**Московский государственный технический**

**университет им. Н. Э. Баумана**

Отчёт по лабораторной работе №5 по курсу «Разработка интернет приложений».

«Работа с СУБД. Обработка данных с использованием Django ORM».

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| Анцифров Н. С.  студент группы ИУ5-51Б |  | Гапанюк Ю. Е. |
|  |  |  |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Содержание**

[**1.** **Задание лабораторной работы.** 2](#_Toc90675458)

[**2.** **Текст программы.** 3](#_Toc90675459)

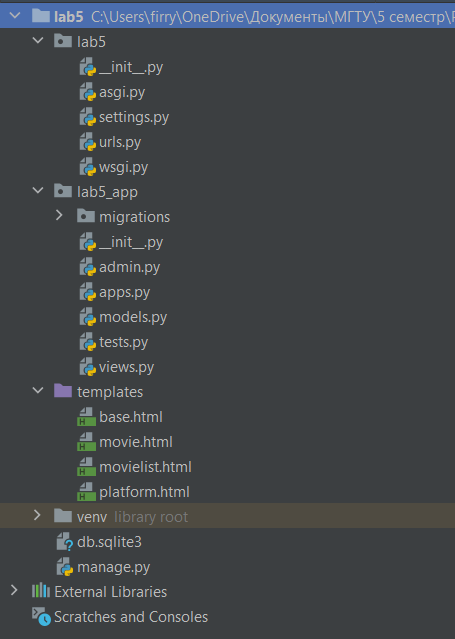
[**3.** **Выполнение программы.** 8](#_Toc90675460)

1. **Задание лабораторной работы.**

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.
4. **Текст программы.**

Структура проекта:

****

*\_\_init\_\_.py*

import pymysql  
  
pymysql.install\_as\_MySQLdb()

*settings.py*

from pathlib import Path  
  
# Build paths inside the project like this: BASE\_DIR / 'subdir'.  
BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = 'django-insecure-c)tpz\*(rrvwa^\_6qielwypt3mw3d1i%@3^&#(ibs!\_!u\*8%@lt'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = []  
  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'lab5\_app.apps.Lab5AppConfig',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'lab5.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [BASE\_DIR / 'templates']  
 ,  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'lab5.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': BASE\_DIR / 'db.sqlite3',  
 }  
}  
  
  
# Password validation  
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
# Internationalization  
# https://docs.djangoproject.com/en/3.2/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_L10N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/3.2/howto/static-files/  
  
STATIC\_URL = '/static/'  
  
# Default primary key field type  
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'moviechoice',  
 'USER': 'root',  
 'PASSWORD': 'root',  
 'HOST': 'localhost',  
 'PORT': 3306,  
 'OPTIONS': {'charset': 'utf8'},  
 'TEST\_CHARSET': 'utf8',  
 }  
}

*urls.py*

from django.contrib import admin  
from django.urls import path  
  
from lab5\_app import views  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', views.MovieList),  
 path('movie/<int:id>/', views.GetMovie, name='movie\_url'),  
 path('platform/<int:id>/', views.GetPlatform, name='platform\_url')  
]

*models.py*

from django.db import models  
  
  
# Create your models here.  
  
class Platform(models.Model):  
 name = models.CharField(max\_length=30)  
 site = models.CharField(max\_length=30)  
  
 class Meta:  
 managed = False  
 db\_table = 'platforms'  
  
  
class Movie(models.Model):  
 name = models.CharField(max\_length=30)  
 description = models.CharField(max\_length=255)  
 platform = models.CharField(max\_length=30)  
  
 class Meta:  
 managed = False  
 db\_table = 'movies'

*views.py*

from django.shortcuts import render  
  
# Create your views here.  
  
from lab5\_app.models import Movie  
from lab5\_app.models import Platform  
  
  
def MovieList(request):  
 return render(request, 'movielist.html', {'data': {  
 'platforms': Platform.objects.all(),  
 'movies': Movie.objects.all(),  
 }})  
  
  
def GetPlatform(request, id):  
 return render(request, 'platform.html', {'data': {  
 'platform': Platform.objects.filter(id=id)[0]  
 }})  
  
  
def GetMovie(request, id):  
 return render(request, 'movie.html', {'data': {  
 'movie': Movie.objects.filter(id=id)[0],  
 'platforms': Platform.objects.all(),  
 }})

