Разработка интернет-приложений РК №2 Карпов И.Е. ИУ5Ц-74Б

Задание

|  |  |  |
| --- | --- | --- |
| Вариант | Предметная область | |
| Класс 1 | Класс 2 |
| 26 | Преподаватель | Учебный курс |

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.

Исходный код

settings.py

*"""  
Django settings for RK\_2 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--!^k&1d=c\*%(np=p8fp#k^!jk$gd!oiulw3=fqn+nuqud5=1e='  
  
*# 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',  
 'rest\_framework',  
 'App.apps.AppConfig',  
]  
  
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 = 'RK\_2\_ls.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 = 'RK\_2\_ls.wsgi.application'  
  
  
*# Database  
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases*DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': BASE\_DIR / 'db.sqlite3',  
 }  
}  
  
  
*# 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 = 'ru-ru'  
  
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'

models.py

from django.db import models  
  
  
*# Create your models here.*class Course(models.Model):  
 name = models.CharField('Название курса', max\_length=50)  
  
  
class Teacher(models.Model):  
 course = models.ForeignKey(Course, on\_delete=models.CASCADE)  
 firstName = models.CharField('Имя', max\_length=50)  
 lastName = models.CharField('Фамилия', max\_length=50)  
 otch = models.CharField('Отчество', max\_length=50)  
  
 def fio(self):  
 return self.lastName + ' ' + self.firstName + ' ' + self.otch  
  
 def \_\_str\_\_(self):  
 return self.fio() + " | Курс: " + self.course.name

serializers.py

from rest\_framework import serializers  
from .models import Course, Teacher  
  
  
class CourseSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Course  
 fields = ['id', 'name']  
  
  
class TeacherSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Teacher  
 fields = ['id', 'course', 'firstName', 'lastName', 'otch']

views.py

from rest\_framework import viewsets  
from .serializers import CourseSerializer, TeacherSerializer  
from .models import Course, Teacher  
from django.shortcuts import render  
from django.http import Http404  
  
  
*# Create your views here.*class CourseViewSet(viewsets.ModelViewSet):  
 queryset = Course.objects.all()  
 serializer\_class = CourseSerializer  
  
  
class TeacherViewSet(viewsets.ModelViewSet):  
 queryset = Teacher.objects.all()  
 serializer\_class = TeacherSerializer  
  
  
def CourseList(request):  
 return render(request, 'course/list.html', {'c\_list': Course.objects.all()})  
  
  
def TeacherList(request):  
 return render(request, 'teacher/list.html', {'t\_list': Teacher.objects.all()})  
  
  
def CourseView(request, course\_id):  
 try:  
 c = Course.objects.get(id=course\_id)  
 except:  
 raise Http404('Курс не найден!')  
 return render(request, 'course/detail.html', {'course': c})  
  
  
def TeacherView(request, teacher\_id):  
 try:  
 t = Teacher.objects.get(id=teacher\_id)  
 except:  
 raise Http404('Преподаватель не найден!')  
 return render(request, 'teacher/detail.html', {'teacher': t})

urls.py

*"""RK\_2 URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/4.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""*from django.contrib import admin  
from django.urls import path, include  
from rest\_framework import routers  
from App import views as app\_views  
  
router = routers.DefaultRouter()  
router.register(r'courses', app\_views.CourseViewSet)  
router.register(r'teachers', app\_views.TeacherViewSet)  
  
*# Wire up our API using automatic URL routing.  
# Additionally, we include login URLs for the browsable API.*urlpatterns = [  
 path('', include(router.urls)),  
 path('admin/', admin.site.urls),  
 path('app/c\_list', app\_views.CourseList),  
 path('app/t\_list', app\_views.TeacherList),  
 path('app/c/<int:course\_id>', app\_views.CourseView, name='course\_detail'),  
 path('app/t/<int:teacher\_id>', app\_views.TeacherView, name='teacher\_detail')  
]

base.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <title>{% block title %}Рубежный контроль № 2{% endblock %}</title>  
</head>  
<body>  
  
{% block content %}{% endblock %}  
  
<hr>  
<center>Карпов И.Е. ИУ5Ц-74Б</center>  
  
</body>  
</html>

course/list.html

{% extends 'base.html' %}  
  
{% block title %}РК №2{% endblock %}  
  
{% block content %}  
  
 <center><h1>Список курсов</h1></center><hr><br>  
  
 {% if с\_list %}  
 {% for с in с\_list %}  
 <a href="{% url 'course\_detail' с.id %}">{{ с.name }}</a><br>  
 {% endfor %}  
 {% else %}  
 Список пуст  
 {% endif %}  
  
