|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 Радиотехнический (РТ)

КАФЕДРА Системы обработки информации и управления (ИУ5)

**Рубежный контроль №2**

**по курсу «Разработка интернет-приложений»**

**17 вариант**

Студентка РТ5-51Б Стадник Е.Р.

(Группа) (Фамилия И.О.)

Преподаватель: Гапанюк Ю. Е.

(Фамилия И.О.)

2021 г.

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Класс 1** | **Класс 2** |
| 17 | Дирижер | Оркестр |

Вариант задания:

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.

Текст программы:

views.py

from django.shortcuts import render

from rest\_framework import viewsets

from orch.serializers import OrchSerializer

from orch.serializers import DirSerializer

from orch.models import Orch

from orch.models import Dir

class OrchViewSet(viewsets.ModelViewSet):

"""

API endpoint, который позволяет просматривать и редактировать акции компаний

"""

# queryset всех пользователей для фильтрации по дате последнего изменения

queryset = Orch.objects.all()

serializer\_class = OrchSerializer # Сериализатор для модели

class DirViewSet(viewsets.ModelViewSet):

"""

API endpoint, который позволяет просматривать и редактировать акции компаний

"""

# queryset всех пользователей для фильтрации по дате последнего изменения

queryset = Dir.objects.all()

serializer\_class = DirSerializer # Сериализатор для модели

def report(request):

return render(request, 'page.html', {'data':{'Dirs':Dir.objects.select\_related('iddirorch').order\_by('iddir')}})

serializers.py

from orch.models import Orch

from orch.models import Dir

from rest\_framework import serializers

class DirSerializer(serializers.ModelSerializer):

class Meta:

# Модель, которую мы сериализуем

model = Dir

# Поля, которые мы сериализуем

fields = ["iddir", "fiodir", "saldir", "iddirorch"]

class OrchSerializer(serializers.ModelSerializer):

class Meta:

model = Orch

fields = ['idorch', 'nameorch']

models.py

from django.db import models

class Dir(models.Model):

iddir = models.IntegerField(db\_column='idDir', primary\_key=True) # Field name made lowercase.

fiodir = models.CharField(db\_column='fioDir', max\_length=100) # Field name made lowercase.

saldir = models.FloatField(db\_column='salDir') # Field name made lowercase.

iddirorch = models.ForeignKey('Orch', models.DO\_NOTHING, db\_column='idDirOrch') # Field name made lowercase.

class Meta:

managed = False

db\_table = 'dir'

class DjangoMigrations(models.Model):

id = models.BigAutoField(primary\_key=True)

app = models.CharField(max\_length=255)

name = models.CharField(max\_length=255)

applied = models.DateTimeField()

class Meta:

managed = False

db\_table = 'django\_migrations'

class Orch(models.Model):

idorch = models.IntegerField(db\_column='idOrch', primary\_key=True) # Field name made lowercase.

nameorch = models.CharField(db\_column='nameOrch', max\_length=45, blank=True, null=True) # Field name made lowercase.

class Meta:

managed = False

db\_table = 'orch'

settings.py

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# DRF

'rest\_framework',

# Наше приложение

'orch'

]

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'MyDB',

'USER': 'root',

'PASSWORD': '123',

'HOST': 'localhost',

'PORT': 3306, # Стандартный порт MySQL

'OPTIONS': {'charset': 'utf8'},

'TEST\_CHARSET': 'utf8',

}

}

urls.py

from django.contrib import admin

from orch import views as dir\_views

from django.urls import include, path

from rest\_framework import routers

router = routers.DefaultRouter()

router.register(r'dirs', dir\_views.DirViewSet)

router.register(r'orchs', dir\_views.OrchViewSet)

# Wire up our API using automatic URL routing.

# Additionally, we include login URLs for the browsable API.

urlpatterns = [

path('', include(router.urls)),

path('api-auth/', include('rest\_framework.urls', namespace='rest\_framework')),

path('page/', dir\_views.report),

path('admin/', admin.site.urls),

]

base.html

<!doctype html>

<html lang="en" class="h-100">

<head>

<meta charset="utf-8">

<title>{% block title %}{% endblock %}</title>

</head>

<body>

{% block content %}{% endblock %}

</body>

</html>

page.html

{% extends 'base.html' %}

{% block title %} Дирижеры {% endblock %}

{% block content %}

<table>

<tr>

<th>ID</th>

<th>ФИО</th>

<th>Зарплата</th>

<th>Оркестр</th>

</tr>

{% for dir in data.Dirs %}

<tr>

<th>{{ dir.iddir }}</th>

<td>{{ dir.fiodir }}</td>

<td>{{ dir.saldir }}</td>

<td>{{ dir.iddirorch.nameorch }}</td>

</tr>

{% endfor %}

</table>

{% endblock %}

Результат выполнения программы:

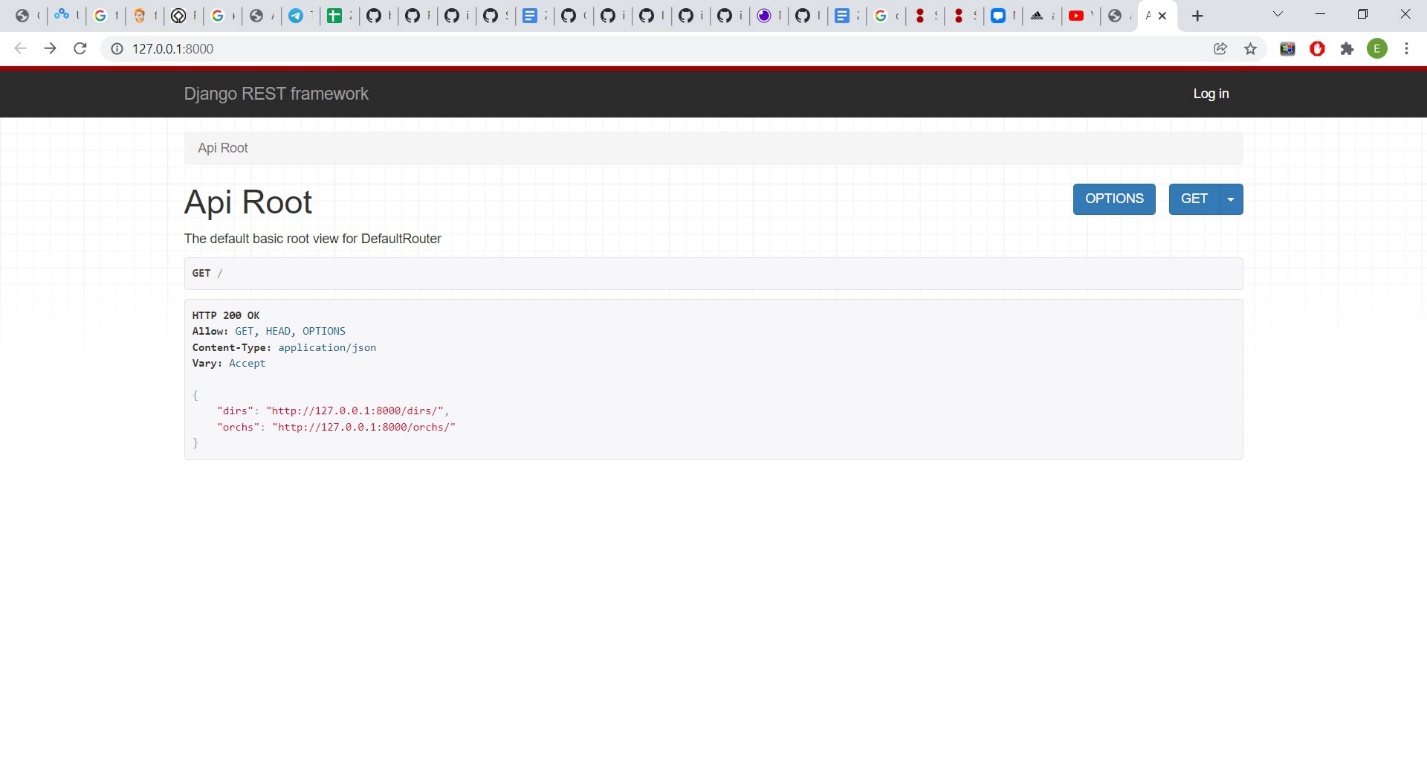
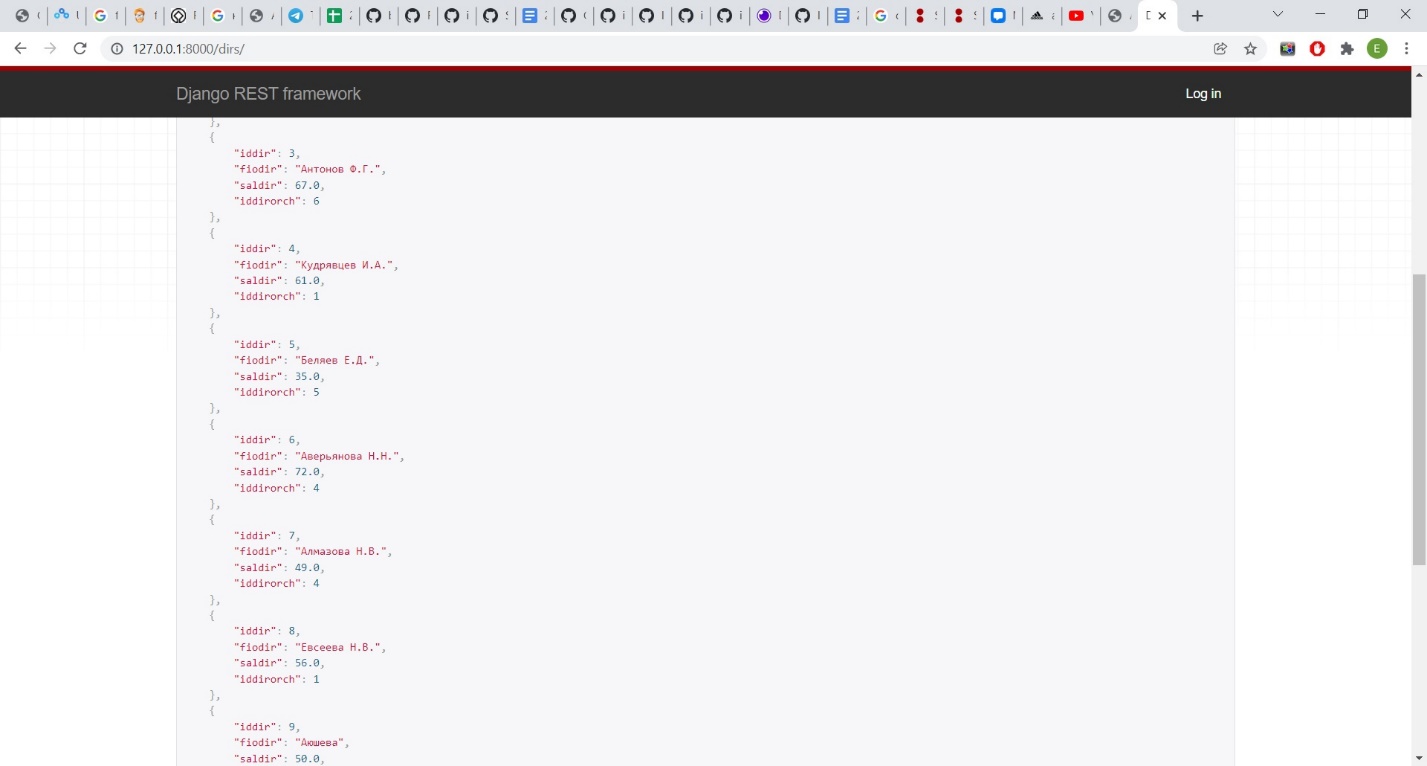


