

**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 45.03.04 - Интеллектуальные системы в гуманитарной  
сфере

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

## **О Т Ч Е Т**

по курсовой работе

Тема задания: Реализация web-сервисов средствами Django REST framework, Vue.js, Muse-UI

Обучающийся Богданова Е.Ю. К3343

Руководитель: Говоров А.И.

Оценка \_\_\_\_

Дата \_\_\_\_

Санкт-Петербург  
20 20

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ.....	5
2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ.....	7
3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ.....	12
4. ИСПОЛЬЗОВАНИЕ DOCKER ДЛЯ РАЗВЕРТЫВАНИЯ WEB-ПРИЛОЖЕНИЙ .....	22
ЗАКЛЮЧЕНИЕ.....	23
СПИСОК ЛИТЕРАТУРЫ.....	24
ПРИЛОЖЕНИЯ .....	25

## ВВЕДЕНИЕ

В качестве варианта курсовой работы было выбрано задание, по которому необходимо было создать программную систему, предназначенную для работников приемной комиссии колледжа. Она должна обеспечивать хранение, просмотр и изменение сведений об абитуриентах.

Основным пользователем данной системы является секретарь приемной комиссии, который регистрирует абитуриентов. Для каждого абитуриента в базу данных заносятся следующие сведения: фамилия, имя, отчество, паспортные данные, какое учебное заведение, где и когда окончил, наличие золотой или серебряной медали, название специальности, на которые поступает абитуриент. При подаче заявления абитуриент указывает форму обучения (очная, очно-заочная (вечерняя), заочная), поступление на бюджет или контракт. Абитуриент может поступать вне конкурса (инвалиды, сироты). Также существуют абитуриенты-целевики, которые поступают по договорам с направляющими организациями, и обучаются на коммерческой основе.

Абитуриенты, поступающие на базе 9 классов, участвуют в конкурсе аттестатов. Для них указывается информация по 4-м профильным дисциплинам и средний балл по всем остальным дисциплинам аттестата. На основе этих данных строится рейтинг абитуриентов.

Абитуриенты, поступающие на базе 11 классов, предоставляют сертификаты ЕГЭ по 2 дисциплинам, на основе чего строится рейтинг абитуриентов.

Конкурс для абитуриентов на базе 9 и 11 классов отдельный, т.к. они поступают на разные курсы.

Абитуриент может не только подать, но и забрать документы, а также перевести их на другую специальность.

Известно количество мест на каждый факультет. Приемная комиссия по результатам экзаменов должна сформировать списки абитуриентов, зачисленных в колледж.

Секретарю приемной комиссии могут потребоваться следующие сведения:

- Список абитуриентов, подавших заявление на заданную специальность.
- Количество абитуриентов, подавших заявления на каждую специальность по каждой форме обучения на бюджет (или контракт).
- Количество абитуриентов на базе 9 и 11 классов, поступающих на бюджет (или контракт).
- Общее количество поданных заявлений ежедневно.

- Конкурс на каждую специальность по каждой форме обучения на бюджет.

Необходимо предусмотреть возможность получения документа, представляющего собой сгруппированный по заданной специальности список абитуриентов по заданной форме обучения, зачисленных в колледж, с указанием набранных ими баллов по аттестату. Отчет должен содержать проходной балл по специальности в целом, а также количество абитуриентов, поступающих на специальность.

# **1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

Предметной областью для курсовой работы является приемная кампания колледжа. Поступать в колледж могут абитуриенты как на базе 9 классов по оценкам в школьном аттестате, так и на базе 11 классов по результатам ЕГЭ. Существует несколько форм обучения в колледже: заочная, очная, очно-заочная; так же есть разные основы обучения: бюджет и контракт, отдельно имеется целевой прием.

Процедура регистрации абитуриента происходит следующим образом: сначала заполняется личный профиль с указанием персональных данных: фамилия, имя, отчество, паспортные данные, какое учебное заведение, где и когда окончил, наличие золотой или серебряной медали; затем формируется заявка на определенную специальность колледжа с указанием формы и основы обучения.

Результатом работы приемной кампании должен быть документ, или сводная таблица, со списком абитуриентов, специальностями, на которые каждый абитуриент подавал документы и статус заявки: зачислен, не зачислен.

Таким образом, функциональные требования к web-приложению должны отвечать всем установленным запросам и максимально удовлетворять потребностям предметной области. Все функциональные требования можно разделить на несколько групп по виду манипулирования информацией, а именно: блок требований, касающийся функций просмотра информации, блок требований для функций добавления информации, блок требований для функций редактирования информации, а также удаления информации в базе данных. Кроме того, отдельной группой являются такие функциональные требования как авторизация, регистрация и выполнение запросов.

Таким образом, были выявлены следующие функциональные требования:

## **1. Функции просмотра**

- a. Информации об абитуриентах
- b. Информации о сертификатах ЕГЭ, поданных абитуриентами
- c. Информации об аттестатах, поданных абитуриентами
- d. Информации о заявках на поступление в колледж
- e. Информации о факультетах, которые есть в колледже
- f. Информации о специальностях, которые есть в колледже
- g. Результаты приемной кампании

2. Функции добавления
  - a. Регистрация новых абитуриентов
  - b. Добавление аттестата, который предоставил абитуриент
  - c. Добавление оценок из аттестата
  - d. Добавление сертификата ЕГЭ, который предоставил абитуриент
  - e. Добавление баллов из сертификата ЕГЭ
3. Функции редактирования
  - a. Профиля абитуриента
  - b. Заявки на поступление в колледж
4. Функции удаления
  - a. Удаление абитуриента
  - b. Удаление заявки
5. Функции регистрации и авторизации
6. Функции выполнения запросов

## 2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ

Первым этапом создания web-приложения было проектирование схемы базы данных, представленной на рисунке 1.

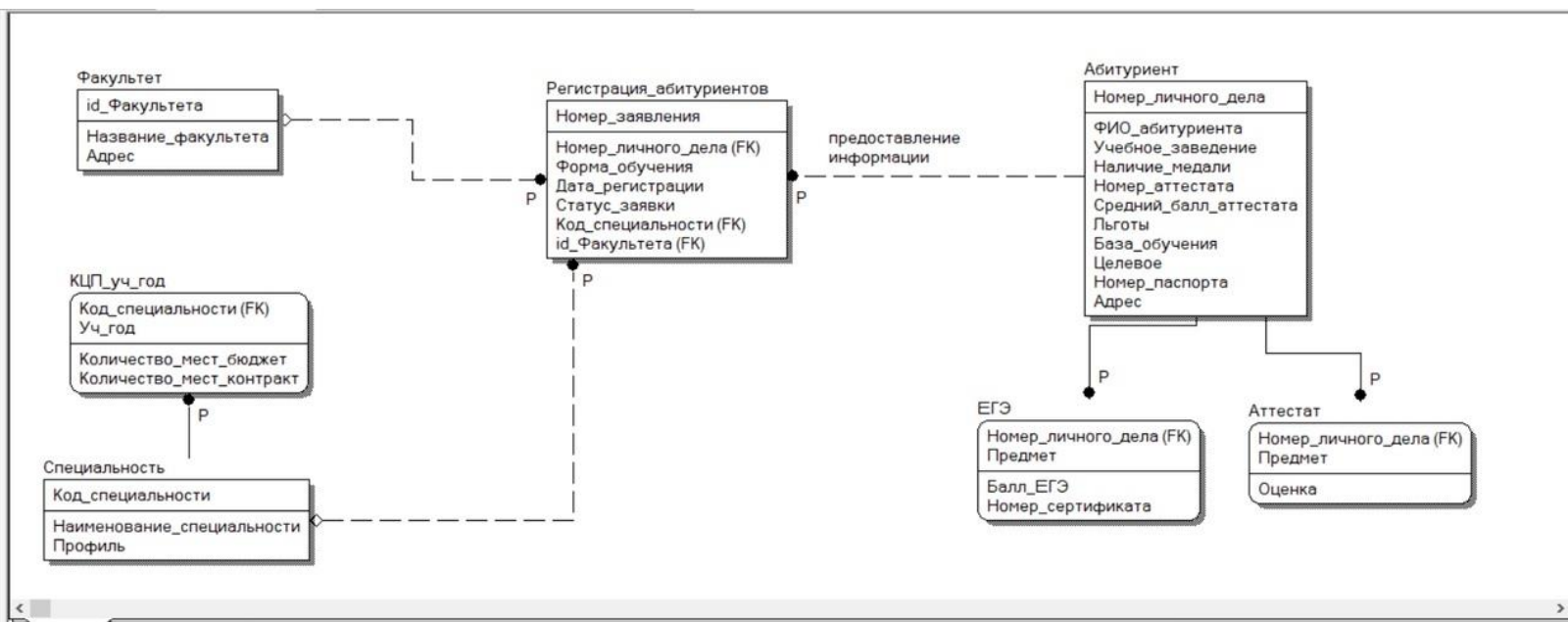


Рисунок 1 – схема базы данных

Для реализации серверной части web-приложения были использованы следующие средства разработки:

- база данных – PostgreSQL
- web-фреймворк Django REST языка программирования Python

Следующим этапом создания web-сервиса была реализация созданной схемы базы данных в качестве моделей Django в файле models.py, представленном в приложении 1. Были созданы следующие модели:

- class Faculty – Факультет. Модель содержит поля: название факультета, адрес деканата факультета.
- class Specialty – Специальность. Модель содержит поля: название специальности, факультет, к которому относится специальность, количество мест на бюджет, количество мест на контракт.
- class Enrollee – Абитуриент. Модель содержит поля: фамилия, имя, отчество (ФИО), учебное заведение, которое окончил абитуриент, дата окончания учебного заведения, отсутствие или наличие медали: золотой или серебряной, номер паспорта абитуриента, адрес, отсутствие или наличие льгот: инвалид, сирота; факт целевого приема.

- class Application – Заявка. Модель содержит поля: абитуриент, факультет, специальность, дата подачи заявки, статус заявки: зачислен, в очереди, не зачислен; форма обучения: очная, очно-заочная, заочная; основа обучения: контракт или бюджет.
- class EGE – Сертификат ЕГЭ. Модель содержит поля: абитуриент и предметы с баллами.
- class EgeSubject – Предмет из сертификата ЕГЭ. Модель содержит поля: сертификат ЕГЭ, к которому относятся баллы, предмет и сами баллы за этот предмет.
- class Attestat – Аттестат. Модель содержит поля: абитуриент, средний балл аттестата и предметы с баллами.
- class AttestatSubject – Предмет из аттестата. Модель содержит поля: аттестат, к которому относятся оценки, предмет и сама оценка за этот предмет.

Далее были созданы сериализаторы для обеспечения обмена данными между серверной частью, написанной на Django REST Framework, и клиентской частью, написанной на Vue.js. Листинг кода файла serializers.py, в котором описаны все необходимые сериализаторы, представлен в приложении 2. Пример сериализатора для отображения модели «Заявка» изображен на рисунке 2.

```
class ApplicationSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Заявка"""
    enrollee = serializers.SlugRelatedField(slug_field="fio", read_only=True)
    specialty = serializers.SlugRelatedField(slug_field="name", read_only=True)
    faculty = serializers.SlugRelatedField(slug_field="name", read_only=True)

    class Meta:
        model = Application
        fields = "__all__"
```

Рисунок 2 – сериализатор для модели «Заявка»

Следующим этапом было создание отображений, для этого были использованы классы ViewSet и APIView. Листинг кода файла views.py, в котором описаны все созданные классы отображений, представлен в приложении 3. Пример отображения, созданного с помощью класса ViewSet, продемонстрирован на рисунке 3.

```
class EnrolleeViewSet(viewsets.ModelViewSet):
    """Отображение для модели Абитуриент"""
    queryset = Enrollee.objects.all()

    def get_serializer_class(self):
        if self.action == 'create':
            return EnrolleeSerializer
        elif self.action != 'create':
            return EnrolleeDetailSerializer
```

Рисунок 3 – отображение для модели «Абитуриент»



Пример отображения, созданного с помощью класса APIView, представлен на рисунке

4.

```
class Query1(APIView):  
    """Список абитуриентов, подавших заявление на заданную специальность"""  
  
    def get(self, request):  
        specialty = request.GET.get('specialty')  
        enrollee_list = Application.objects.filter(specialty=specialty)  
        serializer = ApplicationSerializer(enrollee_list, many=True)  
        return Response({'result': serializer.data})
```

Рисунок 4 – отображение запроса

Последним этапом написания серверной части было создание путей, прописанных в файле urls.py/

Результат работы серверной части web-приложения можно проверить в панели Django REST. Некоторые из реализованных интерфейсов представлены далее:

#### 1. Регистрация нового пользователя

Django REST framework Log in

## User Create OPTIONS

Use this endpoint to register new user.

