Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Радиотехнический»

Кафедра ИУ5 «Информатика и вычислительная техника»

**Курс «Разработка интернет-приложений»**

**Отчет по рубежному контролю № 2**

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-51 |  | преподаватель |
| Бушуев В.М. |  | Гапанюк Ю. Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

Задание

Рубежный контроль представляет собой разработку веб-приложения с использованием фреймворка Django. Веб-приложение должно выполнять следующие функции:

1. Создайте проект Python Django с использованием стандартных средств Django.
2. Создайте модель Django ORM, содержащую две сущности, связанные отношением один-ко-многим в соответствии с Вашим вариантом из условий рубежного контроля №1.
3. С использованием стандартного механизма Django сгенерируйте по модели макет веб-приложения, позволяющий добавлять, редактировать и удалять данные.
4. Создайте представление и шаблон, формирующий отчет, который содержит соединение данных из двух таблиц.

Ход работы:

* Models.py

from django.db import models

import os

# Create your models here.

class Computer(models.Model):

    model = models.CharField(max\_length=50, verbose\_name=("Модель"), null=True)

    count = models.CharField(max\_length=10, verbose\_name=("Количество"), default='0')

    classroom = models.ForeignKey("class", verbose\_name=("Класс"), on\_delete=models.CASCADE, null=True)

    class Meta:

        verbose\_name = ("Computer")

        verbose\_name\_plural = ("Computers")

    def \_\_str\_\_(self):

        return self.model

    def get\_absolute\_url(self):

        return reverse("Computer\_detail", kwargs={"pk": self.pk})

class Class(models.Model):

    name = models.CharField(max\_length=50, verbose\_name=("Название"), null=True)

    class Meta:

        verbose\_name = ("Class")

        verbose\_name\_plural = ("Classes")

    def \_\_str\_\_(self):

        return self.name

    def get\_absolute\_url(self):

        return reverse("Class\_detail", kwargs={"pk": self.pk})

* Serializers.py

from .models import Computer, Class

from rest\_framework import serializers

class ComputerSerializer (serializers.ModelSerializer):

    class Meta:

        model = Computer

        fields = ["model", "count", "classroom"]

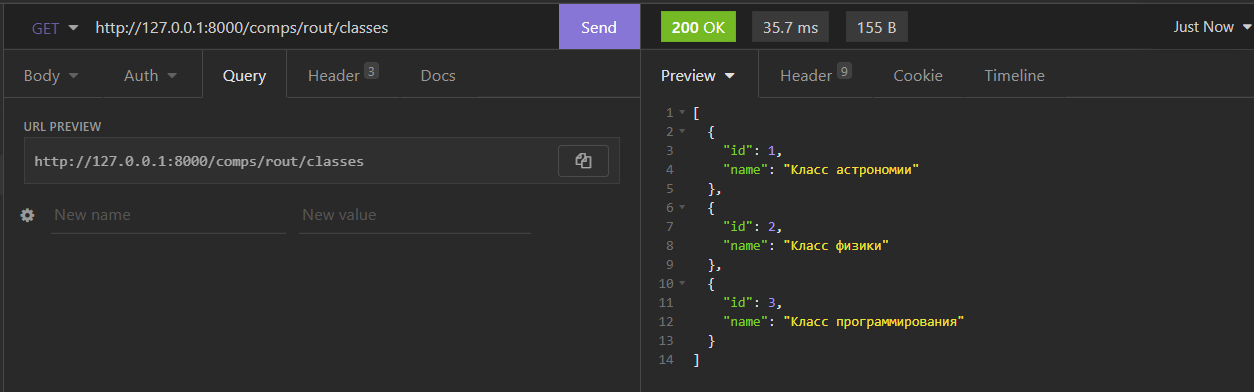
class ClassSerializer (serializers.ModelSerializer):

