

**Московский государственный технический
университет им. Н.Э. Баумана**

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Разработка интернет приложений»

**Отчет по рубежному контролю №2
Вариант В-6**

Выполнил:
студент группы ИУ5-53Б
Ветошкин А.А.,

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2021 г.

Ветошкин Артём Алексеевич ИУ5-53Б

Задание:

Вариант В-6.

Есть сущности "Процессор" и "Компьютер". *Требуется сделать:*

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

Создание моделей и "ручек" для работы с ними

Создание моделей:

```
class PCManager(models.Manager):
    def get_pc_by_processor_id(self, id):
        return self.filter(id_processor=id).all()

class Processor(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=30, blank=True, null=False)
    frequency = models.IntegerField()
    memory_cash = models.IntegerField()

    class Meta:
        db_table = 'processor'

class PC(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=30, blank=True, null=False)
    price = models.IntegerField()
    id_processor = models.ForeignKey('Processor', models.DO_NOTHING,
db_column='id_processor', blank=True, null=True)

    objects = PCManager()

    class Meta:
```

```
db_table = 'pc'
```

Для возможности добавления, редактирования и удаления данных были использован `rest_framework`.

Соответственно были прописаны сериализаторы:

```
class PCSerializer(serializers.ModelSerializer):
    class Meta:
        model = models.PC
        fields = ["id", "name", "price", "id_processor"]

class ProcessorSerializer(serializers.ModelSerializer):
    class Meta:
        model = models.Processor
        fields = ["id", "name", "frequency", "memory_cash"]
```

И представления:

```
class PCViewSet(viewsets.ModelViewSet):
    queryset = models.PC.objects.all()
    serializer_class = serializers.PCSerializer

class ProcessorViewSet(viewsets.ModelViewSet):
    queryset = models.Processor.objects.all()
    serializer_class = serializers.ProcessorSerializer
```

И настроен роутер:

```
router = routers.DefaultRouter()
router.register('processor', views.ProcessorViewSet)
router.register('pc', views.PCViewSet)
```

Проверка выполнения запросов на созданные ручки

Для записей о компьютерах

Получение всех компьютеров GET запросом:

http://127.0.0.1:8000/pc/

GET http://127.0.0.1:8000/pc/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 [{"name": "Intel i5-9600",
2    "frequency": 3600,
3    "memory_cash": 128
4  },
5  ]
```

Body Cookies Headers (10) Test Results Status: 200 OK Time: 98 ms Size: 880 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "name": "Lenovo IdeaCentre G5 14IMB05",
4   "price": 88000,
5   "id_processor": 8
6 },
7 {
8   "id": 2,
9   "name": "HP Pavilion Gaming TG01",
10  "price": 110000,
11  "id_processor": 1
12 },
13 {
14   "id": 3,
15   "name": "ZOTAC MAGNUS ONE",
16   "price": 187000
17 }
```

Получение конкретного компьютера GET запросом:

http://127.0.0.1:8000/pc/8/

GET http://127.0.0.1:8000/pc/8/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

