

Министерство образования и науки Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ

**“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 отслеживания распределения газет»**

Обучающийся: Зангиева В.Т., группа К3342

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

Оценка за курсовую работу \_\_\_\_

Подпись преподавателя:

\_\_\_\_\_ (\_\_\_\_\_)  
(подпись)

Дата \_\_\_\_

Санкт-Петербург  
2020

## Содержание

ВВЕДЕНИЕ.....	3
ГЛАВА 1 – ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА .....	4
1.1. Анализ предметной области .....	4
1.2. Проектирование приложения .....	5
1.2.1. Разработка модели «Распределение газет» .....	5
1.3. Состав реквизитов сущностей .....	6
1.4. Теоретические сведения об используемых технологиях .....	6
ГЛАВА 2 – РЕАЛИЗАЦИЯ WEB-СЕРВИСА .....	8
2.1. Серверная часть приложения.....	8
2.1.1. Полученные интерфейсы в панели Django Admin .....	11
2.1.2. Полученные интерфейсы в панели Django REST .....	13
2.2. Клиентская часть.....	15
ВЫВОДЫ.....	20
ЗАКЛЮЧЕНИЕ .....	21
СПИСОК ИСТОЧНИКОВ.....	22

## ВВЕДЕНИЕ

Курсовая работа посвящена созданию web-сервиса, предназначенного для отслеживания распределения по почтовым отделениям газет, печатающихся в типографиях города. Программная система должна обеспечивать хранение, просмотр и изменение сведений о газетах, почтовых отделениях, получающих газеты и о типографиях, выпускающих газеты.

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 – ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА

## 1.1. Анализ предметной области

Предметной областью для курсовой работы является отслеживание распределения по почтовым отделениям газет, печатающихся в типографиях города.

### **Формулировка задания:**

Создать программную систему, позволяющую отслеживать распределение по почтовым отделениям газет, печатающихся в типографиях города.

Система должна обеспечивать хранение, просмотр и изменение сведений о газетах, почтовых отделениях, получающих газеты и о типографиях, выпускающих газеты.

В типографии разными тиражами печатаются газеты нескольких наименований.

Сведения о газетах включают в себя: название газеты, индекс издания, фамилию, имя и отчество редактора, цену экземпляра газеты. Цены могут меняться. Возможно появление новых газет и изменения индекса существующего издания.

Для типографий указываются их названия и адреса.

Типография может быть закрыта, тогда необходимо скорректировать работу других типографий с учетом потребностей почтовых отделений в газетах.

Почтовое отделение имеет номер и адрес. На каждое почтовое отделение поступают в определенных количествах газеты разных наименований, причем часть экземпляров одной и той же газеты может быть напечатана в одной типографии, а часть – в другой.

Пользователям системы может потребоваться следующая информация:

- По каким адресам печатаются газеты данного наименования?
- Фамилия редактора газеты, которая печатается в указанной типографии самым большим тиражом?
- На какие почтовые отделения (адреса) поступает газета, имеющая цену, больше указанной?
- Какие газеты и куда (номер почты) поступают в количестве меньшем, чем заданное?
- Куда поступает данная газета, печатающаяся по данному адресу.

## 1.2. 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.

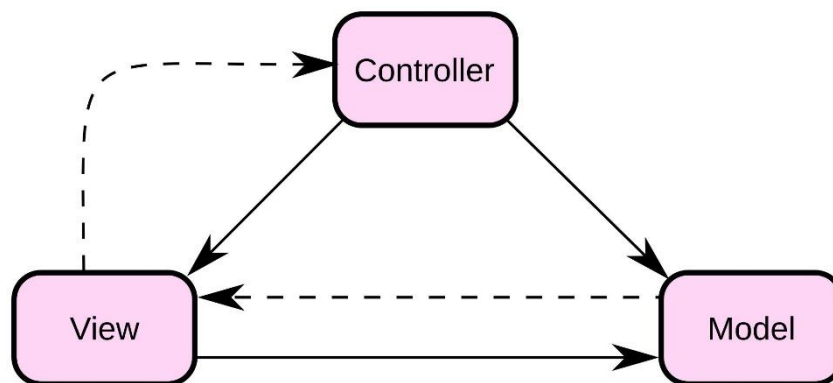


Рисунок 1 – концепция MVC

### 1.2.1. Разработка модели «Распределение газет»

В соответствии с индивидуальным заданием была разработана модель БД, представленная на Рисунке 2.

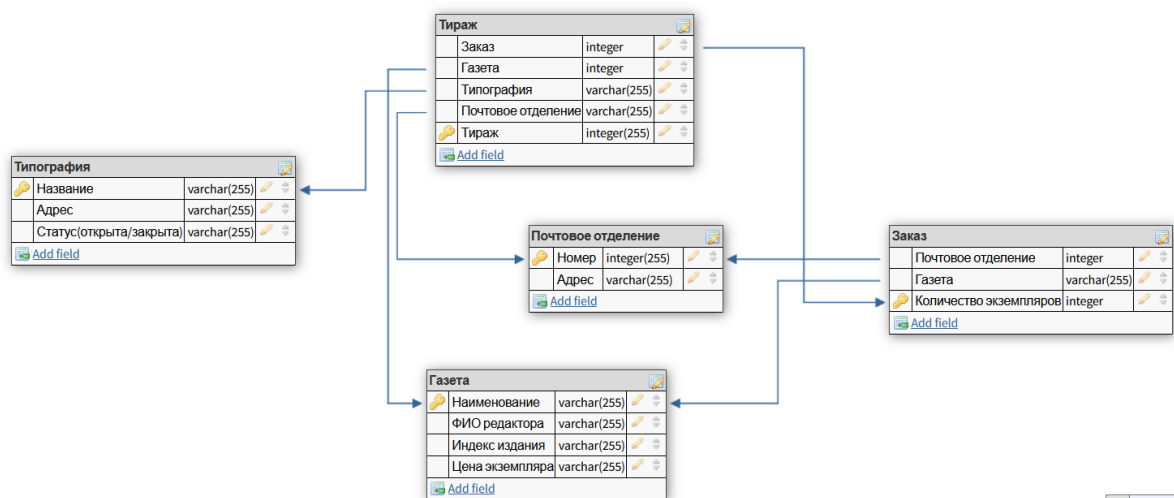


Рисунок 2 – модель базы данных «Распределение газет»

### **1.3. Состав реквизитов сущностей**

- Почтовое отделение (номер почтового отделения, адрес почтового отделения).
- Газета (наименование газеты, ФИО редактора газеты, индекс издания газеты, цена экземпляра газеты).
- Типография (название типографии, адрес типографии, статус типографии – открыта/закрыта).
- Заказ (почтовое отделение (номер), газета (наименование), количество экземпляров).
- Тираж (заказ (количество экземпляров), почтовое отделение (номер), газета (наименование), типография (название), тираж).

### **1.4. Теоретические сведения об используемых технологиях**

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций). Требуемое приложение будет реализовываться с помощью трехзвенной клиент-серверной архитектуры, изображенной на Рис.3. Трехзвенная клиент-серверная архитектура - сетевое приложение разделено на две и более частей, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате.

Третьим звеном в трехзвенной архитектуре становится сервер приложений, т.е. компоненты распределяются следующим образом:

1. Представление данных — на стороне клиента.
2. Прикладной компонент — на выделенном сервере приложений (как вариант, выполняющем функции промежуточного ПО).
3. Управление ресурсами — на сервере БД, который и представляет запрашиваемые данные.

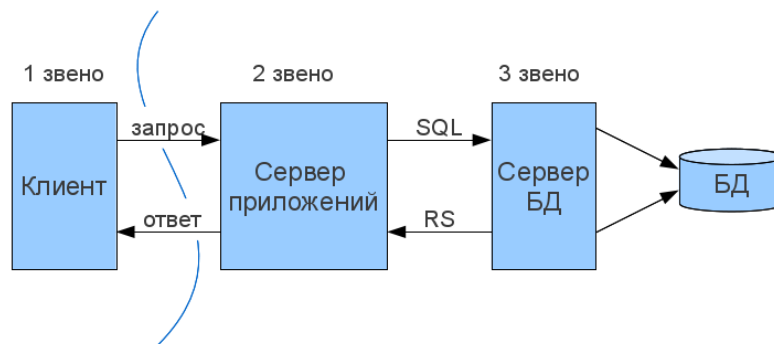


Рисунок 3 – Трехзвенная архитектура

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 (ClassBased-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 – РЕАЛИЗАЦИЯ WEB-СЕРВИСА

