

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,  
МЕХАНИКИ И ОПТИКИ”**

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**

по курсовой работе по дисциплине «Основы Web-программирования»

Тема задания: **«Web-сервис для администрации аэропорта некоторой компании-авиаперевозчика»**

Обучающийся: Назаренко У. К., группа К3343

Преподаватель дисциплины: Говоров А.И., ассистент кафедры ИТГС Университета ИТМО

Оценка за курсовую работу \_\_\_\_

Подпись преподавателя:

\_\_\_\_ (\_\_\_\_\_)  
(подпись)

Дата \_\_\_\_

## Содержание

ВВЕДЕНИЕ.....	3
ГЛАВА 1 .....	4
1.1. Анализ предметной области .....	4
1.2. Модель базы данных «Компания-авиаперевозчик» .....	5
1.3. Состав реквизитов сущностей .....	5
1.4. Теоретические сведения об используемых технологиях .....	6
ГЛАВА 2 .....	7
2.1. Проектирование приложения .....	7
2.2. Серверная часть приложения.....	7
2.2.1. Полученные интерфейсы в панели Django Admin .....	12
2.2.2. Полученные интерфейсы в панели Django REST.....	16
2.3. Клиентская часть.....	22
2.4. Использование Docker для развертывания web-приложения.....	27
ВЫВОДЫ .....	29
ЗАКЛЮЧЕНИЕ .....	29
СПИСОК ИСТОЧНИКОВ.....	30

## **ВВЕДЕНИЕ**

Курсовая работа посвящена созданию web-сервиса, предназначенного для администрации аэропорта некоторой компании-авиаперевозчика.

Цель выполнения курсовой работы в рамках изучения дисциплины «Основы веб-программирования»: овладеть практическими навыками и умениями реализации web-сервисов средствами Django REST framework, Vue.js, Muse-UI.

Для реализации сайта были использованы вышеуказанные технологии, в соответствии с индивидуальным заданием, а также были поставлены следующие задачи:

1. Проанализировать предметную область.
2. Создать модель данных.
3. Реализовать серверную часть средствами Django REST framework.
4. Реализовать клиентскую часть средствами Vue.js.

## ГЛАВА 1

### 1.1. Анализ предметной области

Предметной областью для курсовой работы является администрация аэропорта некоторой компании-авиаперевозчика.

#### **Формулировка задания:**

Создать программную систему, предназначенную для администрации аэропорта некоторой компании-авиаперевозчика.

Рейсы обслуживаются бортами, принадлежащими разным авиаперевозчикам. О каждом самолете необходима следующая минимальная информация: номер самолета, тип, число мест, скорость полета, компания-авиаперевозчик. Один тип самолета может летать на разных маршрутах и по одному маршруту могут летать разные типы самолетов.

О каждом рейсе необходима следующая информация: номер рейса, расстояние до пункта назначения, пункт вылета, пункт назначения; дата и время вылета, дата и время прилета, транзитные посадки (если есть), пункты посадки, дата и время транзитных посадок и дат и времени их вылета, количество проданных билетов. Каждый рейс обслуживается определенным экипажем, в состав которого входят командир корабля, второй пилот, штурман и стюардессы или стюарды. Каждый экипаж может обслуживать разные рейсы на разных самолетах. Необходимо предусмотреть наличие информации о допуске члена экипажа к рейсу.

Администрация компании-владельца аэропорта должна иметь возможность принять работника на работу или уволить. При этом необходима следующая информация: ФИО, возраст, образование, стаж работы, паспортные данные. Эта же информация необходима для сотрудников сторонних компаний

Перечень возможных запросов:

- Выбрать марку самолета, которая чаще всего летает по маршруту.
- Выбрать маршрут/маршруты, по которым летают рейсы, заполненные менее чем на 70%.
- Определить наличие свободных мест на заданный рейс.
- Определить количество самолетов, находящихся в ремонте.
- Определить количество работников компания-авиаперевозчика.

Необходимо предусмотреть возможность получения отчета о бортах компании-владельца по маркам с характеристикой марки. Указать общее количество бортов и количество бортов по каждой марке.

На основе данных сведений можно составить схему для будущей базы данных.

## 1.2. Модель базы данных «Компания-авиаперевозчик»

В соответствии с индивидуальным заданием была разработана модель БД, представленная на Рисунке 1.

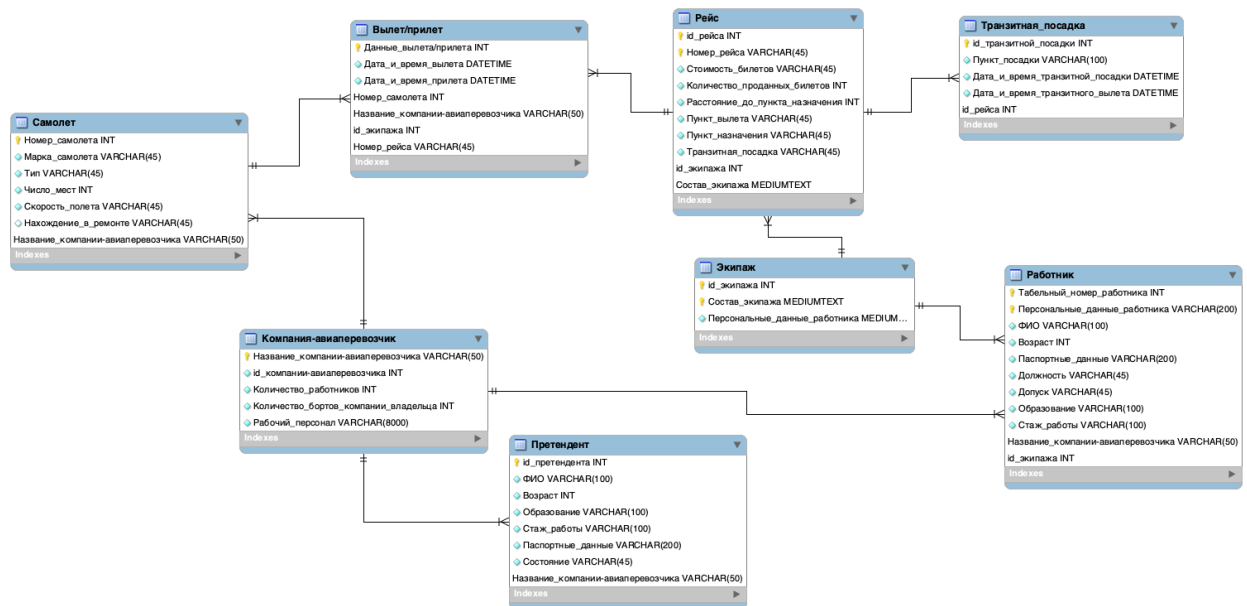


Рисунок 1 – модель базы данных «Компания-авиаперевозчик»

## 1.3. Состав реквизитов сущностей

- **Самолет** (номер самолета, название компании-авиаперевозчика, марка самолета, тип, число мест, скорость полета, нахождение в ремонте)
- **Компания-авиаперевозчик** (id\_компании-авиаперевозчика, название компании-авиаперевозчика, количество-работников, количество бортов компании-владельца, рабочий персонал)
- **Рейс** (номер рейса, id\_рейса, количество проданных билетов, стоимость билетов, расстояние до пункта назначения, пункт вылета, пункт назначения, транзитные посадки (есть/нет), id\_экипажа, состав экипажа)
- **Транзитные посадки** (id\_транзитной\_посадки, пункты посадки, дата и время транзитных посадок и дата и времени их вылета, id\_рейса)
- **Экипаж** (id\_экипажа, состав экипажа, персональные данные работника)
- **Вылет/прилет** (данные\_вылета/прилета, номер самолета, состав экипажа, id\_рейса, номер рейса, дата и время вылета, дата и время прилета, название компании-авиаперевозчика)
- **Работник** (табельный номер работника, персональные данные работника, название компании-авиаперевозчика, ФИО, возраст, образование, стаж работы, паспортные данные, должность, допуск, название компании-авиаперевозчика, id\_экипажа)

#### 1.4. Теоретические сведения об используемых технологиях

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.

