Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

ЛАБОРАТОРНАЯ РАБОТА №3
«РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ СРЕДСТВАМИ DJANGO И DJANGORESTFRAMEWORK»

Выполнил:
Студент III курса ИМРиП
Группы: D33101
Ф.И.О.: Чжан Цзяи

Проверил:
Говоров Антон Игоревич

Санкт-Петербург
2021

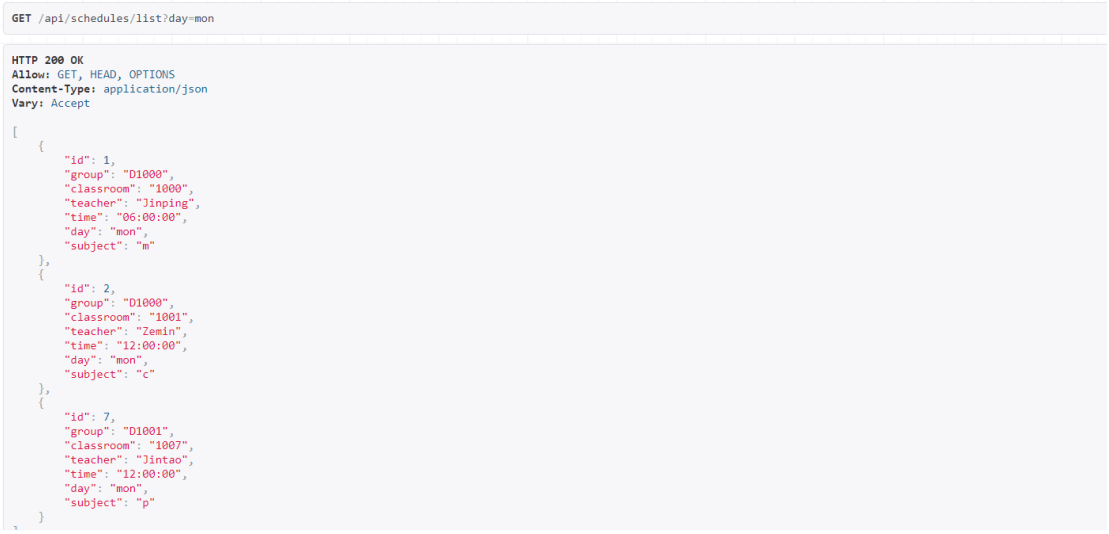
Задание3

Задание 3

Создать программную систему, предназначенную для завуча школы. Она должна обеспечивать хранение сведений о каждом учителе, классном руководстве, о предметах, которые он преподает в заданный период, номере закрепленного за ним кабинета, о расписании занятий. Существуют учителя, которые не имеют собственного кабинета. Об учениках должны храниться следующие сведения: фамилия и имя, в каком классе учится, какую оценку имеет в текущей четверти по каждому предмету.
Завуч должен иметь возможность добавить сведения о новом учителе или ученике, внести в базу данных четвертные оценки учеников каждого класса по каждому предмету, удалить данные об уволившемся учителе и отчисленном из школы ученике, внести изменения в данные об учителях и учениках, в том числе поменять оценку ученика по тому или иному предмету. В задачу завуча входит также составление расписания.

Завучу могут потребоваться следующие сведения:

Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?

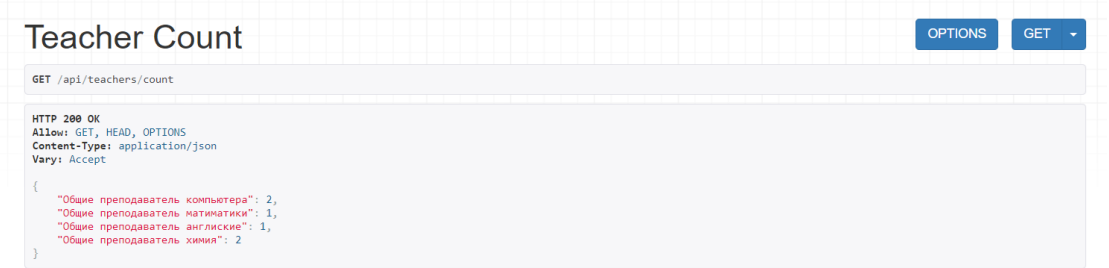


```
GET /api/schedules/list?day=mon
```

