Московский государственный технический университет имени Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5

Отчёт по

Рубежному контролю № 2

Выполнил:

Бахрамов Н.А

Группа ИУ5-53Б

Москва

2021

Задание:

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.

Setting.py

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

    'rest_framework',
    'rk2_rest'
]
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'rk2',
        'USER': 'dbuser2',
        'PASSWORD': '1234',
        'HOST': 'localhost',
        'PORT': 3306,  # Стандартный порт MySQL
        'OPTIONS': {'charset': 'utf8'},
        'TEST_CHARSET': 'utf8',
    }
}
```

Views.py

```python
from django.shortcuts import render
from rest_framework import viewsets

from rk2_rest.models import Musicians, Orchestrys
from rk2_rest.serializers import MusicianSerializer, OrchestrySerializer
# Create your views here.


class MusicianViewSet(viewsets.ModelViewSet):
    queryset = Musicians.objects.all()
    serializer_class = MusicianSerializer


class OrchestryViewSet(viewsets.ModelViewSet):
    queryset = Orchestrys.objects.all()
    serializer_class = OrchestrySerializer


def report(req):
    return render(req, 'report.html', {
        "data": Musicians.objects.all(),
        "dataork": Orchestrys.objects.all()
    })
```

Models.py

```python
from django.db import models

# Create your models here.
class Musicians(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=50, verbose_name="Имя")
    role = models.CharField(max_length=32, verbose_name="Роль в оркестре")
    id_orkestry = models.ForeignKey('Orchestrys', models.DO_NOTHING, db_column="id_orkestry", blank=True, null=False)

    class Meta:
        db_table = 'musician'

class Orchestrys(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=32, verbose_name="Название оркестра")

    class Meta:
        db_table = 'orchestry'
```

Serializers.py

```python
from rest_framework import serializers
from rk2_rest.models import Musicians, Orchestrys


class MusicianSerializer(serializers.ModelSerializer):
    class Meta:
        model = Musicians
        fields = ['id', 'name', 'role', 'id_orkestry']


class OrchestrySerializer(serializers.ModelSerializer):
    class Meta:
        model = Orchestrys
        fields = ['id', 'name']
```

Urls.py

```python
from django.contrib import admin
from django.urls import path, include
from rest_framework import routers
from rk2_rest import views

router = routers.DefaultRouter()
router.register('musicians', views.MusicianViewSet)
router.register('orchestry', views.OrchestryViewSet)

urlpatterns = [
    path('', include(router.urls)),
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework')),
    path('report/', views.report)
]
```

rk2_rest.views

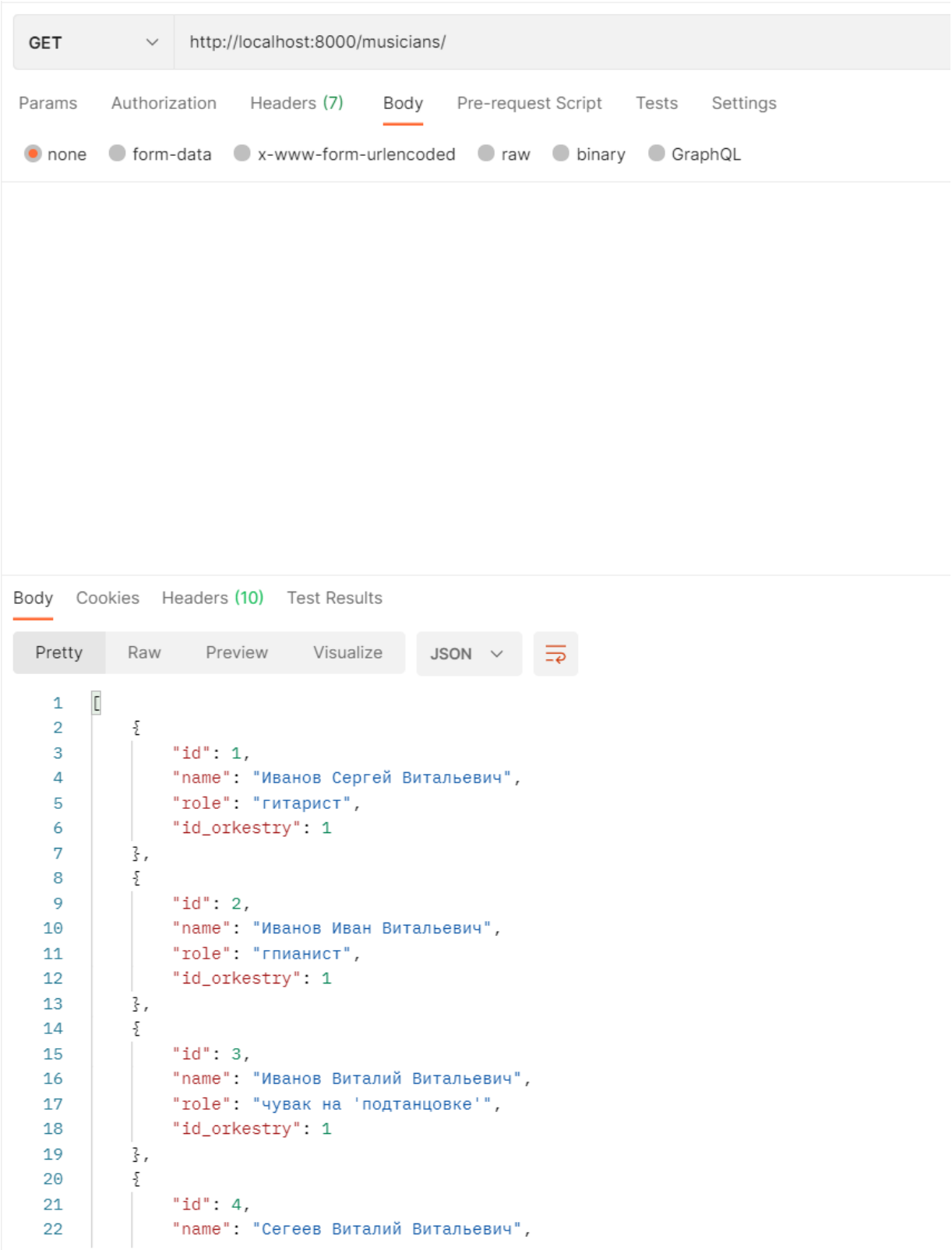
Report html

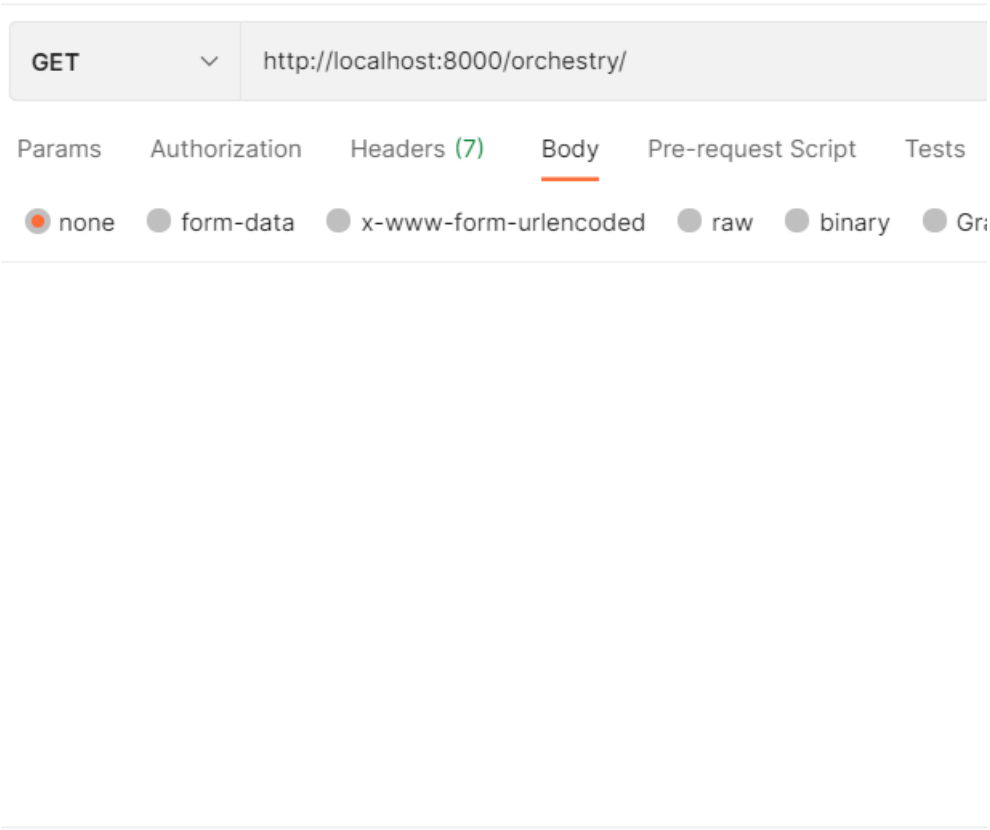
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

<div>
    <div>
        музыканты:
        {% for value in data %}
            <div>
            <span>{{ value.name }} | имя оркестра: {{ value.id_orkestry.name }} | </span>
            <span>{{ value.role }}</span>
            </div>
        {% endfor %}
    </div>
</div>
</body>
</html>
```

музыканты:
Иванов Сергей Витальевич | имя оркестра: крутой оркестр | гитарист
Иванов Иван Витальевич | имя оркестра: крутой оркестр | гпианист
Иванов Виталий Витальевич | имя оркестра: крутой оркестр | чувак на 'подтанцовке'
Сегеев Виталий Витальевич | имя оркестра: мега крутой оркестр | чувак на 'подтанцовке'
Сегеев Анатолий Витальевич | имя оркестра: мега крутой оркестр | гитарист
Сегеев Никита Витальевич | имя оркестра: мега крутой оркестр | пианист
Анатолиев Никита Витальевич | имя оркестра: супер мега крутой оркестр | пианист
Анатолиев Артем Витальевич | имя оркестра: супер мега крутой оркестр | гитарист
Анатолиев Алеша Витальевич | имя оркестра: супер мега крутой оркестр | чувак для мега поддержки

Postman


GET
http://localhost:8000/musicians/
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Body
Cookies
Headers (10)
Test Results
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"name": "Иванов Сергей Витальевич",
"role": "гитарист",
"id_orkestry": 1
},
{
"id": 2,
"name": "Иванов Иван Витальевич",
"role": "гпианист",
"id_orkestry": 1
},
{
"id": 3,
"name": "Иванов Виталий Витальевич",
"role": "чувак на 'подтанцовке'",
"id_orkestry": 1
},
{
"id": 4,
"name": "Сегеев Виталий Витальевич",


GET
http://localhost:8000/orchestry/
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
none
form-data
x-www-form-urlencoded
raw
binary
Body
Cookies
Headers (10)
Test Results
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"name": "крутой оркестр"
},
{
"id": 2,
"name": "мега крутой оркестр"
},
{
"id": 3,
"name": "супер мега крутой оркестр"
}
]