Django REST framework — удобный инструмент для работы с rest основанный на идеологии фреймворка Django, который предоставляет готовую архитектуру для разработки как простых RESTful API, так и более сложных конструкций. Его ключевая особенность, это четкое разделение на сериализаторы, которые описывают соответствие между моделью и ее форматом представления (будь то JSON, XML или любой другой формат), и на отдельный набор универсальных представлений на основе классов (Class-Based-Views), которые могут быть по необходимости расширены. Также можно определить пользовательскую ссылочную структуру, вместо использования дефолтной. Это то, что отличает Django Rest Framework от других фреймворков, таких как Tastypie и Piston, которые автоматизируют формирование API на основе моделей, но это происходит за счет снижения гибкости и применимости к различным нестандартным требованиям.

Vue (произносится /vju:/, примерно как view) — 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.

## ГЛАВА 2

### 2.1. Проектирование приложения

Веб-приложение будет спроектировано на основе шаблона MVC. Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

### 2.2. Серверная часть приложения

Серверная часть приложения реализуется с помощью фреймворка Django REST. Создан проект django в отдельной директории со своим виртуальным окружением, в которое были установлены django, django rest framework, psycopg2, django cors headers.

Был создан файл, содержащий описание таблиц базы данных, представленное в виде класса Python – модель. Файл models.py содержит модели в соответствии с составленной схемой БД (Рисунок 1).

```
from django.db import models
from django.contrib.auth.models import User

class Company(models.Model):
    name = models.CharField(max_length=250)
    workers_amount = models.IntegerField()
    sides_amount = models.IntegerField() # количество бортов

    def __str__(self):
        return self.name
```

Рисунок 2 – модель «Компания-авиаперевозчик»

```
class Plane(models.Model):
    PLANES_TYPE = [
        ('1', 'Passenger'),
        ('2', 'Cargo')
    ]

    company = models.ForeignKey(Company, on_delete=models.CASCADE)
    plane_model = models.CharField(max_length=250)
    seats_num = models.IntegerField()
    plane_type = models.CharField(choices=PLANES_TYPE, max_length=1, default='1')
    plane_speed = models.IntegerField()
    is_repair = models.BooleanField(default=False) # в ремонте или нет?
```

Рисунок 3 – модель «Самолет»

```

class Flight(models.Model):
    saled_tickets_amount = models.IntegerField()
    distance = models.IntegerField()
    arrival_point = models.CharField(max_length=250)
    departure_point = models.CharField(max_length=250)
    is_transit = models.BooleanField(default=False) # наличие транзитных точек
    price = models.IntegerField()

    def __str__(self):
        return f'{self.arrival_point} – {self.departure_point}'

```

Рисунок 4 – модель «Рейс»

```

class TransitLanding(models.Model):
    flight = models.ForeignKey(Flight, on_delete=models.CASCADE)
    arrival_date = models.DateTimeField()
    departure_date = models.DateTimeField()
    landing_point = models.CharField(max_length=250)

    def __str__(self):
        return f'{self.landing_point}, Arrival: {self.arrival_date}, Departure: {self.departure_date}'

```

Рисунок 5 – модель «Транзитная посадка»

```

class ArrivalDeparture(models.Model):
    company = models.ForeignKey(Company, on_delete=models.CASCADE)
    flight = models.ForeignKey(Flight, on_delete=models.CASCADE)
    plane = models.ForeignKey(Plane, on_delete=models.CASCADE)
    arrival_date = models.DateTimeField()
    departure_date = models.DateTimeField()

```

Рисунок 6 – модель дополнительной информации о рейсе



```

class Worker(models.Model):
    POSITIONS = [
        ('1', 'Captain'),
        ('2', 'Second pilot'),
        ('3', 'Navigator'), # штурман
        ('4', 'Steward'),
        ('5', 'Stewardess')
    ]

    company = models.ForeignKey(Company, on_delete=models.CASCADE)

    first_name = models.CharField(max_length=250)
    last_name = models.CharField(max_length=250)
    patronymic = models.CharField(max_length=250)
    age = models.IntegerField()
    education = models.CharField(max_length=250)
    work_experience = models.IntegerField()
    position = models.CharField(choices=POSITIONS, max_length=1, default='1')
    is_allow = models.BooleanField(default=True) # допущен ли к полёту

    def __str__(self):
        return f'{self.last_name} {self.first_name} {self.patronymic}'

```

Рисунок 7 – модель «Работник компании»

```

class Challenger(models.Model):
    POSITIONS = [
        ('1', 'Captain'),
        ('2', 'Second pilot'),
        ('3', 'Navigator'), # штурман
        ('4', 'Steward'),
        ('5', 'Stewardess')
    ]

    company = models.ForeignKey(Company, on_delete=models.CASCADE)

    first_name = models.CharField(max_length=250)
    last_name = models.CharField(max_length=250)
    patronymic = models.CharField(max_length=250)
    age = models.IntegerField()
    education = models.CharField(max_length=250)
    work_experience = models.IntegerField()
    position = models.CharField(choices=POSITIONS, max_length=1, default='1')
    passport = models.IntegerField()
    is_hired = models.BooleanField(default=False)

    def __str__(self):
        return f'{self.last_name} {self.first_name} {self.patronymic}'

```

Рисунок 8 – модель «Претендент на должность в компании»

```
class Crew(models.Model):
    flight = models.ForeignKey(Flight, on_delete=models.CASCADE)
    company = models.ForeignKey(Company, on_delete=models.CASCADE)
```

Рисунок 9 – модель «Экипаж»

```
class Ticket(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    flight = models.ForeignKey(Flight, on_delete=models.CASCADE)
```

Рисунок 10 – модель «Билеты»

Создание админ панели для разработанной модели данных.

```
from django.contrib import admin
from .models import *

# Register your models here.
admin.site.register(Company)
admin.site.register(Plane)
admin.site.register(Flight)
admin.site.register(TransitLanding)
admin.site.register(ArrivalDeparture)
admin.site.register(Worker)
admin.site.register(Challenger)
admin.site.register(Crew)
```

Рисунок 11 – содержимое файла «admin.py»

```

from rest_framework import serializers
from .models import *
from django.contrib.auth.models import User

class UserSerializer(serializers.ModelSerializer):
    class Meta():
        model = User
        fields = ['username', 'email', 'id']

class CompanySerializer(serializers.ModelSerializer):
    class Meta():
        model = Company
        fields = '__all__'

class PlaneSerializer(serializers.ModelSerializer):
    class Meta():
        model = Plane
        fields = '__all__'

class FlightSerializer(serializers.ModelSerializer):
    class Meta():
        model = Flight
        fields = '__all__'

class TransitLandingSerializer(serializers.ModelSerializer):
    class Meta():
        model = TransitLanding
        fields = '__all__'

```

Рисунок 12 – фрагмент кода файла «serializers.py»

Были созданы контроллеры для обработки данных. Представления размещены в файле views.py.

```

from rest_framework import generics
from django.contrib.auth.models import User

from .models import *
from .serializers import *

# Create your views here.
class FlightsView(generics.ListAPIView):
    serializer_class = FlightSerializer

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

        query_params = self.request.query_params

        arrival = query_params.get('arrival', None)
        departure = query_params.get('departure', None)
        is_transit = query_params.get('is_transit', None)
        tickets_saled = query_params.get('tickets', None)
        price = query_params.get('price', None)

        print('IS TRANSIT', is_transit)

```

Рисунок 13 – фрагмент кода файла «views.py»

Файл `urls.py` содержит пути для доступа к страницам.

```
from django.urls import path, include
from rest_framework.authtoken.views import obtain_auth_token

from .views import *

app_name = "airport_app"

urlpatterns = [
    path('auth/', include('djoser.urls')),
    path('auth/token/', obtain_auth_token, name='token'),
    path('get_user_info/', GetUserInfo.as_view()),

    path('flights/', FlightsView.as_view()),
    path('flights/<int:pk>', FlightView.as_view()),
    path('flights/tickets/<int:pk>', UpdateFlightView.as_view()),

    path('companies/', CompanysView.as_view()),
    path('companies/<int:pk>', CompanyView.as_view()),

    path('planes/', PlanesView.as_view()),
    path('planes/<int:pk>', PlaneView.as_view()),

    path('flights/transit/', TransitLandingsView.as_view()),
    path('flights/transit/<int:pk>', TransitLandingView.as_view()),

    path('flights/arrival/', ArrivalDeparturesView.as_view()),
    path('flights/arrival/<int:pk>', ArrivalDepartureView.as_view()),

    path('companies/workers/', WorkersView.as_view()),
    path('companies/workers/<int:pk>', WorkerView.as_view()),

    path('flights/crew/', CrewsView.as_view()),
    path('flights/crew/<int:pk>', CrewView.as_view()),

    path('challengers/', ChallengerView.as_view()),
```

Рисунок 14 – пути к страницам в файле «`urls.py`»

### 2.2.1. Полученные интерфейсы в панели Django Admin

1. Просмотр, редактирование и удаление всех резюме.
2. Просмотр, редактирование, добавление и удаление всех сотрудников.
3. Добавление рейса, маршрута, самолёта, компании и редактирование, удаление.
4. Возможность изменения состояния самолёта (в ремонте или нет).

Были добавлены модели (Рисунок 15). Ниже приведены некоторые полученные интерфейсы в панели `django-admin`.

## Django administration

### Site administration

AIRPORT_APP		
Arrival departures	<a href="#">+ Add</a>	<a href="#">Change</a>
Challengers	<a href="#">+ Add</a>	<a href="#">Change</a>
Companyys	<a href="#">+ Add</a>	<a href="#">Change</a>
Crews	<a href="#">+ Add</a>	<a href="#">Change</a>
Flights	<a href="#">+ Add</a>	<a href="#">Change</a>
Planes	<a href="#">+ Add</a>	<a href="#">Change</a>
Transit landings	<a href="#">+ Add</a>	<a href="#">Change</a>
Workers	<a href="#">+ Add</a>	<a href="#">Change</a>
AUTH TOKEN		
Tokens	<a href="#">+ Add</a>	<a href="#">Change</a>
AUTHENTICATION AND AUTHORIZATION		
Groups	<a href="#">+ Add</a>	<a href="#">Change</a>
Users	<a href="#">+ Add</a>	<a href="#">Change</a>

Рисунок 15 – все имеющиеся модели

### Add company

<b>Name:</b>	<input type="text" value="POBEDA"/>
<b>Workers amount:</b>	<input type="text" value="7000"/>
<b>Sides amount:</b>	<input type="text" value="120"/>

Рисунок 16 – добавление компании с соответствующей информацией

## Add flight

Saled tickets amount:	34
Distance:	1 362
Arrival point:	Moscow
Departure point:	Sochi
<input type="checkbox"/> Is transit	
Price:	6348

Рисунок 17 – добавление рейса

## Add plane

Company:	POBEDA
Plane model:	A30
Seats num:	320
Plane type:	Passenger
Plane speed:	917
<input type="checkbox"/> Is repair	
<div>SAVE</div>	

Рисунок 18 – добавление самолета и возможность изменения состояния самолёта (в ремонте или нет)

### Add arrival departure

Company:	POBEDA	✎	+
Flight:	Moscow - Sochi	✎	+
Plane:	A30, Passenger	✎	+
Arrival date:	Date: 2020-07-01 Today   📅		
	Time: 08:05:35 Now   ⌚		
	Note: You are 3 hours ahead of server time.		
Departure date:	Date: 2020-07-01 Today   📅		
	Time: 10:25:37 Now   ⌚		
	Note: You are 3 hours ahead of server time.		







Рисунок 19 – добавление дополнительной информации о рейсе

### Change worker

Company:	WayW	✎	+
First name:	Иван		
Last name:	Иванов		
Patronymic:	Иванович		
Age:	45		
Education:	ЧТО-ТО		
Work experience:	10		
Position:	Captain		
<input checked="" type="checkbox"/> Is allow			
<div>Delete</div>			

Рисунок 20 – просмотр, редактирование, добавление и удаление всех сотрудников.

## Change challenger

Company:	WayW  
First name:	Юрий
Last name:	Иванов
Patronymic:	Михайлович
Age:	35 
Education:	...
Work experience:	5 
Position:	Navigator 
Passport:	4638900 
<input checked="" type="checkbox"/> Is hired	

[Delete](#)

Рисунок 21 – просмотр резюме претендентов

### 2.2.2. Полученные интерфейсы в панели Django REST

```
1. admin/  
2. api/ auth/  
3. api/ auth/token/ [name='token']  
4. api/ get_user_info/  
5. api/ flights/  
6. api/ flights/<int:pk>  
7. api/ flights/tickets/<int:pk>  
8. api/ companies/  
9. api/ companies/<int:pk>  
10. api/ planes/  
11. api/ planes/<int:pk>  
12. api/ flights/transit/  
13. api/ flights/transit/<int:pk>  
14. api/ flights/arrival/  
15. api/ flights/arrival/<int:pk>  
16. api/ companies/workers/  
17. api/ companies/workers/<int:pk>  
18. api/ flights/crew/  
19. api/ flights/crew/<int:pk>  
20. api/ challengers/  
21. api/ challengers/list  
22. api/ tickets/  
23. api/ tickets/list/
```

Рисунок 22 – все полученные интерфейсы в панели Django REST



Ниже приведены некоторые из имеющихся интерфейсов.

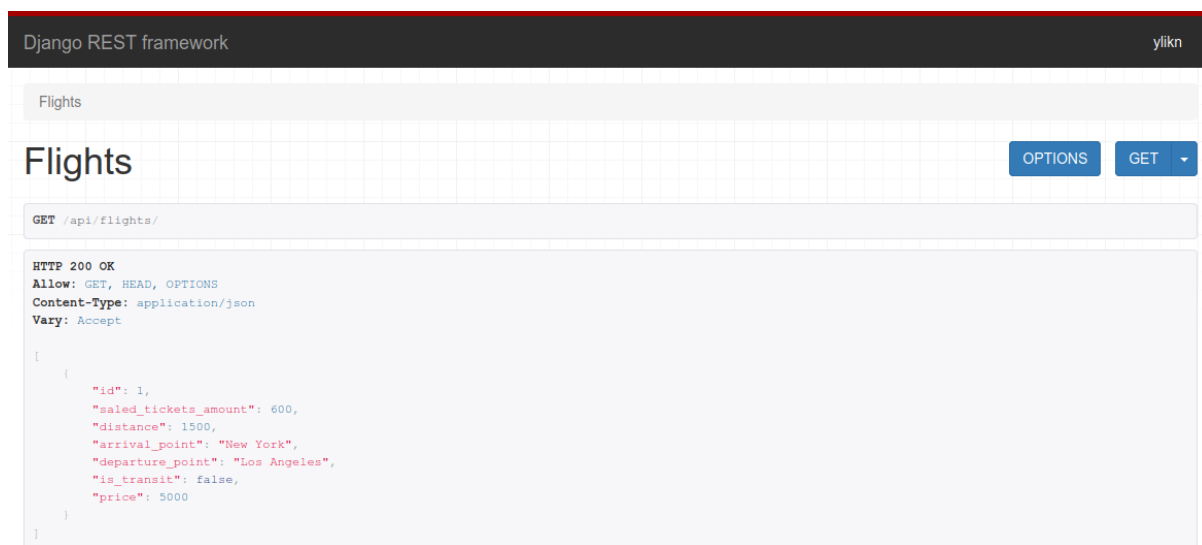


Рисунок 23 – интерфейс «Табло с рейсами»