GET /auth/users/create/

HTTP 405 Method Not Allowed  
Allow: POST, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{  
  "detail": "Метод \"/auth/users/create/
```

Raw data HTML form

Адрес электронной почты

Имя пользователя   
Обязательное поле. Не более 150 символов. Только буквы, цифры и символы @/./+/\_-

Password

POST

Рисунок 5 –регистрация нового пользователя

## 2. Список абитуриентов с их заявлениями на обучение в колледже



Рисунок 6 –вывод абитуриентов

## 3. Факультеты и специальности

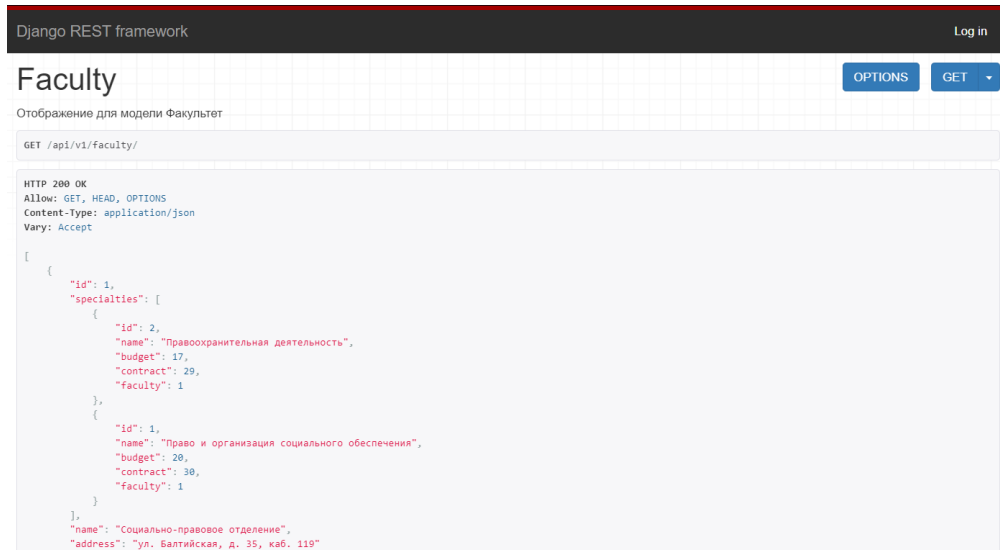


Рисунок 7 –вывод факультетов и специальностей

## 4. Создание заявки

Raw data HTML form

Дата регистрации заявки	<input type="text" value="ДД.ММ.ГГГГ"/>
Статус заявки	<input type="text" value="-----"/>
Форма обучения	<input type="text" value="-----"/>
Поступление на	<input type="text" value="-----"/>
Enrollee	<input type="text" value="Степанов Иван Сергеевич"/>
Specialty	<input type="text" value="Реклама"/>
Faculty	<input type="text" value="Социально-правовое отделение"/>

POST

Рисунок 8 –создание заявки

## 5. Сертификат ЕГЭ с предметами и баллами

Ege

CRUD для модели ЕГЭ

OPTIONS

GET

GET /api/v1/ege/1/

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

{
  "id": 1,
  "enrollee": "Ципленкова Анна Юрьевна",
  "marks": [
    {
      "id": 1,
      "subject": "Математика",
      "mark": 90,
      "ege": 1
    },
    {
      "id": 2,
      "subject": "Информатика",
      "mark": 99,
      "ege": 1
    }
  ]
}
```

