

**Министерство образования и науки Российской Федерации**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  
**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**  
**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**  
**МЕХАНИКИ И ОПТИКИ”**

Факультет ИКТ

Образовательная программа Интеллектуальные системы в гуманитарной сфере

Направление подготовки (специальность) Интеллектуальные системы в гуманитарной  
сфере (45.03.04)

## **О Т Ч Ё Т**

по курсовой работе

Тема задания: РЕАЛИЗАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ СРЕДСТВАМИ Django REST  
framework.

Обучающийся: Бережнова Марина Юрьевна К3343

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

Оценка за курсовую работу \_\_\_\_

Подписи членов комиссии:

\_\_\_\_\_  
(подпись) **Ф.И.О.**

\_\_\_\_\_  
(подпись) **Ф.И.О.**

\_\_\_\_\_  
(подпись) **Ф.И.О.**

Дата \_\_\_\_

Санкт-Петербург  
2020

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	3
<b>1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ</b>	4
1.1 Описание предметной области	4
1.2 Описание функциональных требований	4
<b>2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА</b>	6
2.1 Описание архитектуры сервиса	6
2.2 Модель данных	6
<b>3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ</b>	8
<b>4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ</b>	10
4.1 Описание средств разработки клиентской части	10
4.2 Разработанные интерфейсы	10
<b>5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ</b>	21
<b>6. ЗАКЛЮЧЕНИЕ</b>	22
<b>7. СПИСОК ЛИТЕРАТУРЫ</b>	23

## **ВВЕДЕНИЕ**

Сегодня путешествия на самолете играют очень важную роль в жизни любого человека, поэтому большим спросом пользуются различные приложения и сайты, в которых возможен поиск авиабилетов на разные даты, в разные города с возможностью выбора даже модели самолета, на котором лететь. Соответственно, они нуждаются в удобных для использования программных системах. Данная курсовая работа описывает программную систему, предназначенную для администрации аэропорта некоторой компании-авиаперевозчика.

Целью данной курсовой работы является разработка веб-сервиса согласно выбранному варианту. В рамках работы нужно было изучить и научиться применять средства создания вебсервисов, 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.

# **1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1. Описание предметной области**

Предметной областью курсовой работы является программная система аэропорта. Информационная система для данной области представляет собой сервис, в котором реализованы стандартные процедуры для данной области: покупка билетов на различные рейсы по необходимым требованиям, подача заявлений на работу в авиакомпании, просмотр купленных билетов в личном кабинете. Такая система должна обеспечивать хранение сведений о зарегистрированных пользователях, о работающих сотрудниках компаний и о временно не работающих, о допуске экипажа к полету, о всех рейсах и маршрутах, о ценах на билеты, о количестве оставшихся билетов и о транзитных точках. Все рейсы можно фильтровать по необходимым пользователю данным: модели самолета, городе вылета или прибытия, цене билета или же доступности его к покупке.

## **1.2. Описание функциональных требований**

На основе анализа предметной области выявлены функциональные требования в разрабатываемой системе. Что должно быть в нашей системе? В первую очередь, должен быть создан и доступен личный кабинет пользователя, в котором будет вся информация о приобретенных билетах. Обязательно отображение рейсов, которые 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 время их вылета, количество проданных билетов. Каждый рейс обслуживается определенным экипажем. 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.

Также необходимо реализовать возможность подачи заявления на работу в компанию, указав свои персональные данные и возможность сортировки необходимого рейса.

## **2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА**

### **2.1 Описание архитектуры сервиса**

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

### **2.1. Модель данных**

В соответствии с вариантом задания и функциональными требованиями была создана модель данных, представленная на рис. 1

В качестве сущностей для разрабатываемой программной системы были выбраны Сотрудник, Претендент, Экипаж, Допуск экипажа, Компания, Билеты, Рейс, Самолет, Маршрут, Состояние самолета и Транзитные посадки.