Рис. 1. Главная страница

Рис. 2. Дирижеры

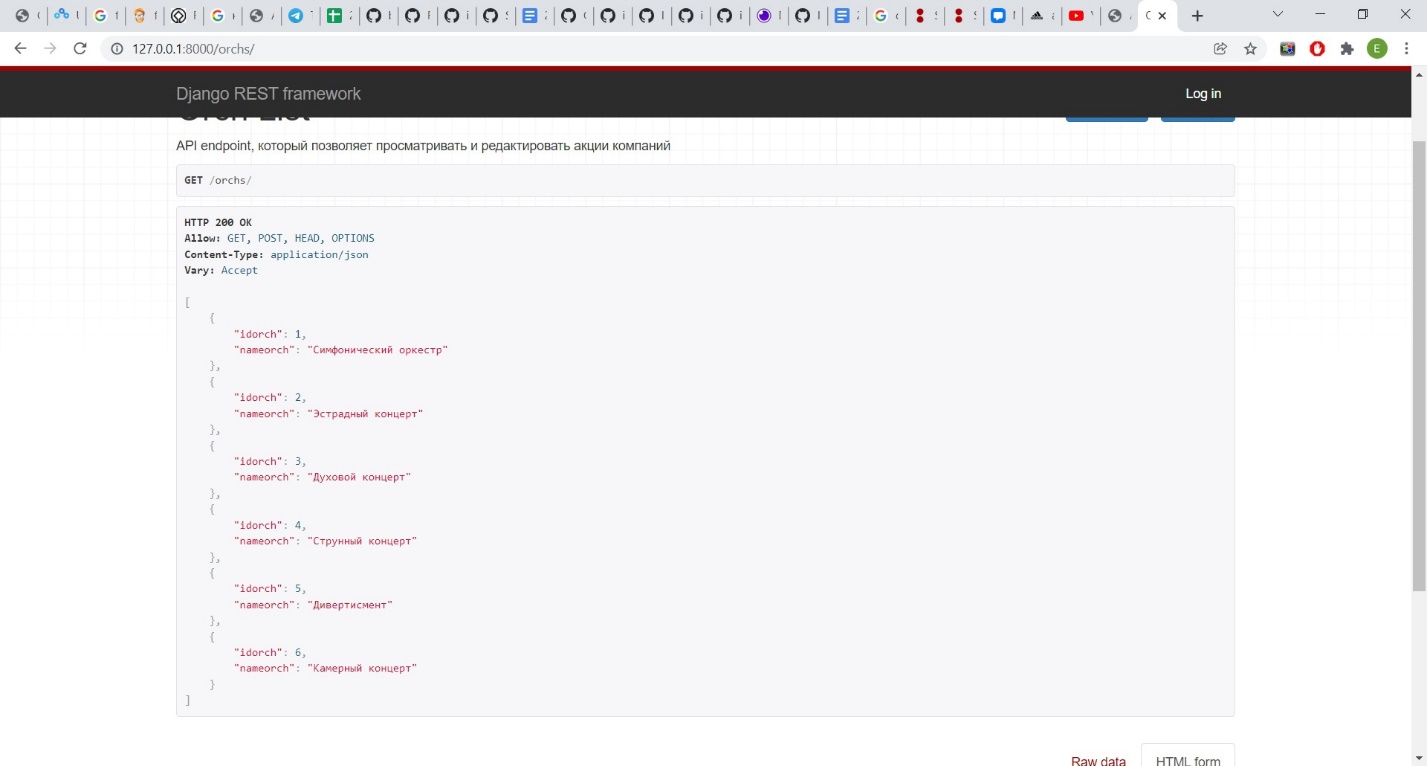


Рис. 3. Оркестры

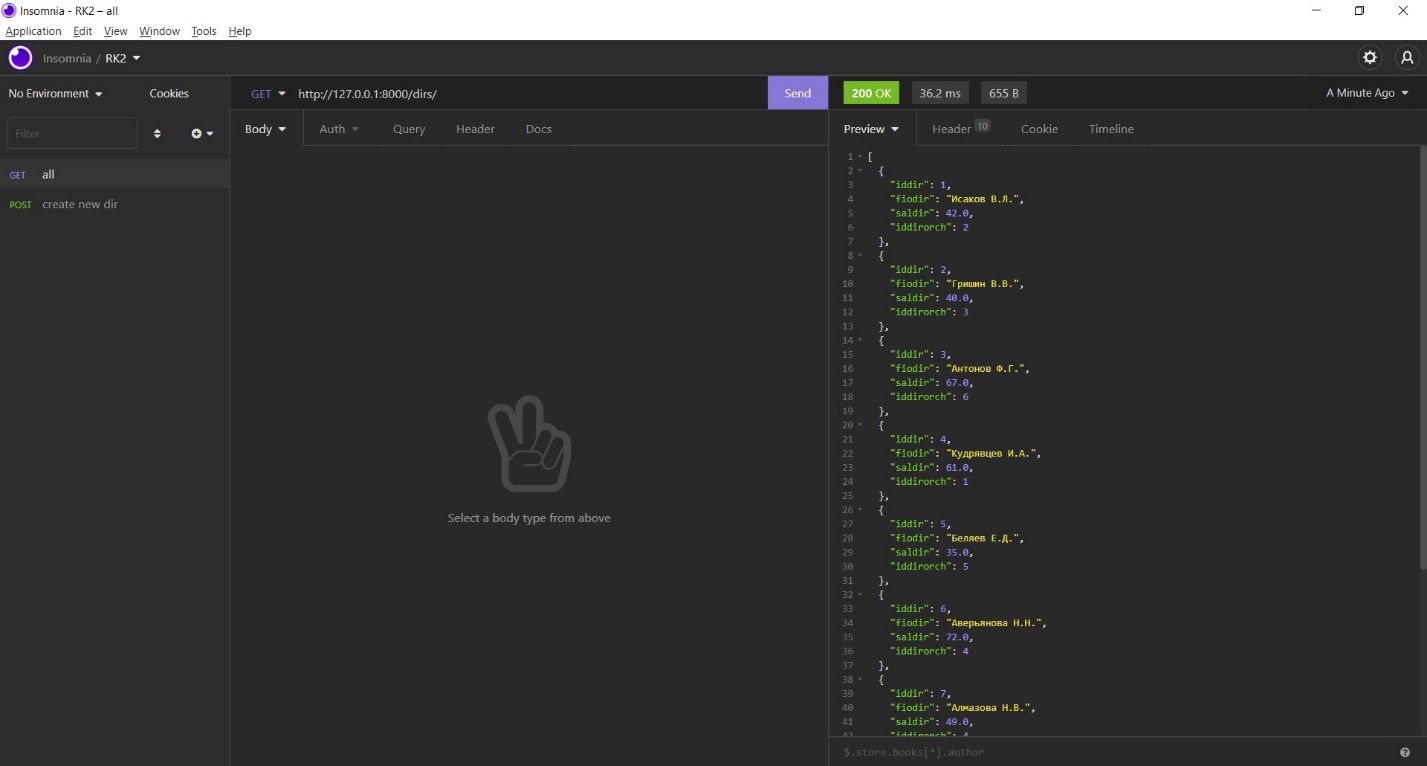


Рис. 3. Список всех дирижеров через Insomnia

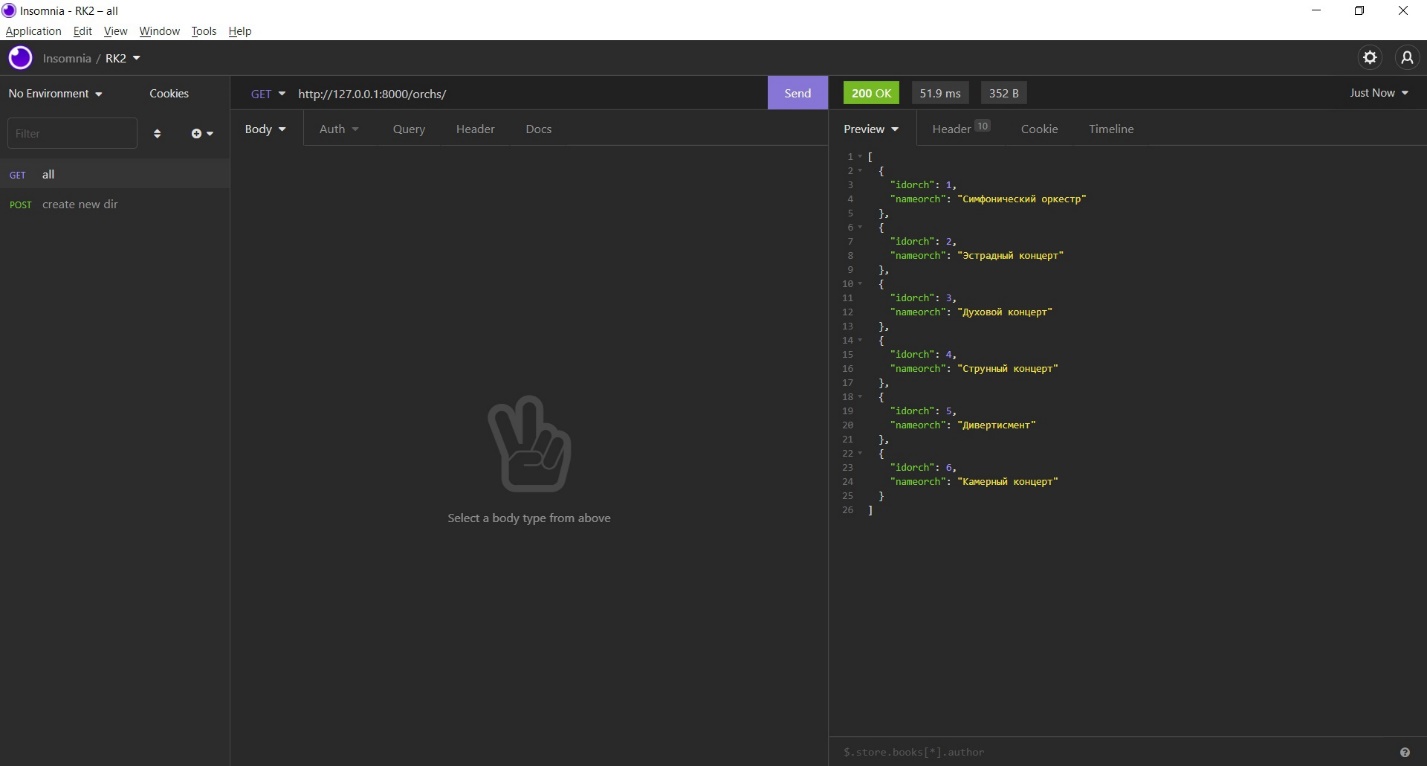
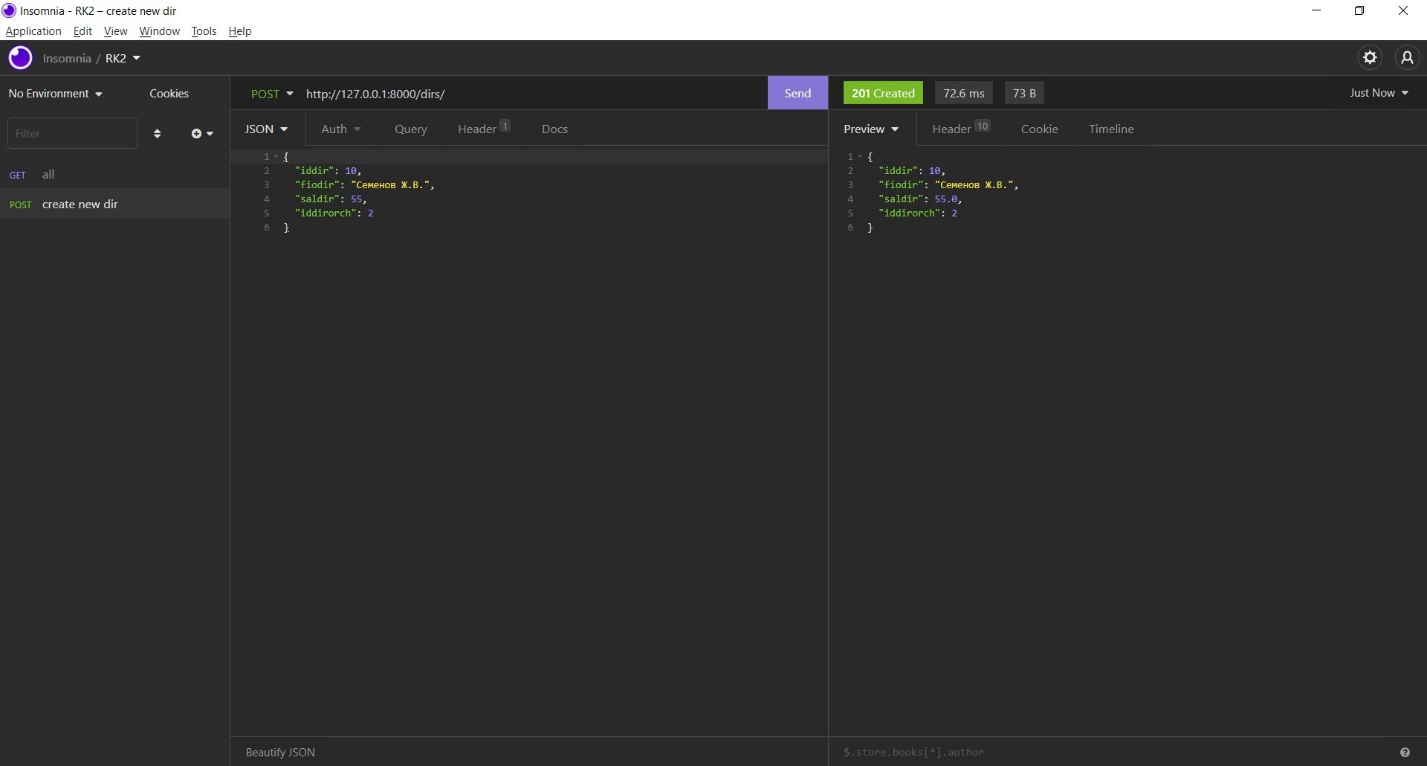