Рисунок 9 –вывод определенного сертификата ЕГЭ

## 6. Аттестат с предметами и оценками

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

{
  "id": 3,
  "enrollee": "Степанов Иван Сергеевич",
  "marks": [
    {
      "id": 6,
      "subject": "Русский язык",
      "mark": 4,
      "attestat": 3
    },
    {
      "id": 7,
      "subject": "Физика",
      "mark": 4,
      "attestat": 3
    },
    {
      "id": 8,
      "subject": "Химия",
      "mark": 4,
      "attestat": 3
    },
    {
      "id": 9,
      "subject": "Математика",
      "mark": 4,
      "attestat": 3
    }
  ],
  "average": 3.9
}
```

Рисунок 10 –вывод определенного аттестата

### 3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ

Клиентская часть была реализована с помощью web-фреймворка Vue.js языка программирования JavaScript и его библиотеки MUSE-UI. Были получены следующие интерфейсы:

#### 1. Главная страница

Главная страница web-сервиса содержит информацию о колледже. При нажатии на кнопку «Зарегистрировать абитуриента» появляется форма добавления записи в модель «Абитуриент» в базе данных. Скриншоты представлены на рисунках 10 и 11.

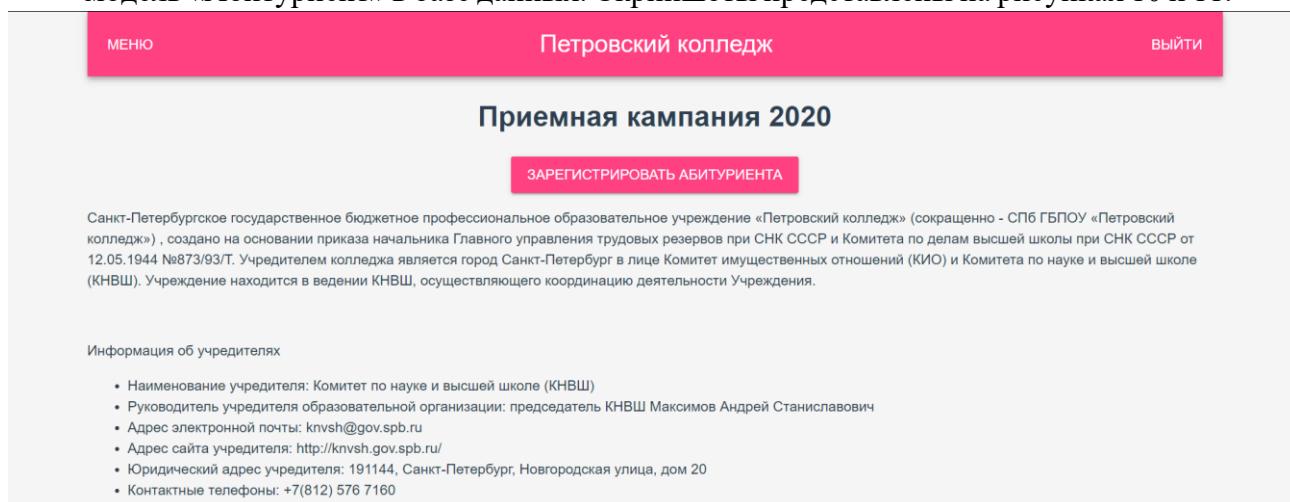


Рисунок 10 –главная страница

Рисунок 11 –форма регистрации абитуриента

## 2. Вход

Страница входа имеет поля для ввода имени пользователя и пароля. При нажатии на кнопку «Войти» данные передаются на сервер Django и в случае корректно введенных данных пользователь перенаправляется на главную страницу, в обратном случае – появляется уведомление о том, что данные введены неверно. При нажатии на активную ссылку «Регистрация» пользователь перенаправляется на страницу регистрации. Скриншот представлен на рисунке 12.

МЕНЮ Петровский колледж ВОЙТИ

Имя пользователя

Пароль

Пожалуйста, введите пароль

ВОЙТИ

Регистрация

Рисунок 12 –форма входа

## 3. Регистрация

Страница регистрации имеет поля для ввода имени нового пользователя и пароля. При правильном вводе данных новый пользователь добавляется в базу данных и происходит перенаправление на страницу входа, если данные введены неверно, то появляется соответствующее предупреждение. Скриншот представлен на рисунке 13.

МЕНЮ Петровский колледж ВОЙТИ

Пожалуйста, заполните следующие поля

Поля должны содержать не менее 8 символов.

Username

Password

Password again

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 13 –форма регистрации

#### 4. Факультеты и специальности

Страница со списком факультетов и их специальностей. Таблица под каждой специальностью содержит информацию о количестве бюджетных и контрактных мест. Скриншот представлен на рисунке 14.

МЕНЮ

Петровский колледж

ВЫЙТИ

Факультеты и специальности

- Социально-правовое отделение
  - Правоохранительная деятельность

Приемная кампания 2020	
Количество бюджетных мест	17
Количество контрактных мест	29
  - Право и организация социального обеспечения

Приемная кампания 2020	
Количество бюджетных мест	20
Количество контрактных мест	30
- Отделение международных программ, туризма и сервиса
  - Реклама

Приемная кампания 2020	
Количество бюджетных мест	25
Количество контрактных мест	40

Рисунок 14 –Факультеты и специальности

#### 5. Абитуриенты

Страница содержит список абитуриентов, при нажатии на определенного абитуриента происходит перенаправление на его личную страницу. При нажатии на кнопку «Абитуриент принес документы» появляется форма, аналогичная той, которая на главной странице. При нажатии на кнопку «Абитуриент забрал документы» под каждым абитуриентом появляется опция выбора, после нажатия на кнопку «Подтвердить удаление» выбранный абитуриент удаляется из базы данных. Скриншоты представлены на рисунках 15 и 16.

МЕНЮ

Петровский колледж

ВЫЙТИ

Список всех абитуриентов

Степанов Иван Сергеевич

Цыпленкова Анна Юрьевна

Тереник Михаил Владимирович

Овчинникова Елена Андреевна

АБИТУРИЕНТ ПРИНЕС ДОКУМЕНТЫ

АБИТУРИЕНТ ЗАБРАЛ ДОКУМЕНТЫ

Рисунок 15 –Абитуриенты

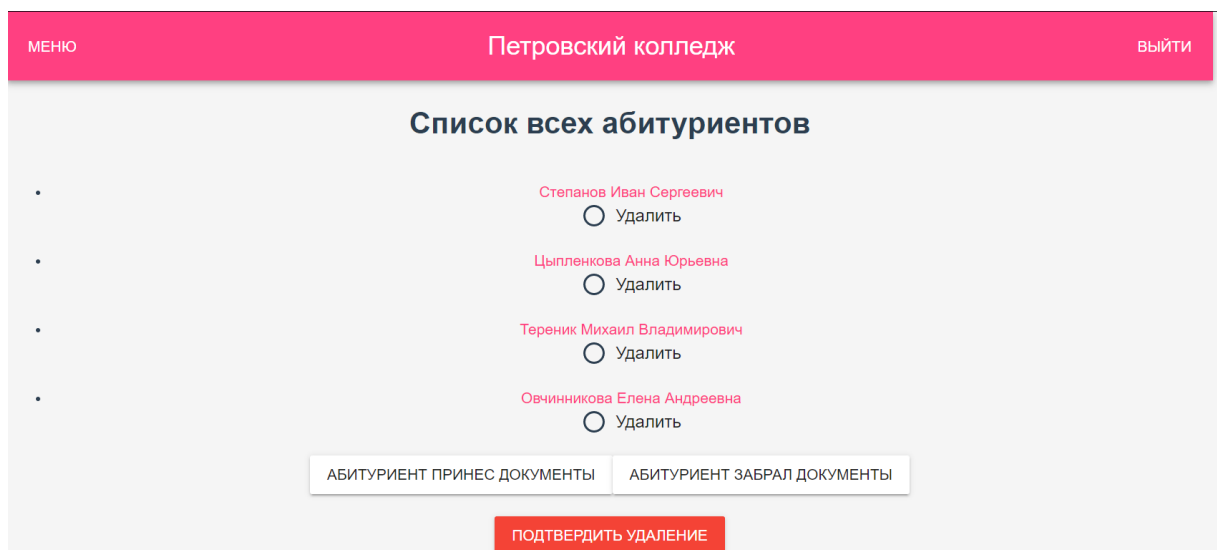


Рисунок 16 –удаление выбранного абитуриента

## 6. Профиль абитуриента

Страница содержит подробную информацию об абитуриенте. При нажатии на кнопку «Показать заявки» появляется таблица с заявками, которые подал этот абитуриент. При нажатии на «Абитуриент забрал документы» данный абитуриент удаляется из базы данных и происходит перенаправление на страницу «Абитуриенты». При нажатии на кнопку «Изменить данные об абитуриенте» появляется форма аналогичная форме добавления на главной странице. При нажатии на «Добавить заявку на поступление» появляется форма создания заявки.

Скриншоты представлены на рисунках 17 и 18.

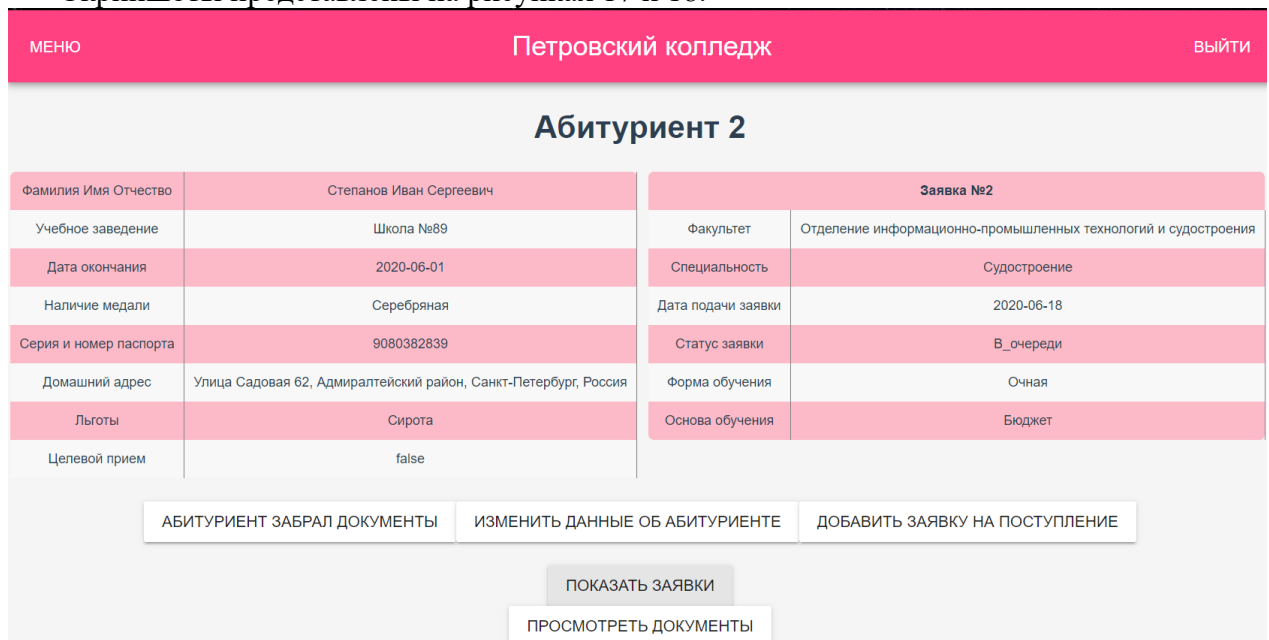


Рисунок 17 –профиль выбранного абитуриента

АБИТУРИЕНТ ЗАБРАЛ ДОКУМЕНТЫ
ИЗМЕНИТЬ ДАННЫЕ ОБ АБИТУРИЕНТЕ
ДОБАВИТЬ ЗАЯВКУ НА ПОСТУПЛЕНИЕ

ПОКАЗАТЬ ЗАЯВКИ
ПРОСМОТРЕТЬ ДОКУМЕНТЫ

Дата регистрации заявки

Статус заявки

Форма обучения

Основа обучения

Факультет

Специальность

ВНЕСТИ ЗАЯВКУ

Рисунок 18 –форма создания заявки на поступление

## 7. Документы

Страница с информацией о документах, которые абитуриенты предоставили при поступлении: аттестат или сертификат ЕГЭ. При нажатии на кнопку «Добавить аттестат/сертификат ЕГЭ» появляется форма с добавлением соответствующего документа. Скриншот представлен на рисунке 19.

МЕНЮ
Петровский колледж
ВЫЙТИ

### Документы, предоставляемые абитуриентами

#### Аттестаты

- Аттестат №3  
Абитуриент - Степанов Иван Сергеевич  
Средний балл - 3.9

Оценки	
Русский язык	4
Физика	4
Химия	4
Математика	4
- Аттестат №4  
Абитуриент - Овчинникова Елена Андреевна  
Средний балл - 4.6

Оценки	
Математика	5
Русский язык	5
Обществознание	5
История	4

ДОБАВИТЬ АТТЕСТАТ

#### Сертификаты ЕГЭ

- Сертификат ЕГЭ №1  
Абитуриент - Цыпленкова Анна Юрьевна

Результаты	
Математика	90
Информатика	99
- Сертификат ЕГЭ №2  
Абитуриент - Тереник Михаил Владимирович

Результаты	
Русский язык	89
История	78

ДОБАВИТЬ СЕРТИФИКАТ ЕГЭ

Рисунок 19 –Документы



