



**Министерство науки и высшего образования
Российской Федерации Федеральное
государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный
технический университет имени Н.Э.
Баумана
(национальный
исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Системы обработки информации и управления

**Лабораторная работа №6
По курсу «Разработка интернет приложений»**

Подготовил:
Студент группы ИУ5-55Б.
Турчин Д.С.
18.12.2020

Проверил:
Преподаватель кафедры ИУ5
Гапанюк Ю.Е.

Москва, 2021 г.

Цель лабораторной работы: изучение возможностей разработки REST API с использованием Django REST Framework.

Задание: С использованием Django REST Framework разработать REST API для одной модели (одной таблицы базы данных).

Ход работы

Создание приложения

Писать сервер с API мы будем на Django Rest Framework (DRF). Это надстройка над Django, которая позволяет нам писать REST приложения.

Войдем в виртуальное окружение и установим Django rest framework

```
env\Scripts\activate.bat  
pip install djangorestframework
```

Создадим приложение stocks с помощью команды `django-admin startapp stocks`

Применим все миграции проекта: `python manage.py migrate`

В файле `lab6/lab6/settings.py` в листе `INSTALLED_APPS` добавим название нашего приложения и название модуля DRF

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
  
    'rest_framework',  
    'stocks',  
]
```

Написание модели

В файле `lab6/stocks/models.py` напишем представление моделей, как в 5 лабораторной и запустим миграции

```

lab-6 > app > models.py > ...
1
2 # ИУ5-55 Турчин Денис. лаб-5. 27.12.2021. Предметная область 14: CD-диск (класс-1) и Библиотека CD-дисков (класс-2). Запросы Д:
3 #
4 #
5 # «Библиотека CD-дисков» и «CD-диск» связаны соотношением многие-ко-многим. <-- сделал многие-ко-многим
6 # Выведите список всех библиотек, у которых название начинается с буквы «А», и список хранящихся в них CD-дисков.
7 #
8
9 from django.conf import settings
10 from django.db import models
11 from django.utils import timezone
12 from django.db.models import Count
13
14 class Disk(models.Model):
15     class Meta:
16         db_table = 'Disks'
17         managed = True
18         verbose_name = 'диск'
19         verbose_name_plural = 'диски'
20
21     title = models.CharField(max_length=200)
22     format = models.CharField(max_length=10)
23     size = models.IntegerField(default = 0)
24
25     @property
26     def description(self):
27         return self.title + '(' + str(self.size) + ' MB, ' + self.format + ')'
28
29     def __str__(self):
30         return self.title
31
32 class Library(models.Model):
33     class Meta:
34         db_table = 'Libraries'
35         managed = True
36         verbose_name = 'библиотека'
37         verbose_name_plural = 'библиотеки'
38
39     fk_storedDisks = models.ManyToManyField(Disk)
40     title = models.CharField(max_length=200)
41     owner = models.CharField(max_length=80)
42
43     @property
44     def disks_num(self):
45         _disks_num = self.fk_storedDisks.all().count()
46         return _disks_num
47
48     @property
49     def description(self):
50         return str(self.disks_num) + ' disks | owner: , ' + self.owner
51
52     def __str__(self):
53         return self.title
54

```

Написание сериализатора

Напишем сериализаторы в файле lab6/stocks/serializers.py



```

lab-6 > app > serializers.py > Disk
1 from app.models import Disk
2 from rest_framework import serializers
3
4
5 class DiskSerializer(serializers.ModelSerializer):
6     class Meta:
7         # Модель, которую мы сериализуем
8         model = Disk
9         # Поля, которые мы сериализуем
10        fields = ["id", "title", "size", "format", "description"]

```

Написание View

Напишем view в файле lab6/stocks/views.py как в пятой лабораторной, но добавим путь для API

lab-6 > app >  views.py >  render