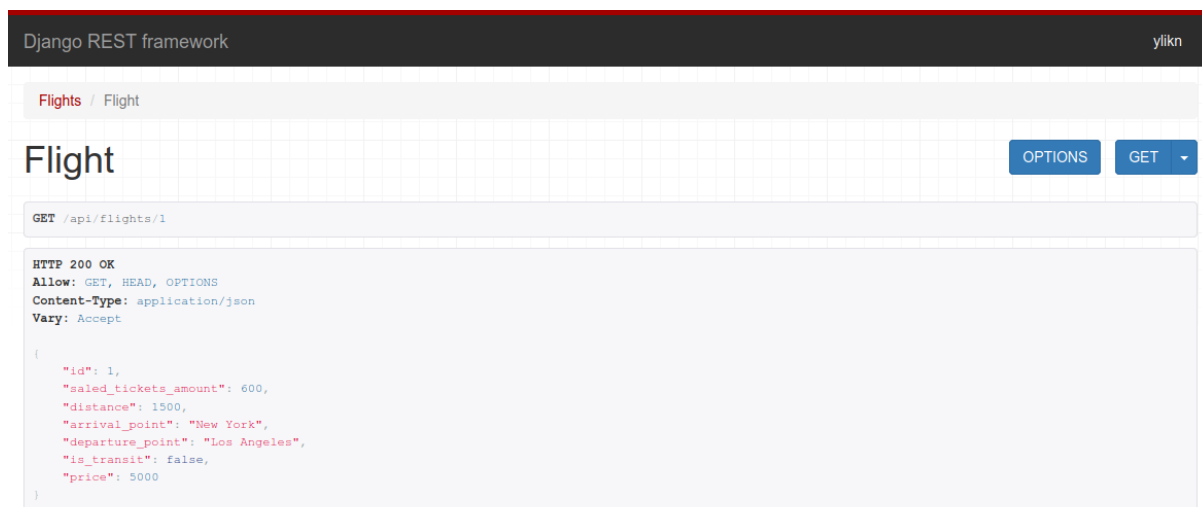


Рисунок 24 – «Рейс»

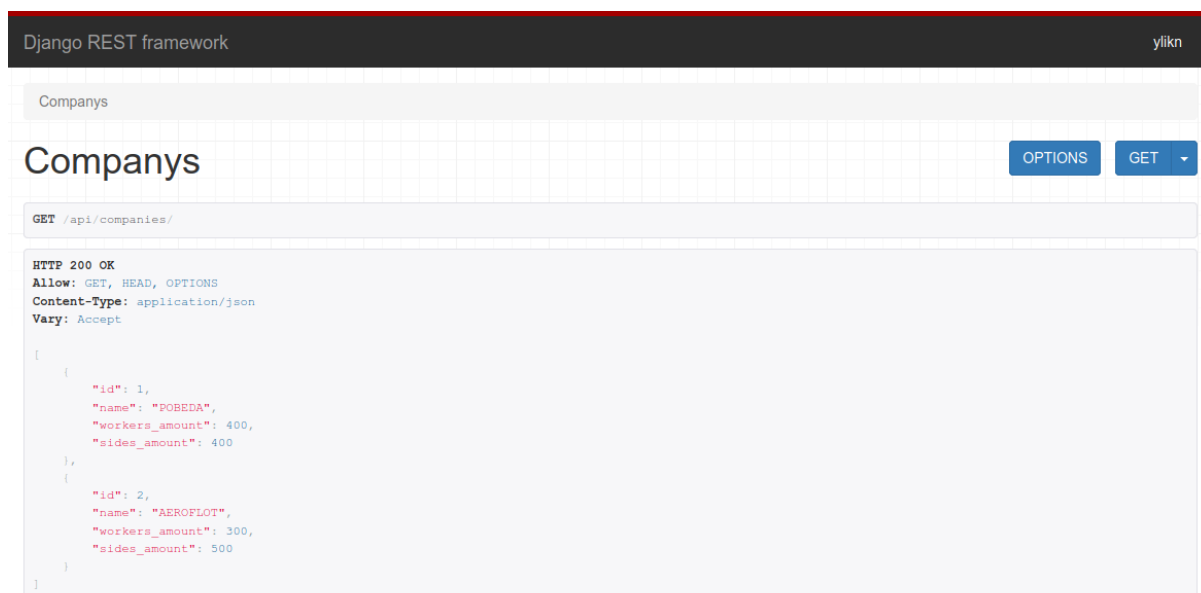


Рисунок 25 – «Компании-авиаперевозчики»

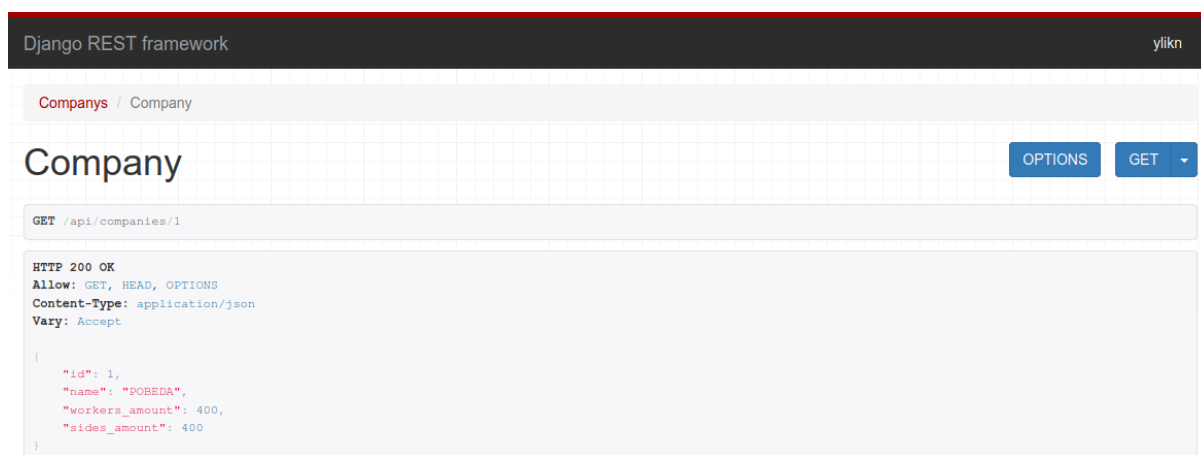


Рисунок 26 – «Компания-авиаперевозчик»

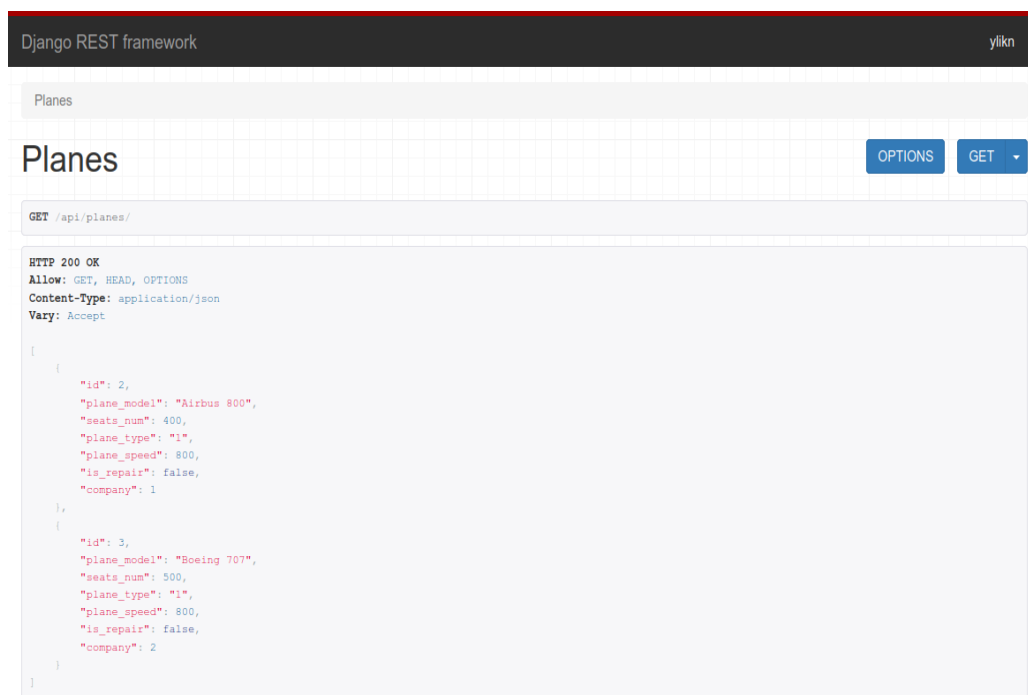


Рисунок 27 – «Самолеты»

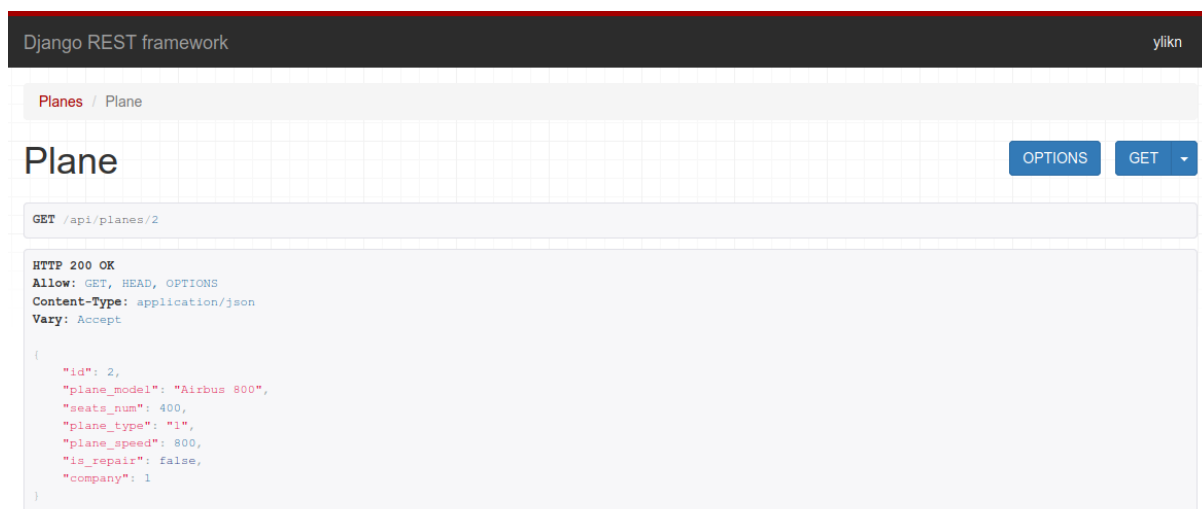


Рисунок 28 – «Самолет»

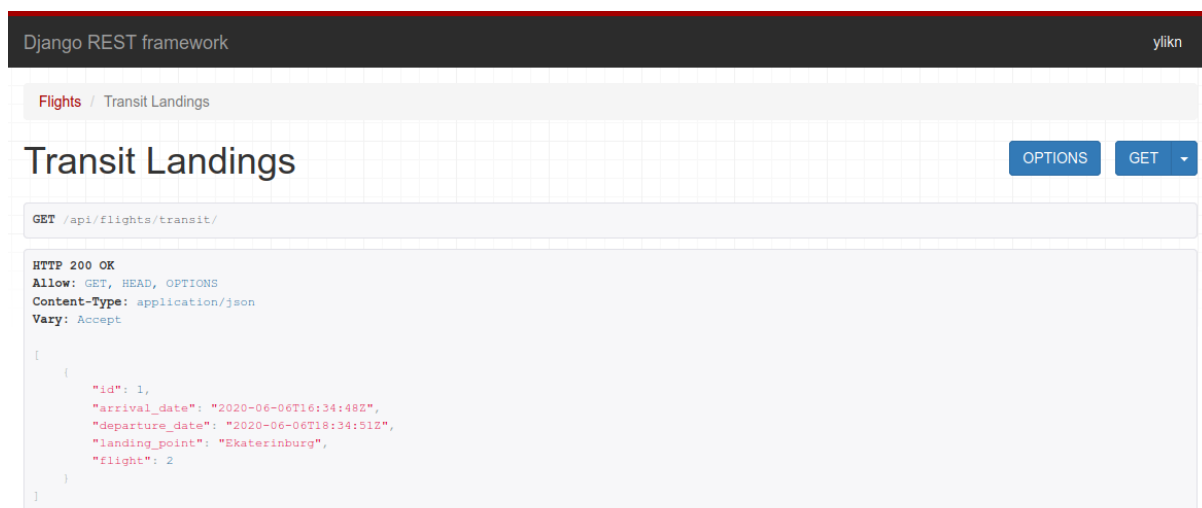


Рисунок 29 – «Транзитные посадки»

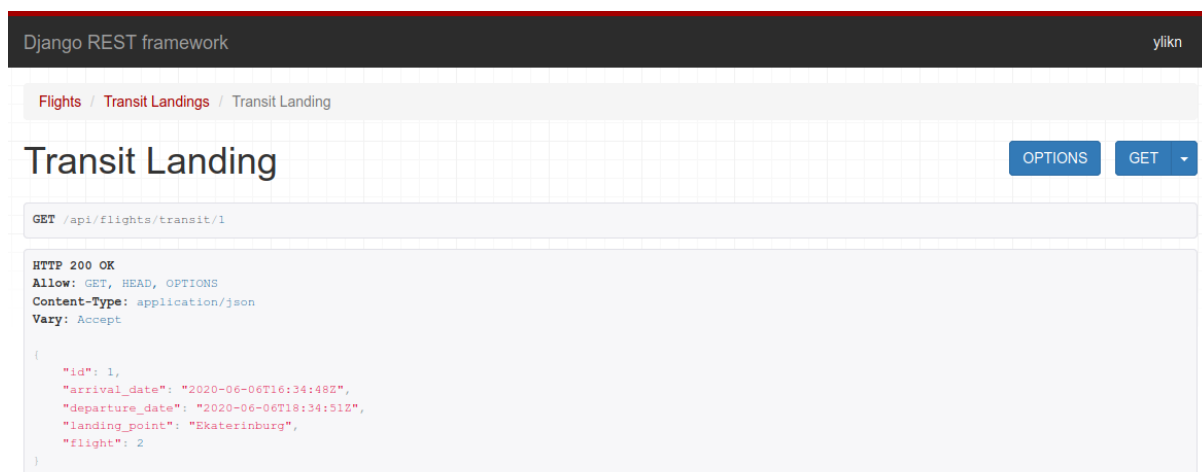


Рисунок 30 – «Транзитная посадка»