## 8. Заявки

Страница содержит список заявок с информацией об абитуриенте, специальности и статусе заявки. Каждая заявка является активной ссылкой, при нажатии на которую происходит перенаправление на страницу с ее подробным описанием. При нажатии на кнопку «Добавить заявку» появляется форма создания заявки. При нажатии на «Удалить заявку» появляется опция выбора под каждой заявкой, после нажатия на кнопку «Подтвердить удаление» выбранная заявка удаляется из базы данных, и страница автоматически обновляется. Скриншоты представлены на рисунках 20, 21 и 22.

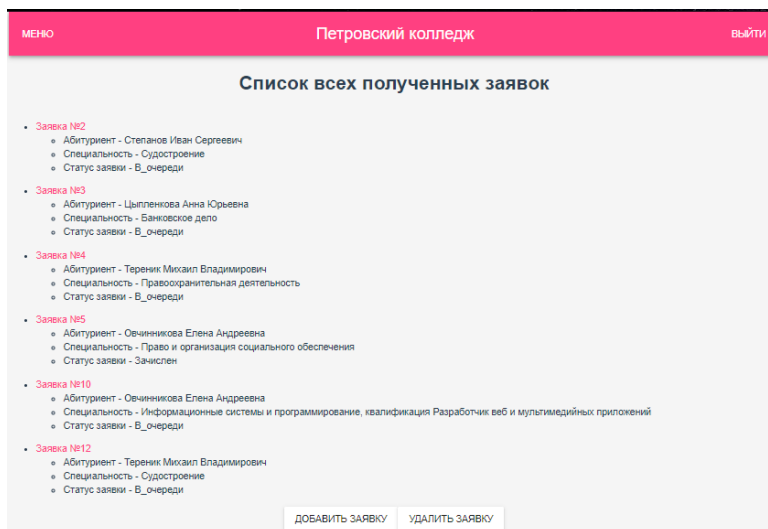


Рисунок 20 –Заявки

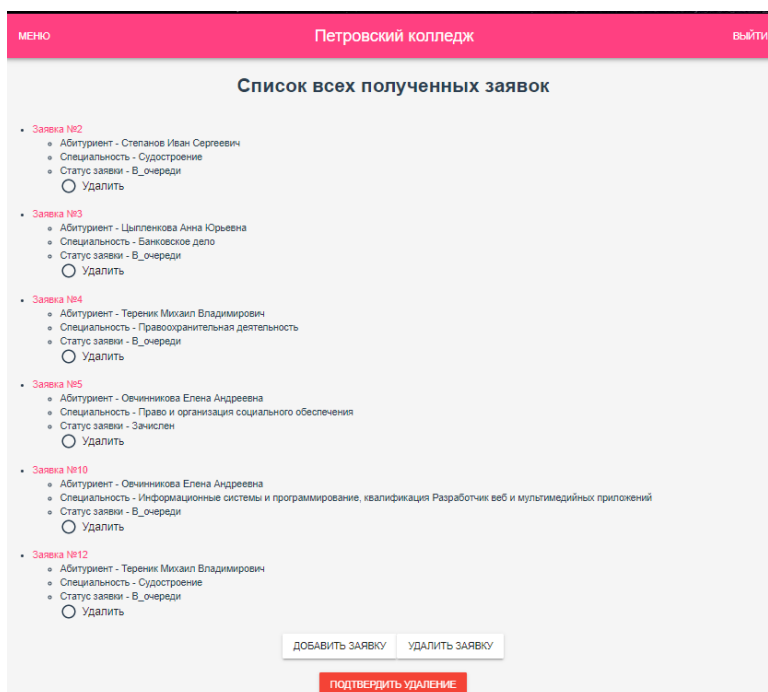


Рисунок 21 –удаление выбранной заявки

Заявка №12

- Абитуриент - Тереник Михаил Владимирович
- Специальность - Судостроение
- Статус заявки - В\_очереди

ДОБАВИТЬ ЗАЯВКУ
УДАЛИТЬ ЗАЯВКУ

Дата регистрации заявки

Статус заявки

Форма обучения

Основа обучения

Абитуриент

Факультет

Специальность

ВНЕСТИ ЗАЯВКУ

Рисунок 22 –создание новой заявки

## 9. Заявка

Страница с подробным описанием выбранной заявки. При нажатии на кнопку «Удалить заявку» данная заявка удаляется и происходит перенаправление на страницу «Заявки». При нажатии на кнопку «Изменить заявку» появляется форма с полями ввода для изменения даты регистрации заявки. После заполнения всех полей при нажатии на кнопку «Внести изменения» изменения вносятся в базу данных и страница автоматически обновляется. Скриншоты представлены на рисунках 23 и 24.

МЕНЮ
Петровский колледж
ВЫЙТИ

### Заявка №3

Статус	В_очереди
ФИО абитуриента	Цыпленкова Анна Юрьевна
Факультет	Отделение экономики и финансов
Специальность	Банковское дело
Форма обучения	Очная
Бюджет/Контракт	Бюджет
Дата подачи заявки	2020-06-18

УДАЛИТЬ ЗАЯВКУ
ИЗМЕНИТЬ ЗАЯВКУ

Рисунок 23 –Заявка

МЕНЮ
Петровский колледж
ВЫЙТИ

### Заявка №3

Статус	В_очереди
ФИО абитуриента	Цыпенкова Анна Юрьевна
Факультет	Отделение экономики и финансов
Специальность	Банковское дело
Форма обучения	Очная
Бюджет/Контракт	Бюджет
Дата подачи заявки	2020-06-18

УДАЛИТЬ ЗАЯВКУ
ИЗМЕНИТЬ ЗАЯВКУ

Дата регистрации заявки

Статус заявки
▼

Форма обучения
▼

Основа обучения
▼

ВНЕСТИ ИЗМЕНЕНИЯ

Рисунок 24 –форма изменения заявки

## 10. Запросы

### а. Запрос 1

Список абитуриентов, подавших заявление на заданную специальность.

Результат выполнения запроса представлен на рисунке 25.

МЕНЮ
Петровский колледж
ВЫЙТИ

### Запросы к курсовой работе

Секретарю приемной комиссии могут потребоваться следующие сведения:

- Запрос №1

Список абитуриентов, подавших заявление на заданную специальность

Специальность
Банковское дело
▼

ВЫПОЛНИТЬ ЗАПРОС

Результат выполнения запроса

Цыпенкова Анна Юрьевна

Рисунок 25 –Запрос 1

b. Запрос 2.

Количество абитуриентов, подавших заявления на каждую специальность по каждой форме обучения на бюджет (или контракт). Результат выполнения запроса представлен на рисунке 26.

Запрос №2

Количество абитуриентов, подавших заявления на каждую специальность по каждой форме обучения на бюджет (или контракт)

Код специальности	Основа обучения	Количество абитуриентов
1	Контракт	1
2	Бюджет	1
5	Бюджет	1
6	Бюджет	2
8	Бюджет	1

Код специальности	Название специальности
3	Реклама
4	Гостиничное дело
8	Банковское дело
7	Страховое дело
6	Судостроение
5	Информационные системы и программирование, квалификация Разработчик веб и мультимедийных приложений
2	Правоохранительная деятельность
1	Право и организация социального обеспечения

Рисунок 26 –Запрос 2

c. Запрос 3.

Количество абитуриентов на базе 9 и 11 классов, поступающих на бюджет (или контракт). Результат выполнения запроса представлен на рисунке 27.

Запрос №3

Количество абитуриентов на базе 9 и 11 классов, поступающих на бюджет (или контракт).

Основа обучения	Количество абитуриентов
Бюджет	5
Контракт	1

Рисунок 27 –Запрос 3

d. Запрос 4.

Общее количество поданных заявлений ежедневно. Результат выполнения запроса представлен на рисунке 28.

Запрос №4

Общее количество поданных заявлений ежедневно

Дата	Количество заявлений
2019-12-12	1
2020-06-18	3
2020-06-19	1
2020-06-28	1

Рисунок 28 –Запрос 4

е. Запрос 5.

Конкурс на каждую специальность по каждой форме обучения на бюджет.

Результат выполнения запроса представлен на рисунке 29.

• Запрос №5

Конкурс на каждую специальность по каждой форме обучения на бюджет

Специальность	Количество бюджетных мест
Реклама	25
Гостиничное дело	30
Банковское дело	60
Страховое дело	20
Судостроение	43
Информационные системы и программирование, квалификация Разработчик веб и мультимедийных приложений	50
Правоохранительная деятельность	17
Право и организация социального обеспечения	20

Рисунок 29 –Запрос 5

11. Отчет

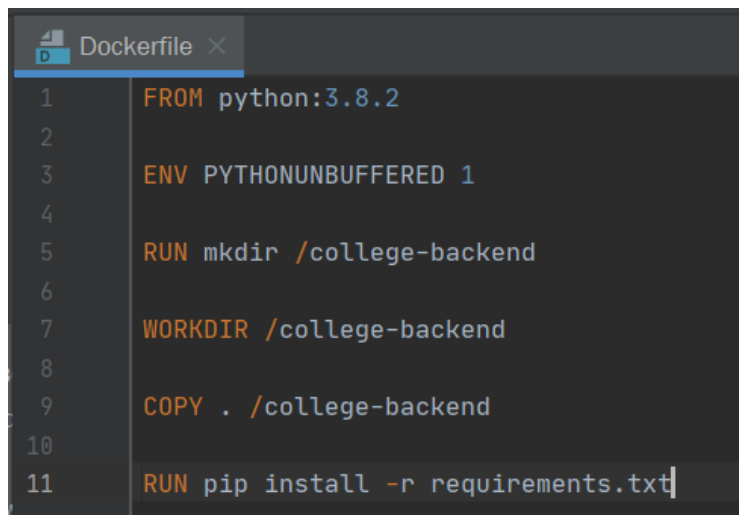
Отчет по результатам приемной кампании				
Факультет	Специальность	Основа обучения	Абитуриент	Статус
Отделение информационно-промышленных технологий и судостроения	Судостроение	Бюджет	Степанов Иван Сергеевич	В_очереди
Отделение экономики и финансов	Банковское дело	Бюджет	Цыпленкова Анна Юрьевна	В_очереди
Социально-правовое отделение	Правоохранительная деятельность	Бюджет	Тереник Михаил Владимирович	В_очереди
Социально-правовое отделение	Право и организация социального обеспечения	Контракт	Овчинникова Елена Андреевна	Зачислен
Отделение информационно-промышленных технологий и судостроения	Информационные системы и программирование, квалификация Разработчик веб и мультимедийных приложений	Бюджет	Овчинникова Елена Андреевна	В_очереди
Отделение информационно-промышленных технологий и судостроения	Судостроение	Бюджет	Тереник Михаил Владимирович	В_очереди

Рисунок 30 –Отчет

## 4. ИСПОЛЬЗОВАНИЕ DOCKER ДЛЯ РАЗВЕРТЫВАНИЯ WEB-ПРИЛОЖЕНИЙ

Docker - это ПО с открытым исходным кодом, который упрощает создание контейнеров и приложений на основе контейнеров. Первоначально разработанный для Linux, Docker теперь работает также на Windows и MacOS. Чтобы понять, как работает Docker, нужно рассмотреть компоненты, которые используются для создания контейнеризованных приложений.

Каждый контейнер Docker начинается с Dockerfile – это текстовый файл, который включает инструкции по созданию образа Docker. Dockerfile определяет операционную систему, которая будет лежать в основе контейнера, а также языки, переменные среды, расположение файлов, сетевые порты, необходимые библиотеки и действия контейнера после его запуска. Dockerfile для бэкенда реализованного web-приложения представлен на рисунке 31.



```
1 FROM python:3.8.2
2
3 ENV PYTHONUNBUFFERED 1
4
5 RUN mkdir /college-backend
6
7 WORKDIR /college-backend
8
9 COPY . /college-backend
10
11 RUN pip install -r requirements.txt
```

Рисунок 31 –Dockerfile

Далее создается файл docker-compose, который используется для управления несколькими контейнерами, входящими в состав приложения.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы по дисциплине «Основы web-программирования» была создана программная система, предназначенная для секретаря приемной комиссии колледжа. Главным интерфейсом разрабатываемого web-приложения стала регистрация абитуриентов, их документов и заявок, которые они подают на поступление в колледж. Для двух основных моделей: «Абитуриент» и «Заявка» - были реализованы все четыре базовые функции, используемые при работе с базами данных: создание, чтение, модификация, удаление.

Стек используемых технологий включал в себя:

- PostgreSQL – для работы с базой данных
- Django и Django REST Framework – web-фреймворк языка программирования Python для создания web-приложений
- Vue.js – web-фреймворк языка программирования JavaScript для создания пользовательских интерфейсов
- MUSE-UI – библиотека Vue.js для дизайна пользовательского интерфейса, основанная на Material Design
- Docker – для автоматизации разворачивания приложения

В качестве дальнейшей разработки данного web-приложения можно было бы поработать над улучшением дизайна, добавлением новых функций (например, фильтрация) и расширить базу факультетов и специальностей.

## СПИСОК ЛИТЕРАТУРЫ

1. Django Rest Framework. Документация Django Rest Framework [Электронный ресурс]. URL: <https://www.django-rest-framework.org> (дата обращения: 29.06.2020).
2. 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).
3. 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).
4. Vue.js. Документация Vue.js [Электронный ресурс]. URL: <https://vuejs.org> (дата обращения: 29.06.2020).



# **ПРИЛОЖЕНИЯ**

