Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»
Рубежный контроль №2
Вариант Е11

Выполнил: студент группы РТ5-51Б
Незаметдинов Тимур
Подпись и дата:

Проверил: преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Описание задания

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

models.py

```
from django.db import models


class PCs(models.Model):
    name = models.CharField('Название', max_length=30)
    op_system = models.CharField('Операционная система',
max_length=50)


class Programs(models.Model):
    name = models.CharField('Название', max_length=30)
    storage_usage = models.DecimalField('Занимаемый объем
памяти', decimal_places=3, max_digits=10)
    pcs = models.ForeignKey(PCs, models.DO_NOTHING)
```

views.py

```
from django.shortcuts import render
from rest_framework import viewsets
from master.models import PCs, Programs
from master.serializers import PCSerializer,
ProgramSerializer


def index(request):
    return render(request, 'index.html')


class PCViewSet(viewsets.ModelViewSet):
    queryset = PCs.objects.all().order_by("name")
    serializer_class = PCSerializer


class ProgramViewSet(viewsets.ModelViewSet):
    queryset = Programs.objects.all().order_by("name")
    serializer_class = ProgramSerializer


def report(request):
    return render(request, 'report.html', {'data':
{'programs': Programs.objects.select_related('pcs')}})
```

urls.py

```
from django.contrib import admin
from django.urls import include, path
from master.views import PCViewSet, ProgramViewSet
import master.views
from rest_framework import routers

router = routers.DefaultRouter()
router.register(r'PC', PCViewSet)
router.register(r'Program', ProgramViewSet)
```

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

master/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index),
    path('pcs/', views.pcs),
    path('programs/', views.programs),
    path('pc/<int:id>/', views.pcs),
    path('program/<int:id>/', views.programs),
]
```

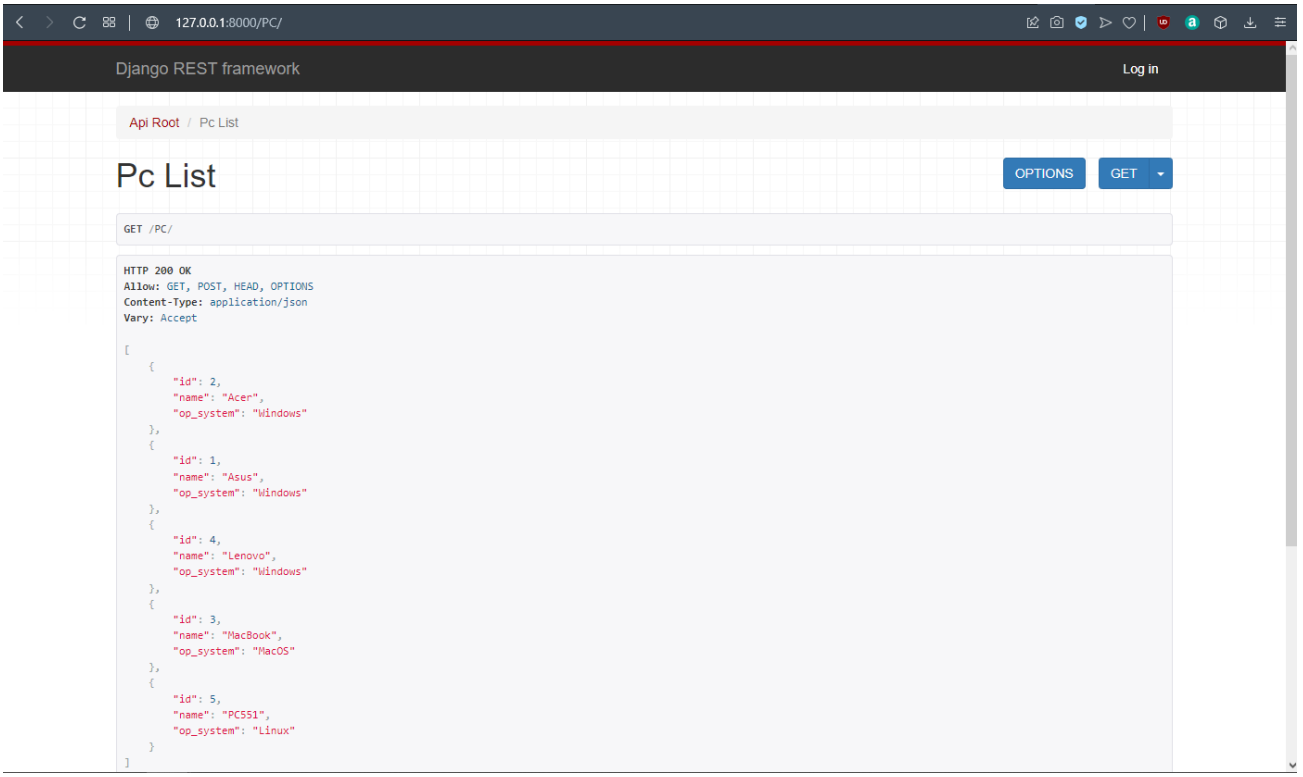
report.html

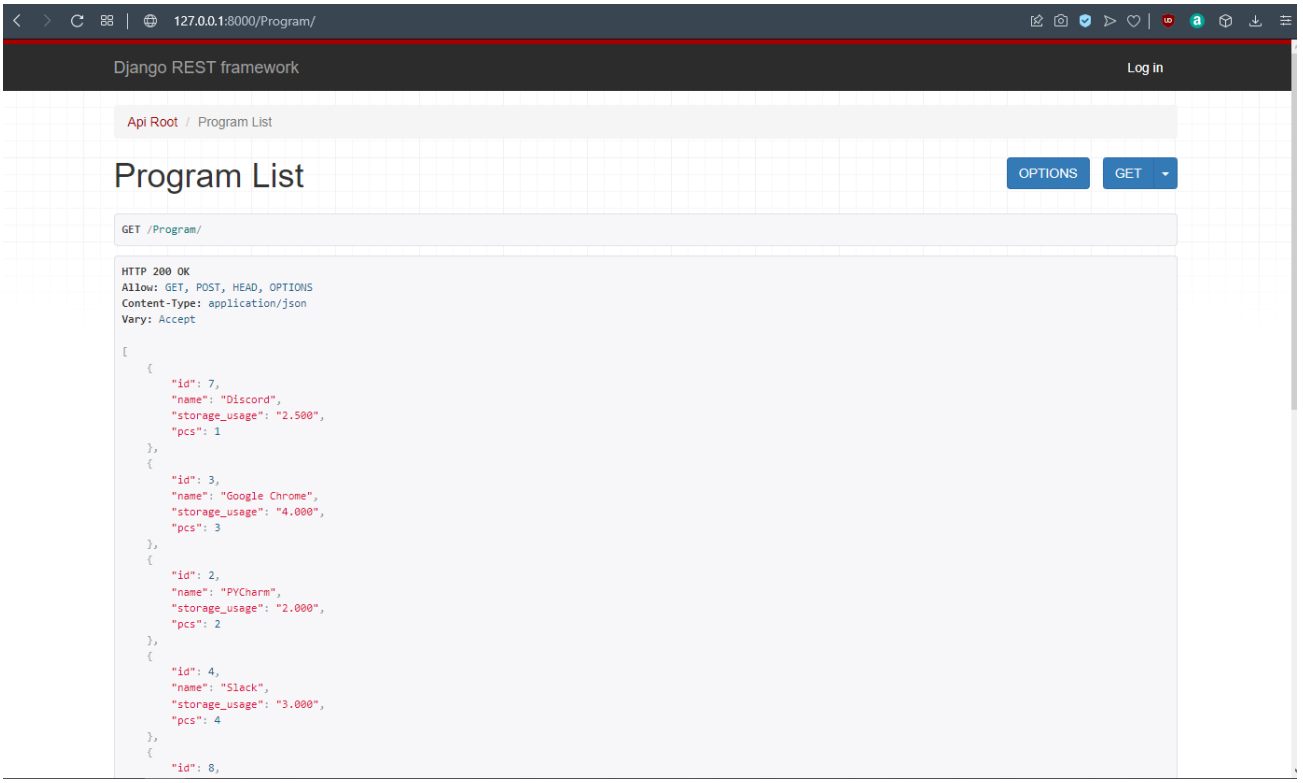
```
