ИУ5-55 Турчин Денис, рк-2 РИП

Задание:

```
«Библиотека CD-дисков» и «CD-диск» связаны соотношением многие-ко-многим

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

Создам модели:

```python
class Disk(models.Model):
    class Meta:
        db_table = 'Disks'
        managed = True
        verbose_name = 'диск'
        verbose_name_plural = 'диски'

    title = models.CharField(max_length=200)
    format = models.CharField(max_length=10)
    size = models.IntegerField(default = 0)

    @property
    def description(self):
        return self.title + '(' + str(self.size) + ' MB, ' + self.format + ')'

    def __str__(self):
        return self.title

class Library(models.Model):
    class Meta:
        db_table = 'Libraries'
        managed = True
        verbose_name = 'библиотека'
        verbose_name_plural = 'библиотеки'

    fk_storedDisks = models.ManyToManyField(Disk)
    title = models.CharField(max_length=200)
    owner = models.CharField(max_length=80)

    @property
    def disks_num(self):
        _disks_num = self.fk_storedDisks.all().count()
        return _disks_num

    @property
    def description(self):
        return str(self.disks_num) + ' disks | owner: , ' + self.owner

    def __str__(self):
        return self.title
```

Обеим определю вычисляемое поле description, которое буду выводить в приложении для идентификации объекта.

Поля класс Мета нужны для более качественного отображения записей в админке джанги.

Сохраняю базу

```
(myvenv) PS C:\Users\Denactive\source\repos\Denactive\WebApplicationDevs\rk-2> python manage.py makemigrations
Migrations for 'rk_app':
  rk_app\migrations\0001_initial.py
    - Create model Disk
    - Create model Library
(myvenv) PS C:\Users\Denactive\source\repos\Denactive\WebApplicationDevs\rk-2> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, rk_app, sessions
Running migrations:
  Applying rk_app.0001_initial... OK
```

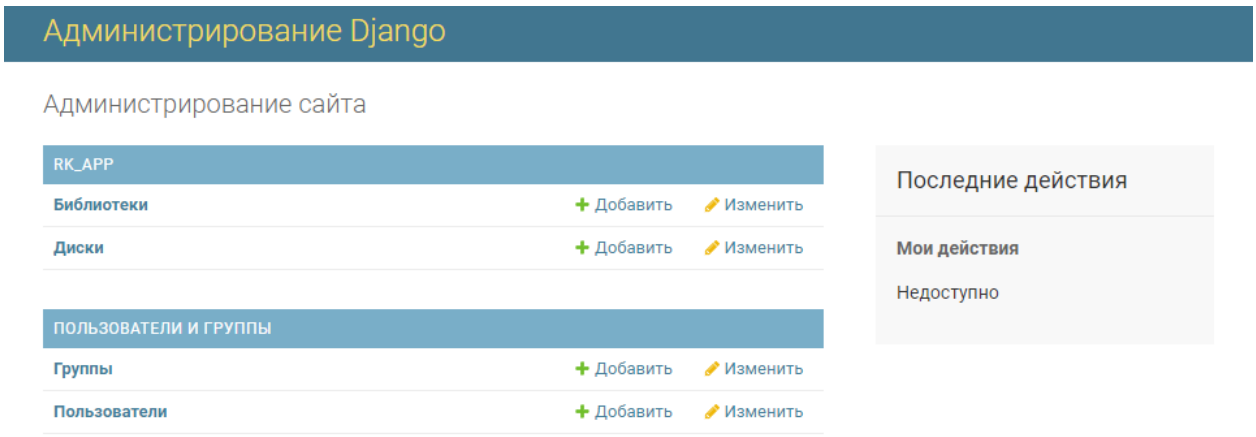
Администрирование:

```
rk_app > admin.py > ...
1   from django.contrib import admin
2   from rk_app.models import Library, Dis
3
4   # Register your models here.
5   admin.site.register(Library)
6   admin.site.register(Disk)
```

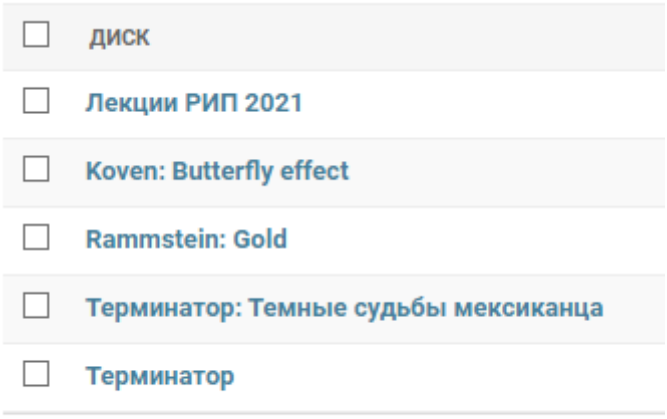
Супер-юзер для администрирования

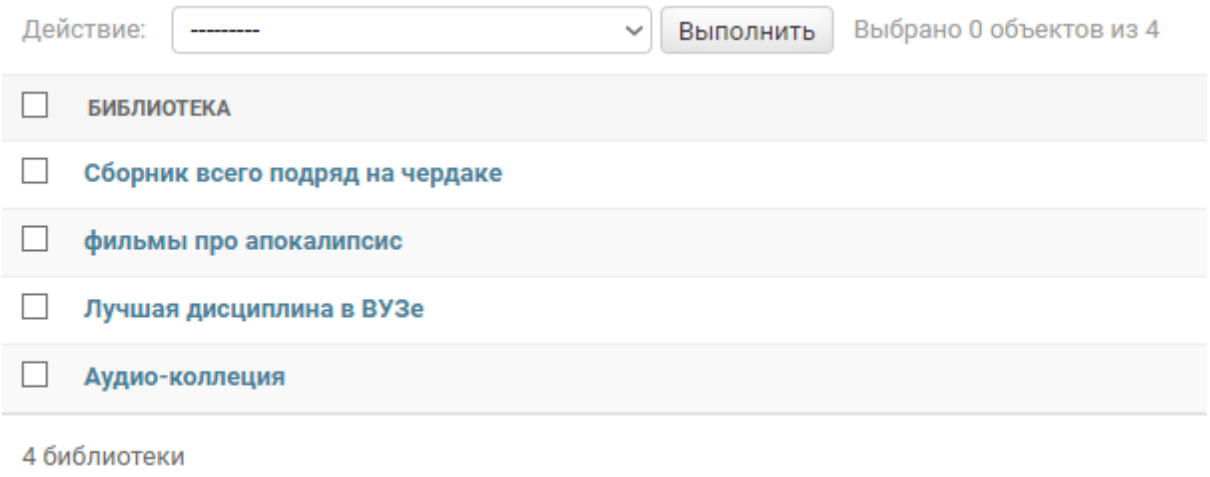
```
(myvenv) PS C:\Users\Denactive\source\repos\Denactive\WebApplicationDevs\rk-2> py manage.py createsuperuser
Имя пользователя (leave blank to use 'denactive'):
Адрес электронной почты:
Password:
Password (again):
Error: Blank passwords aren't allowed.
Password:
Password (again):
Введённый пароль слишком короткий. Он должен содержать как минимум 8 символов.
Введённый пароль состоит только из цифр.
Bypass password validation and create user anyway? [y/N]:
Password:
Password (again):
Введённый пароль слишком короткий. Он должен содержать как минимум 8 символов.
Введённый пароль состоит только из цифр.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Теперь по локейшену /admin/ мне доступно добавление записей в обе таблицы



Создам несколько записей





Интерфейс моего приложения – это 3 странички. Просмотр дисков, просмотр библиотек, просмотр содержимого конкретной библиотеки

Заведу следующие локейшены:

```python
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index_page_router),
    path('libraries', views.libs_page_router),
    path('library/<int:pk>/', views.lib_page_router, name='library'),
]
```

Последний локейшн содержит url-переменную и собственное имя для реверс-роутинга, чтобы информацию об объекте (айди) можно было использовать для составления ссылки на страницу, на которой он используется.