```

from django.db import models

class Faculty(models.Model):
    name = models.CharField('Название факультета', max_length=100)
    address = models.CharField('Адрес деканата факультета', max_length=100)

    class Meta:
        verbose_name = 'Факультет'
        verbose_name_plural = 'Факультеты'

    def __str__(self):
        return self.name

class Specialty(models.Model):
    name = models.CharField('Название специальности', max_length=100)
    faculty = models.ForeignKey(Faculty, on_delete=models.CASCADE, related_name='specialties')
    budget = models.IntegerField('Количество мест на бюджет', null=True)
    contract = models.IntegerField('Количество мест на контракт', null=True)

    class Meta:
        verbose_name = 'Специальность'
        verbose_name_plural = 'Специальности'

    def __str__(self):
        return self.name

class Enrollee(models.Model):
    fio = models.CharField('ФИО', max_length=200)
    school = models.CharField('Учебное заведение', max_length=100)
    finish_school = models.DateField('Дата окончания учебного заведения')
    medal_type = models.TextChoices('medal_type', 'Золотая Серебряная Отсутствует')
    medal = models.CharField('Медаль', blank=True, choices=medal_type.choices, max_length=100)
    passport_number = models.CharField('Номер паспорта', max_length=10)
    address = models.CharField('Адрес', max_length=200)
    privileges_type = models.TextChoices('privileges_type', 'Инвалид Сирота Нет')
    privileges = models.CharField('Льготы', blank=True, choices=privileges_type.choices, max_length=100)
    target = models.BooleanField('Целевой прием')

    class Meta:
        verbose_name = 'Абитуриент'
        verbose_name_plural = 'Абитуриенты'

    def __str__(self):
        return self.fio

class Application(models.Model):
    enrollee = models.ForeignKey(Enrollee, on_delete=models.CASCADE, related_name='apps')
    specialty = models.ForeignKey(Specialty, on_delete=models.CASCADE)
    faculty = models.ForeignKey(Faculty, on_delete=models.CASCADE)
    date = models.DateField('Дата регистрации заявки')
    status_type = models.TextChoices('status_type', 'Зачислен В_очереди Не_зачислен')
    status = models.CharField('Статус заявки', blank=True, choices=status_type.choices, max_length=100)

```

```

form_type = models.TextChoices('form_type', 'Очная Очно-заочная Заочная')
form_types = models.CharField('Форма обучения', blank=True, choices=form_type.choices, max_length=100,
default='Очная')
budget_type = models.TextChoices('budget_type', 'Бюджет Контракт')
form = models.CharField('Поступление на', blank=True, choices=budget_type.choices, max_length=100)

class Meta:
    verbose_name = 'Заявка'
    verbose_name_plural = 'Заявки'

class EGE(models.Model):
    enrollee = models.OneToOneField(Enrollee, on_delete=models.CASCADE, related_name='ege')

    class Meta:
        verbose_name = 'Сертификат ЕГЭ'
        verbose_name_plural = 'Сертификаты ЕГЭ'

class EgeSubject(models.Model):
    ege = models.ForeignKey(EGE, on_delete=models.CASCADE, related_name='marks')
    subject = models.CharField('Дисциплина', max_length=50)
    mark = models.IntegerField('Балл')

    class Meta:
        verbose_name = 'Дисциплина ЕГЭ'
        verbose_name_plural = 'Дисциплины ЕГЭ'

class Attestat(models.Model):
    enrollee = models.OneToOneField(Enrollee, on_delete=models.CASCADE, related_name='attestat')
    average = models.FloatField('Средний балл аттестата')

    class Meta:
        verbose_name = 'Аттестат'
        verbose_name_plural = 'Аттестаты'

class AttestatSubject(models.Model):
    attestat = models.ForeignKey(Attestat, on_delete=models.CASCADE, related_name='marks')
    subject = models.CharField('Дисциплина', max_length=50)
    mark = models.IntegerField('Балл')

    class Meta:
        verbose_name = 'Дисциплина аттестата'
        verbose_name_plural = 'Дисциплины аттестата'
# Create your models here.

```

## Приложение 2. Файл serializers.py