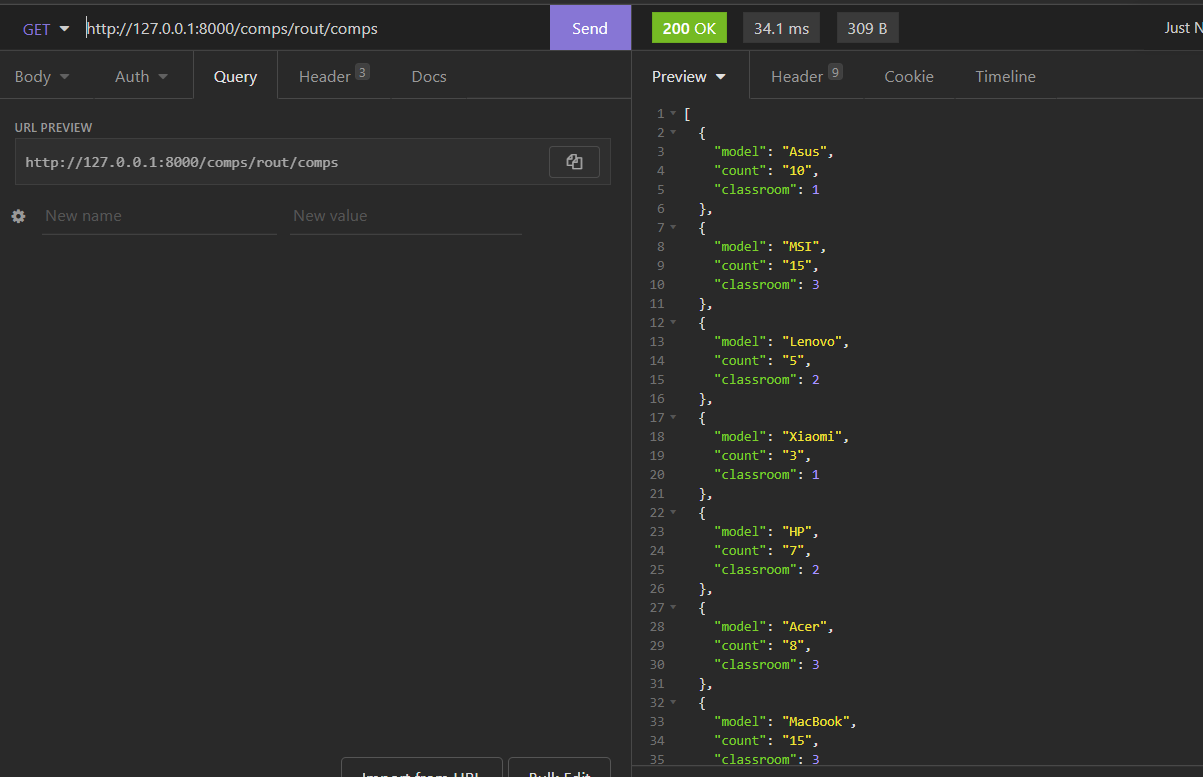
    class Meta:

        model = Class

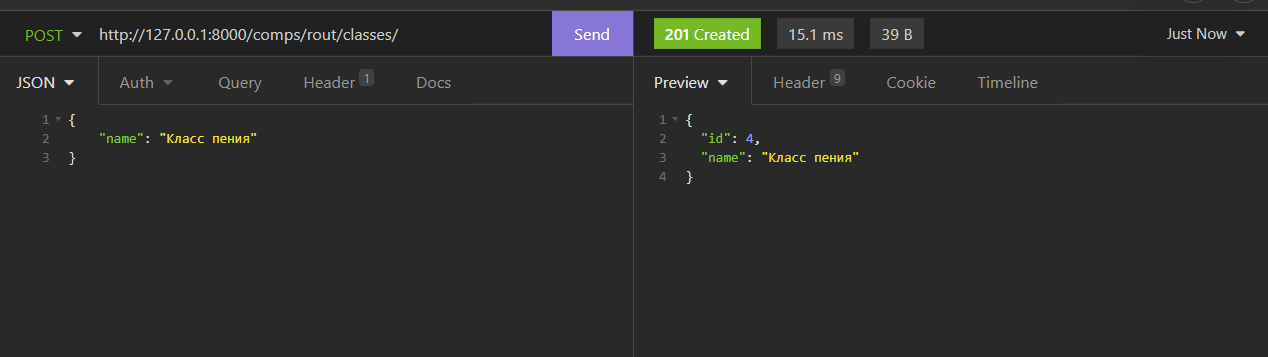
        fields = ["id", "name"]

* Urls.py
* from django.urls import include, path
* from . import views
* from rest\_framework import routers
* router = routers.DefaultRouter()
* router.register('classes', views.ClassesViewSet)
* urlpatterns = [
* path('', views.main),
* path('rout/', include(router.urls)),
* path('api-auth/', include('rest\_framework.urls', namespace='rest\_framework'))
* ]
* Vievs.py
* from django.shortcuts import render
* from .models import Computer, Class
* from rest\_framework import viewsets
* from .serializers import ComputerSerializer, ClassSerializer
* # Create your views here.
* def main(request):
* return render(request, 'index.html', { 'data' : {
* 'comps': Computer.objects.all(),
* 'classes': Class.objects.all()
* }})
* class CompsViewSet(viewsets.ModelViewSet):
* """
* API endpoint, который позволяет просматривать и редактировать акции компаний
* """
* # queryset всех пользователей для фильтрации по дате последнего изменения
* queryset = Computer.objects.all()
* serializer\_class = ComputerSerializer  # Сериализатор для модели
* class ClassesViewSet(viewsets.ModelViewSet):
* """
* API endpoint, который позволяет просматривать и редактировать акции компаний
* """
* # queryset всех пользователей для фильтрации по дате последнего изменения
* queryset = Class.objects.all()
* serializer\_class = ClassSerializer  # Сериализатор для модели
* GET-запрос

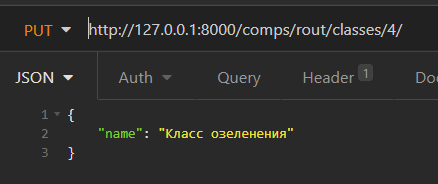


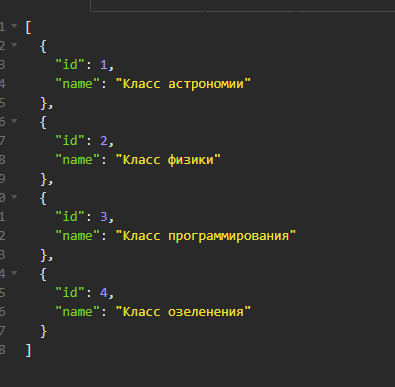


* POST-запрос



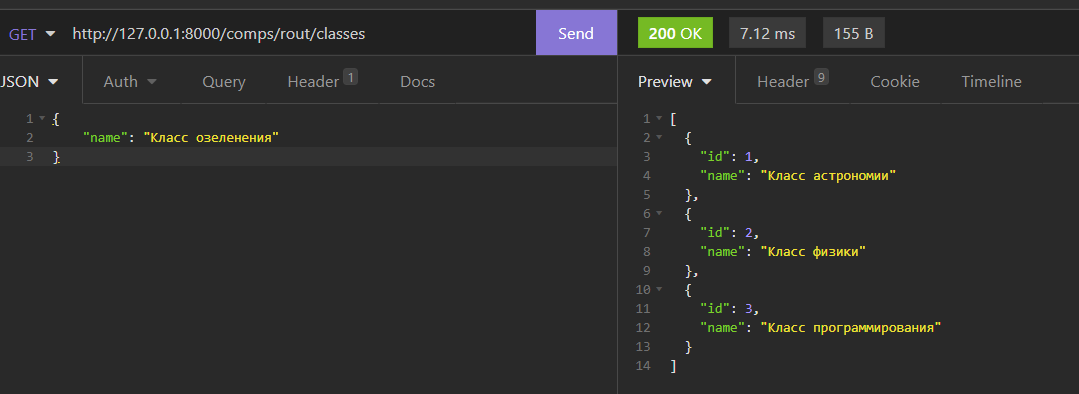
* PUT-запрос





* DELETE-запрос



****

Отчет