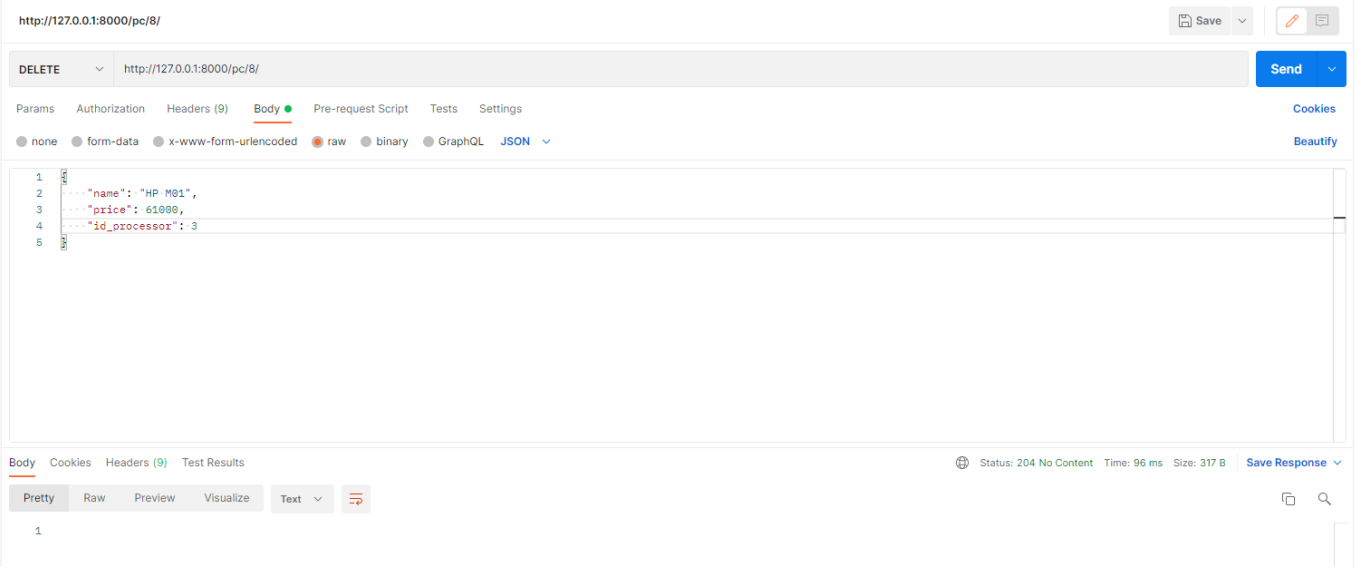
```
1 [{"name": "Intel i5-9600",
2    "frequency": 3600,
3    "memory_cash": 128
4  },
5  ]
```

Body Cookies Headers (10) Test Results Status: 200 OK Time: 250 ms Size: 397 B Save Response

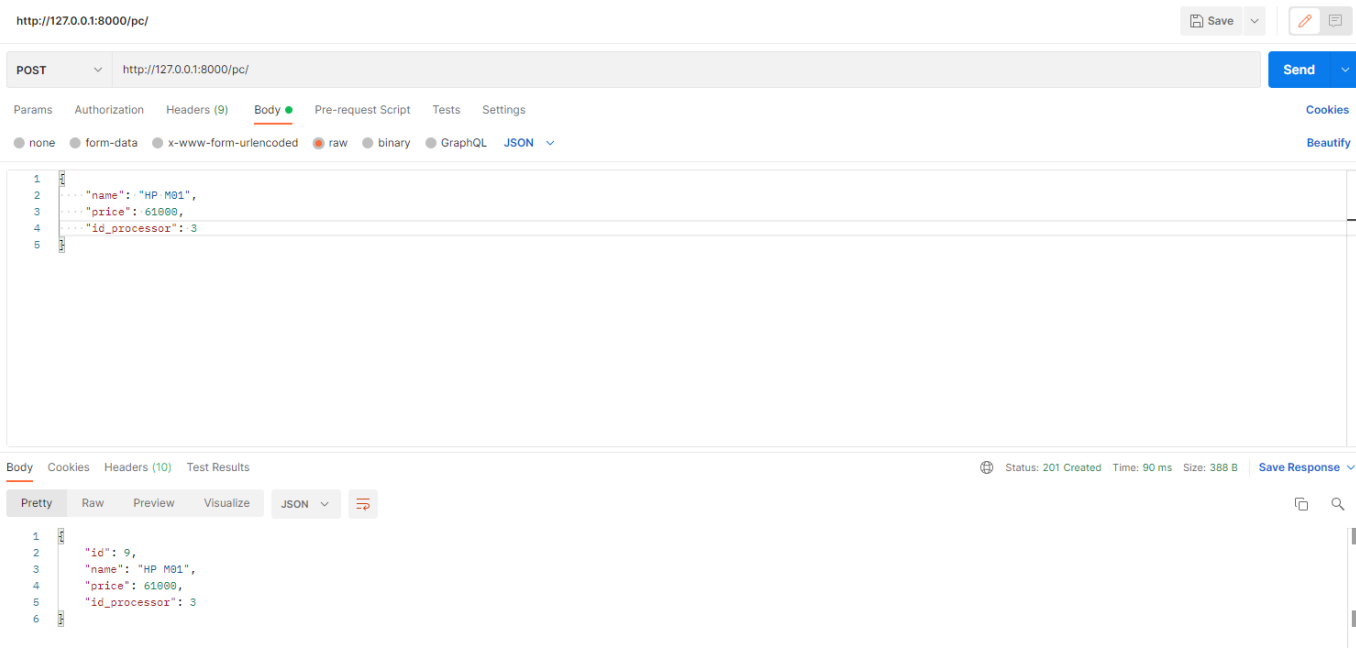
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 8,
3   "name": "HP M01",
4   "price": 61000,
5   "id_processor": 3
6 }
```

Удаление конкретного компьютера DELETE запросом:



Создание записи о компьютере POST запросом:



Для записей о процессорах

Получение всех процессоров GET запросом:

http://127.0.0.1:8000/processor/

Save

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