Рис. 4. Список всех оркестров через Insomnia

Рис. 5. Добавление нового дирижера

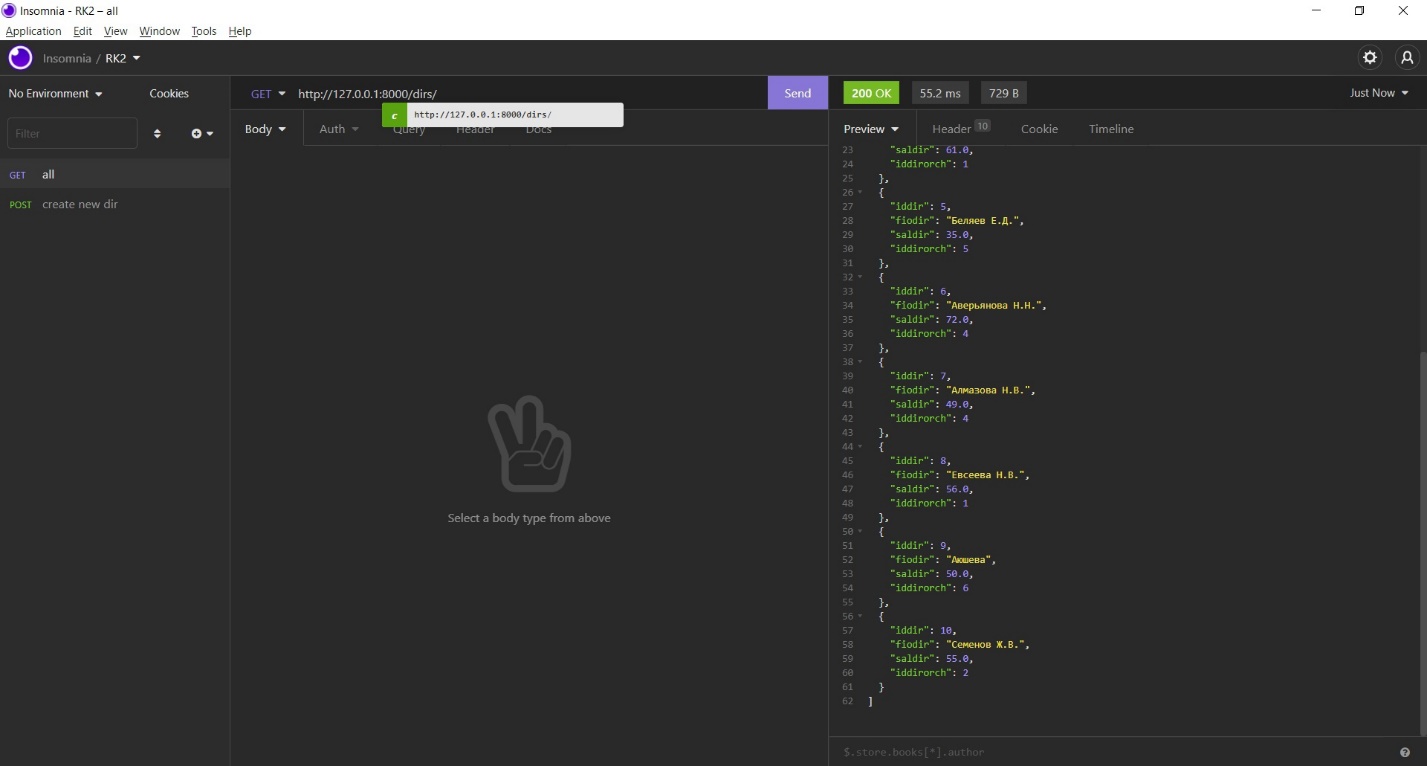


Рис. 6. Список дирижеров