{% endblock %}

course/detail.html

{% extends 'base.html' %}  
  
{% block title %}{{ course.name }}{% endblock %}  
  
{% block content %}  
  
 <center><h1>{{ course.name }}</h1></center><hr><br>  
  
 <table border="1" cellpadding="5">  
 <tr>  
 <th>Номер курса</th>  
 <th>{{ course.name }}</th>  
 </tr>  
 </table>  
 <br><hr><br>  
 <a href="/App/c\_list">Назад</a>  
  
{% endblock %}

teacher/list.html

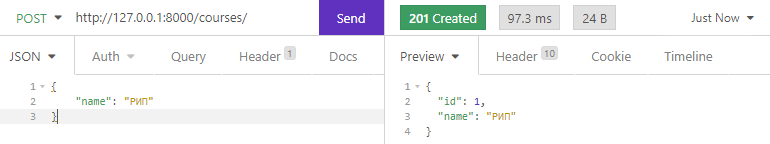
{% extends 'base.html' %}  
  
{% block title %}РК №2{% endblock %}  
  
{% block content %}  
  
 <center><h1>Список преподавателей</h1></center><hr><br>  
  
 {% if t\_list %}  
 {% for t in t\_list %}  
 <a href="{% url 'teacher\_detail' t.id %}">{{ t.fio }}</a><br>  
 {% endfor %}  
 {% else %}  
 Список пуст  
 {% endif %}  
  
{% endblock %}

teacher/detail.html

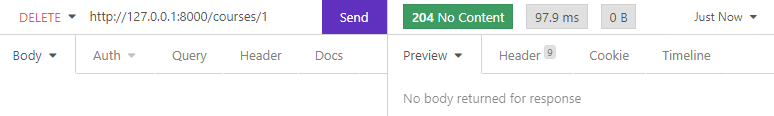
{% extends 'base.html' %}  
  
{% block title %}{{ group.number}}{% endblock %}  
  
{% block content %}  
  
 <center><h1>{{ teacher.fio }}</h1></center><hr><br>  
  
 <table border="1" cellpadding="5">  
 <tr>  
 <th>Курс</th>  
 <th>{{ teacher.course.name }}</th>  
 </tr>  
 <tr>  
 <th>Фамилия</th>  
 <th>{{ teacher.lastName }}</th>  
 </tr>  
 <tr>  
 <th>Имя</th>  
 <th>{{ teacher.firstName }}</th>  
 </tr>  
 <tr>  
 <th>Отчество</th>  
 <th>{{ teacher.otch }}</th>  
 </tr>  
 </table>  
 <br><hr><br>  
 <a href="/app/g\_list">Назад</a>  
  
{% endblock %}

Добавление, изменение и удаление данных через API

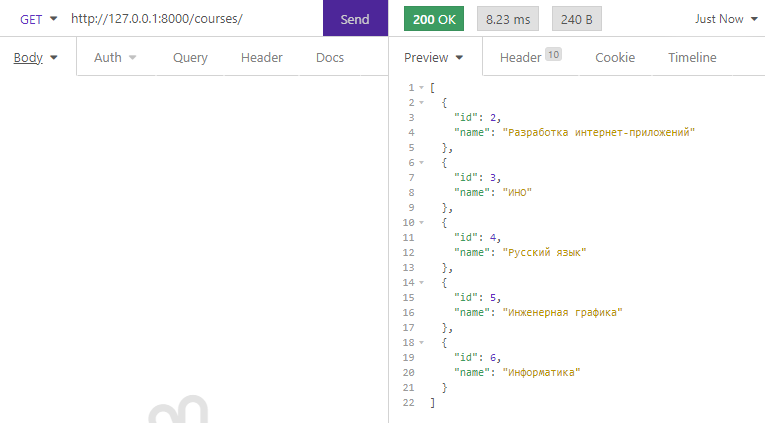
Добавление курса



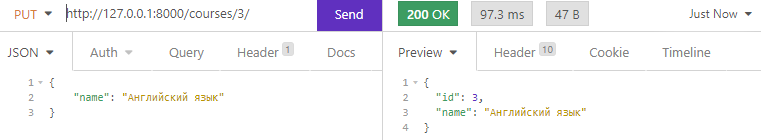
Удаление курса



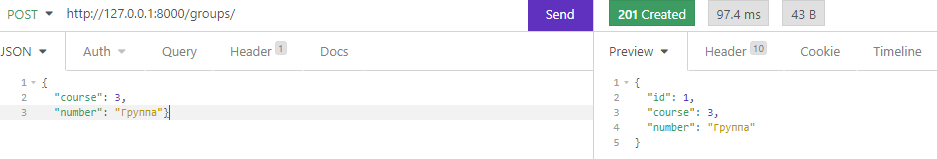
Список курсов



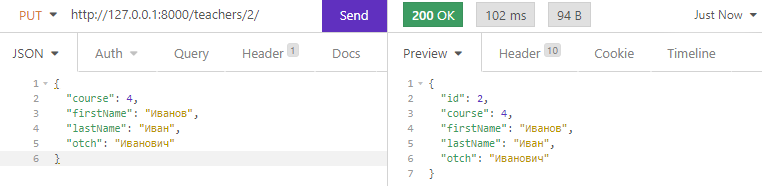
Изменение курса



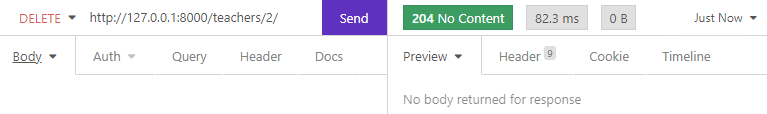
Добавление преподавателя



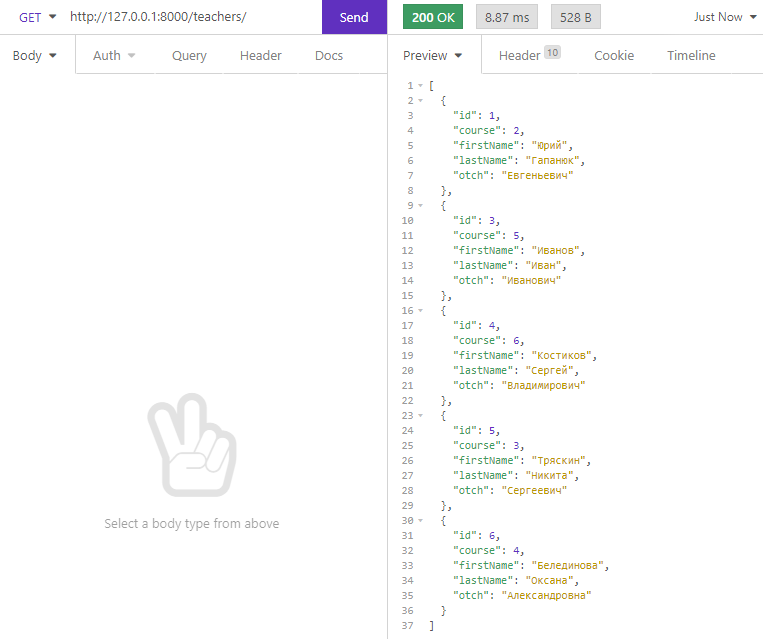
Изменение преподавателя



Удаление преподавателя

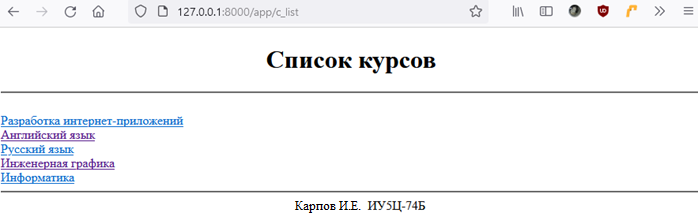


Список преподавателей



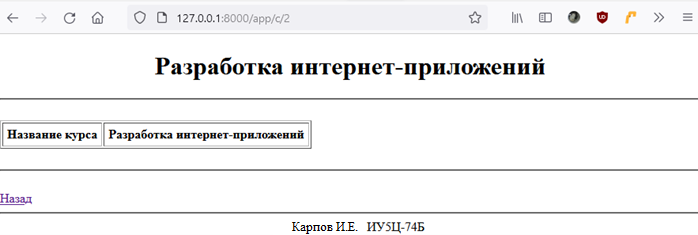
Просмотр данных через браузер

Список курсов



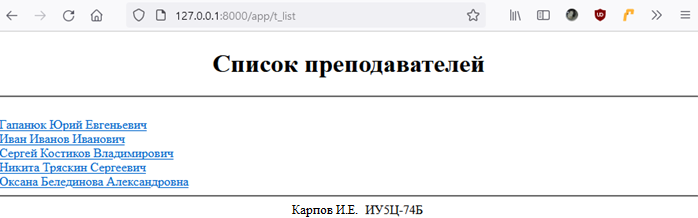


Информация о курсе





Список преподавателей





Информация о преподавателе. Соединение двух таблиц.