```
from rest_framework import serializers
from .models import Faculty, Specialty, Enrollee, Application, EGE, EgeSubject, Attestat, AttestatSubject

class AttestatSubjectSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Дисциплина аттестата"""
    class Meta:
        model = AttestatSubject
        fields = "__all__"

class AttestatSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Аттестат"""
    enrollee = serializers.SlugRelatedField(slug_field="fio", read_only=True)
    marks = AttestatSubjectSerializer(many=True)

    class Meta:
        model = Attestat
        fields = "__all__"

class AttestatCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для добавления нового аттестата"""

    class Meta:
        model = Attestat
        fields = "__all__"

class EgeSubjectSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Дисциплина ЕГЭ"""

    class Meta:
        model = EgeSubject
        fields = "__all__"

class EgeSerializer(serializers.ModelSerializer):
    """Сериализатор для модели ЕГЭ"""
    enrollee = serializers.SlugRelatedField(slug_field="fio", read_only=True)
    marks = EgeSubjectSerializer(many=True)

    class Meta:
        model = EGE
        fields = "__all__"

class EgeCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для добавления нового сертификата ЕГЭ"""

    class Meta:
        model = EGE
        fields = "__all__"

class ApplicationSerializer(serializers.ModelSerializer):
```

```

"""Сериализатор для модели Заявка"""
enrollee = serializers.SlugRelatedField(slug_field="fio", read_only=True)
specialty = serializers.SlugRelatedField(slug_field="name", read_only=True)
faculty = serializers.SlugRelatedField(slug_field="name", read_only=True)

class Meta:
    model = Application
    fields = "__all__"

class ApplicationCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для добавления новой заявки"""
    class Meta:
        model = Application
        fields = "__all__"

class EnrolleeSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Абитуриент"""
    class Meta:
        model = Enrollee
        fields = "__all__"

class EnrolleeDetailSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Абитуриент"""
    apps = ApplicationSerializer(many=True)

    class Meta:
        model = Enrollee
        fields = "__all__"

class SpecialtySerializer(serializers.ModelSerializer):
    """Сериализатор для модели Специальность"""
    class Meta:
        model = Specialty
        fields = "__all__"

class FacultySerializer(serializers.ModelSerializer):
    """Сериализатор для модели Факультет"""
    specialties = SpecialtySerializer(many=True)

    class Meta:
        model = Faculty
        fields = "__all__"

```

```

from django.shortcuts import render
from rest_framework import viewsets
from rest_framework.views import APIView
from rest_framework.response import Response
from collections import Counter
from django.db.models import Count, Avg
from .models import Faculty, Specialty, Enrollee, Application, EGE, EgeSubject, Attestat, AttestatSubject
from .serializers import FacultySerializer, SpecialtySerializer, EnrolleeSerializer, ApplicationSerializer, \
    ApplicationCreateSerializer, EgeSerializer, EgeCreateSerializer, AttestatSerializer, AttestatCreateSerializer, \
    EgeSubjectSerializer, AttestatSubjectSerializer, EnrolleeDetailSerializer

class AttestatViewSet(viewsets.ModelViewSet):
    """CRUD для модели Аттестат"""
    queryset = Attestat.objects.all()

    def get_serializer_class(self):
        if self.action == 'create':
            return AttestatCreateSerializer
        elif self.action != 'create':
            return AttestatSerializer

class EgeViewSet(viewsets.ModelViewSet):
    """CRUD для модели ЕГЭ"""
    queryset = EGE.objects.all()

    def get_serializer_class(self):
        if self.action == 'create':
            return EgeCreateSerializer
        elif self.action != 'create':
            return EgeSerializer

class ApplicationViewSet(viewsets.ModelViewSet):
    """CRUD для модели Заявка"""
    queryset = Application.objects.all()

    def get_serializer_class(self):
        if self.action == 'create':
            return ApplicationCreateSerializer
        elif self.action != 'create':
            return ApplicationSerializer

class EnrolleeViewSet(viewsets.ModelViewSet):
    """Отображение для модели Абитуриент"""
    queryset = Enrollee.objects.all()

    def get_serializer_class(self):
        if self.action == 'create':
            return EnrolleeSerializer
        elif self.action != 'create':
            return EnrolleeDetailSerializer

```

```

class EgeSubjectViewSet(viewsets.ModelViewSet):
    """Отображение для модели Дисциплина ЕГЭ"""
    queryset = EgeSubject.objects.all()
    serializer_class = EgeSubjectSerializer

class AttestatSubjectViewSet(viewsets.ModelViewSet):
    """Отображение для модели Дисциплина аттестата"""
    queryset = AttestatSubject.objects.all()
    serializer_class = AttestatSubjectSerializer

class SpecialtyViewSet(viewsets.ModelViewSet):
    """Отображение для модели Специальность"""
    queryset = Specialty.objects.all()
    serializer_class = SpecialtySerializer

class FacultyViewSet(viewsets.ModelViewSet):
    """Отображение для модели Факультет"""
    queryset = Faculty.objects.all()
    serializer_class = FacultySerializer

"""Запросы к курсовой работе"""

class Query1(APIView):
    """Список абитуриентов, подавших заявление на заданную специальность"""

    def get(self, request):
        specialty = request.GET.get('specialty')
        enrollee_list = Application.objects.filter(specialty=specialty)
        serializer = ApplicationSerializer(enrollee_list, many=True)
        return Response({'result': serializer.data})

class Query2(APIView):
    """Количество абитуриентов, подавших заявления на каждую специальность"""

    def get(self, request):
        results = Application.objects.values('specialty', 'form').order_by('specialty').annotate(Count('enrollee'))
        return Response({'result': results})

class Query3(APIView):
    """Количество абитуриентов на базе 9 и 11 классов, поступающих на бюджет (или контракт)."""

    def get(self, request):
        results = Application.objects.values('form').order_by('form').annotate(Count('enrollee'))
        return Response({'result': results})

class Query4(APIView):
    """Общее количество поданных заявлений ежедневно"""

    def get(self, request):

```

```
results = Application.objects.values('date').order_by('date').annotate(Count('enrollee'))  
return Response({'result': results})
```

# Create your views here.