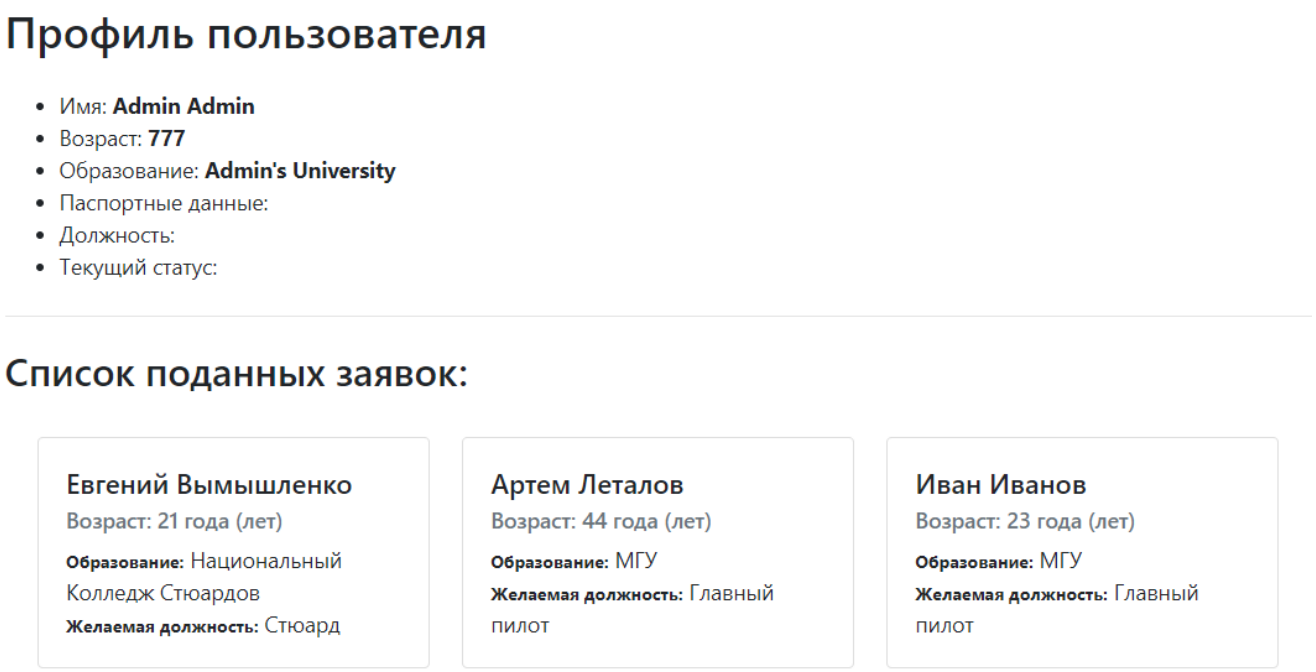


Рисунок 7 – Список поданных заявок в личном кабинете администратора

2.3.7 Реализация возможности записаться на конкретные рейсы

На данном этапе стояла задача дать пользователям возможность записываться на рейсы. Это было реализовано во вкладке детального просмотра вылета. Также были учтены такие факторы, как, например, если человек не авторизован или не зарегистрирован, то ему возможность записи не доступна. Помимо этого, если количество проданных билетов на рейс равно общему количеству мест, то зарегистрироваться на такой вылет тоже нельзя. Плюсом ко всему, при успешной отправке запроса на регистрацию количество проданных билетов динамически меняется согласно числу людей, записанных на рейс.

Теперь во вкладке детального просмотра рейса появилась кнопка «Записаться на рейс», а также кнопка «Назад», которая возвращает пользователя назад (подобный функционал был реализован с помощью библиотеки маршрутизации *Vue Router*, легко интегрируемой с *Vue.js*):

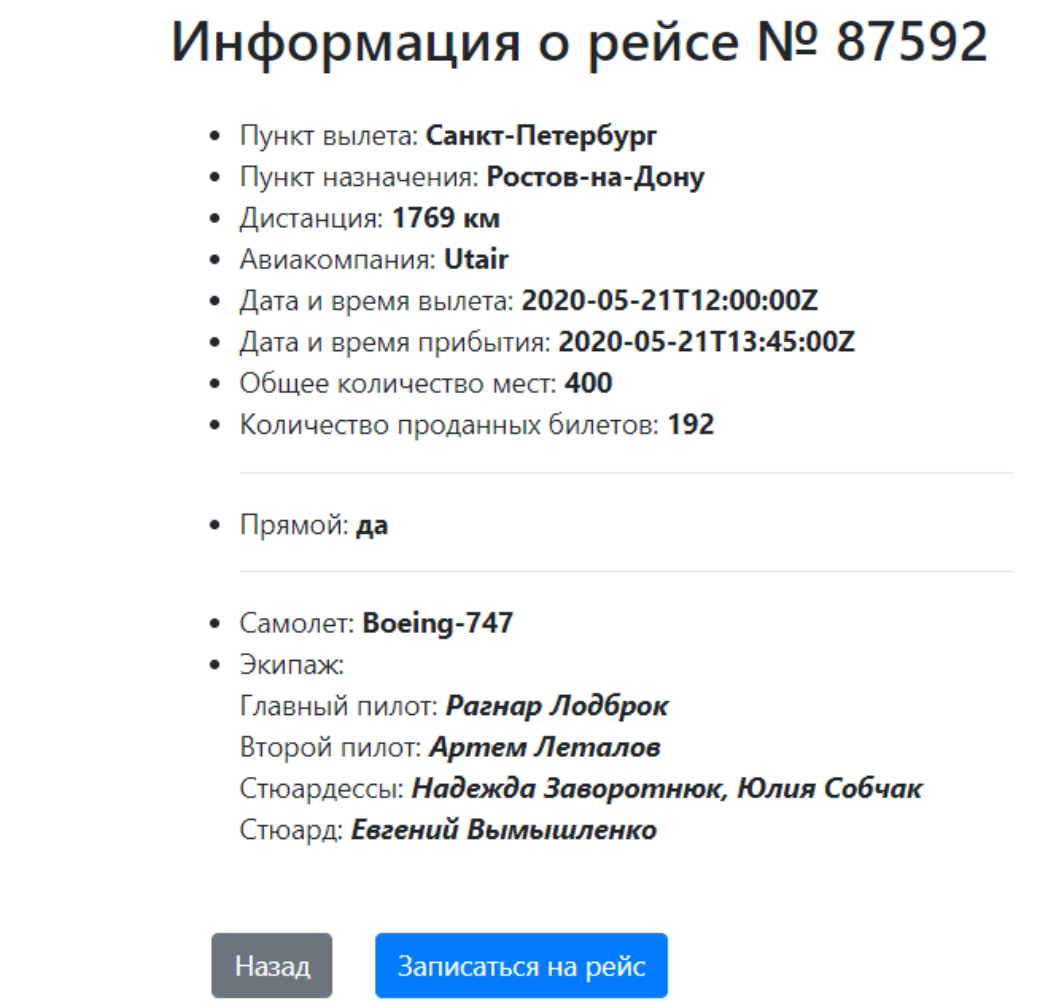


Рисунок 8 – Детальное отображение рейса с возможностью записаться на рейс

После успешной записи на рейс пользователи могут получить доступ к их рейсам в личном кабинете, нажав на кнопку «Мои рейсы»:

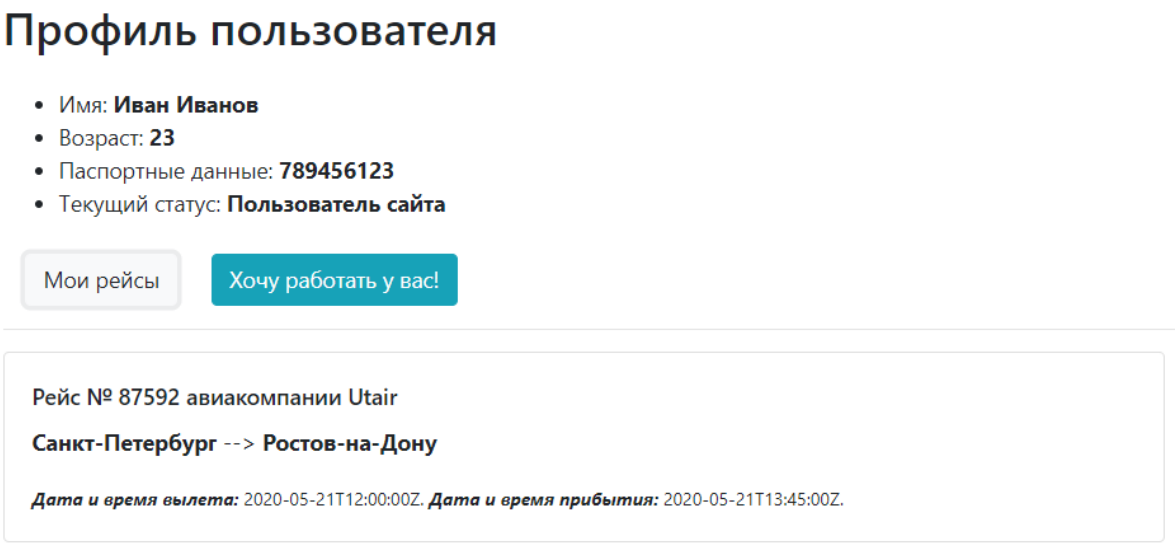


Рисунок 8 – Профиль пользователя с рейсами

2.3.8 Контейнеризация итогового проекта для удобства его развертывания в дальнейшем

В данном пункте было создано два *Docker file* отдельно для бэк-энда и фронт-энда. В каждом файле содержатся инструкции по созданию конкретного образа. Они представляют из себя следующее:

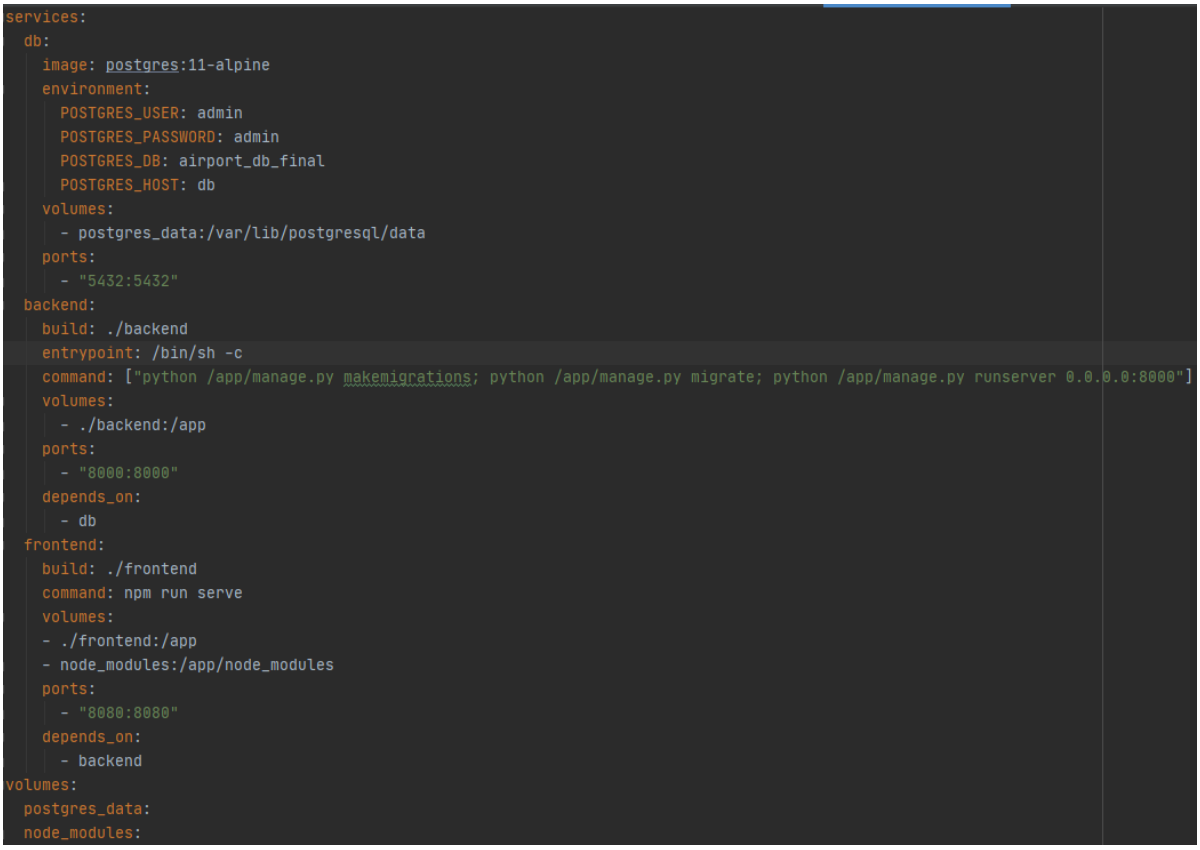
```
FROM python:3.7-alpine
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1
RUN mkdir /app
WORKDIR /app
COPY . /app
RUN apk update
RUN apk add postgresql-dev gcc python3-dev musl-dev
RUN pip install -r requirements.txt
```

Рисунок 9 – Docker file для бэк-энда

```
FROM node:lts-alpine
WORKDIR /app
COPY . .
RUN npm install
```

Рисунок 10 – Docker file для фронт-энда

Затем из образов были созданы три контейнера: база данных, Django и Vue.js. Создание и запуск данных контейнеров осуществлялся с помощью такого инструмента, как *Docker Compose*. В нем использовался специальный конфигурационный файл (*docker-compose.yml*). Его содержание представлено ниже.



```yaml
services:
  db:
    image: postgres:11-alpine
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin
      POSTGRES_DB: airport_db_final
      POSTGRES_HOST: db
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
  backend:
    build: ./backend
    entrypoint: /bin/sh -c
    command: ["python /app/manage.py makemigrations; python /app/manage.py migrate; python /app/manage.py runserver 0.0.0.0:8000"]
    volumes:
      - ./backend:/app
    ports:
      - "8000:8000"
    depends_on:
      - db
  frontend:
    build: ./frontend
    command: npm run serve
    volumes:
    - ./frontend:/app
    - node_modules:/app/node_modules
    ports:
      - "8080:8080"
    depends_on:
      - backend
volumes:
  postgres_data:
  node_modules:
```

Рисунок 11 – Конфигурационный файл docker-compose.yml

Финальным этапом после внесения всех необходимых инструкций стала сборка проекта (команда *docker-compose build*) с дальнейшим его запуском (команда *docker-compose up*).

2. ВЫВОДЫ

3.1. Результаты

В ходе выполнения курсовой работы по дисциплине «Основы web-программирования» была создана программная система, предназначенная для администрации аэропорта. Главным интерфейсом разрабатываемого сайта стали рейсы, включающие в себя следующую информацию: пункт вылета и назначения, дистанцию, авиакомпанию-перевозчика, дату и время вылета и прибытия, общее количество мест и количество проданных билетов. Также были включены такие данные, как пересадки, тип самолета и состав летного экипажа. Помимо этого, была разработана система регистрации и авторизации, позволяющая пользователям удобно взаимодействовать с сайтом. Плюсом ко всему стала реализованная система фильтров, позволяющая получить нужную и актуальную информацию о вылетах. Была предусмотрена возможность подачи пользователями заявок на работу, а также потенциальный сценарий с записью на рейс. Финальным этапом в разработке проекта стала его контейнеризация для удобства его развертывания в дальнейшем.

3.2. Список использованной литературы

1. Введение — Vue.js [Электронный ресурс]. – URL: https://ru.vuejs.org/v2/guide/
2. Введение — Bootstrap [Электронный ресурс]. – URL: https://bootstrap-4.ru/docs/4.4/getting-started/introduction/
3. Django REST framework [Электронный ресурс]. – URL: https://www.django-rest-framework.org/
4. Современный учебник JavaScript [Электронный ресурс]. – URL: https://learn.javascript.ru/
5. Docker documentation [Электронный ресурс]. – URL: https://docs.docker.com/