{% block title %} Рубежный контроль №2. Незаметдинов
Тимур РТ5-51Б. Вариант Программа - Компьютер {% endblock
%}
{% block body %}
    <div>
    <div>
    <table>
        <thead>
        <tr>
            <th scope="col">ID</th>
            <th scope="col">Название</th>
            <th scope="col">Название компьютера</th>
            <th scope="col">Занимаемая память</th>
```

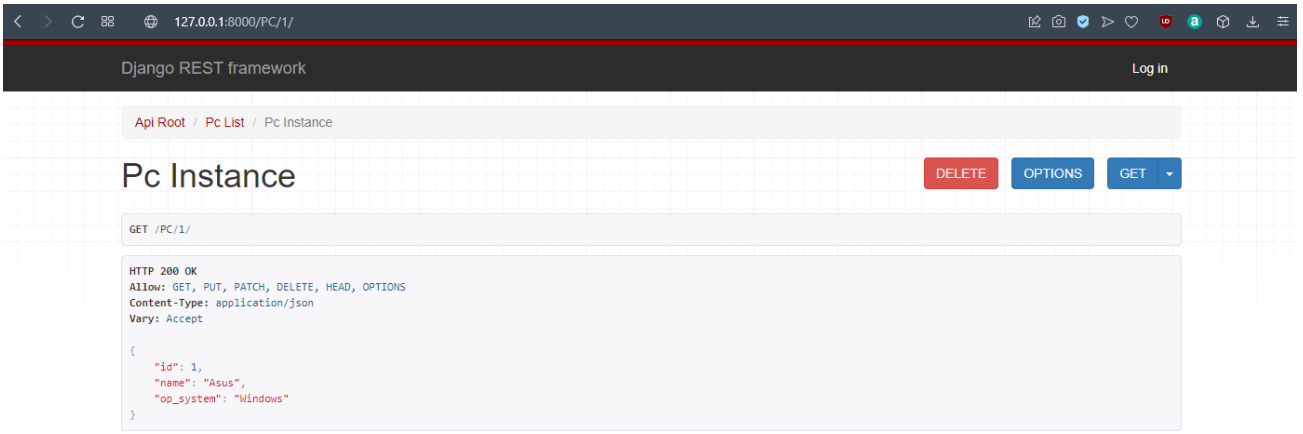
```
            <th scope="col">Операционная система</th>
        </tr>
        </thead>
        <tbody>
        {% for prog in data.programs %}
            <tr>
                <th scope="row">{{ prog.id }}</th>
                <td>{{ prog.name }}</td>
                <td>{{ prog.pcs.name }}</td>
                <td>{{ prog.storage_usage }}</td>
                <td>{{ prog.pcs.op_system }}</td>
            </tr>
        {% endfor %}
        </tbody>
    </table>
    </div>
    </div>
{% endblock %}
```

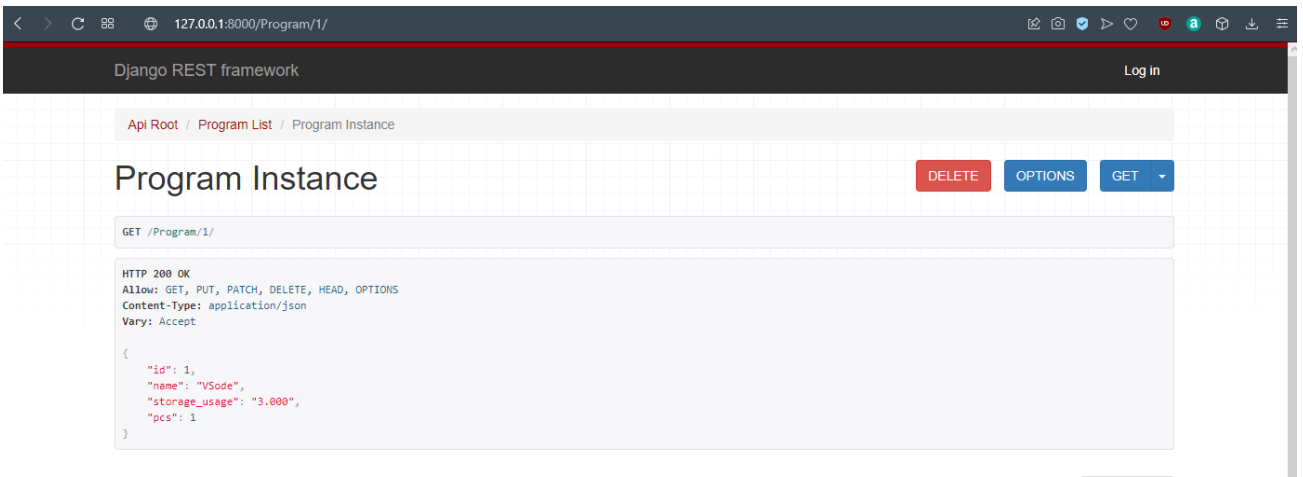
Результат работы программы

Get-запросы:

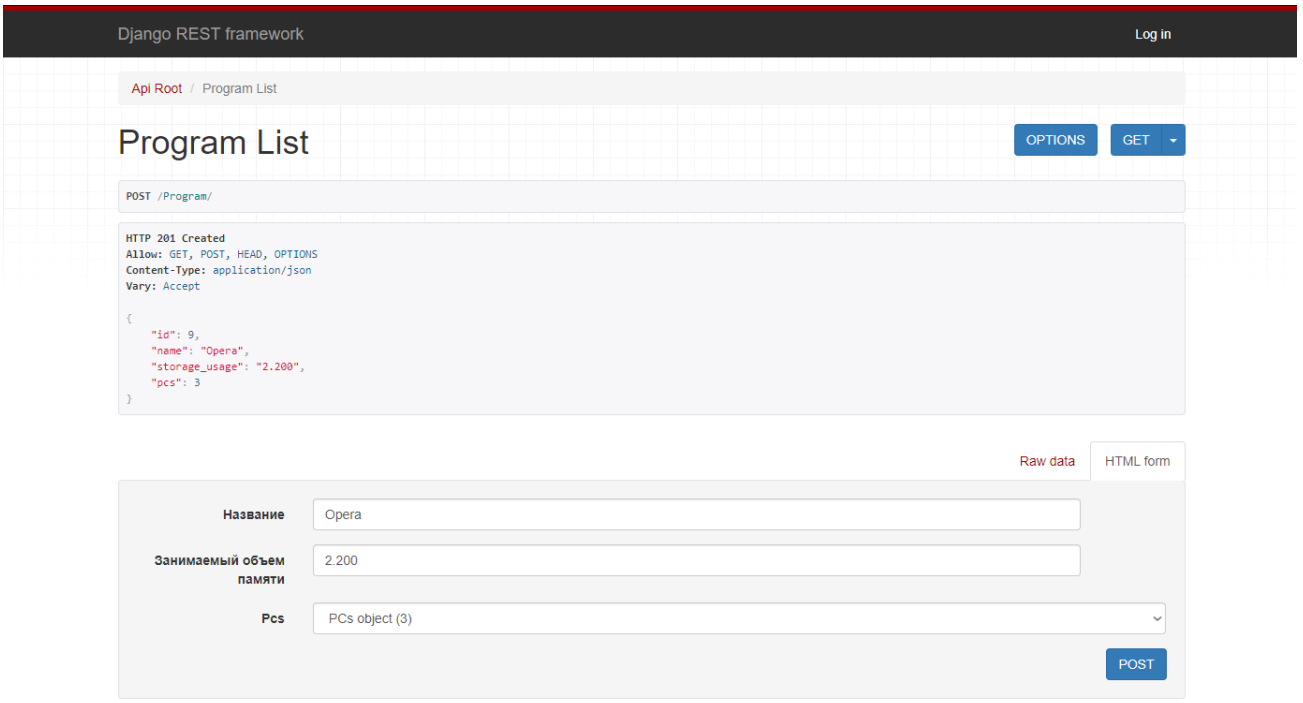


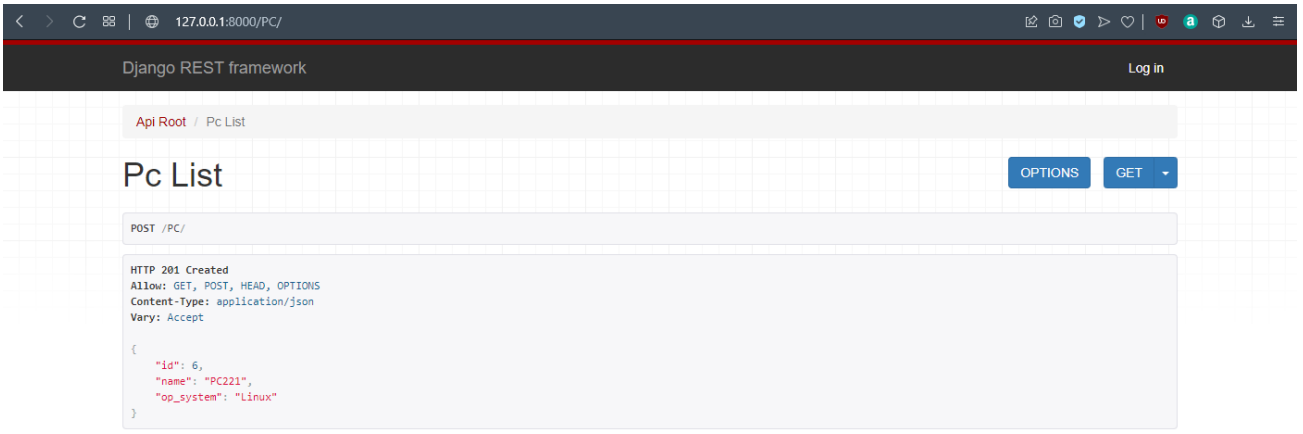

127.0.0.1:8000/Program/
Django REST framework
Log in
Api Root / Program List
Program List
OPTIONS
GET
GET /Program/
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 7,
"name": "Discord",
"storage_usage": "2.500",
"pcs": 1
},
{
"id": 3,
"name": "Google Chrome",
"storage_usage": "4.000",
"pcs": 3
},
{
"id": 2,
"name": "PYCharm",
"storage_usage": "2.000",
"pcs": 2
},
{
"id": 4,
"name": "Slack",
"storage_usage": "3.000",
"pcs": 4
},
{
"id": 8,


127.0.0.1:8000/PC/1/
Django REST framework
Log in
Api Root / Pc List / Pc Instance
Pc Instance
DELETE
OPTIONS
GET
GET /PC/1/
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 1,
"name": "Asus",
"op_system": "Windows"
}

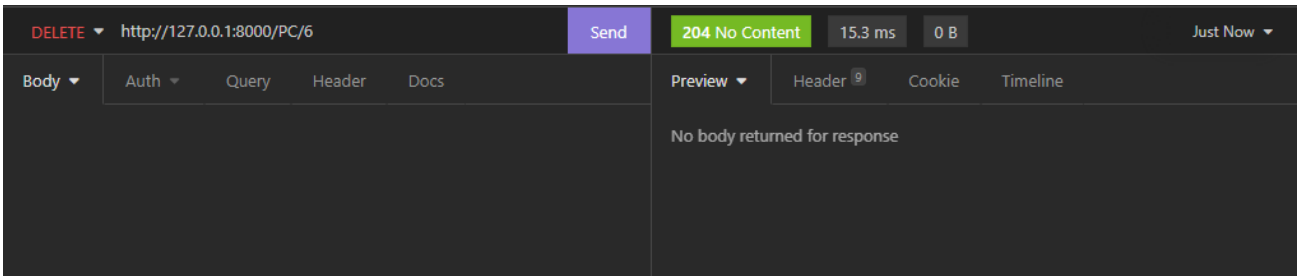

127.0.0.1:8000/Program/1/
Django REST framework
Log in
Api Root / Program List / Program Instance
Program Instance
DELETE
OPTIONS
GET
GET /Program/1/
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 1,
"name": "VSode",
"storage_usage": "3.000",
"pcs": 1
}

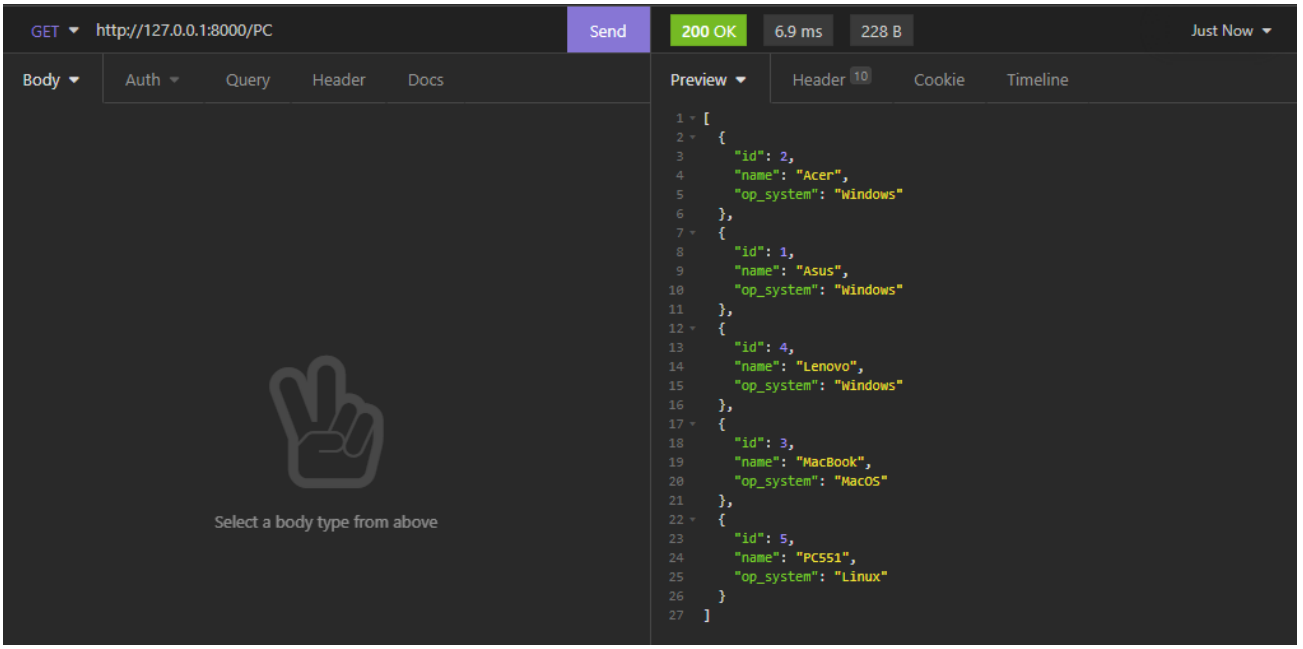
Post-запросы:



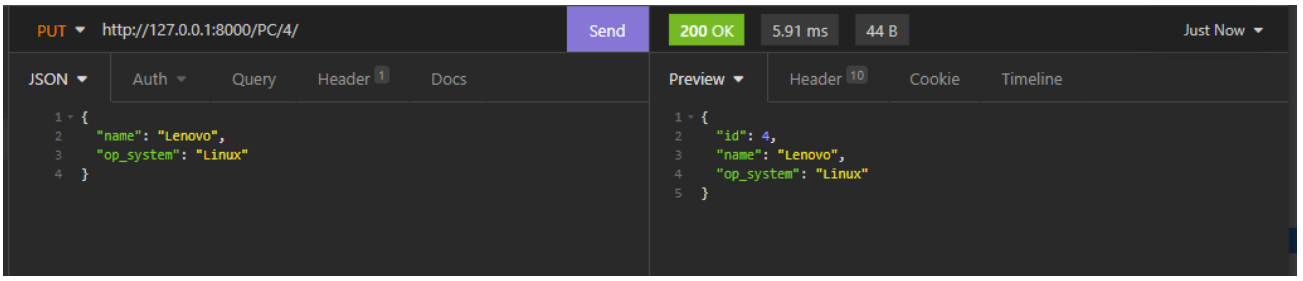


Delete-запрос:

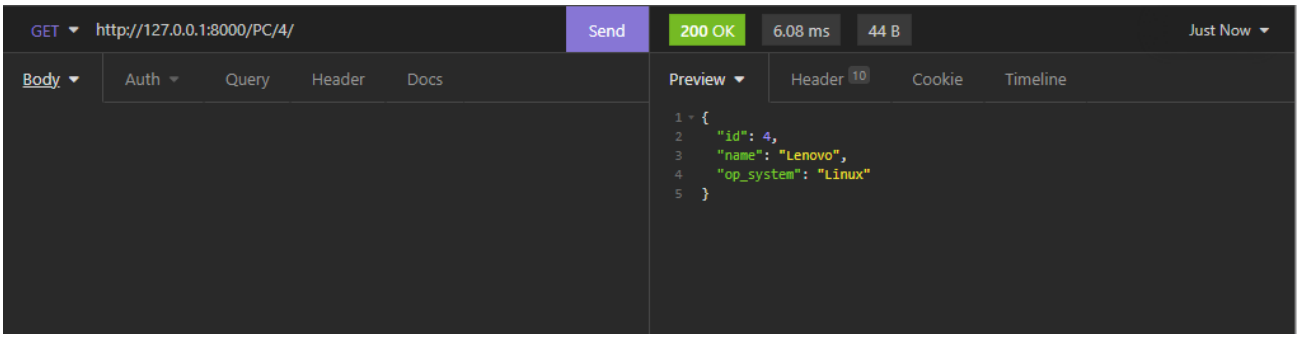




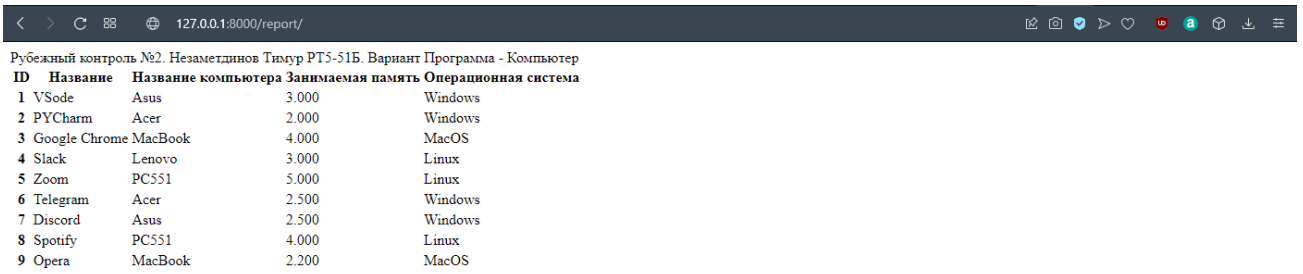
PUT-запросы:



Результат PUT-запроса:



Отчет, соединяющий данные из двух таблиц:



Рубежный контроль №2. Незаметдинов Тимур РТ5-51Б. Вариант Программа - Компьютер

ID	Название	Название компьютера	Занимаемая память	Операционная система
1	VSode	Asus	3.000	Windows
2	PYCharm	Acer	2.000	Windows
3	Google Chrome	MacBook	4.000	MacOS
4	Slack	Lenovo	3.000	Linux
5	Zoom	PC551	5.000	Linux
6	Telegram	Acer	2.500	Windows
7	Discord	Asus	2.500	Windows
8	Spotify	PC551	4.000	Linux
9	Opera	MacBook	2.200	MacOS