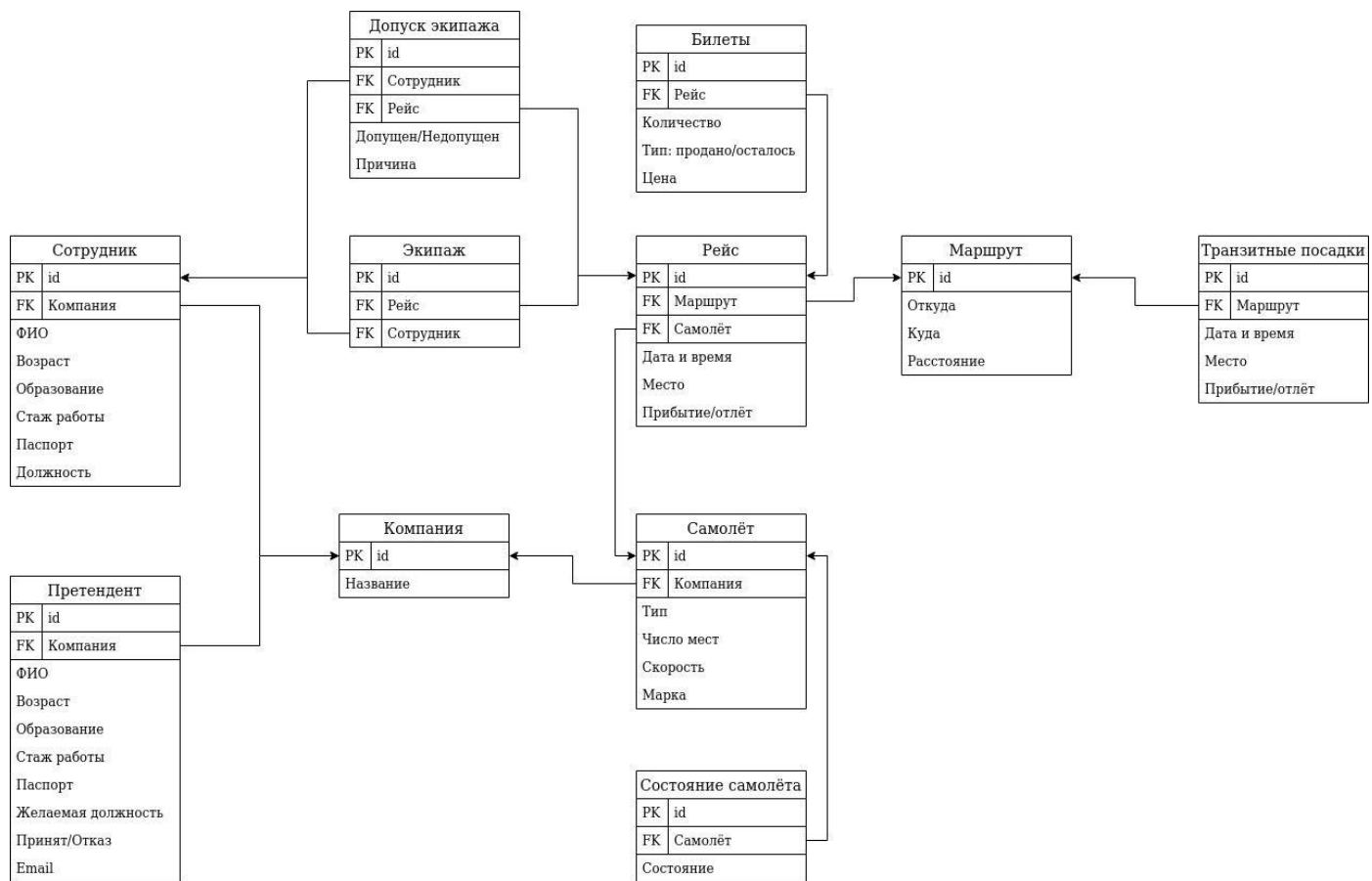


Рисунок 1 - Диаграмма базы данных программной системы

### 3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ

Для реализации программной системы, предназначенной для администрации аэропорта некоторой компании-авиаперевозчика, за основу был взят Django Rest Framework. С помощью него было реализовано сервисное отображение данных, выборка необходимой информации, добавление новой информации и исполнение запросов в соответствии с поставленными задачами функционала.