```
1 GET
2 ..... "name": "Intel Pentium",
3 ..... "frequency": 3000,
4 ..... "memory_cash": 32
5 POST
```

BodyCookiesHeaders (10)Test Results

64 "name": "Ryzen 9 5900X",
65 "frequency": 3700,
66 "memory_cash": 32
67 },
68 {
69 "id": 12,
70 "name": "Ryzen 9 5900X",
71 "frequency": 3400,
72 "memory_cash": 64
73 },
74 {
75 "id": 13,
76 "name": "Intel Pentium",
77 "frequency": 3000,
78 "memory_cash": 32
79 }
80]

Status: 200 OKTime: 62 msSize: 1.17 KBSave Response

PrettyRawPreviewVisualizeJSON

Получение конкретного процессора GET запросом:

http://127.0.0.1:8000/processor/13/

Save

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

```
1 GET
2 ..... "name": "Intel Pentium",
3 ..... "frequency": 3000,
4 ..... "memory_cash": 32
5 POST
```

BodyCookiesHeaders (10)Test Results

1 "id": 13,
2 "name": "Intel Pentium",
3 "frequency": 3000,
4 "memory_cash": 32
5
6 POST

Status: 200 OKTime: 77 msSize: 408 BSave Response

PrettyRawPreviewVisualizeJSON

Удаление конкретного процессора DELETE запросом:

http://127.0.0.1:8000/processor/13/

DELETE http://127.0.0.1:8000/processor/13/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "Intel Pentium",
3   "frequency": 3000,
4   "memory_cache": 32
5 }
```

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize Text

Status: 204 No Content Time: 124 ms Size: 317 B Save Response

Создание записи о процессоре **POST** запросом:

http://127.0.0.1:8000/processor/

POST http://127.0.0.1:8000/processor/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "Intel Pentium",
3   "frequency": 3000,
4   "memory_cache": 32
5 }
```

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 13,
3   "name": "Intel Pentium",
4   "frequency": 3000,
5   "memory_cache": 32
6 }
```

Status: 201 Created Time: 79 ms Size: 399 B Save Response

Создание отчёта

Для показа отчёта было добавлено представление:

```
def index(request):
    processors = models.Processor.objects.all()
    data = []
    for processor in processors:
        data.append((processor,
models.PC.objects.get_pc_by_processor_id(processor.id)))

    return render(request, "index.html", {"data": data})
```

И сверстана страница:

```

<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Report</title>
  <style>
    table.collapse {
      border: 2px solid black;
      border-collapse: collapse;
      margin-top: 10px;
      display: inline-block;
    }

    td.upper {
      border: 2px solid black;
      padding: 10px;
    }

    td {
      border: 2px solid black;
      padding: 5px;
    }

    .center-text {
      text-align: center;
    }
  </style>
</head>
<body>
<h1 class="center-text"> Отчёт по данным базы о процессорах и компьютерах</h1>
<div class="center-text">
  {% for value in data %}
    <table class="collapse center-text">
      <tr>
        <td class="upper" colspan="12">Процессор</td>
      </tr>
      <tr>
        <td class="upper" colspan="3">ID</td>
        <td class="upper" colspan="3">Название</td>
        <td class="upper" colspan="3">Частота</td>
        <td class="upper" colspan="3">Размер кэша</td>
      </tr>
      <tr>
        <td class="upper" colspan="3">{{ value.0.id }}</td>
        <td class="upper" colspan="3">{{ value.0.name }}</td>
        <td class="upper" colspan="3">{{ value.0.frequency }}</td>
        <td class="upper" colspan="3">{{ value.0.memory_cash }}</td>
      </tr>
      <tr>
        <td class="upper" colspan="12">Используется в компьютерах</td>
      </tr>
      <tr>
        <td colspan="4">ID</td>

