|  |  |
| --- | --- |
|  | **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 РУБЕЖНОМУ КОНТРОЛЮ №2**

Студент Столярова Ольга Денисовна .

*фамилия, имя, отчество*

Группа РТ5-51Б .

Название предприятия МГТУ им. Н. Э. Баумана ю

Студент Столярова О.Д.

Преподаватель Гапанюк Ю.Е.

*2021 г.*

**Задание:**

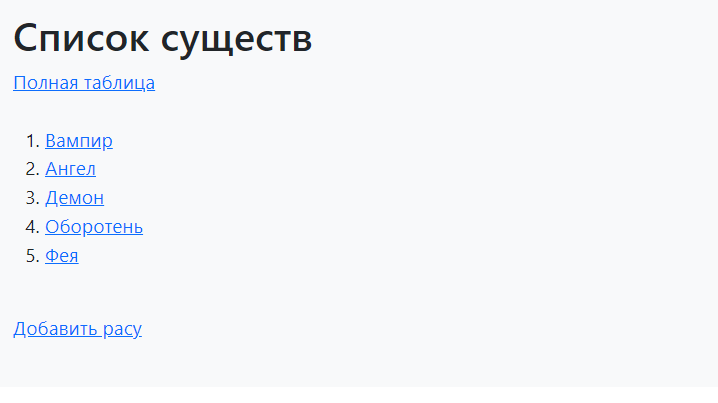
Рубежный контроль представляет собой разработку веб-приложения с использованием фреймворка Django. Веб-приложение должно выполнять следующие функции:

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

Экранные формы с примерами выполнения программы:

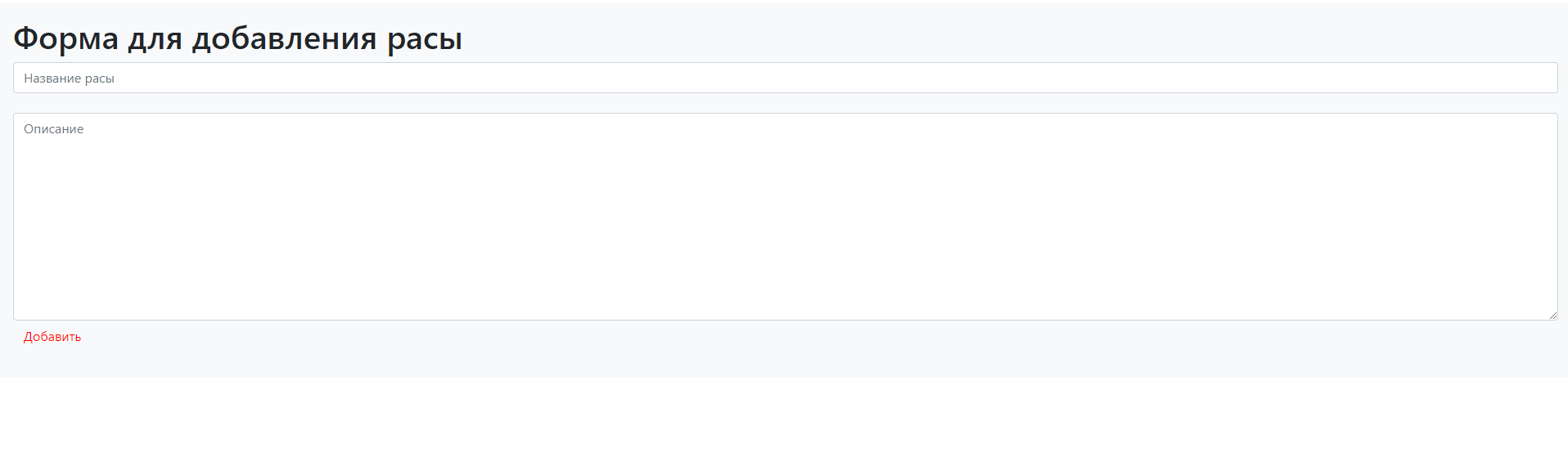
Форма для просмотра списка рас:

<http://127.0.0.1:8000/>



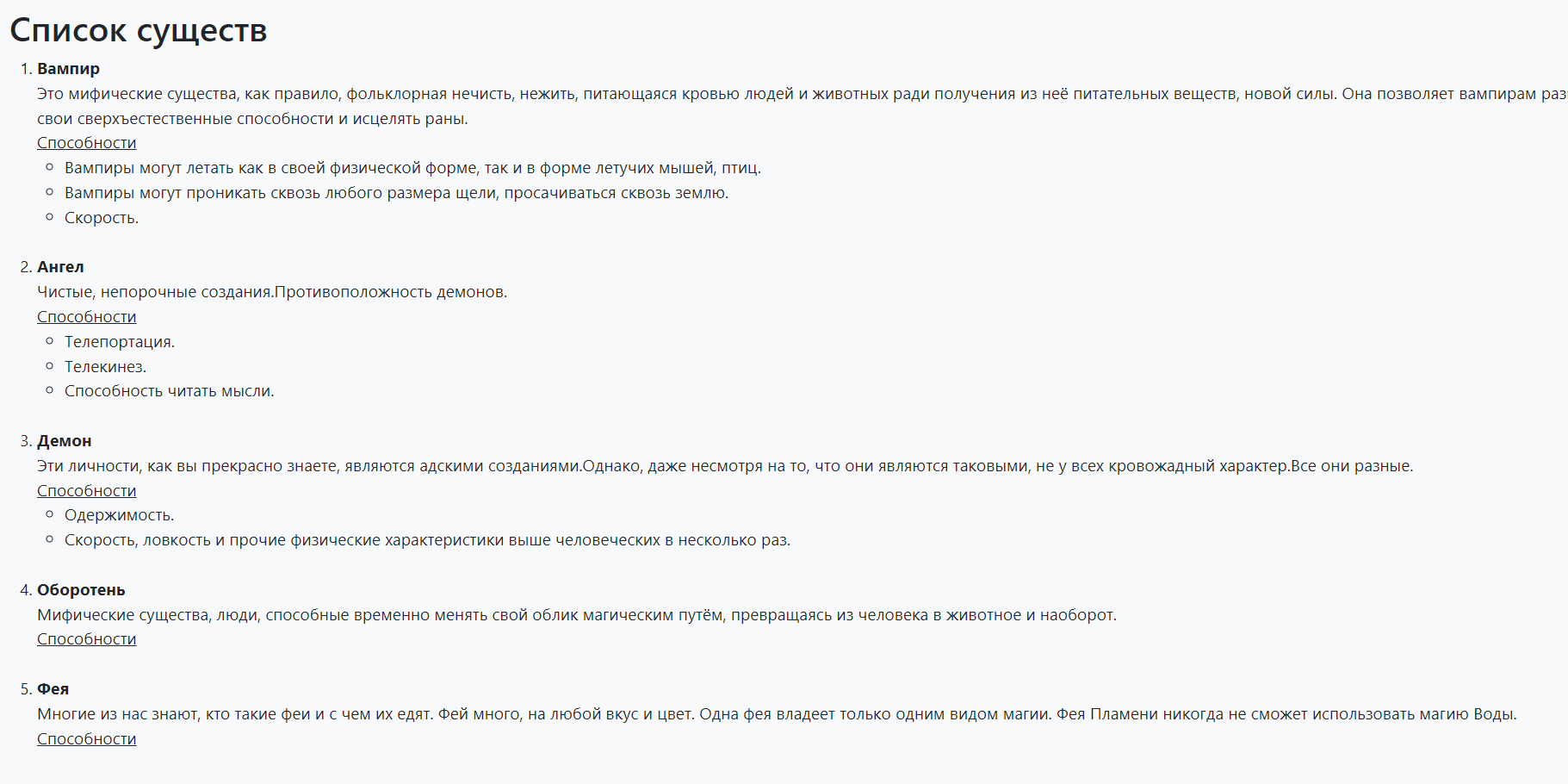
Форма для добавления расы:

http://127.0.0.1:8000/create/



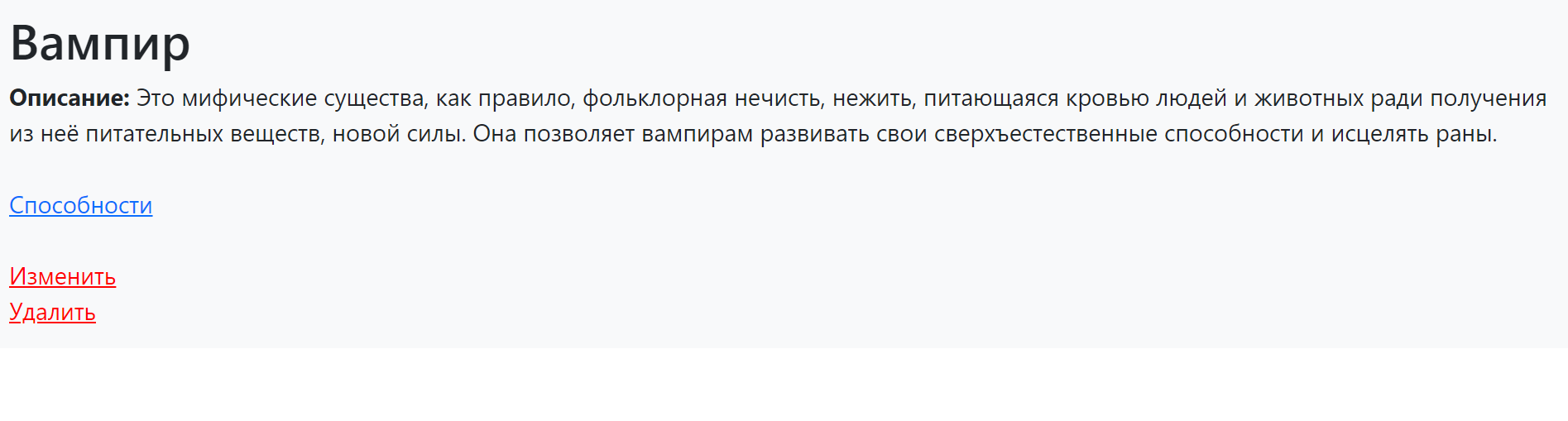
Форма для просмотра отчета со всей информацией из бд:

http://127.0.0.1:8000/rep/



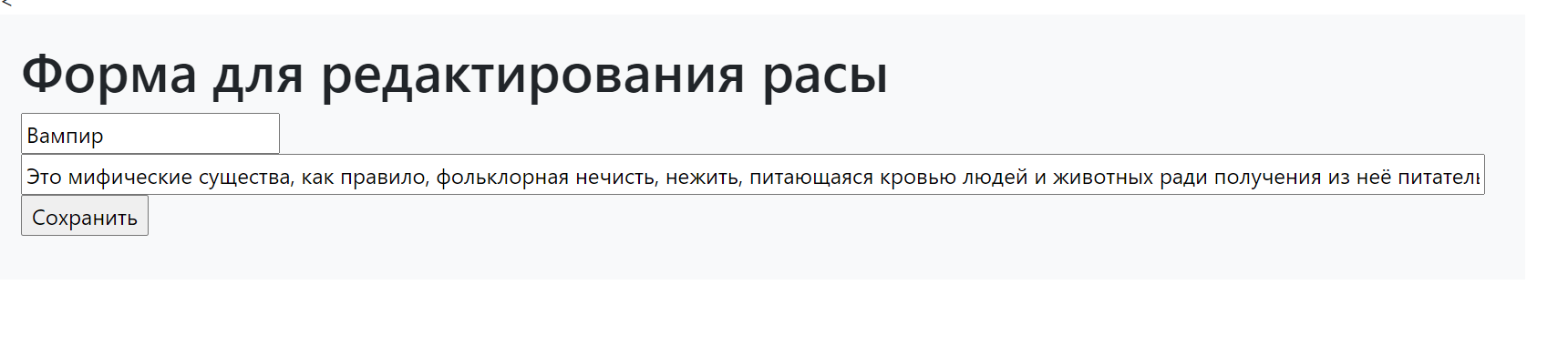
Форма для просмотра информации об определенной расе:

<http://127.0.0.1:8000/2/>



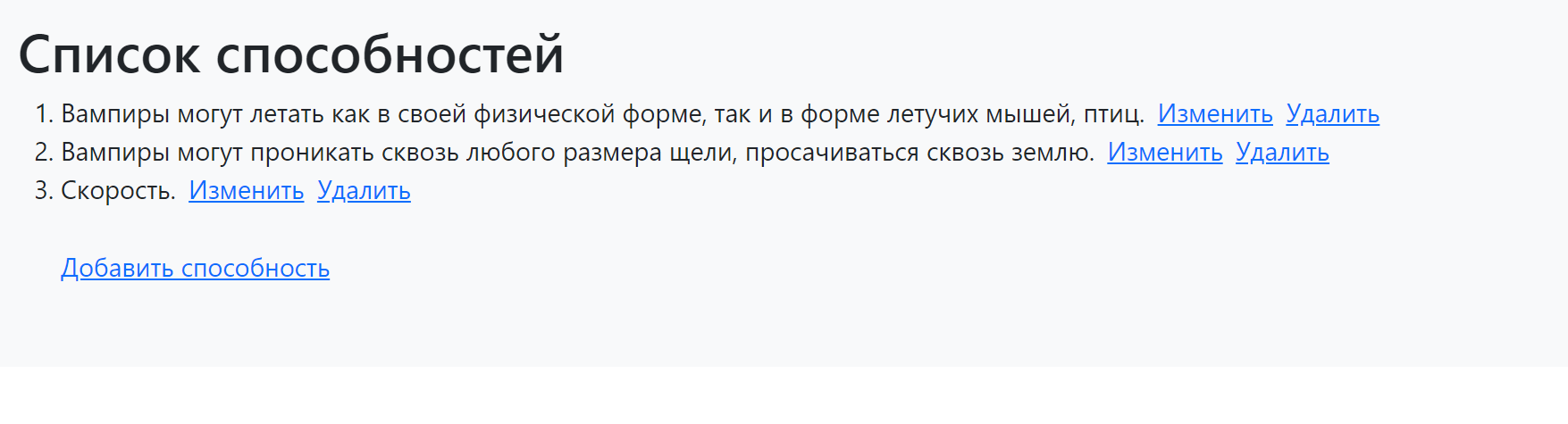
Форма для изменения информации о расе:

<http://127.0.0.1:8000/2/update/>



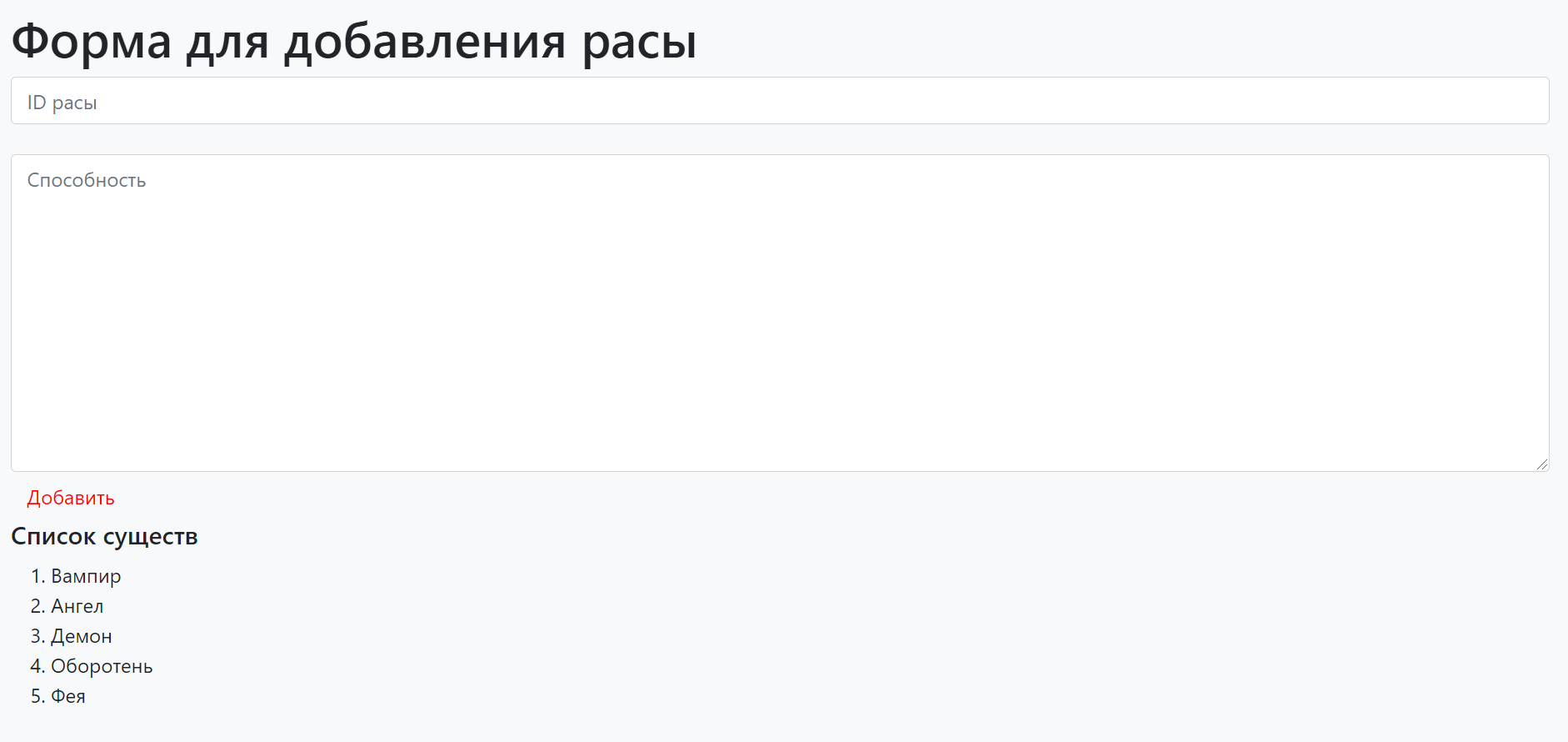
Форма для просмотра способностей расы (связь один ко многим: одна раса – несколько способностей):