Для осуществления фронтенда приложения использовалось Vue.js с плагином Vuetify. Vuetify был выбран по причине того, что он предоставляет быструю скорость сборки клиентской части, а также удобство и простоту дизайна. Представления программной системы отображены в папке Components и полностью удовлетворяют запрашиваемому функционалу.

В качестве базы данных была использована база данных PostgreSQL; настройка базы данных, а также последующая работа и сборка программной системы производилась с помощью редактора PyCharm. Для удобства запуска, а также быстроты дальнейшей сборки на желаемом сервисе были добавлены Docker-контейнеры для каждой части архитектуры приложения.

Проект разделён на следующие главные интерфейсы:

- Подача заявки на работу(все заявления видны через админку)
- Регистрация пользователя
- Личный кабинет пользователя (содержит все его билеты со всей информацией по ним. Есть возможность отметить задачу выполненной, выполнение отмечается обеими сторонами: исполнителем и руководителем)
- Главная страница с выбором необходимого рейса со всеми фильтрами

Вся навигация в проекте и дизайн реализованы с помощью плагина Vue.js Vuetify. Для реализации программной системы были использованы следующие модули: django, django rest framework, psycopg2, django cors headers.

Добавленные модели представлены на Рисунке 2.



## Site administration

AIRPORT		
Challengerss	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Companyys	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Employee states	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Employeeess	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Flight teams	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Flightss	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Plane states	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Planes	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Routes	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Ticketss	<a href="#">+ Add</a>	<a href="#">✎ Change</a>
Transit pointss	<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 2 – Модели

## **4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ**

### **4.1. Описание средств разработки клиентской части**

Для разработки клиентской части системы был использован фреймворк Vue.js и библиотека Muse UI. Vue.js — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками. [3] Библиотека Muse UI представляет собой набор компонентов для Vue, которые используют Material Design [4]. Это фреймворк для быстрого создания и запуска пользовательского интерфейса с приятным и удобным дизайном.

### **4.2. Разработанные интерфейсы**

#### **Реализация интерфейса «Регистрация пользователя»**

Данный интерфейс позволяет пользователям зарегистрироваться и обзавестись личным кабинетом для дальнейших действий. Пользователю представлена также кнопка «Уже есть аккаунт» для быстрого входа в личный кабинет.

Реализация интерфейса User представлена на рисунке 3

## Регистрация

Имя пользователя

Адрес электронной почты

Пароль



ЗАРЕГИСТРИРОВАТЬСЯ

Уже есть аккаунт? [Вход](#)

Рисунок 3 – Регистрация

Всех зарегистрированных пользователей можно увидеть в админке.

Select user to change

Q  Search

Action:  Go 0 of 4 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS
<input type="checkbox"/>	marina	
<input type="checkbox"/>	Marina	7marinabe@gmail.com
<input type="checkbox"/>	marina7	7marina@mail.ru
<input type="checkbox"/>	Marina77	7martina@mail.ru

Рисунок 4 – Пользователи

### Реализация интерфейса “Личный кабинет”

Данный интерфейс отображает всю информацию о купленных билетах и ближайших рейсах.

Реализация интерфейса представлена на рисунке 5.

## Кабинет пользователя

Ваши билеты:

Tarjei stjerne: Saint P - Moscow, 633 км, 2020-06-07

svendsen leti: London - Paris, 344 км, 2020-06-09

Рисунок 5 – Личный кабинет


Вход в личный кабинет осуществляется путем заполнения личных данных. (Рисунок 6)

## Вход

Имя пользователя

|

Пароль

 ВОЙТИ

Ещё нет аккаунта? [Регистрация](#)










Рисунок 6 - Вход

### Реализация интерфейса “Подача заявки на работу”

В данном интерфейсе отображены все данные по заявке, которую пользователь оставил на сайте. Чтобы создать заявку, пользователь должен заполнить следующие поля: дать

информацию о себе, указать опыт работы, желаемую должность и выбрать компанию, в которой он хочет работать. Соответственно, вся эта информация отображается в Create Request. К ней добавляются ещё следующие данные: статус заявки (выполнена или не выполнена), дата создания заявки. В данном интерфейсе важную роль играют права доступа. Администратор может просмотреть список заявок, нанять и уволить человека.