*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>

*movie.html*

{% extends 'base.html' %}  
  
{% block title %}{{ data.movie.name }}{% endblock %}  
  
{% block content %}  
 <div>Название: {{ data.movie.name }}</div>  
 {% for platform in data.platforms %}  
 {% if data.movie.platform == platform.id %}  
 <div>Платформа: {{ platform.name }}</div>  
 {% endif %}  
 {% endfor %}  
 <div>Описание: {{ data.movie.description }}</div>  
{% endblock %}

*movielist.html*

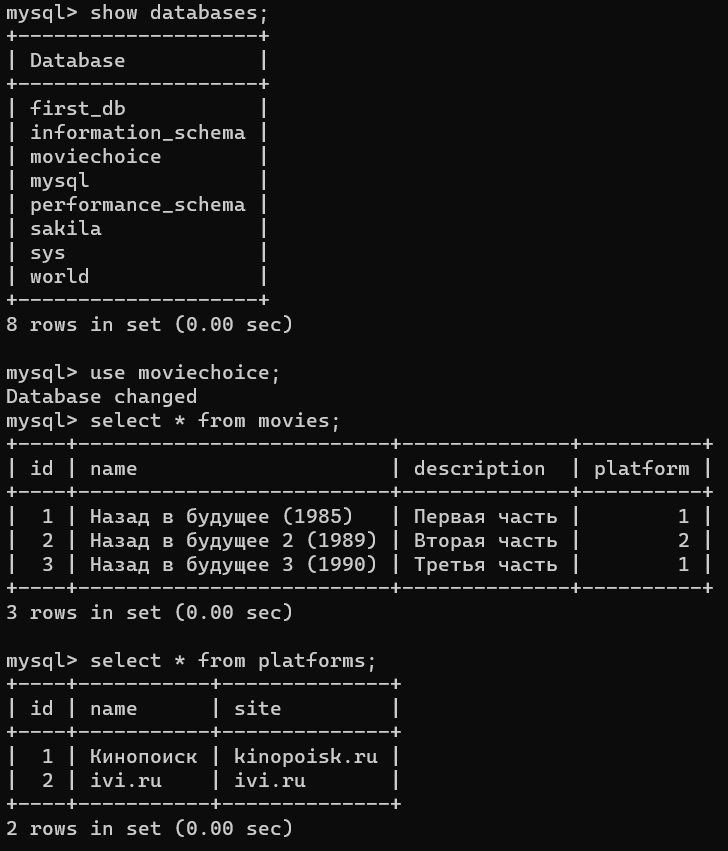
{% extends 'base.html' %}  
{% load static %}  
  
{% block title %}Лабораторная работа №5{% endblock %}  
  
{% block content %}  
 <h2>Фильмы:</h2>  
<h3><ul>  
 {% for movie in data.movies %}  
 <li><a href="{% url 'movie\_url' movie.id %}">{{movie.name}}</a></li>  
 {% empty %}  
 <li>Список пуст</li>  
 {% endfor %}  
</ul>  
</h3>  
 <h2>Платформы:</h2>  
 <h3>  
 <ul>  
 {% for platform in data.platforms %}  
 <li><a href="{% url 'platform\_url' platform.id %}">{{platform.name}}</a></li>  
 {% empty %}  
 <li>Список пуст</li>  
 {% endfor %}  
 </ul>  
 </h3>  
{% endblock %}

*platform.html*

{% extends 'base.html' %}  
  
{% block title %}{{ data.platform.name }}{% endblock %}  
  
{% block content %}  
 <div>Название: {{ data.platform.name }}</div>  
 <div>Сайт: {{ data.platform.site }}</div>  
{% endblock %}

1. **Выполнение программы.**

База данных:



Работа программы:

