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 лабораторной работе №5
«Работа с СУБД. Обработка данных с использованием Django ORM.»

Выполнил:
студент группы ИУ5-52Б
Барышников Михаил

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2021 г.

Описание задания:

В этой лабораторной работе Вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также Вам нужно будет дополнить свои классы предметной области, связав их с созданной БД. После этого Вы создадите свои модели с помощью Django ORM, отобразите объекты из БД с помощью этих моделей.

1. Создайте сценарий с подключением к БД и несколькими запросами, примеры рассмотрены в методических указаниях.
2. Реализуйте модели Вашей предметной области из предыдущей ЛР (минимум две модели, т.е. две таблицы).
3. Создайте представления и шаблоны Django для отображения списка данных по каждой из сущностей.

Создание таблиц базы данных:

```
USE [Computer]
GO

/****** Object:  Table [dbo].[CPU]    Script Date: 15.12.2021 18:29:21 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[CPU](
    [ID] [bigint] IDENTITY(1,1) NOT NULL,
    [Name] [nchar](30) NOT NULL,
    [Frequency] [nchar](10) NULL,
    [Architecture] [nchar](20) NULL,
 CONSTRAINT [PK_CPU] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
) ON [PRIMARY]
GO
```

```
USE [Computer]
GO

/****** Object:  Table [dbo].[DiskStorage]    Script Date: 15.12.2021 18:31:17 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DiskStorage](
    [ID] [bigint] IDENTITY(1,1) NOT NULL,
    [Type] [nchar](20) NOT NULL,
    [RS] [nchar](20) NULL,
    [WS] [nchar](20) NULL,
    [Volume] [nchar](20) NOT NULL,
 CONSTRAINT [PK_DiskStorage] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
) ON [PRIMARY]
GO
```

```
USE [Computer]
GO

/****** Object:  Table [dbo].[RAM]    Script Date: 15.12.2021 18:31:11 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[RAM](
    [ID] [bigint] IDENTITY(1,1) NOT NULL,
    [Name] [nchar](50) NULL,
    [Type] [nchar](20) NOT NULL,
    [Frequency] [nchar](20) NOT NULL,
    [Volume] [tinyint] NULL,
 CONSTRAINT [PK_RAM] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
) ON [PRIMARY]
GO
```

```
USE [Computer]
GO

/****** Object:  Table [dbo].[Hardware]    Script Date: 15.12.2021 18:29:04 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Hardware](
    [ID] [bigint] IDENTITY(1,1) NOT NULL,
    [CPU_ID] [bigint] NOT NULL,
    [RAM_ID] [bigint] NOT NULL,
    [DiskStor_ID] [bigint] NOT NULL,
 CONSTRAINT [PK_Hardware] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Hardware]  WITH CHECK ADD FOREIGN KEY([CPU_ID])
REFERENCES [dbo].[CPU] ([ID])
GO

ALTER TABLE [dbo].[Hardware]  WITH CHECK ADD FOREIGN KEY([DiskStor_ID])
REFERENCES [dbo].[DiskStorage] ([ID])
GO

ALTER TABLE [dbo].[Hardware]  WITH CHECK ADD FOREIGN KEY([RAM_ID])
REFERENCES [dbo].[RAM] ([ID])
GO
```

Скрипт подключения к базе данных, добавления записей в таблицу и выборки данных:

```
import pyodbc

connectionString = ("Driver={SQL Server Native Client 11.0};"
                    "Server=localhost\\MSSQLSERVERDEV;"
                    "Database=Computer;"
                    "Trusted_Connection=yes;")

request1 = """INSERT INTO dbo.CPU (Name, Frequency, Architecture) VALUES
('Intel core i5', '5Ghz', 'x86');"""

request2 = "SELECT * FROM dbo.CPU"

connection = pyodbc.connect(connectionString, autocommit=True)
dbCursor = connection.cursor()
#dbCursor.execute(request1)
dbCursor.execute(request2)
for row in dbCursor:
    print(f"{row.ID} {row.Name} {row.Frequency} {row.Architecture}")
```

```
connection.commit()
dbCursor.close()
connection.close()
```

Результат:

```
"D:\bmstu_studing\5 sem\RIP\lab5\Scripts\python.exe" "D:/bmstu_studing/5 sem/RIP/lab5/computercomponents/connection_script.py"
1 Intel core i5                 5Ghz       x86
2 Intel core i3                 3.2Ghz     x86
3 Apple M1 PRO                  3.9Mhz     arm
4 Ryzen 5                       4.6Ghz     x86
```

Файл main/models.py с созданием классов моделей таблиц на основе кода, сгенерированного с помощью команды «python manage.py inspectdb»:

```
from django.db import models


class Cpu(models.Model):
    id = models.BigAutoField(db_column='ID', primary_key=True)  # Field
name made lowercase.
    name = models.CharField(db_column='Name', max_length=30)  # Field
name made lowercase.
    frequency = models.CharField(db_column='Frequency', max_length=10,
blank=True, null=True)  # Field name made lowercase.
    architecture = models.CharField(db_column='Architecture',
max_length=20, blank=True, null=True)  # Field name made lowercase.

    class Meta:
        db_table = 'CPU'


class Diskstorage(models.Model):
    id = models.BigAutoField(db_column='ID', primary_key=True)  # Field
name made lowercase.
    type = models.CharField(db_column='Type', max_length=20)  # Field
name made lowercase.
    rs = models.CharField(db_column='RS', max_length=20, blank=True,
null=True)  # Field name made lowercase.
    ws = models.CharField(db_column='WS', max_length=20, blank=True,
null=True)  # Field name made lowercase.
```

```
    volume = models.CharField(db_column='Volume', max_length=20)  # Field
name made lowercase.

    class Meta:
        db_table = 'DiskStorage'


class Hardware(models.Model):
    id = models.BigAutoField(db_column='ID', primary_key=True)  # Field
name made lowercase.
    cpu = models.ForeignKey(Cpu, models.DO_NOTHING, db_column='CPU_ID')
# Field name made lowercase.
    ram = models.ForeignKey('Ram', models.DO_NOTHING, db_column='RAM_ID')
# Field name made lowercase.
    diskstor = models.ForeignKey(Diskstorage, models.DO_NOTHING,
db_column='DiskStor_ID')  # Field name made lowercase.

    class Meta:
        db_table = 'Hardware'


class Ram(models.Model):
    id = models.BigAutoField(db_column='ID', primary_key=True)  # Field
name made lowercase.
    name = models.CharField(db_column='Name', max_length=50, blank=True,
null=True)  # Field name made lowercase.
    type = models.CharField(db_column='Type', max_length=20)  # Field
name made lowercase.
    frequency = models.CharField(db_column='Frequency', max_length=20)  #
Field name made lowercase.
    volume = models.SmallIntegerField(db_column='Volume', blank=True,
null=True)  # Field name made lowercase.

    class Meta:
        db_table = 'RAM'
```

Файл computercomponents/urls.py:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls'))
]
```

Файл main/urls.py:

```
from django.urls import path
```

```
from . import views

urlpatterns = [
    path('', views.index),
    path('<str:model_name>/', views.list, name='list')
]
```

Файл main/views.py:

```
from django.shortcuts import render
from .models import *
from django.apps import apps


def index(request):
    models = apps.get_app_config('main').get_models()
    models_names = [model._meta.db_table for model in models]
    models_dict = {model_id: models_name for model_id, models_name in
zip(range(len(models_names)), models_names)}
    params = {'models_dict': models_dict}
    return render(request, 'main/index.html', params)


def list(request, model_name):
    models = apps.get_app_config('main').get_models()
    model = ''
    for elem in models:
        if elem._meta.db_table == model_name:
            model = elem
    params = {'model_name': model._meta.db_table, 'objects':
model.objects.values()}
    return render(request, 'main/list.html', params)
```

Файл main/templates/main/base.html:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <a href='/'>Главная</a>
    {% block content %}{% endblock %}
</body>
```

```
</html>
```

Файл main/templates/main/index.html:

```
{% extends 'main/base.html' %}


{% block title %}
Главная
{% endblock %}


{% block content %}
<h2>Подключенная база данных содержит следующие сущности:</h2>
<ol>
    {% for model_id, model_name in models_dict.items %}
    <li><a href="{% url 'list' model_name %}">{{ model_name }}</a></li>
    <br>
    {% endfor %}
</ol>
{% endblock %}
```

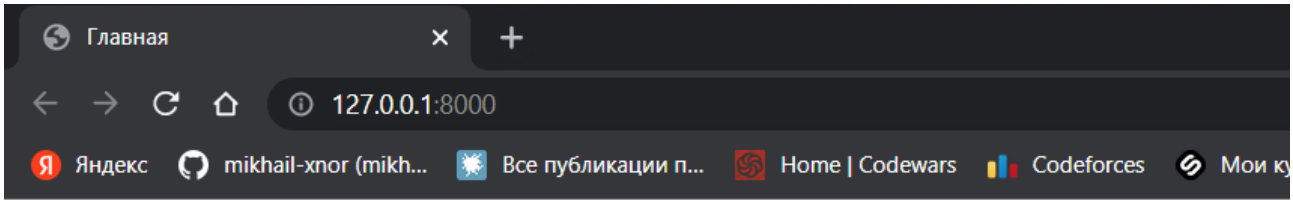
Файл main/templates/main/list.html:

```
{% extends 'main/base.html' %}


{% block title %}
{{ model_name }}
{% endblock %}


{% block content %}
<h2>Сущность <i>{{ model_name }}</i></h2>
<ul>
    {% for object in objects %}
    <li>
        {% for key, value in object.items %}
            <i>{{key}}</i>: {{value}}
            <br>
        {% endfor %}
    </li>
    <br>
    {% endfor %}
</ul>
{% endblock %}
```

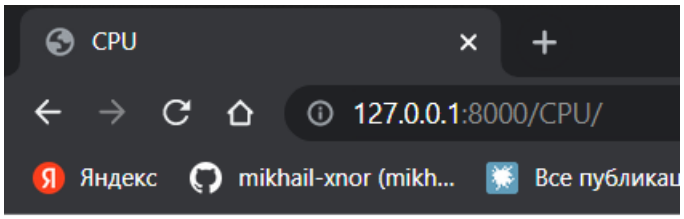
Прототип веб-приложения:



Главная

Подключенная база данных содержит следующие сущности:

1. CPU
2. DiskStorage
3. Hardware
4. RAM



Главная

Сущность *CPU*

- *id*: 1
  name: Intel core i5
  frequency: 5Ghz
  architecture: x86

- *id*: 2
  name: Intel core i3
  frequency: 3.2Ghz
  architecture: x86

- *id*: 3
  name: Apple M1 PRO
  frequency: 3.9Mhz
  architecture: arm

- *id*: 4
  name: Ryzen 5
  frequency: 4.6Ghz
  architecture: x86