### Change challengers

Company:	S7  
First name:	Vasily
Last name:	Petrov
Middle name:	Vladimirovich
Age:	28  
Experience:	3  
Passport:	4141414  
Position:	Captain
State:	Rejection 
Email:	valisiy@bk.ru

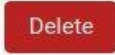


Рисунок 7 – Инфо



Рисунок 8 - Заявка

Проверяем в админке:

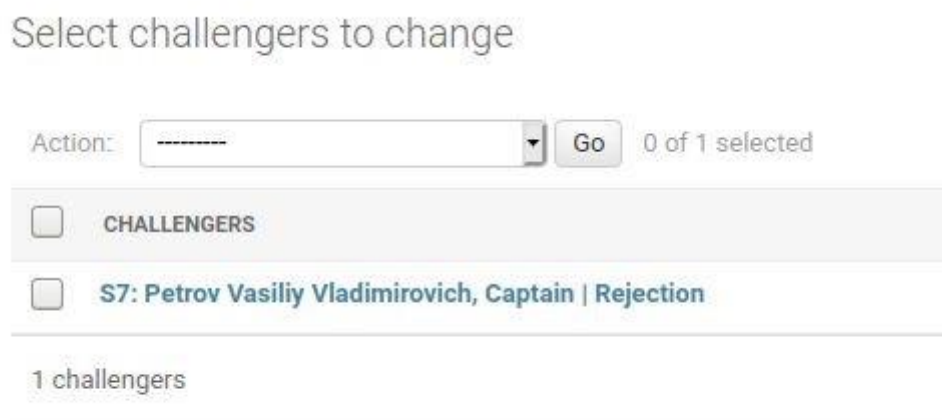


Рисунок 9 - заявка

Также есть возможность увидеть какую-либо конкретную заявку с помощью админки; заявка отображается дате подачи. На рисунке 10 показано, как отображаются все заявки, а также возможность их удаления.

## Select challengers to change

Action:   1 of 2 selected

<input type="checkbox"/>	CHALLENGER	<input type="button" value="Delete selected challengers"/>
<input checked="" type="checkbox"/>	Tarjei stjerne: Berezhnova Marina Yurievna, Pilot   Rejection	
<input type="checkbox"/>	Marina7: Federer Roger Roger, Pilot   Rejection	

2 challengers

Рисунок 10

## Реализация интерфейса “Главная страница”

Данный интерфейс отображает информацию о рейсах, соответствующую поставленным задачам. Сортировка и выборка необходимых данных происходит на бэкенде.

Все рейсы

Хочу работать

Вход

Регистрация

≡

✈ Airport

Откуда	Куда	Цена билета 0
Модель самолёта	Количество оставшихся билетов 0	<input type="checkbox"/> Нет транзитных точек

⌵ ОТФИЛЬТРОВАТЬ

🔄 ОЧИСТИТЬ

Рисунок 11 – главная страница

Слева отображаются опции регистрации нового пользователя, входа в личный кабинет и подача заявки на работу.

На главной странице показаны все опции фильтрации поиска нужного билета: откуда, куда, цена билета, модель самолета, отсутствие транзитных точек и количество оставшихся билетов. На рисунке 12 показано, как работает фильтрация по запросу.

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "flights": [
    {
      "id": 1,
      "date": "2020-05-21",
      "route": {
        "id": 1,
        "from_point": "Saint-Petersburg",
        "to_point": "Kazan",
        "distance": 1500
      },
      "plane": {
        "id": 1,
        "plane_type": "2",
        "seats_amonut": 400,
        "speed": 400,
        "model": "Airbus-3000",
        "company": 1
      }
    }
  ],
  "transit_points": [
    {
      "id": 1,
      "date": "2020-05-21",
      "action": "1",
      "point": "Moscow",
      "route": {
        "id": 1,
        "from_point": "Saint-Petersburg",
        "to_point": "Kazan",
        "distance": 1500
      }
    }
  ]
}
```