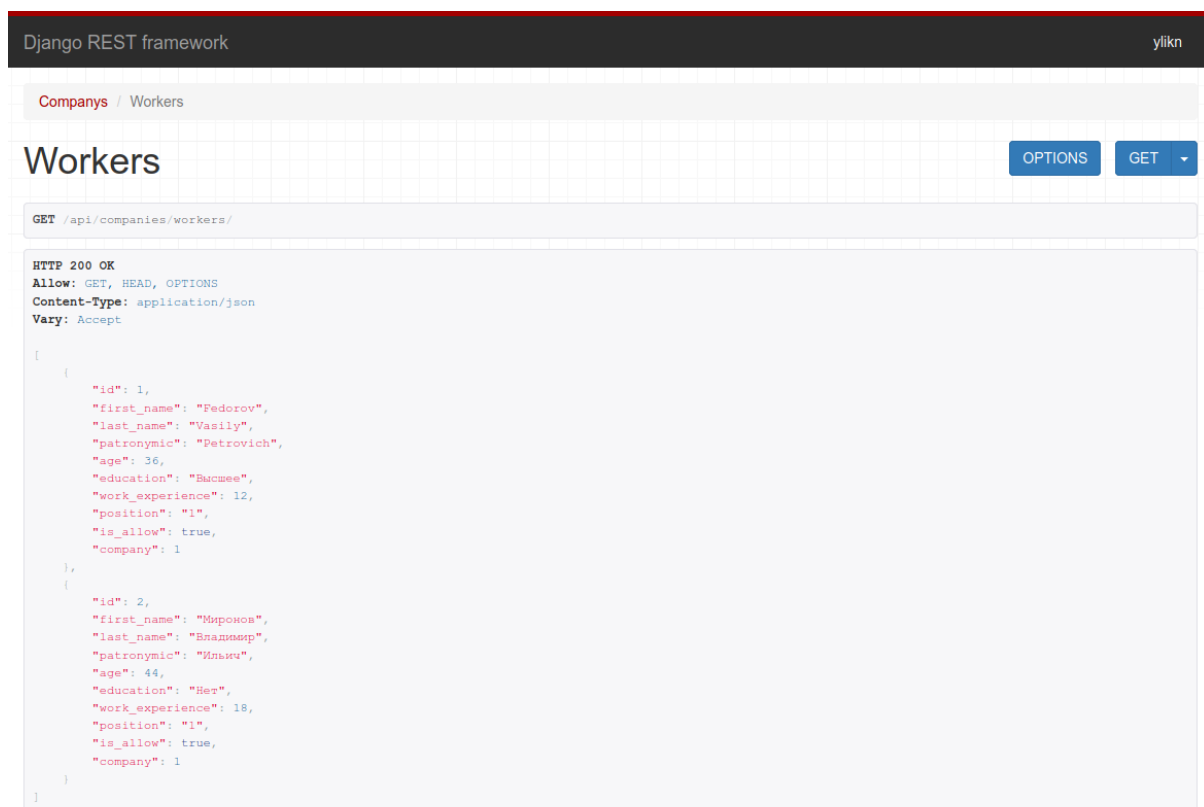


Рисунок 31 – «Работники»

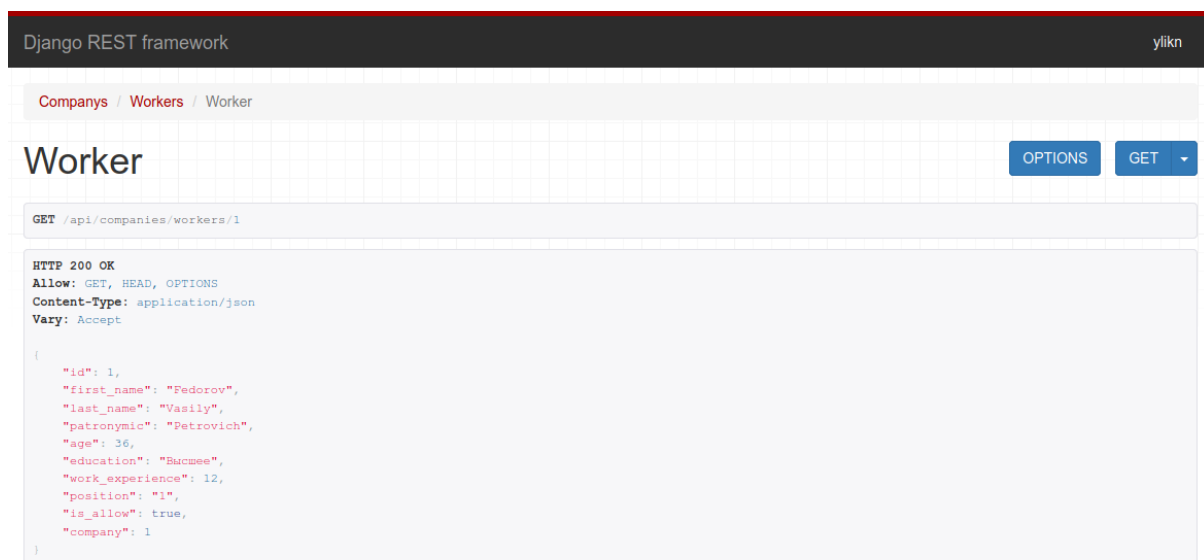


Рисунок 32 – «Работник»

The image shows a web browser window. The top part displays an HTTP 405 error message: "HTTP 405 Method Not Allowed", "Allow: POST, OPTIONS", "Content-Type: application/json", and "Vary: Accept". Below this is a JSON response: {"detail": "Method \"/>

Raw data HTML form

First name

Last name

Patronymic

Age

Education

Work experience

Position

Passport

Is hired ☐

Company

POST

Рисунок 33 – интерфейс «Резюме претендента»

### 2.3. Клиентская часть

С преподавателем были согласованы следующие интерфейсы:

1. Главная страница – табло со всеми рейсами и возможностью фильтрации по столбцам:
  - Пункт отправления;
  - Пункт прибытия;
  - Количество билетов;
  - Цена билета;
  - Есть ли транзитные пересадки.
2. Интерфейс для просмотра подробной информации о рейсе.
3. Интерфейс для входа пользователя в личный кабинет.
4. Интерфейс для регистрации нового пользователя.
5. Интерфейс личного кабинета пользователя.
6. Интерфейс для подачи резюме от претендента.

7. Интерфейс для просмотра оставленных резюме.
8. Возможность покупки билета и интерфейс для просмотра купленных билетов.

Интерфейсы реализуются с помощью технологий Vue.js и Bootstrap 4. В качестве UI-библиотеки была использована Vuetify.

Файл Bootstrap подключаются в файле index.html проекта, интерфейсы описываются как template в файлах-компонентах формата .vue.

```
<template>
  <main>
    <div class="d-flex flex-column">
      <div class="filter d-flex flex-row">
        <div class="m-auto">
          <b-form inline>
            <div class="m-1">
              <label for="arrival">Отправление</label>
              <b-form-input id="arrival" v-model="arrival"></b-form-input>
            </div>
            <div class="m-1">
              <label for="departure">Прибытие</label>
              <b-form-input id="departure" v-model="departure"></b-form-input>
            </div>
            <div class="m-1">
              <label for="tickets">Количество билетов</label>
              <b-form-input id="tickets" type="number" v-model="tickets"></b-form-input>
            </div>
            <div class="m-1">
              <label for="price">Цена</label>
              <b-form-input type="number" id="price" v-model="price"></b-form-input>
            </div>
            <div class="m-1">
              <b-form-checkbox type="checkbox" id="transit" v-model="transit">Транзитные посадки</b-form-checkbox>
            </div>
            <div class="m-1 mt-4">
              <b-button variant="info" @click="filterFlights()">Найти</b-button>
              <b-button variant="warning" @click="clear()">Сброс</b-button>
            </div>
          </b-form>
        </div>
      </div>
      <div class="content container mt-5">
        <h2 class="mb-5" v-if="flights.length > 0">Найдено {{ flights.length }} полётов</h2>
        <h2 class="mb-5" v-if="flights.length === 0">Ничего не найдено :( </h2>
      </div>
    </div>
  </main>
</template>
```

Рисунок 34 – фрагмент кода файла «Airport.vue»

```

<template>
  <main>
    <div class="container">
      <h1>Мои билеты</h1>
      <p>Здесь Вы можете увидеть все свои билеты.</p>
    </div>
    <div class="container">
      <h2 v-if="tickets.length">Найдено {{ tickets.length }} билетов</h2>
      <h2 v-if="!tickets.length">К сожалению, билетов пока нет, но Вы можете их купить!</h2>
      <p>Общие затраты: {{ ticketsSum }} руб.</p>
      <b-card class="mt-2 mb-2" v-for="ticket in tickets" :key="ticket.id">
        <b-card-title>
          {{ ticket.arrival }} – {{ ticket.departure }}
        </b-card-title>
        <b-card-sub-title>Цена: {{ ticket.price }} рублей</b-card-sub-title>
        <b-card-text>
          Расстояние: {{ ticket.distance }} км
        </b-card-text>
        <b-card-text>
          Есть ли транзитные посадки: <span v-if="ticket.transit">Да</span><span v-if="!ticket.transit">Нет</span>
        </b-card-text>
      </b-card>
    </div>
  </main>
</template>
<script>
export default {
  name: 'MyTickets',
  data () {
    return {
      tickets: [],
      ticketsSum: 0
    }
  }
}

```

Рисунок 35 – фрагмент кода файла «MyTickets.vue»

AviaCom
Отправить резюме
Вход
Регистрация

Отправление

Прибытие


Количество билетов


Цена

☐ Транзитные посадки

Найти

Сбросить

Найденные рейсы: 3




**New York - Los Angeles**  
 Цена: 5000 рублей  
 Расстояние: 1500 км  
 Количество оставшихся билетов: 249  
 Есть ли транзитные посадки: Нет  

Купить билет
Подробнее

**Saint Petersburg - Barcelona**  
 Цена: 7400 рублей  
 Расстояние: 2823 км  
 Количество оставшихся билетов: 179  
 Есть ли транзитные посадки: Нет  

Купить билет
Подробнее

Рисунок 36 – Главная страница сервиса «AviaCom» с табло рейсов и поиском



**POBEDA, Moscow - Sochi**

Цена: 6348 рублей  
Расстояние: 1384 км

Количество оставшихся билетов: 34

Самолёт: A30

Отправка: 2020-07-01T08:05:35Z

Прибытие: 2020-07-01T10:25:37Z

Есть ли транзитные посадки: Нет

[Купить билет](#)

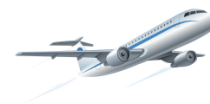


Рисунок 37 – Подробная информация о выбранном рейсе

**Стань частью нашей команды!**



Имя:

Фамилия:

Отчество:

Стаж работы:

Возраст:

Образование:

Паспорт:

Компания:

Выберите компанию



Позиция:

Выберите должность



[Отправить](#)

[Сбросить](#)



Рисунок 38 – Добавление резюме

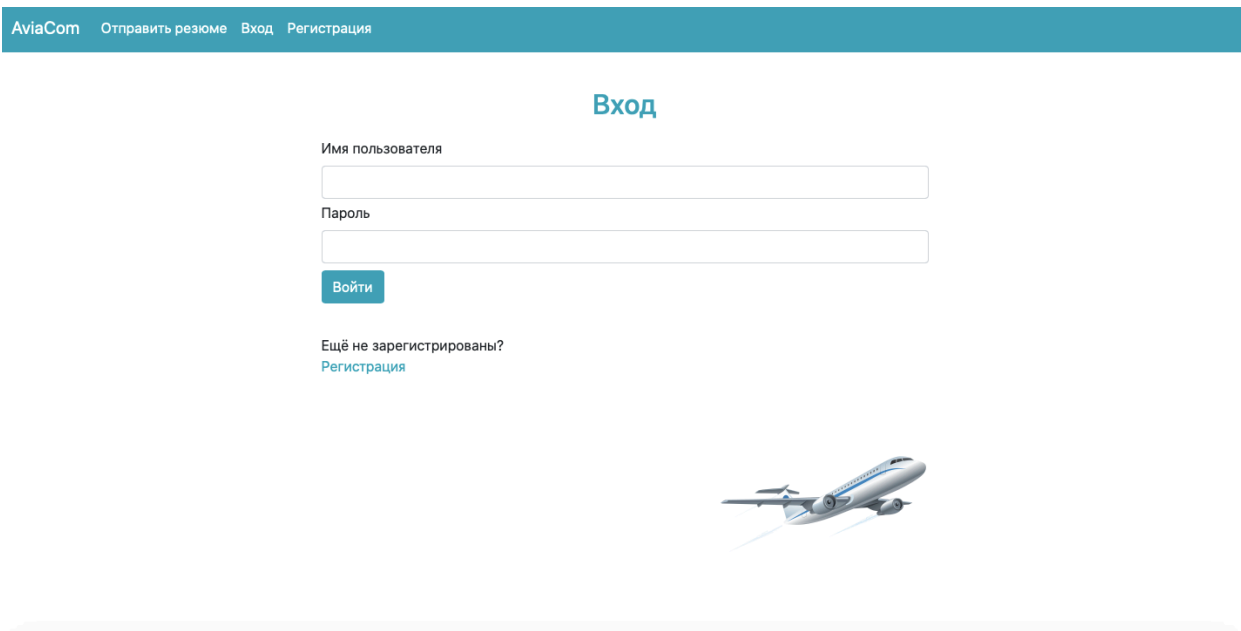


Рисунок 39 – Вход в личный кабинет сервиса «AviaCom»

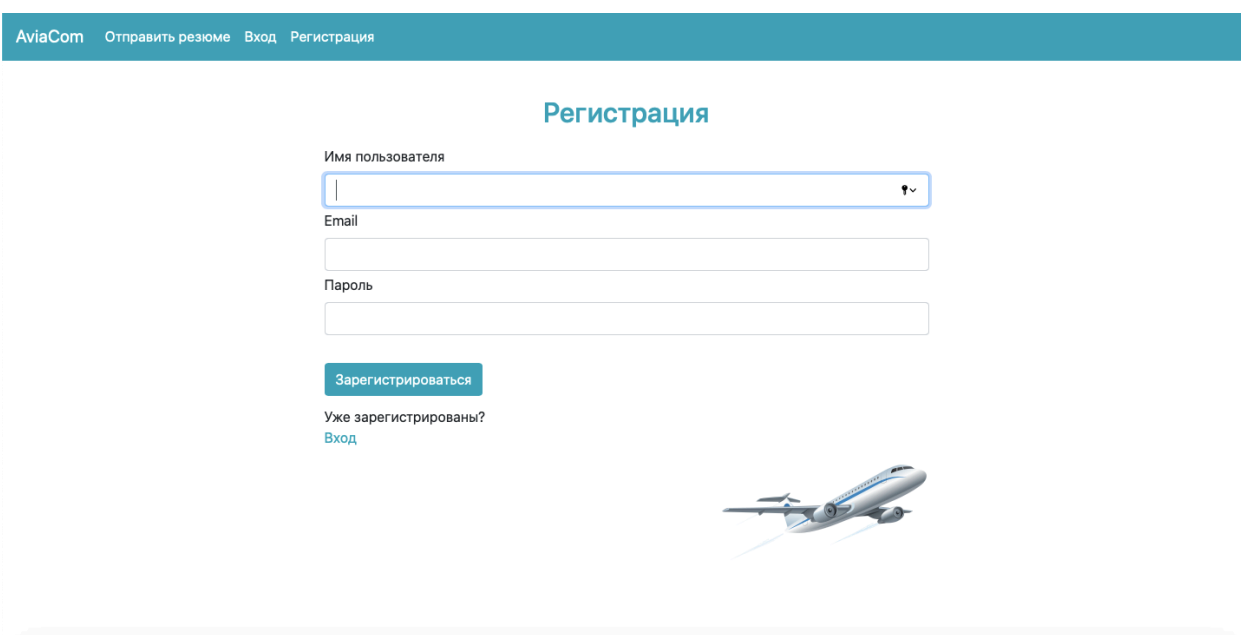


Рисунок 40 – Регистрация нового пользователя сервиса «AviaCom»