### 2.1. 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, 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 2).

```
from django.db import models

class PostOffice(models.Model):
    poNum = models.PositiveIntegerField('Номер почтового отделения')
    poAddress = models.CharField('Адрес почтового отделения', max_length = 500)
```

Рисунок 4 – модель «Почтовое отделение»

```
class PrintingHouse(models.Model):
    CType = (
        ('открыта', 'открыта'),
        ('закрыта', 'закрыта'),
    )
    phName = models.CharField('Название типографии', max_length = 50)
    phAddress = models.CharField('Адрес типографии', max_length = 500)
    phWorkStatus = models.CharField('Статус', choices=CType, max_length=100)
```

Рисунок 5 – модель «Типография»

```
class Newspaper(models.Model):
    npName = models.CharField('Название газеты', max_length = 50)
    npEditorName = models.CharField('ФИО редактора', max_length = 100)
    npEditionIndex = models.IntegerField('Индекс издания')
    npPrice = models.PositiveIntegerField('Цена экземпляра')
```

Рисунок 6 – модель «Газета»



```
class Order(models.Model):
    oPoCode = models.ForeignKey(PostOffice, on_delete=models.CASCADE, verbose_name='Почтовое отделение')
    oNpCode = models.ForeignKey(Newspaper, on_delete=models.CASCADE, verbose_name='Газета',)
    oNpCount = models.IntegerField('Количество экземпляров')
```

Рисунок 7 – модель «Заказ»

```
class PrintRun(models.Model):
    prPoCode = models.ForeignKey(PostOffice, on_delete=models.CASCADE)
    prOCode = models.ForeignKey(Order, on_delete=models.CASCADE)
    prNpCode = models.ForeignKey(Newspaper, on_delete=models.CASCADE)
    prPhCode = models.ForeignKey(PrintingHouse, on_delete=models.CASCADE)
    prPrintRun = models.PositiveIntegerField('Тираж')
```

Рисунок 8 – модель «Тираж»

Создание админ панели для разработанной модели данных.

```
from django.contrib import admin

from .models import PrintingHouse, PostOffice, Newspaper, Order, PrintRun

admin.site.register(PrintingHouse)
admin.site.register(PostOffice)
admin.site.register(Newspaper)
admin.site.register(Order)
admin.site.register(PrintRun)
```

Рисунок 9 – содержимое файла «admin.py»

```
class OrderSerializer(serializers.ModelSerializer):
    oPoCode = getPostOfficeNumSerializer()
    oNpCode = getNewspaperNameSerializer()
    class Meta:
        model = Order
        fields = "__all__"

class OrderSerializer_raw(serializers.ModelSerializer):
    class Meta:
        model = Order
        fields = "__all__"

class PrintRunSerializer(serializers.ModelSerializer):
    class Meta:
        model = PrintRun
        fields = "__all__"

class PrintRunFunc1Ser(serializers.ModelSerializer):
    phAddress = serializers.CharField(source='prPhCode__phAddress')
    class Meta:
        model = PrintRun
        fields = ('phAddress',)
```

Рисунок 10 – фрагмент кода файла «sterializers.py»

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

```
class CreatePrintRunsViewSet(ModelViewSet):
    """Формирование тиражей по последнему заказу"""
    queryset = PrintRun.objects.all()
    serializer_class = PrintRunSerializer
    filter_backends = (DjangoFilterBackend,)
    filterset_class = PrintRunFilter

    def get_queryset(self):
        lastOrder = Order.objects.last()
        postOffice = lastOrder.oPoCode
        newspaper = lastOrder.oNpCode
        orderNpCount = lastOrder.oNpCount
        openedPH = PrintingHouse.objects.filter(phWorkStatus='открыта')
        printRuns = splitCount(orderNpCount, len(openedPH))

        i = 0
        for printingHouse in openedPH:
            PrintRun.objects.create(
                prOCode=lastOrder,
                prPoCode=postOffice,
                prNpCode=newspaper,
                prPhCode=printingHouse,
                prPrintRun=printRuns[i]
            )
            i += 1
```

Рисунок 11 – фрагмент кода файла «views.py»

Файл urls.py содержит пути для доступа к страницам.

```
from django.urls import path
from rest_framework.routers import DefaultRouter
from mainapp.views import (
    PostOfficeViewSet,
    NewspaperViewSet,
    PrintingHouseViewSet,
    OrderViewSet,
    OrderViewSet_raw,
    PrintRunViewSet,
    CreatePrintRunsViewSet,
    UpdatePrintRuns,
    PHAddress,
    EditorName,
    PostOfficeAddress,
    NewspaperNameAndPostOfficeNum,
    PostOfficeInfo,
)

router = DefaultRouter()
router.register("po_list", PostOfficeViewSet)
router.register("np_list", NewspaperViewSet)
router.register("ph_list", PrintingHouseViewSet)
router.register("o_list", OrderViewSet)
router.register("o_list2", OrderViewSet_raw)
router.register("pr_create", CreatePrintRunsViewSet)
router.register("pr_update", UpdatePrintRuns)
router.register("pr_list", PrintRunViewSet)
router.register("func1", PHAddress)
router.register("func2", EditorName)
router.register("func3", PostOfficeAddress)
router.register("func4", NewspaperNameAndPostOfficeNum)
router.register("func5", PostOfficeInfo)

urlpatterns = router.urls
```

Рисунок 12 – пути к страницам в файле «urls.py»

### 2.1.1. Полученные интерфейсы в панели Django Admin

1. Хранение, добавление, удаление, просмотр информации о газетах.
2. Возможность изменения статуса типографии (открыта/закрыта).
3. Добавление новых заказов.

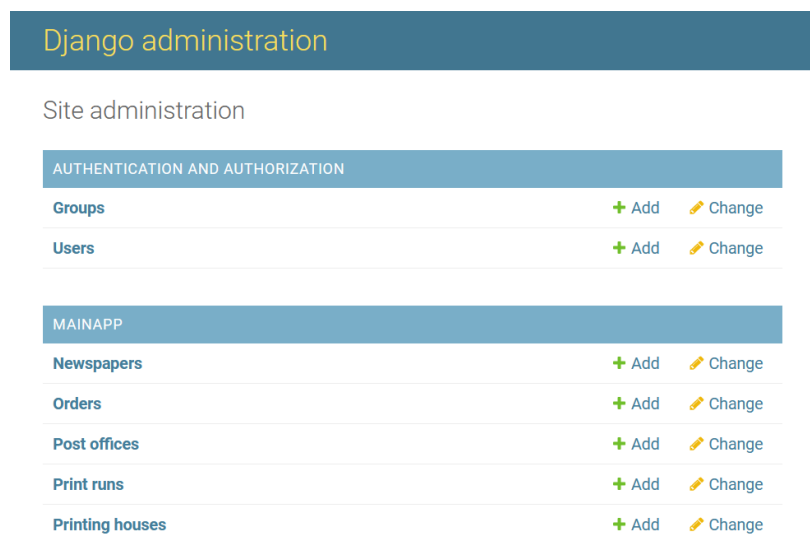


Рисунок 13 – все имеющиеся модели

Были добавлены модели (Рисунок 13). Ниже приведены некоторые полученные интерфейсы в панели django-admin.

The screenshot shows the 'Change newspaper' form in the Django Admin interface. It contains four input fields: 'Название газеты:' (Newspaper name) with the value 'Такие дела', 'ФИО редактора:' (Editor's full name) with the value 'Сысоев Петр Антонинович', 'Индекс издания:' (Edition index) with the value '547896', and 'Цена экземпляра:' (Price per copy) with the value '15'. At the bottom, there is a red 'Delete' button.