```
1  from django.shortcuts import render
2  from app.models import Library
3  from app.models import Disk
4  # V DRF V
5  from rest_framework import viewsets
6  from app.serializers import DiskSerializer
7
8  def index_page_router(request):
9      return render(request, 'index.html', {"items": Disk.objects.all()})
10
11 def libs_page_router(request):
12     return render(request, 'libraries.html', {"items": Library.objects.all()})
13
14 def lib_page_router(request, pk):
15     lib = Library.objects.get(pk=pk)
16     disks = lib.fk_storedDisks.all()
17     return render(request, 'lib_details.html', {'item': lib, 'items': disks})
18
19 # V DRF V
20 class DiskViewSet(viewsets.ModelViewSet):
21     """
22     API endpoint, который позволяет просматривать и редактировать
23     """
24     # queryset всех пользователей для фильтрации по дате последнего изменения
25     queryset = Disk.objects.all()
26     serializer_class = DiskSerializer # Сериализатор для модели
```

Добавление View в URL'ы нашего приложения

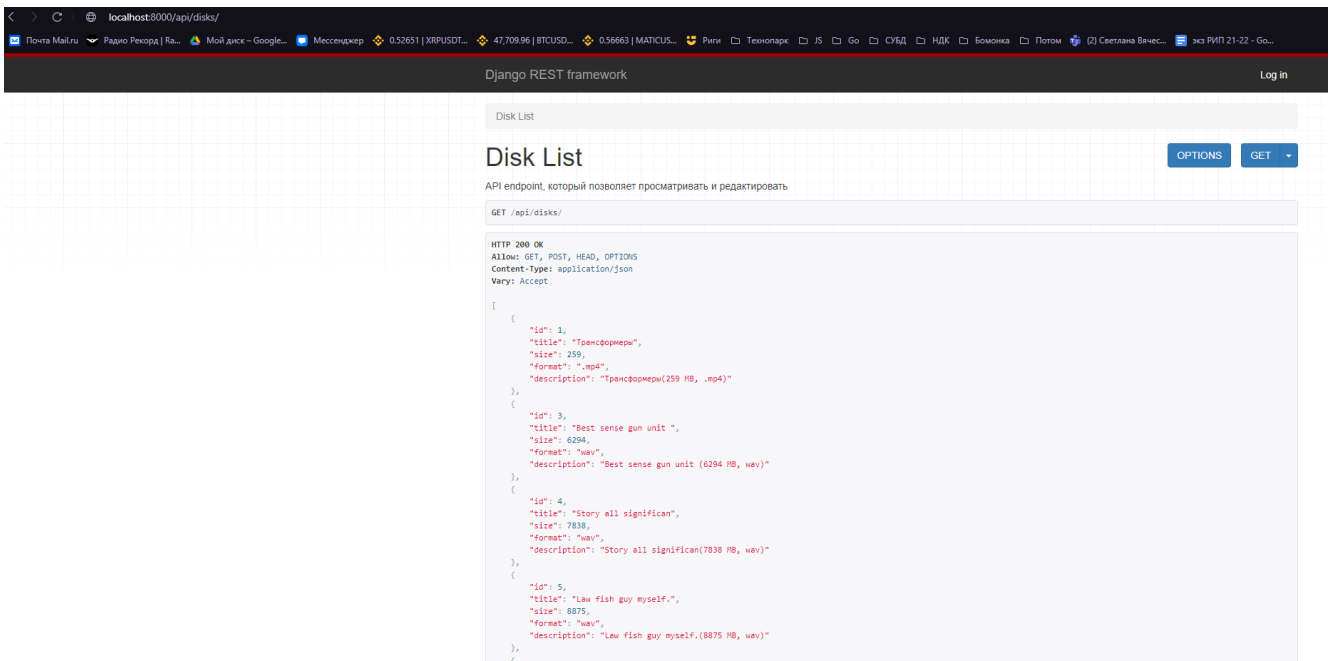
Добавим роутер нашего view в URL'ы приложения.
Для этого в файле lab6/lab6/urls.py напомним:

```

16 from django.contrib import admin
17 from django.urls import path, include
18 from app import views
19 # V DRF V
20 from rest_framework import routers
21
22 router = routers.DefaultRouter()
23 router.register(r'api/disks', views.DiskViewSet)
24
25 urlpatterns = [
26     path('admin/', admin.site.urls),
27     path('', views.index_page_router),
28     path('libraries', views.libs_page_router),
29     path('library/<int:pk>', views.lib_page_router, name='library'),
30
31     # V API V
32     path('', include(router.urls)),
33     path('api/', include('rest_framework.urls', namespace='rest_framework'))
34 ]
35

```

Проверить работоспособность API можно, перейдя по url API в браузере. При этом Джанго заботливо отдаст нам не сам JSON, а веб-приложение, где можно посмотреть результат запроса.



Django REST framework

Log in

Disk List

API endpoint, который позволяет просматривать и редактировать

GET /api/disks/

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

```

[
  {
    "id": 1,
    "title": "Трансформеры",
    "size": 259,
    "format": ".mp4",
    "description": "Трансформеры(259 MB, .mp4)"
  },
  {
    "id": 3,
    "title": "Best sense gun unit ",
    "size": 6294,
    "format": ".wav",
    "description": "Best sense gun unit (6294 MB, wav)"
  },
  {
    "id": 4,
    "title": "Story all significant",
    "size": 7838,
    "format": ".wav",
    "description": "Story all significant(7838 MB, wav)"
  },
  {
    "id": 5,
    "title": "Law fish guy myself.",
    "size": 8875,
    "format": ".wav",
    "description": "Law fish guy myself.(8875 MB, wav)"
  }
]

```

В следующих ЛР я буду использовать это API уже со своим интерфейсом