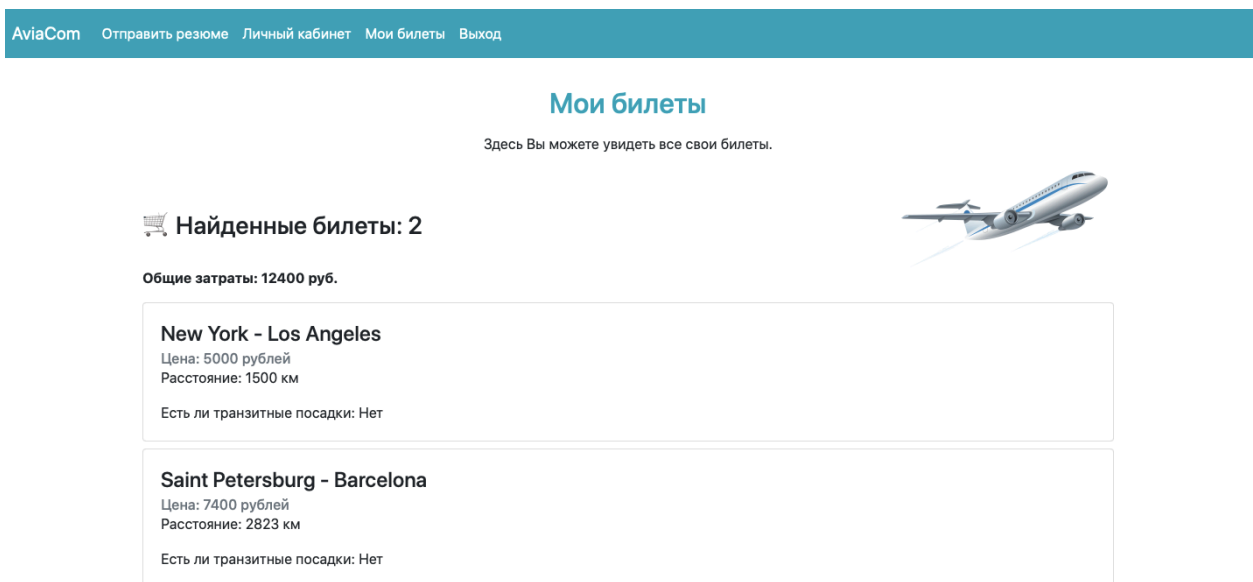


Рисунок 41 – Просмотр купленных билетов

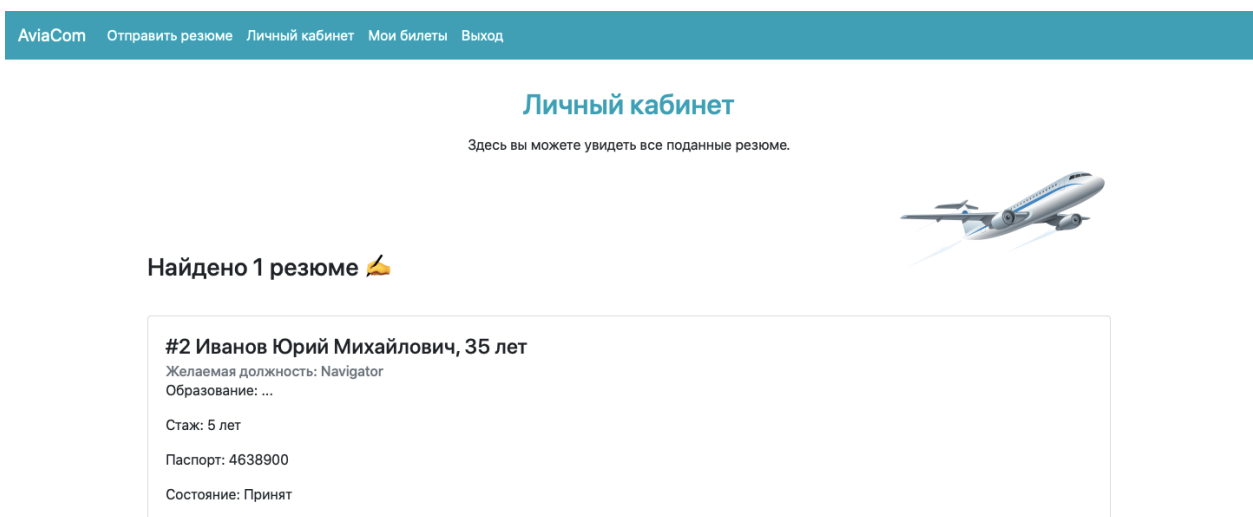


Рисунок 42 – Просмотр оставленных резюме

## 2.4. Использование Docker для развертывания web-приложения

Docker — это открытая платформа для разработки, доставки и эксплуатации приложений. Docker разработан для более быстрого выкладывания приложений. С помощью docker можно отделить приложение от инфраструктуры и обращаться с инфраструктурой как управляемым приложением. Docker помогает выкладывать код быстрее, быстрее тестировать, быстрее выкладывать приложения и уменьшить время между написанием кода и запуском кода. Docker делает это с помощью легковесной платформы контейнерной виртуализации, используя процессы и утилиты, которые помогают управлять и выкладывать приложения.

В своем ядре docker позволяет запускать практически любое приложение, безопасно изолированное в контейнере. Безопасная изоляция позволяет запускать на одном хосте много контейнеров одновременно. Легковесная природа контейнера, который запускается без дополнительной нагрузки гипервизора, позволяет добиваться больше от железа.

Dockerfile – файл, в котором прописываются пути и названия виртуальных папок контейнера, а также происходит установка важных частей проекта – различных библиотек, без которых работа с веб сервисом невозможна. Эти требования описаны в файле requirements.txt.

```
FROM python:3.6.9

ENV PYTHONUNBUFFERED 1

RUN mkdir /airport_project

WORKDIR /airport_project

COPY . /airport_project

RUN pip3 install -r requirements.txt
```

Рисунок 43 – содержимое файла «Dockerfile»

Docker Compose используется для одновременного управления несколькими контейнерами, входящими в состав приложения.

```
services:
  db:
    image: postgres
    ports:
      - "5436:5432"
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=airport_db
    volumes:
      - ./dbs/postgres-data:/var/lib/postgresql
  backend:
    container_name: airport_backend_container
    build: ./airport/airport_project
    command: bash -c "
      sleep 3 &&
      python3 manage.py makemigrations && python3 manage.py migrate &&
      python3 manage.py runserver --insecure 0.0.0.0:8000";
    volumes:
      - ./airport/airport_project:/airport_project
    ports:
      - "8000:8000"
    depends_on:
      - db
  frontend:
    container_name: airport_frontend_container
    build:
      context: ./airport-frontend
      dockerfile: Dockerfile
    command: npm start --start;
    volumes:
      - ./airport-frontend:/airport-frontend
      - /airport-frontend/node_modules
    ports:
      - "8080:8080"
```

Рисунок 44 – фрагмент кода файла «docker-compose.yml»

## **ВЫВОДЫ**

В ходе выполнения курсовой работы были получены практические навыки и умения реализации web-сервисов средствами Django 2.2., реализации REST API, используя DRF и SPA-приложение на Vue.js.

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы реализовано web-приложение «AviaCom» в соответствии с индивидуальным заданием, с использованием технологий Vue.js и Django REST Framework.

В приложении реализован набор интерфейсов:

1. Главная страница – табло со всеми рейсами и возможностью фильтрации по столбцам:
  - Пункт отправления;
  - Пункт прибытия;
  - Количество билетов;
  - Цена билета;
  - Есть ли транзитные пересадки.
2. Интерфейс для просмотра подробной информации о рейсе.
3. Интерфейс для входа пользователя в личный кабинет.
4. Интерфейс для регистрации нового пользователя.
5. Интерфейс личного кабинета пользователя.
6. Интерфейс для подачи резюме от претендента.
7. Интерфейс для просмотра оставленных резюме.
8. Возможность покупки билета и интерфейс для просмотра купленных билетов.

Получен опыт разработки frontend и backend частей приложения, практические навыки web-разработки.

## СПИСОК ИСТОЧНИКОВ

1. Официальный сайт Django REST Framework  
URL: <https://www.django-rest-framework.org> (Дата обращения 23.06.2020)
2. Документация Django на русском языке [Электронный ресурс]  
URL: <https://djbook.ru/rel1.9/> (Дата обращения 24.06.2020)
3. Vue.js. Введение [Электронный ресурс]. Режим доступа: <https://ru.vuejs.org/v2/guide/index.html> (Дата обращения 24.06.2020)
4. Хабр. Архитектура мобильного клиент-серверного приложения [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/246877/>. (Дата обращения 25.06.2020)