Название газеты:	Такие дела
ФИО редактора:	Сысоев Петр Антонинович
Индекс издания:	547896
Цена экземпляра:	15

[Delete](#)

Рисунок 14 – Добавление, просмотр, удаление, хранение информации о газетах

**Django administration**


Home › Mainapp › Printing houses › PrintingHouse object (3)

Change printing house

Название типографии:

Адрес типографии:

Статус: 

открыта 


Delete



-----  
открыта  
закрыта

Рисунок 15 – Возможность изменения статуса типографии (открыта/закрыта)


Change order



Почтовое отделение: 

PostOffice object (5) 



 

Газета: 

Newspaper object (13) 

Количество экземпляров: 

500  

Delete

Рисунок 16 – Добавление новых заказов

### 2.1.2. Полученные интерфейсы в панели Django REST

Список полученных интерфейсов в панели Django REST:

1. /admin
2. /api/po\_list – список почтовых отделений
3. /api/np\_list – список газет
4. /api/ph\_list – список типографий
5. /api/o\_list – список заказов
6. /api/pr\_create – формирование тиражей по последнему заказу
7. /api/pr\_update – обновление тиражей по последнему заказу
8. /api/pr\_list – список тиражей
9. /api/func1 – запрос «По каким адресам печатаются газеты данного наименования?»
10. /api/func2 – запрос «Фамилия редактора газеты, которая печатается самым большим тиражом?»  
/api/func3 – запрос «На какие почтовые отделения (адреса) поступает газета, имеющая цену, больше указанной?»
11. /api/func4 – запрос «Какие газеты и куда (номер почты) поступают в количестве меньшем, чем заданное?»
12. /api/func5 – запрос «Куда поступает данная газета, печатающаяся по данному адресу.

Ниже приведены некоторые из имеющихся интерфейсов.

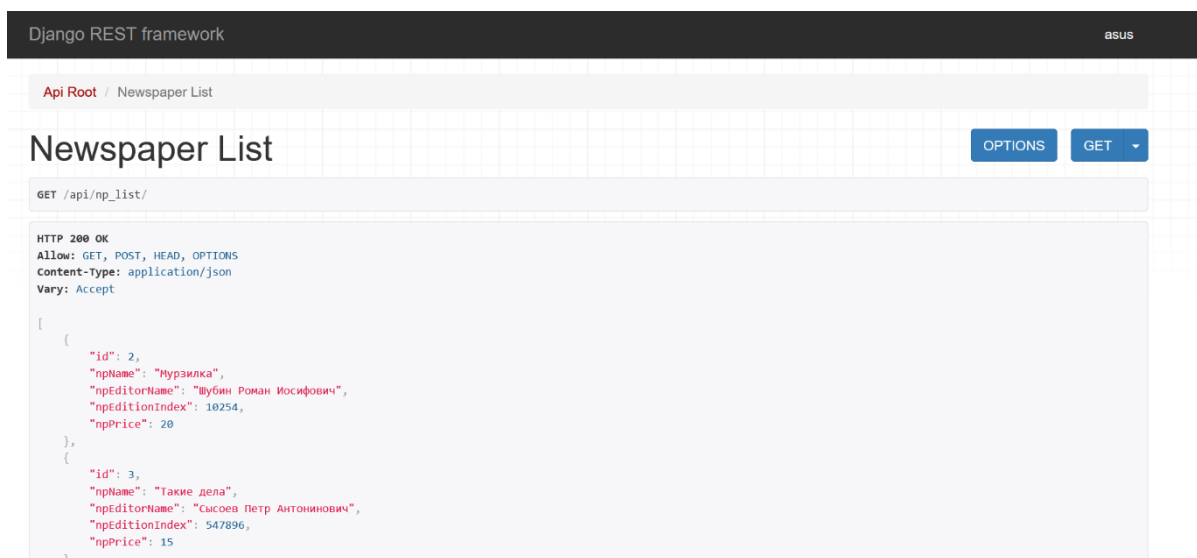


Рисунок 17 – интерфейс «список газет»

Django REST frameworkasus

```

prPhCode": 3
},
{
  "id": 191,
  "prPrintRun": 250,
  "prPoCode": 5,
  "prOCode": 50,
  "prNpCode": 13,
  "prPhCode": 4
}
]

```

Raw data HTML form

Тираж

PrPoCode

PostOffice object (3)

PrOCode

Order object (43)

PrNpCode

Newspaper object (2)

PrPhCode

PrintingHouse object (1)

POST

Рисунок 18 – интерфейс «формирование тиражей по последнему заказу»

