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: | Проверил: |
| студентка группы ИУ5-52Б | преподаватель каф. ИУ5 |
| Заузолков Денис | Гапанюк Ю.Е. |

Москва, 2021 г.

# Описание задания:

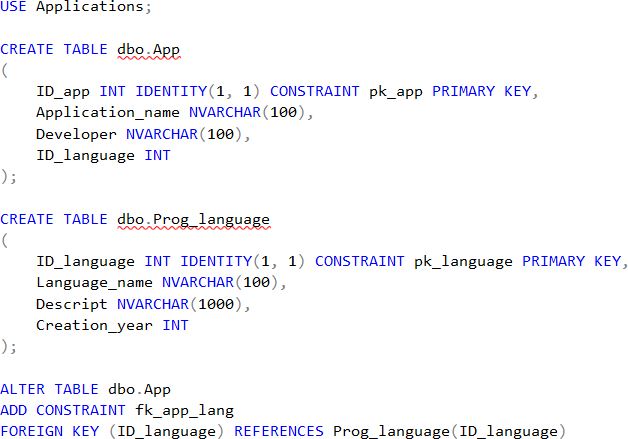
В этой лабораторной работе Вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также Вам нужно будет дополнить свои классы

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.](https://github.com/iu5team/iu5web-fall-2021/blob/main/tutorials/lab5/lab5_tutorial.md)
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 двух таблиц базы данных:



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

import pyodbc

connectionString = ("Driver={SQL Server Native Client 11.0};"

"Server=localhost\\SQLEXPRESS;" "Database=Applications;" "Trusted\_Connection=yes;")

request1 = """ INSERT INTO dbo.Prog\_language

SELECT 'C++', 'C++ is the best programming language!', 19

83

ng.', 1991 """

UNION ALL

SELECT 'Python', 'Python is often ised for machine learni

request2 = "SELECT \* FROM dbo.Prog\_language"

connection = pyodbc.connect(connectionString, autocommit=True) dbCursor = connection.cursor()

dbCursor.execute(request1) dbCursor.execute(request2) for row in dbCursor:

print(f"{row.ID\_language} {row.Language\_name} {row.Descript} {row.Cre ation\_year}")

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

# Результат:



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

from django.db import models

class ProgLanguage(models.Model):

id\_language = models.AutoField(db\_column='ID\_language', primary\_key=T rue) # Field name made lowercase.

language\_name = models.CharField(db\_column='Language\_name', max\_lengt h=100, blank=True, null=False) # Field name made lowercase.

descript = models.CharField(db\_column='Descript', max\_length=1000, bl ank=True, null=True) # Field name made lowercase.

creation\_year = models.IntegerField(db\_column='Creation\_year', blank= True, null=True) # Field name made lowercase.

class Meta:

db\_table = 'Prog\_language'

class App(models.Model):

id\_app = models.AutoField(db\_column='ID\_app', primary\_key=True) # Fi eld name made lowercase.

application\_name = models.CharField(db\_column='Application\_name', max

\_length=100, blank=True, null=False) # Field name made lowercase. developer = models.CharField(db\_column='Developer', max\_length=100, b

lank=True, null=True) # Field name made lowercase.

id\_language = models.ForeignKey('ProgLanguage', models.DO\_NOTHING, db

\_column='ID\_language', blank=True, null=True) # Field name made lowercas e.

class Meta:

db\_table = 'App'

# Файл Lab5/urls.py:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls), path('', include('main\_app.urls'))]

# Файл main\_app/urls.py:

from django.urls import path from . import views

urlpatterns = [

path('', views.index),

path('<str:model\_name>/', views.list, name='list')]

# Файл main\_app/views.py:

from django.shortcuts import render from .models import \*

from django.apps import apps

def index(request):

models = apps.get\_app\_config('main\_app').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} print(params)

return render(request, 'index.html', params)

def list(request, model\_name):

models = apps.get\_app\_config('main\_app').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, 'list.html', params)

# Файл main\_app/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\_app/index.py:

{% extends '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\_app/list.py:

{% extends 'base.html' %}

{% block title %}

{{ model\_name }}

{% endblock %}

{% block content %}

<h2>Cущность <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:

