|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | РАДИОТЕХНИЧЕСКИЙ |

|  |  |
| --- | --- |
| КАФЕДРА | СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ |

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

Студент Шкарин Егор Алексеевич .

*фамилия, имя, отчество*

Группа ИУ5-55Б .

Название предприятия МГТУ им. Н. Э. Баумана ю

Студент Шкарин Е.А.

Преподаватель Гапанюк Ю.Е.

*2021 г.*

**Цель работы**

Изучение возможностей СУБД MySQL для разработки веб-приложений. Изучение возможностей обработки данных с использованием Django ORM.

**Задание**

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.

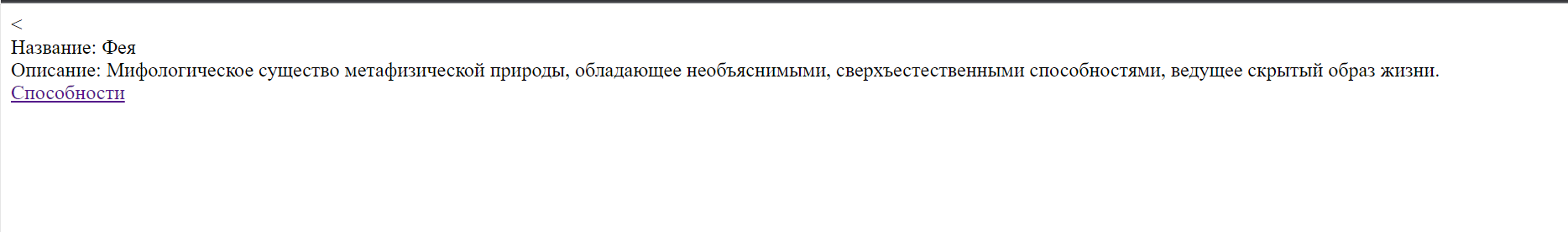
1. Создайте сценарий с подключением к БД и несколькими запросами, примеры рассмотрены в методических указаниях.
2. Реализуйте модели Вашей предметной области из предыдущей ЛР (минимум две модели, т.е. две таблицы).
3. Создайте представления и шаблоны Django для отображения списка данных по каждой из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

**Пример выполнения**

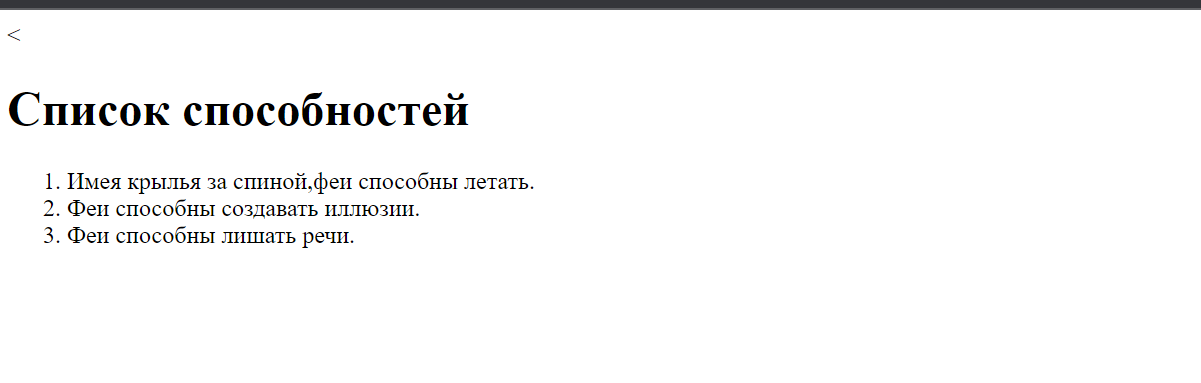
**http://127.0.0.1:8000/**

****

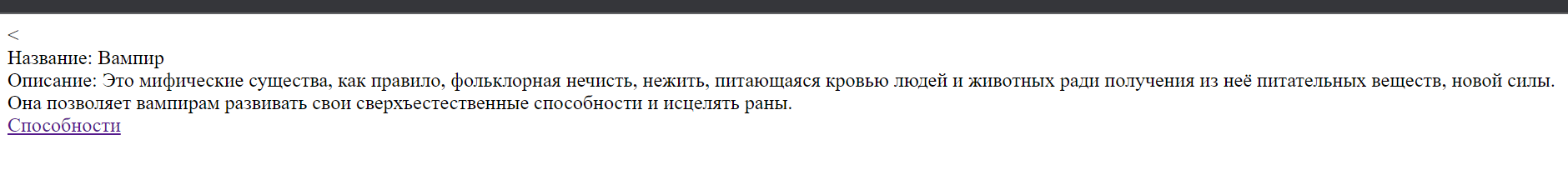
**http://127.0.0.1:8000/1/**

****

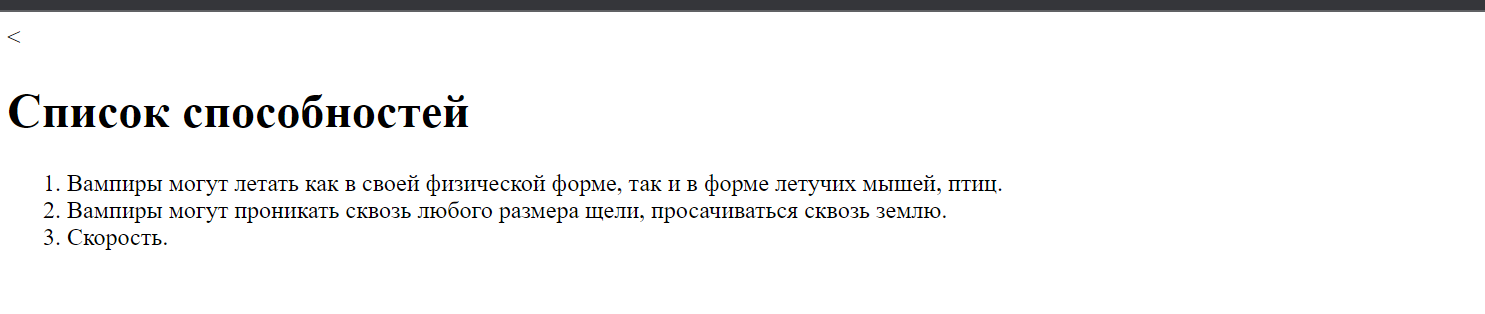
**http://127.0.0.1:8000/1/cans/**

****

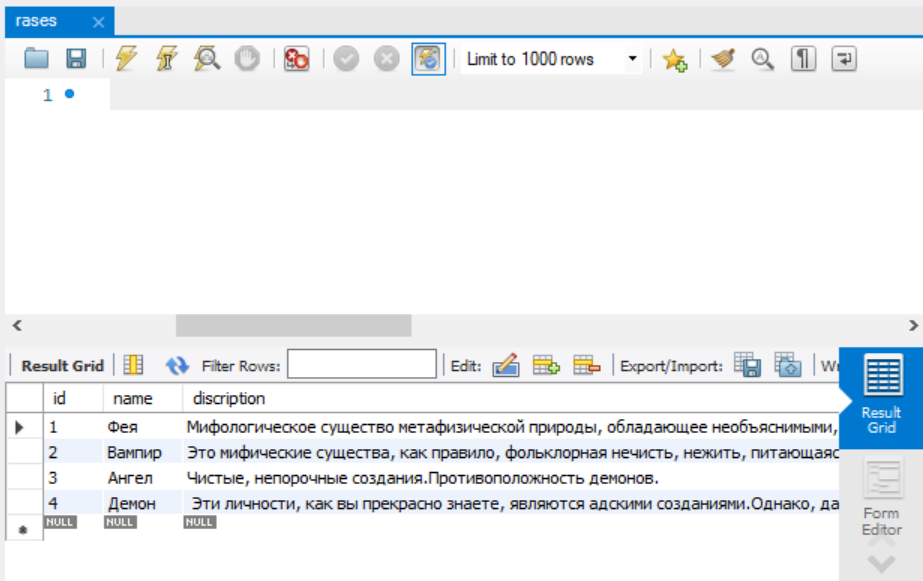
[**http://127.0.0.1:8000/2/**](http://127.0.0.1:8000/2/)

****

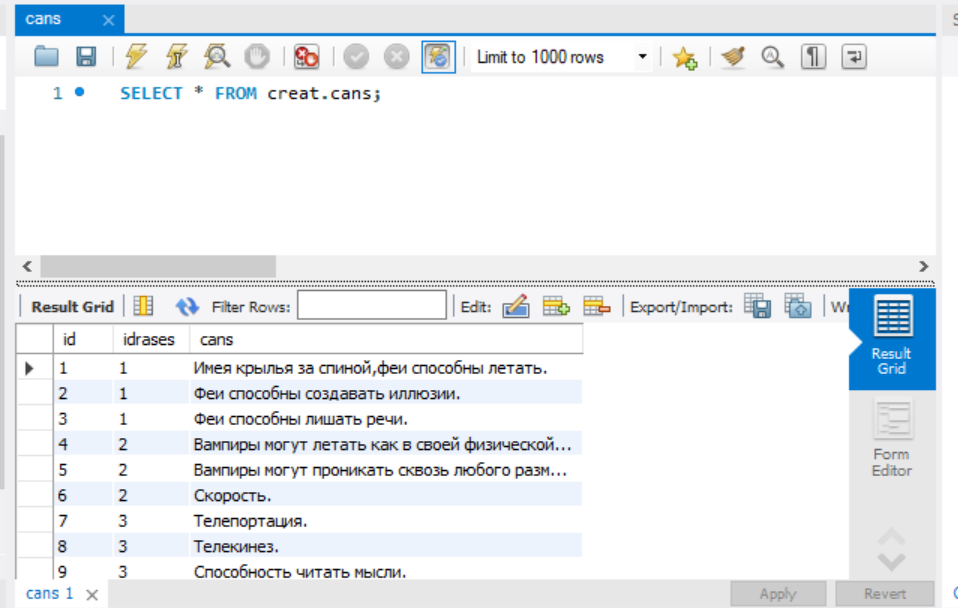
**http://127.0.0.1:8000/2/cans/**

****

Созданная в sql таблица Rases



Созданная в sql таблица cans



**Код программы**

Lr5/urls.py

from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('main.urls')),  
]

lr5/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/4.0/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = 'django-insecure-p(z8&hr)=ax2xxz4@##\*%&3c3=46)v0\_sunb#\*v=!qp2\_=tx5t'  
  
# 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',  
 'main'  
]  
  
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 = 'lr5.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [],  
 '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 = 'lr5.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'creat',  
 'USER': 'root',  
 'PASSWORD': '20012011',  
 'HOST': 'localhost',  
 'PORT': '3306',  
 }  
}  
# Password validation  
# https://docs.djangoproject.com/en/4.0/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/4.0/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/4.0/howto/static-files/  
  
STATIC\_URL = 'static/'  
  
# Default primary key field type  
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

main/models.py

from django.db import models  
  
  
class Rases(models.Model):  
 name = models.CharField(max\_length=30)  
 discription = models.CharField(max\_length=255)  
  
 class Meta:  
 managed = False  
 db\_table = 'rases'  
  
  
class Cans(models.Model):  
 idrases = models.IntegerField()  
 cans = models.CharField(max\_length=100)  
  
 class Meta:  
 managed = False  
 db\_table = 'cans'

main/urls.py

from django.urls import path  
from . import views  
from django.conf.urls.static import static  
from django.conf import settings  
  
urlpatterns = [  
 path('', views.master, name='master'),  
 path('<int:cr\_id>/', views.detail, name='detail'),  
 path('<int:cr\_id>/cans/', views.can, name='cann')  
  
]

main/views.py

from django.shortcuts import render  
from .models import Rases  
from .models import Cans  
  
  
def master(request):  
 crs = {'crs': Rases.objects.order\_by('id')}  
 return render(request, 'main/list.html', crs)  
  
  
def detail(request, cr\_id):  
 cr = Rases.objects.get(id=cr\_id)  
 return render(request, 'main/detail.html', {'cr': cr})  
  
  
def can(request, cr\_id):  
 cr = Rases.objects.get(id=cr\_id)  
 c = Cans.objects.all()  
 return render(request, 'main/second.html', {'c': c, 'cr': cr})

base.html

<!DOCTYPE html>  
<html lang="ru" class="h-100">  
<head>  
 <meta charset="UTF-8">  
 <title>{% block title %}{% endblock%}</title>  
</head>  
<body>  
 {% block content %}{% endblock %}  
</body>  
</html>

List.html

{% extends 'main/base.html' %}  
  
{% block title %} Фантастические существа {% endblock %}  
  
{% block styles %}  
body {  
 background-color: orange;  
 color: black;  
 font-family: 'Calibri';  
}  
li {  
 font-size: 24px;  
 margin-bottom: 15px;  
}  
li:hover {  
 font-weight: bold;  
}  
a {  
 text-decoration: none;  
 color: black;  
}  
{% endblock %}  
  
  
{% block content %}  
 <h1>Список существ</h1>  
 <ol>  
 {% for i in crs %}  
 <li><a href="{% url 'detail' i.id %}">{{i.name}}</a></li>  
 {% endfor %}  
 </ol>  
{% endblock %}

Detail.html

<{% extends 'main/base.html' %}  
  
{% block title %} Раса{% endblock %}  
  
{% block content %}  
 <div>Название: {{ cr.name }}</div>  
 <div>Описание: {{ cr.discription }}</div>  
 <div><a href="{% url 'cann' cr.id %}">Способности</a></div>  
{% endblock %}

Second.html

<{% extends 'main/base.html' %}  
  
{% block title %} Раса{% endblock %}  
  
{% block content %}  
 <h1>Список способностей</h1>  
 <ol>  
 {% for i in c %}  
 {% if i.idrases == cr.id %}  
 <li>{{i.cans}}</li>  
 {% endif %}  
 {% endfor %}  
 </ol>  
{% endblock %}