```



```

        <td colspan="4">Название</td>
        <td colspan="4">Цена</td>
    </tr>
    {% for pc in value.1 %}
        <tr>
            <td colspan="4">{{ pc.id }}</td>
            <td colspan="4">{{ pc.name }}</td>
            <td colspan="4">{{ pc.price }}</td>
        </tr>
    {% empty %}
        <tr>
            <td colspan="12">Не найдено</td>
        </tr>
    {% endfor %}
</table>
{% endfor %}
</div>
</body>
</html>

```

Для заполнения данных был написан sql скрипт:

```

INSERT INTO processor (name, frequency, memory_cash) VALUES
('Intel i7-10600', 3700, 128),
('Intel i5-6500', 3600, 12),
('Intel i5-6600', 3700, 32),
('Intel i7-8600', 3500, 64),
('Intel i7-10600', 3800, 256),
('Intel i3-5600', 3500, 128),
('Intel i7-4790', 3200, 128),
('Ryzen 5 5600X', 3800, 32),
('Ryzen 7 5800X', 3700, 32),
('Ryzen 9 5900X', 3700, 32),
('Ryzen 9 5950X', 3400, 64);

INSERT INTO PC (name, price, id_processor) VALUES
('Lenovo IdeaCentre G5 14IMB05', 88000, 8),
('HP Pavilion Gaming TG01', 110000, 1),
('ZOTAC MAGNUS ONE', 187000, 7),
('Apple Mac mini 2020 M1', 173000, 6),
('Acer Aspire TC-895', 34000, 7),
('Apple iMac 24 2021', 248000, 11),
('Gigabyte GB-BR', 93000, 4),
('HP M01', 61000, 3);

```

← ↻ 127.0.0.1:8000 Report ● 207 отзывов

Отчёт по данным базы о процессорах и компьютерах

Процессор				Процессор				Процессор				Процессор				Процессор			
ID	Название	Частота	Размер кэша	ID	Название	Частота	Размер кэша	ID	Название	Частота	Размер кэша	ID	Название	Частота	Размер кэша	ID	Название	Частота	Размер кэша
1	Intel i5-9600	3600	128	2	Intel i7-10600	3700	128	3	Intel i5-6500	3600	12	4	Intel i5-6600	3700	32	5	Intel i7-8600	3500	64
Используется в компьютерах				Используется в компьютерах				Используется в компьютерах				Используется в компьютерах				Используется в компьютерах			
ID	Название	Цена		ID	Название	Цена		ID	Название	Цена		ID	Название	Цена		ID	Название	Цена	
2	HP Pavilion Gaming TG01	110000			Не найдено			9	HP M01	61000		7	Gigabyte GB-BR	93000			Не найдено		

Процессор			
ID	Название	Частота	Размер кэш
6	Intel i7-10600	3800	256
Используется в компьютерах			
ID	Название	Цена	
4	Apple Mac mini 2020 M1	173000	

Процессор			
ID	Название	Частота	Размер кэш
7	Intel i3-5600	3500	128
Используется в компьютерах			
ID	Название	Цена	
3	ZOTAC MAGNUS ONE	187000	
5	Acer Aspire TC-895	34000	

Процессор			
ID	Название	Частота	Размер кэш
8	Intel i7-4790	3200	128
Используется в компьютерах			
ID	Название	Цена	
1	Lenovo IdeaCentre G5 I41MB05	88000	

Процессор			
ID	Название	Частота	Размер кэш
9	Ryzen 5 5600X	3800	32
Используется в компьютерах			
ID	Название	Цена	
	Не найдено		

Процессор				Процессор				Процессор				
ID	Название	Частота	Размер кэша	ID	Название	Частота	Размер кэша	ID	Название	Частота	Размер кэша	
10	Ryzen 7 5800X	3700	32	11	Ryzen 9 5900X	3700	32	12	Ryzen 9 5950X	3400	64	
Используется в компьютерах				Используется в компьютерах				Используется в компьютерах				
ID	Название	Цена		ID	Название	Цена		ID	Название	Цена		
Не найдено				6	Apple iMac 24 2021	248000		Не найдено				