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



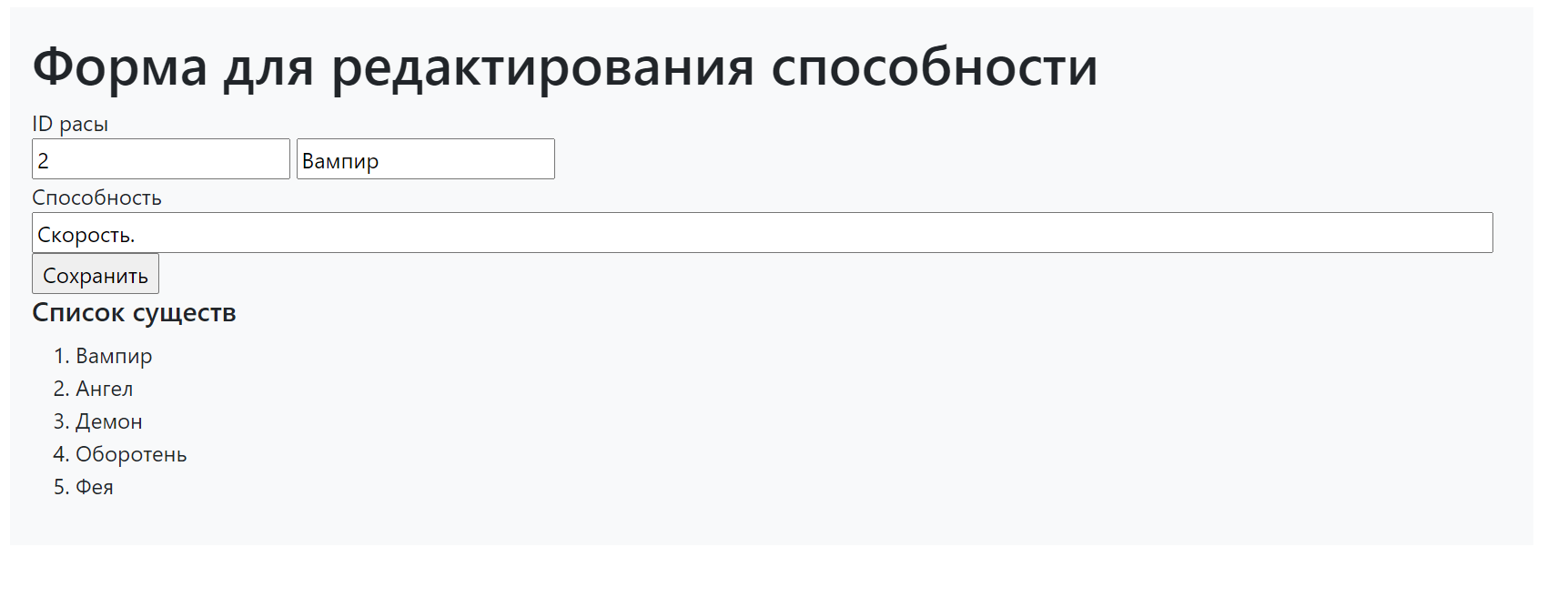
Форма для добавления способностей:

<http://127.0.0.1:8000/2/cans/create/>



Форма для редактирования способности:

<http://127.0.0.1:8000/update/4/>



**Код программы:**

RK/urls.py

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

RK/settings.py

*"""  
Django settings for lr5 project.  
  
Generated by 'django-admin startproject' using Django 4.0.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/4.0/topics/settings/  
  
For the full list of settings and their values, see  
https://docs.djangoproject.com/en/4.0/ref/settings/  
"""*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/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'),  
 path('rep/', views.report, name='report'),  
 path('create/', views.create, name='create'),  
 path('<int:cr\_id>/update/', views.update, name='update'),  
 path('<int:cr\_id>/delete/', views.delete, name='delete'),  
 path('<int:cr\_id>/cans/create/', views.create1, name='create1'),  
 path('update/<int:cr\_id>/', views.update1),  
 path('delete/<int:cr\_id>/', views.delete1)  
  
]

main/views.py

from django.shortcuts import render, redirect, HttpResponseRedirect  
from .models import Rases  
from .models import Cans  
from .forms import RasesForm, CansForm  
from django.http import HttpResponse, HttpResponseNotFound  
  
  
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})  
  
  
def report(request):  
 cr = Rases.objects.all()  
 c = Cans.objects.all()  
 return render(request, 'main/rep.html', {'c': c, 'cr': cr})  
  
  
def create(request):  
 error= ''  
 if request.method == 'POST':  
 form = RasesForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('master')  
 else:  
 error = 'Данные введены некорректно!'  
  
 form = RasesForm()  
  
 data = {  
 'form': form,  
 'error': error  
 }  
 return render(request, 'main/create.html', data)  
  
  
def update(request, cr\_id):  
 try:  
 cr = Rases.objects.get(id=cr\_id)  
  
 if request.method == "POST":  
 cr.name = request.POST.get("name")  
 cr.discription = request.POST.get("discription")  
 cr.save()  
 return HttpResponseRedirect("/")  
 else:  
 return render(request, "main/update.html", {"cr": cr})  
 except cr.DoesNotExist:  
 return HttpResponseNotFound("<h2>Person not found</h2>")  
  
  
def delete(request, cr\_id):  
 try:  
 cr = Rases.objects.get(id=cr\_id)  
 cr.delete()  
 return HttpResponseRedirect("/")  
 except cr.DoesNotExist:  
 return HttpResponseNotFound("<h2>Person not found</h2>")  
  
  
def create1(request, cr\_id):  
 cr = Rases.objects.all()  
 error = ''  
 if request.method == 'POST':  
 form = CansForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 else:  
 error = 'Данные введены некорректно!'  
  
 form = CansForm()  
  
 data = {  
 'form': form,  
 'error': error,  
 'cr': cr,  
 }  
 return render(request, 'main/create1.html', data)  
  
def update1(request, cr\_id):  
 try:  
 cr = Cans.objects.get(id=cr\_id)  
 ras = Rases.objects.all()  
 if request.method == "POST":  
 cr.idrases = request.POST.get("idrases")  
 cr.cans = request.POST.get("cans")  
 cr.save()  
 return HttpResponseRedirect("/")  
 else:  
 return render(request, "main/update2.html", {"cr": cr, "c": ras})  
 except cr.DoesNotExist:  
 return HttpResponseNotFound("<h2>Person not found</h2>")  
  
  
def delete1(request, cr\_id):  
 try:  
 cr = Cans.objects.get(id=cr\_id)  
 cr.delete()  
 return HttpResponseRedirect("/")  
 except cr.DoesNotExist:  
 return HttpResponseNotFound("<h2>Person not found</h2>")

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/forms.py

from .models import Rases, Cans  
from django.forms import ModelForm, TextInput, Textarea  
  
  
class RasesForm(ModelForm):  
 class Meta:  
 model = Rases  
 fields = ['name', 'discription']  
  
 widgets = {  
 "name": TextInput(attrs = {  
 'class': 'form-control',  
 'placeholder' : 'Название расы'  
 }),  
 "discription": Textarea(attrs={  
 'class': 'form-control',  
 'placeholder': 'Описание'  
 }),  
  
 }  
  
class CansForm(ModelForm):  
 class Meta:  
 model = Cans  
 fields = ['idrases', 'cans']  
  
 widgets = {  
 "idrases": TextInput(attrs = {  
 'class': 'form-control',  
 'placeholder' : 'ID расы'  
 }),  
 "cans": Textarea(attrs={  
 'class': 'form-control',  
 'placeholder': 'Способность'  
 }),  
  