Django REST framework

Content-Type: application/json  
Vary: Accept

```

[
  {
    "npName": "Желток",
    "poNum": "2",
    "prPrintRun": 600
  },
  {
    "npName": "Желток",
    "poNum": "74",
    "prPrintRun": 500
  },
  {
    "npName": "Еженеделька",
    "poNum": "2",
    "prPrintRun": 250
  },
  {
    "npName": "Муразилка",
    "poNum": "25",
    "prPrintRun": 250
  },
  {
    "npName": "News",
    "poNum": "2",
    "prPrintRun": 250
  },
]

```

Рисунок 19 – интерфейс запроса «Какие газеты и куда (номер почты) поступают в количестве меньшем, чем заданное?»

## 2.2. Клиентская часть

Были реализованы следующие интерфейсы:

1. Интерфейс для просмотра подробной информации о газетах и добавления газеты.
2. Интерфейс для просмотра подробной информации о типографиях и изменения статуса типографии (открыта/закрыта).
3. Интерфейс для просмотра подробной информации о почтовых отделениях.
4. Интерфейс для просмотра подробной информации о заказах, а также добавления нового заказа.
5. Интерфейс с информацией, которая может потребоваться пользователю (запросы).

Интерфейсы реализуются с помощью технологий Vue.js. В качестве UI-библиотеки была использована Muse-UI.

Библиотека Muse-UI подключаются в файле main.js проекта, интерфейсы реализуются с помощью тегов в представлениях frontend формата .vue.

```
1  import Vue from 'vue';
2  import MuseUI from 'muse-ui';
3  import theme from 'muse-ui/lib/theme';
4  import App from './App.vue';
5  import router from './router';
6  // import MuseUI from 'muse-ui';
7  import 'muse-ui/dist/muse-ui.css';
8
9  theme.use('light');
10
11 Vue.use(MuseUI);
12
13 Vue.config.productionTip = false;
14
15 new Vue({
16   router,
17   render: (h) => h(App),
18 }).$mount('#app');
```

Рисунок 20 – подключение библиотеки Muse-UI в main.js





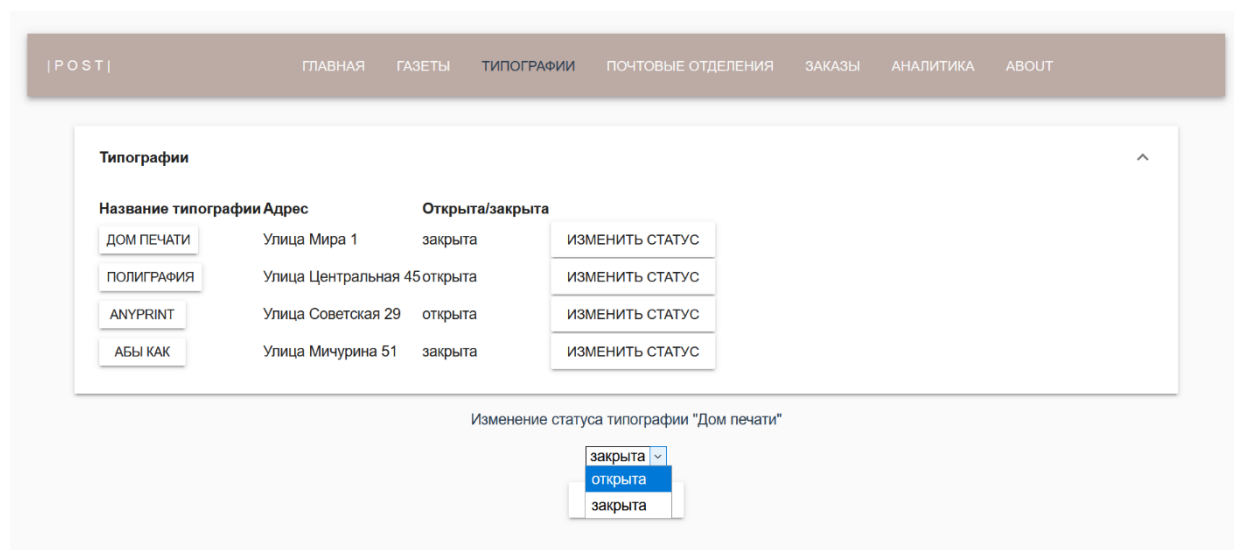


Рисунок 23 – «Типографии»

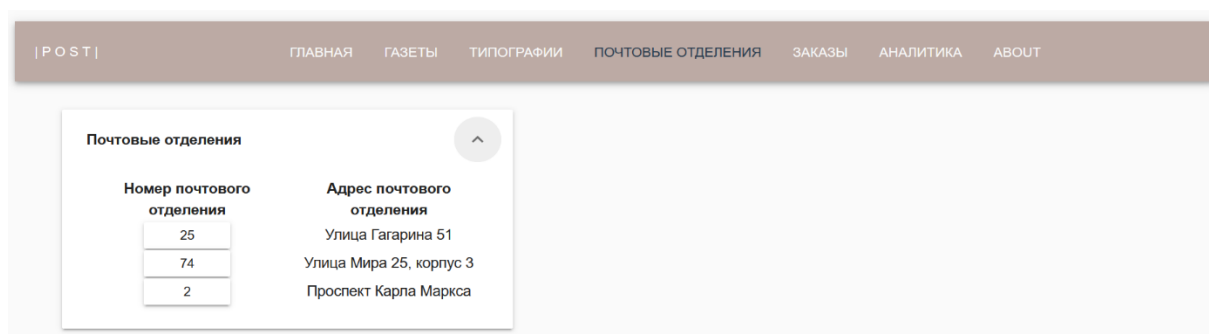


Рисунок 24 – «Почтовые отделения»

По каким адресам печатается газета

Введите название газеты

НАЙТИ

Вывод ФИО Редактора с самым крупным тиражом

НАЙТИ

Найти адреса почтовых отделений с ценой газеты выше чем:

Введите цену газеты

НАЙТИ

Какие газеты и куда поступают в количестве меньшем чем

Введите количество газет

НАЙТИ

Номер отделения, куда поступает газета, которая печатается по адресу

Введите название газеты

Введите адрес типографии

НАЙТИ

Рисунок 25 – «Аналитика»

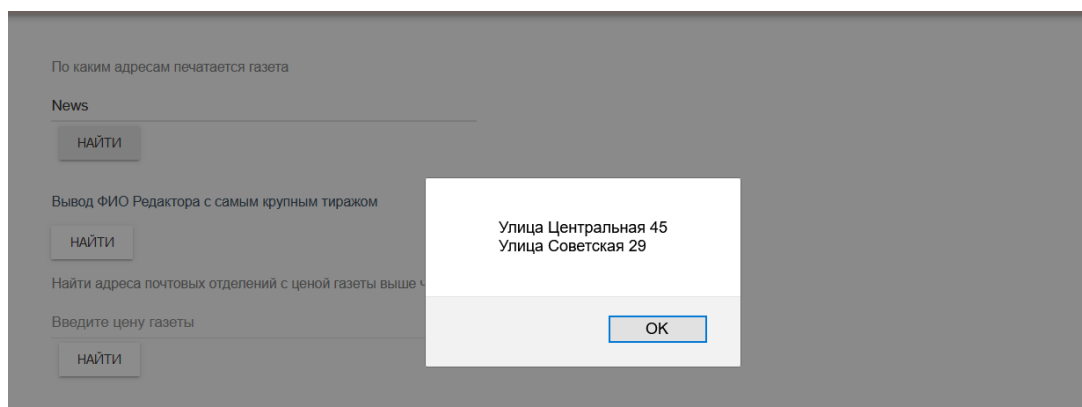


Рисунок 26 – пример выполнения запроса в интерфейсе «Аналитика»

### Заказы

Номер почтового отделения	Название газеты	Количество газет
25	Мурзилка	250
74	Желток	1000
2	Еженеделька	500
2	Желток	1200
25	Мурзилка	500
2	News	500

СДЕЛАТЬ НОВЫЙ ЗАКАЗ

Почтовое отделение:

Номер почтового отделения

25

74

2

ВыбРАТЬ

ВыбРАТЬ

ВыбРАТЬ

Газета:

Название газеты

Мурзилка

Такие дела

Сканворды

Еженеделька

Желток

News

ВыбРАТЬ

ВыбРАТЬ

ВыбРАТЬ

ВыбРАТЬ

ВыбРАТЬ

ВыбРАТЬ

Выбрано почтовое отделение № 74

Выбрана газета Такие дела

Количество газет:

700

ДОБАВИТЬ

СКРЫТЬ

Рисунок 27 – «Заказы»

```

1 <template>
2 <div class="order">
3   <mu-container>
4     <mu-row gutter>
5       <mu-col span="5" align-self="left">
6         <mu-expansion-panel>
7           <div slot="header"><th>Заказы</th></div>
8           <th><div slot="header">Номер почтового отделения</div></th>
9           <th><div slot="header">Название газеты</div></th>
10          <th><div slot="header">Количество газет</div></th>
11          <tr v-for="element in elements" :key="element.id">
12            <td><mu-button small=true>{{element.oPoCode.poNum}}</mu-button></td>
13            <td>{{element.oNpCode.npName}}</td>
14            <td>{{element.oNpCount}}</td>
15          </tr>
16        </mu-expansion-panel>
17      </mu-col>
18    </mu-row>
19  </mu-container>
20 <p>
21   <mu-container>
22     <mu-row gutter align="left">
23       <mu-col span="5" align-self="left">
24         <mu-button @click="showNewOrderForm = true" color="#8d6e63">Сделать новый заказ
25       </mu-button>
26       <mu-form v-show="showNewOrderForm">
27         <mu-form>
28           <label>Почтовое отделение:</label>
29           <mu-div></mu-div>
30           <tr>
31             <th>Номер почтового отделения</th>
32           </tr>
33           <tr v-for="postOffice in postOffices"
34             :key="postOffice.id">
35             <td>{{postOffice.poNum}}</td>
36             <td><mu-button @click="fetchCurPO(postOffice)">Выбрать</mu-button></td>
37           </tr>

```

Рисунок 28 – фрагмент кода Order.vue (Заказы)

```

1 <template>
2 <div class="postOffice">
3   <mu-container>
4     <mu-row gutter align="left">
5       <mu-col span="5" align-self="left">
6         <mu-expansion-panel>
7           <div slot="header"><th>Почтовые отделения</th></div>
8           <th><div slot="header">Номер почтового отделения</div></th>
9           <th><div slot="header">Адрес почтового отделения</div></th>
10          <tr v-for="element in elements" :key="element.id">
11            <td><mu-button small=true>{{element.poNum}}</mu-button></td>
12            <td>{{element.poAddress}}</td>
13          </tr>
14        </mu-expansion-panel>
15      </mu-col>
16    </mu-row>
17  </mu-container>
18 </div>
19 </template>
20 <script>
21 export default {
22   data() {
23     return {
24       elements: [],
25     };
26   },
27   async created() {
28     await this.fetchElements();
29   },
30   methods: {
31     async fetchElements(url = 'http://127.0.0.1:8000/api/po_list/') {
32       const response = await fetch(url);
33       this.elements = await response.json();
34     },
35   },
36 };
37 </script>

```

Рисунок 29 – код PostOffice.vue (Почтовые отделения)

## **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 Muse-UI на Vue.js.

## **ЗАКЛЮЧЕНИЕ**

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

В приложении реализован набор интерфейсов:

1. Интерфейс для просмотра подробной информации о газетах и добавления газеты.
2. Интерфейс для просмотра подробной информации о типографиях и изменения статуса типографии (открыта/закрыта).
3. Интерфейс для просмотра подробной информации о почтовых отделениях.
4. Интерфейс для просмотра подробной информации о заказах, а также добавления нового заказа.
5. Интерфейс с информацией, которая может потребоваться пользователю (запросы).

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 26.06.2020)
2. Документация Django на русском языке [Электронный ресурс]  
URL: <https://djbook.ru/rel1.9/> (Дата обращения 28.06.2020)
3. Vue.js. Введение [Электронный ресурс]. Режим доступа: <https://ru.vuejs.org/v2/guide/index.html> (Дата обращения 29.06.2020)
4. Документация Muse UI на английском языке. [Электронный ресурс].  
URL: <https://muse-ui.org/#/en-US> (Дата обращения 27.06.2020)