```

    {
      "id": 2,
      "date": "2020-05-22",
      "action": "2",
      "point": "Moscow",
      "route": {
        "id": 1,
        "from_point": "Saint-Petersburg",
        "to_point": "Kazan",
        "distance": 1500
      }
    }
  ],
  "teams": [
    {
      "id": 1,
      "employee": {
        "id": 1,
        "first_name": "Ivan",
        "last_name": "Ivanov",
        "middle_name": "Ivanovich",
        "age": 35,
        "experience": 10,
        "passport": 10101010,
        "position": "Captain",
        "company": 1
      },
      "flight": {
        "id": 1,
        "date": "2020-05-21",
        "route": {
          "id": 1,
          "from_point": "Saint-Petersburg",
          "to_point": "Kazan",
          "distance": 1500
        },
        "plane": {
          "id": 1,
          "plane_type": "2",
          "seats_amonut": 400,
          "speed": 400,
          "model": "Airbus-3000",
          "company": 1
        }
      }
    }
  ]
},

```

```

{
  "id": 2,
  "employee": {
    "id": 2,
    "first_name": "Masha",
    "last_name": "Vasileva",
    "middle_name": "Ivanovna",
    "age": 23,
    "experience": 2,
    "passport": 1010010101,
    "position": "Stuard",
    "company": 1
  },
  "flight": {
    "id": 1,
    "date": "2020-05-21",
    "route": {
      "id": 1,
      "from_point": "Saint-Petersburg",
      "to_point": "Kazan",
      "distance": 1500
    },
    "plane": {
      "id": 1,
      "plane_type": "2",
      "seats_amonut": 400,
      "speed": 400,
      "model": "Airbus-3000",
      "company": 1
    }
  }
}

```

Рисунок 12 – рейсы

На рисунке 13 представлена сортировка билетов по цене: каждый билет можно купить при помощи кнопки «Купить» в левом углу каждого билета. Также представлена кнопка «Очистить», что позволяет сбросить все ранее выбранные фильтры

Все рейсы

Хочу работать

Вход

Регистрация

≡ ≥ Airport

Откуда

Куда

Цена билета

12000

Модель самолёта

Количество оставшихся билетов

☐ Нет транзитных точек

ОТФИЛЬТРОВАТЬ

ОЧИСТИТЬ

Tarjei stjerne: Saint P - Moscow | Plane: Boeing 747

Осталось 100 билетов

2300 руб./билет

КУПИТЬ

svendsen leti: London - Paris | Plane: Boenig 700

Осталось 136 билетов

4500 руб./билет

КУПИТЬ

Bob72: Saint P - München | Plane: Airbus A320

Осталось 230 билетов

9200 руб./билет

КУПИТЬ

Рисунок 13 – Покупка билета

При покупке билета выходит следующее окно (рисунок 14):

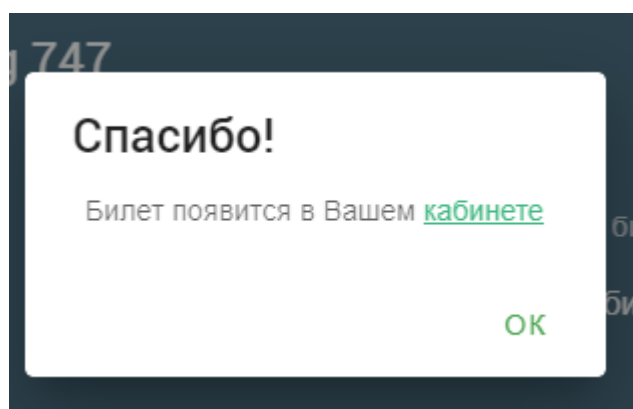


Рисунок 14 – удачная покупка билета



## 5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ

Под контейнеризацией понимается такой подход к разработке программного обеспечения, при котором все части приложения упаковываются вместе в образ контейнера. Преимущество такого подхода в том, что он обеспечивает работоспособность приложения в различных средах. Оркестрация представляет собой метод координации нескольких контейнеров. Оркестрацию проекта можно опустить, но если ее использовать, то приложение получает гибкость, масштабируемость и взаимосвязанность процессов в контейнерах. Использование оркестрации позволяет создавать системы из множества контейнеров, где каждый контейнер отвечает только за свою задачу. Кроме того, каждый их контейнеров можно заменить другим, не перезапуская весь проект. В качестве средства контейнеризации и оркестрации проекта используется Docker. Docker – это открытая платформа для разработки, доставки и эксплуатации приложений. С помощью технологии Docker (контейнеризации) можно разделить исходное приложение на несколько компонентов, которые взаимодействие между которыми возможно реализовать. Подобный подход может привести разные методы реализации компонентов, тем самым предоставляя разработчику широкий спектр возможностей. Благодаря этому, разработчику предоставляется возможность создания микросервисных архитектур. Для контейнеризации проекта были созданы файлы `Dockerfile` в каждой из директорий для будущих контейнеров. Так как проект состоит из трех основных частей (база данных, фронтенд и бэкенд), то и контейнеров будет в итоге три. Для оркестрации проекта в корневой папке необходимо создать файл `dockercompose.yml`, который отвечает за оркестрацию. В нем описаны детали для запуска каждого контейнера, указаны порты и необходимые команды. Первым этапом запуска проекта является сборка всех контейнеров с помощью команды `docker-compose build`. Затем необходимо выполнить команду `docker-compose up`, чтобы запустить проект.

## 6. ЗАКЛЮЧЕНИЕ

Цель курсовой работы заключалась в реализации программной системы для аэропорта которая позволила бы пользователям легко и просто покупать билеты на сайте, фильтруя необходимым образом, просматривать свои покупки и оставлять заявку на работу.

Для решения данной задачи был выбран следующий стек технологий: Django Rest Framework, Vue.js, PostgreSQL; это позволило достаточно быстро и удобно реализовать задуманный проект. В результате выполнения курсовой удалось выполнить все поставленные задачи, которые соответствуют запросам технического задания. Цель курсовой работы достигнута в полном размере.

В качестве дальнейшего развития проекта можно усовершенствовать дизайн веб-сайта, например, добавить фотографии аэропорта. Также можно добавить возможность оставлять отзывы, что также будет содействовать повышению доверия пользователей.

## 7. СПИСОК ЛИТЕРАТУРЫ

*Django Rest Framework*. Документация Django Rest Framework [Электронный ресурс]. URL: <https://www.django-rest-framework.org> (дата обращения: 29.06.2020).

*WebDevBlog*. Создание Django API используя Django Rest Framework [Электронный ресурс]. URL: <https://webdevblog.ru/sozдание-django-api-ispolzuya-django-rest-framework-apiview/> (дата обращения: 29.06.2020).

*Evantotuts+*. JWT Аутентификация в Django [Электронный ресурс] URL: <https://code.tutsplus.com/ru/tutorials/how-to-authenticate-with-jwt-in-django--cms-30460> (дата обращения: 29.06.2020).

*Vue.js*. Документация Vue.js [Электронный ресурс]. URL: <https://vuejs.org> (дата обращения: 29.06.2020).

*Vuetify*. Документация Vuetify [Электронный ресурс]. URL: <https://vuetifyjs.com/ru/> (дата обращения: 29.06.2020).

*Stack OverFlow*. Set initial vuetify vue-select value [Электронный ресурс]. URL: <https://stackoverflow.com/questions/51392719/set-initial-vuetify-v-select-value> (дата обращения: 29.06.2020).

*Asyncee*. Продвинутые запросы в Django: сортировка по дате [Электронный ресурс]. URL: <https://asyncee.github.io/posts/advanced-django-querying-sorting-events-by-date/> (дата обращения: 29.06.2020).

Документация *Docker* [Электронный ресурс] — <https://docs.docker.com/engine/install/>. Дата обращения: 20.06.2020.

«Клиент-серверная архитектура в картинках» [Электронный ресурс] — <https://habr.com/ru/post/495698/>. Дата обращения: 20.05.2020.