 }

base.html

<!DOCTYPE html>  
<html lang="ru" class="h-100">  
<head>  
 <meta charset="UTF-8">  
 <title>{% block title %}{% endblock%}</title>  
 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">  
</head>  
<body>  
<div class="p-3 mb-2 bg-light text-dark">  
 {% block content %}{% endblock %}  
 </div>  
</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: 35px;  
 margin-bottom: 30px;  
}  
li:hover {  
 font-weight: bold;  
}  
a {  
 text-decoration: none;  
 color: black;  
}  
{% endblock %}  
  
  
{% block content %}  
<body bgcolor="#ffffe0">  
  
 <big><h1>Список существ</h1>  
 <div><a href="{% url 'report' %}">Полная таблица</a></div>  
<br>  
 <ol>  
 {% for i in crs %}  
 <li><a href="{% url 'detail' i.id %}">{{i.name}}</a></li>  
 {% endfor %}  
 </ol>  
 <br>  
 <div><a href="{% url 'create' %}">Добавить расу</a></div> <br>  
  
 </big>  
  
{% endblock %}

Detail.html

<{% extends 'main/base.html' %}  
  
{% block styles %}  
body {  
 color: black;  
 font-family: 'Calibri';  
}  
li {  
 font-size: 30px;  
 margin-bottom: 15px;  
}  
li:hover {  
 font-weight: bold;  
}  
a {  
 text-decoration: none;  
 color: black;  
}  
 h1 {  
 font-family: 'Times New Roman', Times, serif; /\* Гарнитура текста \*/  
 font-size: 500%; /\* Размер шрифта в процентах \*/  
 }  
  
{% endblock %}  
  
{% block title %} Раса{% endblock %}  
  
{% block content %}  
<body bgcolor="#ffffe0">  
 <big><h1>{{ cr.name }}</h1>  
 <div><strong>Описание:</strong> {{ cr.discription }}</div>  
<br>  
 <div><a href="{% url 'cann' cr.id %}">Способности</a></div> <br>  
 <div><a href="{% url 'update' cr.id %}"><font color = "red">Изменить </font></a></div>  
 <div><a href="delete"><font color = "red">Удалить</font></a></div>  
 </big>  
{% endblock %}

Rep.html

{% extends 'main/base.html' %}  
  
{% block title %} Фантастические существа {% endblock %}  
  
{% block styles %}  
body {  
 color: black;  
 font-family: 'Calibri';  
}  
li {  
 font-size: 30px;  
 margin-bottom: 15px;  
}  
li:hover {  
 font-weight: bold;  
}  
a {  
 text-decoration: none;  
 color: black;  
}  
{% endblock %}  
  
  
{% block content %}  
<body bgcolor="#ffffe0">  
 <big><h1>Список существ</h1>  
 <ol>  
 {% for i in cr %}  
 <li>  
 <dev><strong>{{i.name}}</strong></dev>  
 <br>  
 <dev>{{i.discription}}</dev>  
 <br>  
 <dev><u>Способности</u></dev>  
 <ul>  
 {% for j in c %}  
 {% if j.idrases == i.id %}  
 <li>{{j.cans}}</li>  
 {% endif %}  
 {% endfor %}  
 </ul>  
 <br>  
 </li>  
 {% endfor %}  
 </ol></big>  
{% endblock %}

Create.html

<{% extends 'main/base.html' %}  
  
{% block styles %}  
body {  
 color: black;  
 font-family: 'Calibri';  
}  
li {  
 font-size: 30px;  
 margin-bottom: 15px;  
}  
li:hover {  
 font-weight: bold;  
}  
a {  
 text-decoration: none;  
 color: black;  
}  
 h1 {  
 font-family: 'Times New Roman', Times, serif; /\* Гарнитура текста \*/  
 font-size: 500%; /\* Размер шрифта в процентах \*/  
 }  
  
{% endblock %}  
  
{% block title %}Добавление записи{% endblock %}  
  
{% block content %}  
  
<h1>Форма для добавления расы</h1>  
<form method = "post">  
 {% csrf\_token %}  
 {{ form.name }}  
 <br>  
 <span>{{error}}</span>  
 {{ form.discription }}  
 <button class="btn btn-succes" type="submit"><font color = "red">Добавить</font></button>  
</form>  
{% endblock %}