после добавления нового

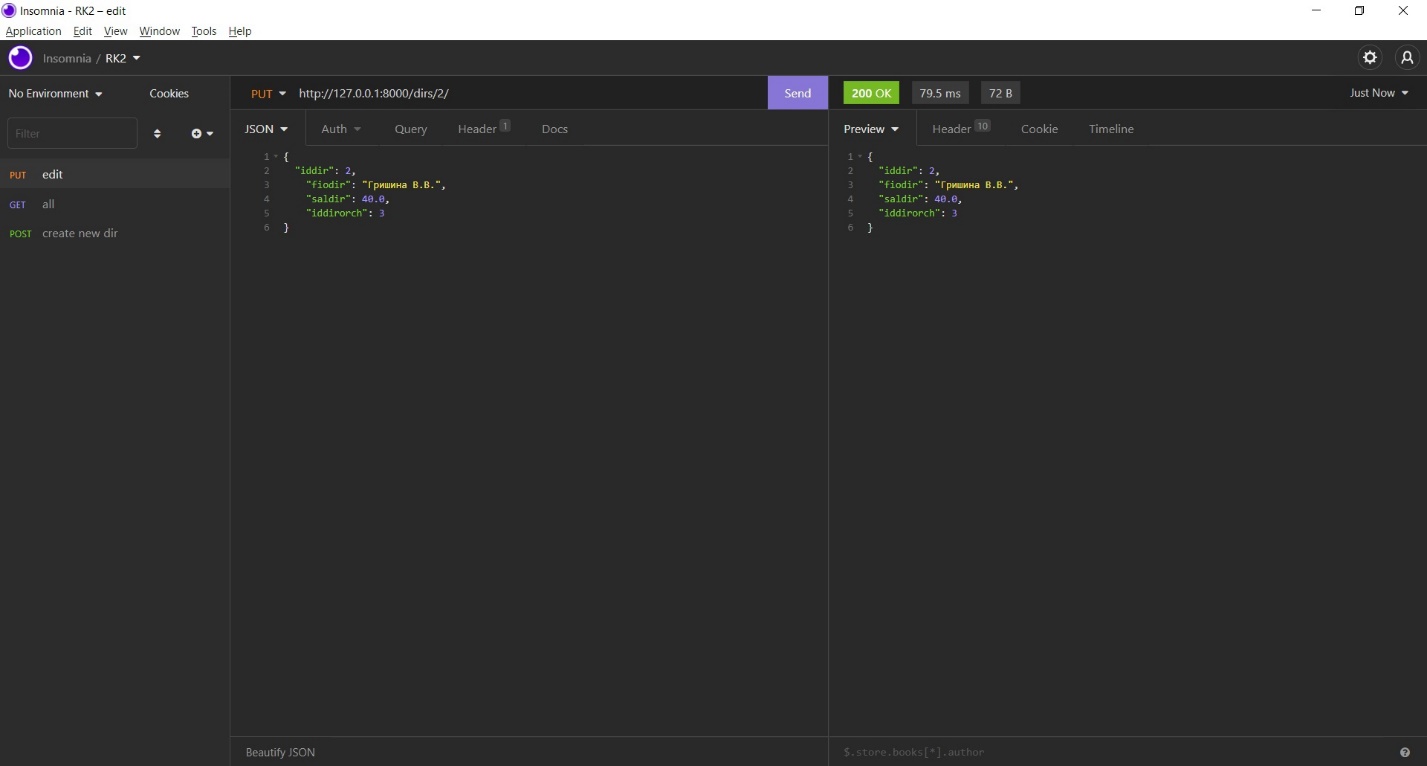


Рис. 7. Изменение 2-ого дирижера

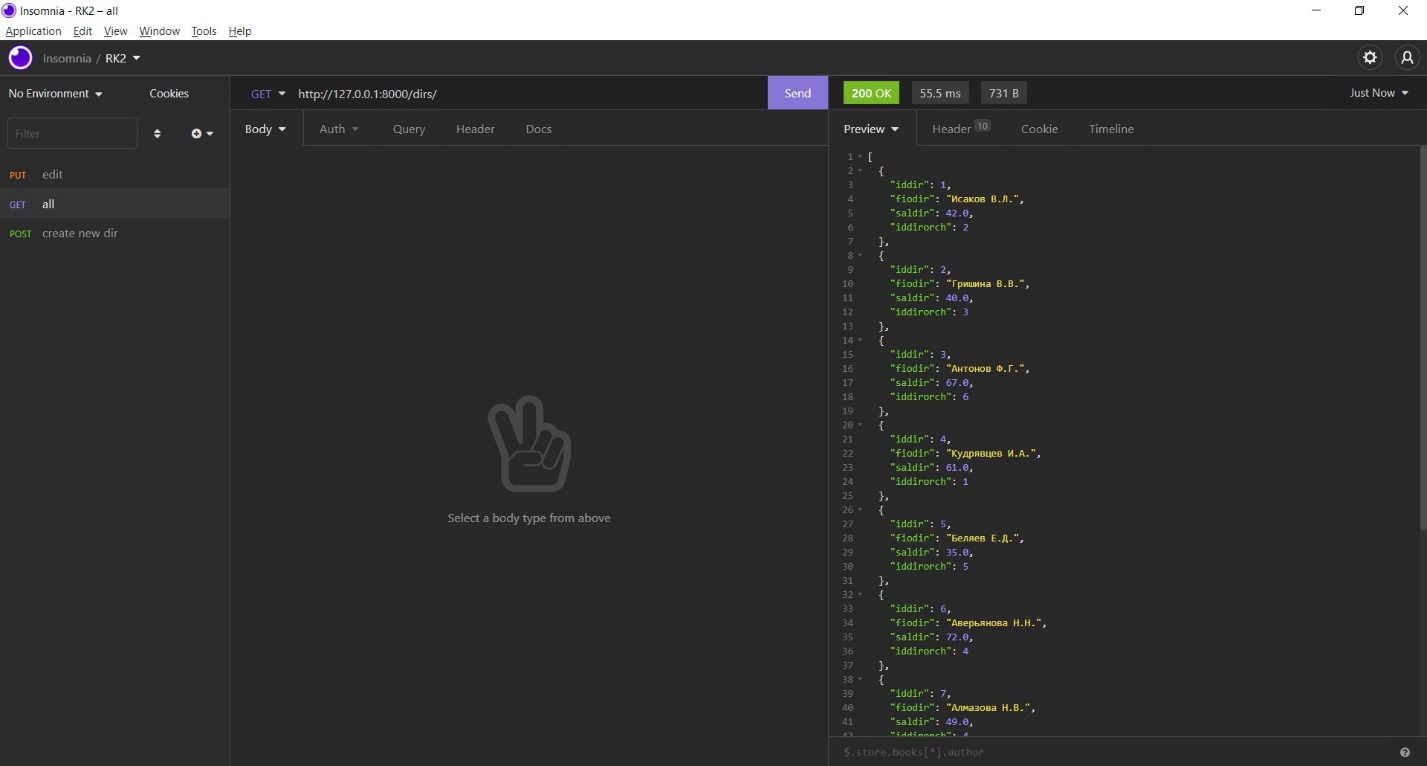