Создам шаблон, используя стили из бутстрапа.

```
<!doctype html>
<html lang="en" class="h-100">
  <head>···
  </head>
  <body class="d-flex flex-column h-100">

<header>···
</header>

<!-- Begin page content -->
<main class="flex-shrink-0 overflow-visible">
<div class="container-fluid">
  <div class="row vh-100">
    <!-- left pic -->
    <div class="col-xl-3 col-sm-1 side-pict-left"></div>
    <div class="col-xl-6 col-sm-10 question-block">
    <!-- central block -->
      <div class="container">
        <!-- Заголовок -->
        <div class="row">
          <div class="d-flex flex-row align-items-center justify-content-start" style="margin-top:100px; z-index: 1;">
            <div class="text-shadowed d-flex">
              <h3>
              {% block page_heading %}
              {% endblock page_heading %}
              </h3>
            </div>
            <div class="d-flex m-4 page-headingurl">
              {% block page_heading_url %}
              {% endblock page_heading_url %}
            </div>
          </div>
        </div>
        <!-- Тело -->
        <div class="row">
          <!-- <div class="col-xl-8 col-sm-8 me-2 mb-5 page_content"> -->
            <div class="col page_content">
            <!-- v Контент страницы v -->
            {% block page_content %}

            {% endblock page_content %}
            <!-- ^ Контент страницы ^ -->
```

Ниже только диваки (не уместились). Хедер тоже из шаблонов бутстрап.

Шаблон нужен, чтобы все страницы смотрелись одинаково. Меняется лишь название и блок с содержимым. Второе поле может использоваться для чего угодно, расположенного рядом с заголовком страницы, например, для ссылки. Его использовать не буду.

С применением шаблона 3 страницы выглядят так:

Страница с дисками

emplates > index.html

```
{% extends "include/base.html" %}
{% load static %}

{% block page_heading %}
  Диски
{% endblock page_heading %}
{% block page_heading_url %}
{% endblock page_heading_url %}

{% block page_content %}
  {% for item in items %}
    {% include "include/disk_unit.html" %}
  {% endfor %}
{% endblock page_content %}
```

Страница с библиотеками

templates > libraries.html

```
{% extends "include/base.html" %}
{% load static %}

{% block page_heading %}
  Библиотеки дисков
{% endblock page_heading %}
{% block page_heading_url %}
{% endblock page_heading_url %}

{% block page_content %}
  {% for item in items %}
    {% include "include/library_unit.html" %}
  {% endfor %}
{% endblock page_content %}
```

Страница библиотеки – ее содержимое чуть сложнее

templates > lib_details.html > ...

```
{% extends "include/base.html" %}
{% load static %}

{% block page_heading %}
  [id={{item.id}}] Просмотр состава библиотеки
{% endblock page_heading %}

{% block page_content %}
<!-- v lib v -->
{% include "include/library_unit.html" %}
<!-- ^ lib ^ -->

<h3>
  Хранимые диски:
</h3>

<!-- v disks v -->
<div id="disks">
  {% for item in items %}
    {% include "include/disk_unit.html" %}
  {% endfor %}
</div>
<!-- ^ disks ^ -->
{% endblock page_content %}
```

Остается роутинг модель-представление (controller из MVC):