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

[
    {
        "id": 1,
        "group": "D1000",
        "classroom": "1000",
        "teacher": "Jinping",
        "time": "06:00:00",
        "day": "mon",
        "subject": "m"
    },
    {
        "id": 2,
        "group": "D1000",
        "classroom": "1001",
        "teacher": "Zemin",
        "time": "12:00:00",
        "day": "mon",
        "subject": "c"
    },
    {
        "id": 7,
        "group": "D1001",
        "classroom": "1007",
        "teacher": "Jintao",
        "time": "12:00:00",
        "day": "mon",
        "subject": "p"
    }
```

Сколько учителей преподает каждую из дисциплин в школе?



Teacher Count

OPTIONS GET

```
GET /api/teachers/count
```

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

{
    "Общие преподаватель компьютера": 2,
    "Общие преподаватель математики": 1,
    "Общие преподаватель английские": 1,
    "Общие преподаватель химия": 2
}
```

Список учителей, преподающих те же предметы, что и учитель, ведущий информатику в заданном классе.

GET /api/schedules/list?subject=p

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

[
    {
        "id": 3,
        "group": "D1000",
        "classroom": "1004",
        "teacher": "Jintao",
        "time": "12:00:00",
        "day": "wed",
        "subject": "p"
    },
    {
        "id": 6,
        "group": "D1000",
        "classroom": "1008",
        "teacher": "Zedong",
        "time": "18:00:00",
        "day": "sat",
        "subject": "p"
    },
    {
        "id": 7,
        "group": "D1001",
        "classroom": "1007",
        "teacher": "Jintao",
        "time": "12:00:00",
        "day": "mon",
        "subject": "p"
    },
```

Сколько мальчиков и девочек в каждом классе?

Student Count Girl

OPTIONS GET

GET /api/students/count/1

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

{
    "Общие студентки": 1,
    "Общие студенты": 2
}
```

Сколько кабинетов в школе для базовых и профильных дисциплин?

Classroom Count Base

OPTIONS GET

GET /api/classrooms/count

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

{
    "Общие  базовые": 4,
    "Общие профильные": 4
}
```

Необходимо предусмотреть возможность получения документа, представляющего собой отчет об успеваемости заданного класса. Отчет включает сведения об успеваемости за четверть по каждому предмету. Необходимо подсчитать средний балл по каждому предмету, по классу в целом, указать общее количество учеников в классе. Для класса указать классного руководителя.

Report

OPTIONS GET

GET /api/report/1

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

{
    "Средний балл": {
        "score_math__avg": 60.333333333333336,
        "score_chemical__avg": 82.33333333333333,
        "score_english__avg": 42.0,
        "score_computer__avg": 70.33333333333333
    },
    "Общие люди": {
        "last_name__count": 3
    },
    "руководитель": [
        {
            "id": 1,
            "head_teacher": "Jinping",
            "number": "D1000"
        }
    ]
}
```

Код:

models.py:

```python
from django.db import models


# Create your models here.
class Teacher(models.Model):
    first_name = models.CharField(max_length=120)
    last_name = models.CharField(max_length=120)
    subject_types = (
        ('m', 'math'),
        ('c', 'chemical'),
        ('e', 'english'),
        ('p', 'computer science'),
    )
    subject = models.CharField(max_length=1, choices=subject_types)
    b_date = models.DateField()
    e_date = models.DateField()
    office = models.CharField(max_length=120, blank=True, null=True)


class Group(models.Model):
    number = models.CharField(max_length=120)
    head_teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE, blank=True, null=True)
```

```python
class Classroom(models.Model):
    room_types = (
        ('b', 'base'),
        ('p', 'professional'),
    )
    room = models.CharField(max_length=1, choices=room_types)
    number = models.CharField(max_length=20)


class Schedule(models.Model):
    time = models.TimeField()
    day_types = (
        ('mon', 'monday'),
        ('tue', 'tuesday'),
        ('wed', 'wednesday'),
        ('thu', 'thursday'),
        ('fri', 'friday'),
        ('sat', 'saturday'),
    )
    day = models.CharField(max_length=3, choices=day_types)
    subject_types = (
        ('m', 'math'),
        ('c', 'chemical'),
        ('e', 'english'),
        ('p', 'computer science'),
    )
    subject = models.CharField(max_length=1, choices=subject_types)
    group = models.ForeignKey(Group, on_delete=models.CASCADE, blank=True, null=True)
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE, blank=True, null=True)
    classroom = models.ForeignKey(Classroom, on_delete=models.CASCADE, blank=True, null=True)
```

```python
class Student(models.Model):
    sex_types = (
        ('m', 'male'),
        ('s', 'female'),
    )
    sex = models.CharField(max_length=1, choices=sex_types)
    first_name = models.CharField(max_length=120)
    last_name = models.CharField(max_length=120)
    score_math = models.IntegerField(blank=True, null=True)
    score_chemical = models.IntegerField(blank=True, null=True)
    score_english = models.IntegerField(blank=True, null=True)
    score_computer = models.IntegerField(blank=True, null=True)
    semester = models.IntegerField()
    group = models.ForeignKey(Group, on_delete=models.CASCADE, blank=True, null=True)
```