Рис. 8. Список дирижеров после изменения 2 дирижера

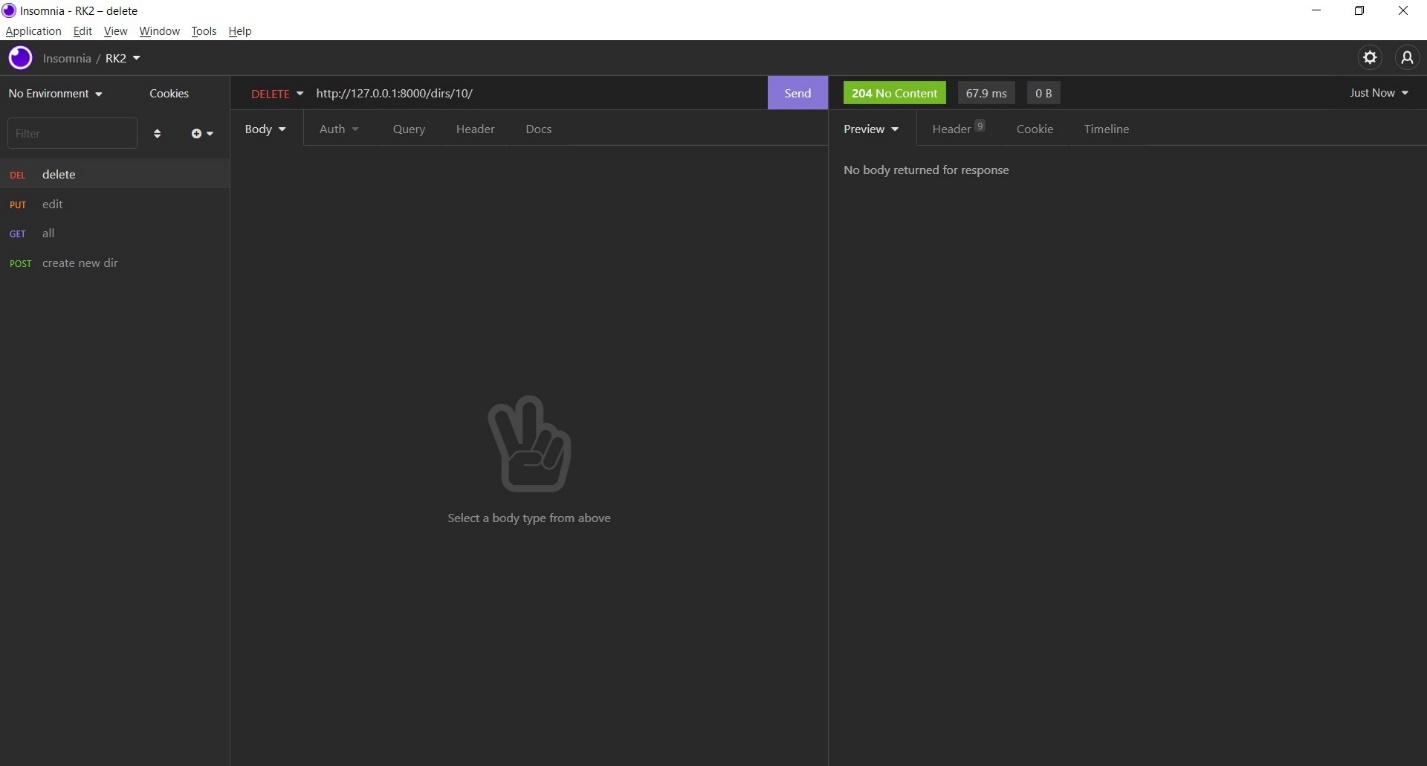


Рис. 9. Удаление 10-ого дирижера

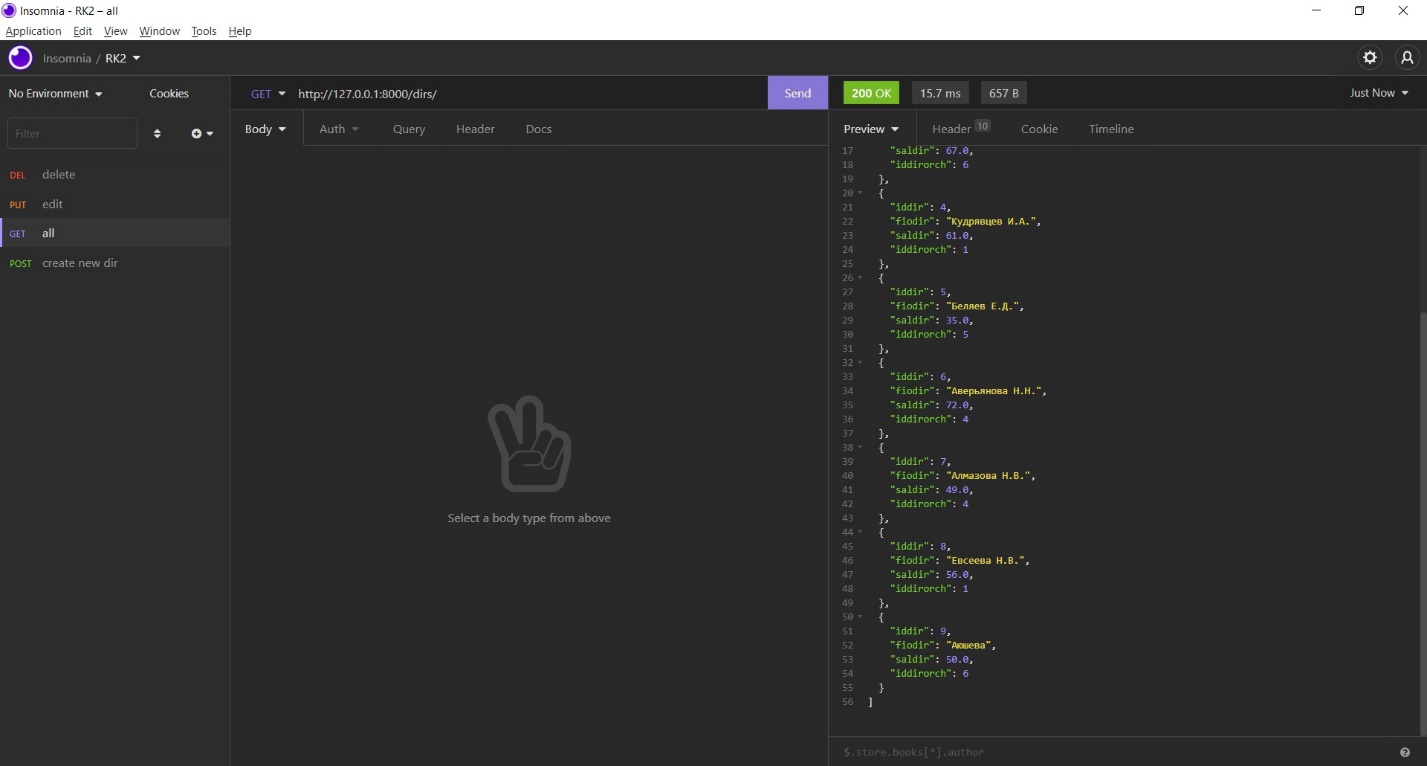


Рис. 10. Список дирижеров после удаления 10-ого дирижера

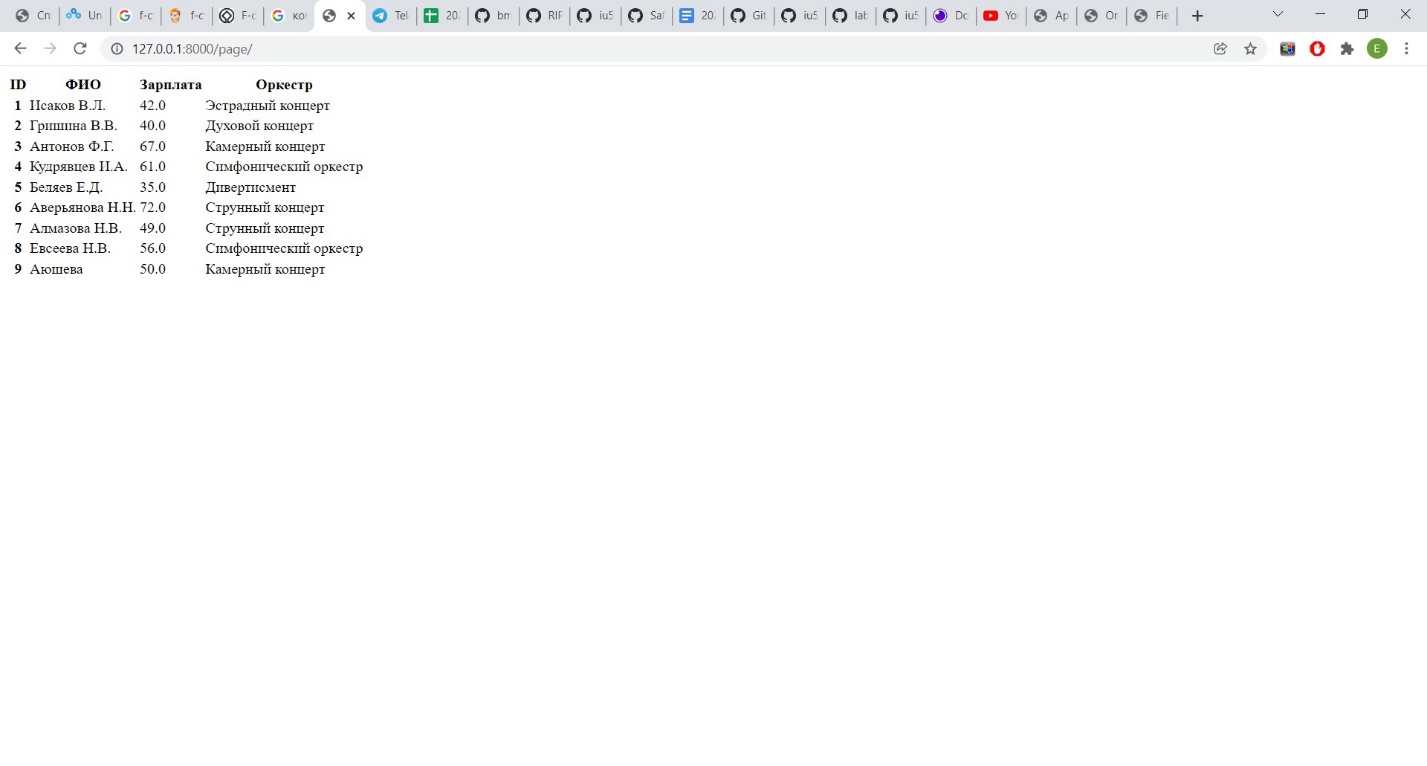


Рис. 11. Отчет с данными из двух таблиц