rk_app > views.py > lib_page_router

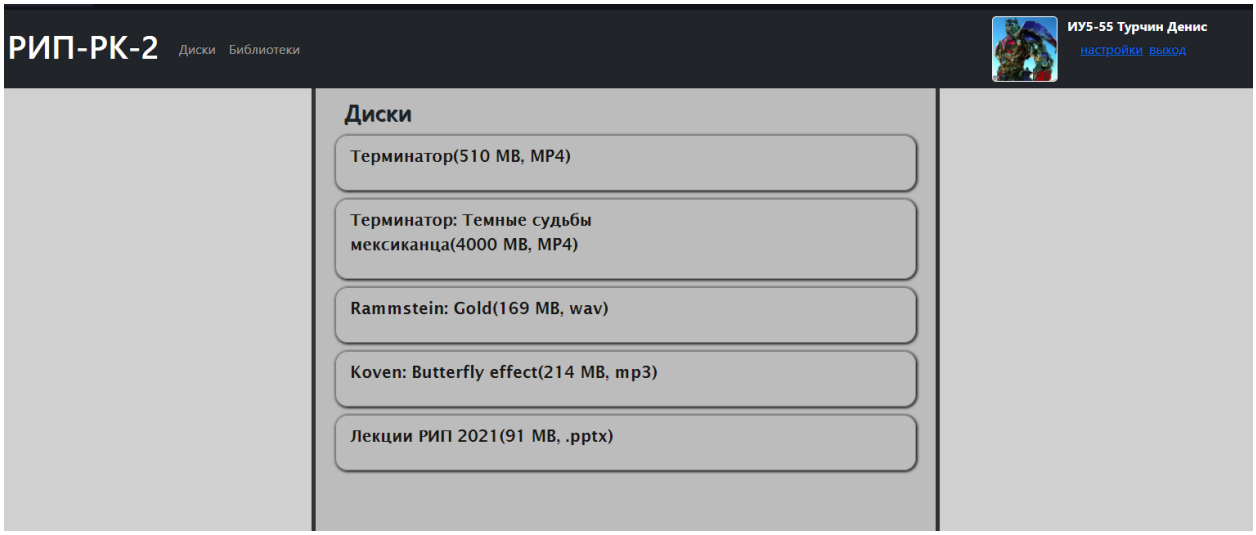
```
from django.shortcuts import render
from rk_app.models import Library
from rk_app.models import Disk

def index_page_router(request):
  return render(request, 'index.html', {"items": Disk.objects.all()})

def libs_page_router(request):
  return render(request, 'libraries.html', {"items": Library.objects.all()})

def lib_page_router(request, pk):
  lib = Library.objects.get(pk=pk)
  disks = lib.fk_storedDisks.all()
  return render(request, 'lib_details.html', {'item': lib, 'items': disks})
```

Данные для страниц берутся из несложных запросов, возвращающих обе таблицы целиком. В продакшн такое использовать не стоит, вдруг записей миллион.

В третьем запросе происходит маппинг двух таблиц через связь многие-ко-многим. Его синтаксис чуть отличается ввиду особенностей джанги.

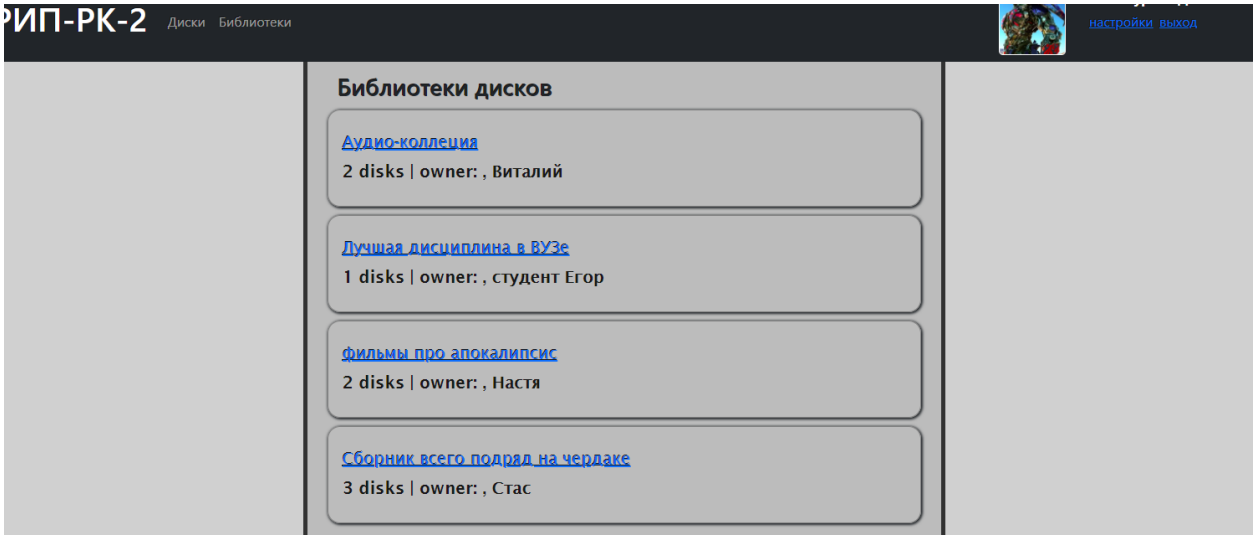
Че получилось

Страница с дисками (она же главная)

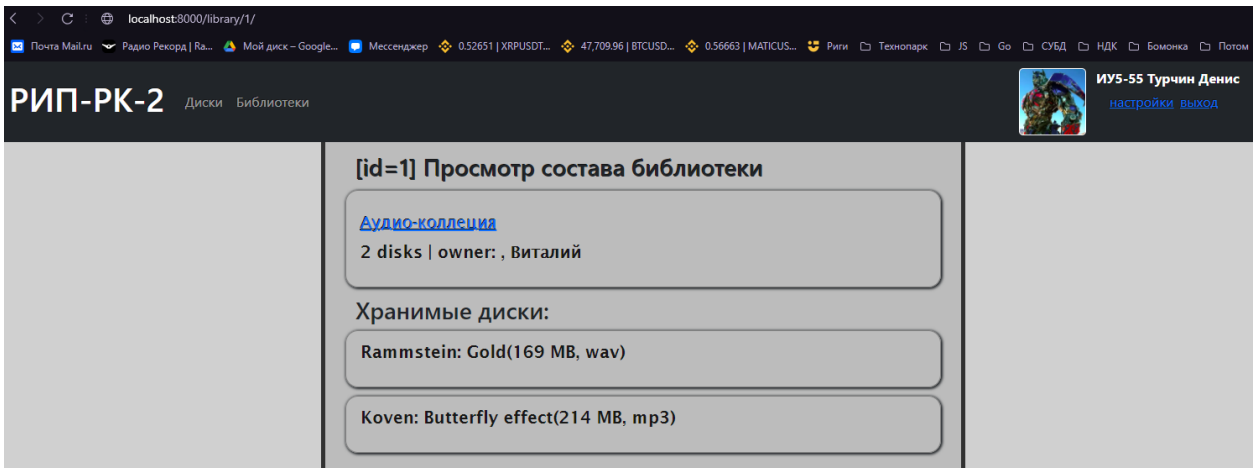


Страница с библиотеками:

Свойство description – вычисляемое и использует еще одно вычисляемое свойство, подсчитывающее количество дисков в ней

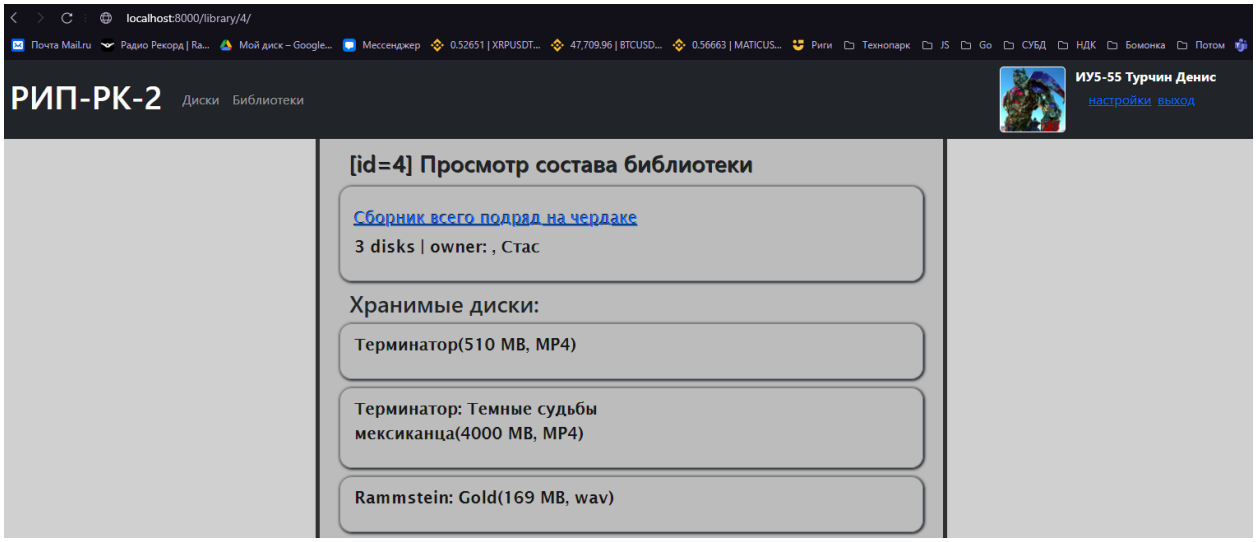


Переходим по ссылкам с названиями коллекций

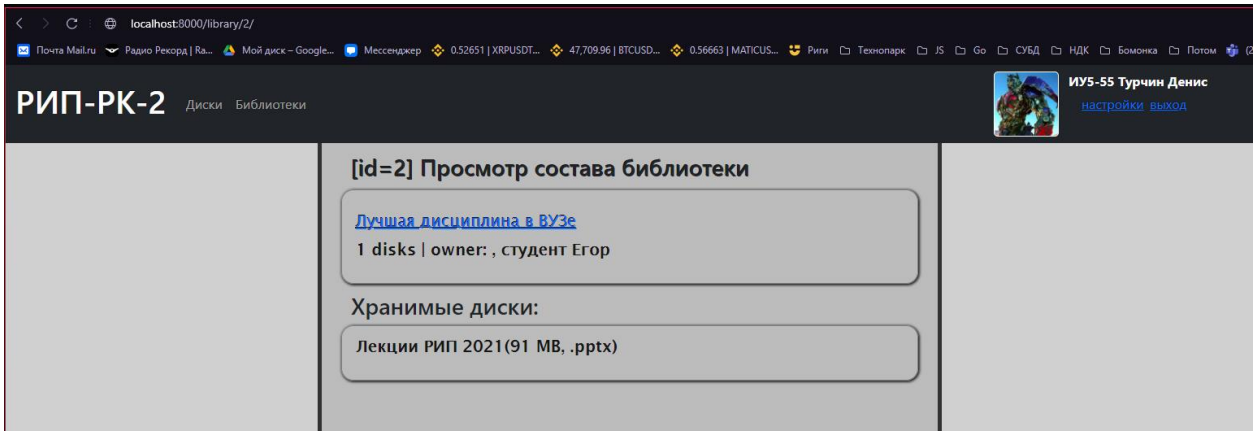


В урле есть параметр. По нему джанга сопоставляет нужную для отображения запись из таблицы

Другая библиотека содержит эти же диски (связь многие-ко-многим):



Есть и библиотека с уникальными записями



Таким образом интерфейс для просмотра содержимого базы готов. Администрировать его можно через встроенное во фреймворк апи /admin/

Можно также добавить формы для заполнения полей сущностей диски и библиотеки.

Отправка таких форм на сервер должна вызывать методы

```
new_disk = Disk(title=”Science Weekly”, format=”.wav”, size=100)

new_disk.save()
```

Удаление:

```
Disk.objects.filter(pk=pk).delete();
```

Изменение:

```
changing_disk = Disk.objects.get(pk=pk)
changing.size = 999
changing.save([“size”])
```

(аналогично для библиотеки, но связи указывать сложнее – надо предварительно достать id дисков с помощью интерфейса».