views.py:

```python
from rest_framework import generics, status
from rest_framework.views import APIView
from .models import *
from .serializers import *
from rest_framework.response import Response
from django.db.models import Avg, Sum, Count


class TeacherListView(generics.ListAPIView):
    serializer_class = TeacherSerializer

    def get_queryset(self):
        queryset = Teacher.objects.all()

        params = self.request.query_params

        subject = params.get('subject', None)

        if subject:
            queryset = queryset.filter(subject__exact=subject)

        return queryset


class TeacherRetrieveView(generics.RetrieveAPIView):
    serializer_class = TeacherSerializer
    queryset = Teacher.objects.all()
```

```python
class TeacherCreateView(generics.CreateAPIView):
    serializer_class = CreateTeacherSerializer
    queryset = Teacher.objects.all()


class TeacherUpdateView(generics.UpdateAPIView):
    serializer_class = CreateTeacherSerializer
    queryset = Teacher.objects.all()


class TeacherCountView(APIView):
    def get(self, request):
        counts = Teacher.objects.count()
        return Response({"Общие преподаватель": counts})


class GroupListView(generics.ListAPIView):
    serializer_class = GroupSerializer
    queryset = Group.objects.all()


class GroupRetrieveView(generics.RetrieveAPIView):
    serializer_class = GroupSerializer
    queryset = Group.objects.all()


class GroupCreateView(generics.CreateAPIView):
    serializer_class = CreateGroupSerializer
    queryset = Group.objects.all()
```

```python
class GroupUpdateView(generics.UpdateAPIView):
    serializer_class = CreateGroupSerializer
    queryset = Group.objects.all()


class ClassroomListView(generics.ListAPIView):
    serializer_class = ClassroomSerializer

    def get_queryset(self):
        queryset = Classroom.objects.all()

        params = self.request.query_params

        room = params.get('room', None)
        if room:
            queryset = queryset.filter(room__exact=room)

        return queryset


class ClassroomRetrieveView(generics.RetrieveAPIView):
    serializer_class = ClassroomSerializer
    queryset = Classroom.objects.all()


class ClassroomCreateView(generics.CreateAPIView):
    serializer_class = CreateClassroomSerializer
    queryset = Classroom.objects.all()
```

```python
class ClassroomUpdateView(generics.UpdateAPIView):
    serializer_class = CreateClassroomSerializer
    queryset = Classroom.objects.all()


class ClassroomCountBaseView(APIView):
    def get(self, request):
        count_base = Classroom.objects.filter(room='b').count()
        count_profession = Classroom.objects.filter(room='p').count()
        return Response({
            "Общие  базовые": count_base,
            "Общие профильные": count_profession
        }
        )
```

```python
class ScheduleListView(generics.ListAPIView):
    serializer_class = ScheduleSerializer

    def get_queryset(self):
        queryset = Schedule.objects.all()

        params = self.request.query_params

        day = params.get('day', None)
        group = params.get('group', None)
        subject = params.get('subject', None)
        if day:
            queryset = queryset.filter(day__exact=day)
        if subject:
            queryset = queryset.filter(subject__exact=subject)
        if group:
            queryset = queryset.filter(group__number=group)

        return queryset


class ScheduleRetrieveView(generics.RetrieveAPIView):
    serializer_class = ScheduleSerializer
    queryset = Schedule.objects.all()


class ScheduleCreateView(generics.CreateAPIView):
    serializer_class = CreateScheduleSerializer
    queryset = Schedule.objects.all()
```

```python
class ScheduleUpdateView(generics.UpdateAPIView):
    serializer_class = CreateScheduleSerializer
    queryset = Schedule.objects.all()


class StudentListView(generics.ListAPIView):
    serializer_class = StudentSerializer

    def get_queryset(self):
        queryset = Student.objects.all()

        params = self.request.query_params

        group = params.get('group', None)
        sex = params.get('sex', None)
        if sex:
            queryset = queryset.filter(sex__exact=sex)
        if group:
            queryset = queryset.filter(group__number=group)

        return queryset


class StudentRetrieveView(generics.RetrieveAPIView):
    serializer_class = StudentSerializer
    queryset = Student.objects.all()
```

```python
class StudentCreateView(generics.CreateAPIView):
    serializer_class = CreateStudentSerializer
    queryset = Student.objects.all()


class StudentUpdateView(generics.UpdateAPIView):
    serializer_class = CreateStudentSerializer
    queryset = Student.objects.all()


class StudentCountGirlView(APIView):
    def get(self, request, pk):
        count_girls = Student.objects.filter(group=pk, sex='s').count()
        counts_boys = Student.objects.filter(group=pk, sex='m').count()
        return Response(
            {"Общие студентки": count_girls,
             "Общие студенты": counts_boys,
             }
        )


class ReportView(APIView):
    def get(self, request, pk):
        aver_grades = Student.objects.filter(group=pk).aggregate(Avg('score_math'),
                                                                 Avg('score_chemical'),
                                                                 Avg('score_english'),
                                                                 Avg('score_computer'))
        all_people = Student.objects.filter(group=pk).aggregate(Count('last_name'))
        head_teacher = Group.objects.filter(id=pk)
        serializer = GroupSerializer(head_teacher, many=True)

        return Response({
            'Средний балл': aver_grades,
            'Обшие люди': all_people,
            'руководитель': serializer.data,
        }
        )
```

serializers.py

```python
from rest_framework import serializers
from .models import *


class TeacherSerializer(serializers.ModelSerializer):
    class Meta:
        model = Teacher
        fields = "__all__"


class CreateTeacherSerializer(serializers.ModelSerializer):
    class Meta:
        model = Teacher
        fields = "__all__"
class GroupSerializer(serializers.ModelSerializer):
    head_teacher = serializers.SlugRelatedField(read_only=True, slug_field='first_name')

    class Meta:
        model = Group
        fields = "__all__"


class CreateGroupSerializer(serializers.ModelSerializer):
    class Meta:
        model = Group
        fields = "__all__"
class ClassroomSerializer(serializers.ModelSerializer):
    class Meta:
        model = Classroom
        fields = "__all__"


class CreateClassroomSerializer(serializers.ModelSerializer):
    class Meta:
        model = Classroom
        fields = "__all__"
```

```python
class ScheduleSerializer(serializers.ModelSerializer):
    group = serializers.SlugRelatedField(read_only=True, slug_field='number')
    classroom = serializers.SlugRelatedField(read_only=True, slug_field='number')
    teacher = serializers.SlugRelatedField(read_only=True, slug_field='first_name')

    # group = serializers.StringRelatedField(read_only=True)

    # group = GroupSerializer()
    class Meta:
        model = Schedule
        fields = "__all__"


class CreateScheduleSerializer(serializers.ModelSerializer):
    class Meta:
        model = Schedule
        fields = "__all__"
class StudentSerializer(serializers.ModelSerializer):
    group = serializers.SlugRelatedField(read_only=True, slug_field='number')

    class Meta:
        model = Student
        fields = "__all__"


class CreateStudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = "__all__"
```

urls.py:

```python
from django.urls import path
from .views import *
from rest_framework import permissions
from drf_yasg.views import get_schema_view
from drf_yasg import openapi
urlpatterns = [
    path('teachers/list', TeacherListView.as_view()),
    path('teachers/<int:pk>', TeacherRetrieveView.as_view()),
    path('teachers/update/<int:pk>', TeacherUpdateView.as_view()),
    path('teachers/new', TeacherCreateView.as_view()),
    path('teachers/count', TeacherCountView.as_view()),

    path('groups/list', GroupListView.as_view()),
    path('groups/<int:pk>', GroupRetrieveView.as_view()),
    path('groups/update/<int:pk>', GroupUpdateView.as_view()),
    path('groups/new', GroupCreateView.as_view()),
```

```
path('classrooms/list', ClassroomListView.as_view()),
path('classrooms/<int:pk>', ClassroomRetrieveView.as_view()),
path('classrooms/update/<int:pk>', ClassroomUpdateView.as_view()),
path('classrooms/new', ClassroomCreateView.as_view()),
path('classrooms/count', ClassroomCountBaseView.as_view()),

path('schedules/list', ScheduleListView.as_view()),
path('schedules/<int:pk>', ScheduleRetrieveView.as_view()),
path('schedules/update/<int:pk>', ScheduleUpdateView.as_view()),
path('schedules/new', ScheduleCreateView.as_view()),

path('students/list', StudentListView.as_view()),
path('students/<int:pk>', StudentRetrieveView.as_view()),
path('students/update/<int:pk>', StudentUpdateView.as_view()),
path('students/new', StudentCreateView.as_view()),
path('students/count/<int:pk>', StudentCountGirlView.as_view()),

path('report/<int:pk>', ReportView.as_view())
```