МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления



**ОТЧЁТ**

**по лабораторной работе №5**

**по теме:** Знакомство с фреймворком Django

**по дисциплине:** Разработка программных приложений и WEB-программирование

Выполнил:

Студент гр. АП-226, АВТФ

Ищенко Сергей Григорьевич

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Проверил: Эстрайх Игорь Викторович

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

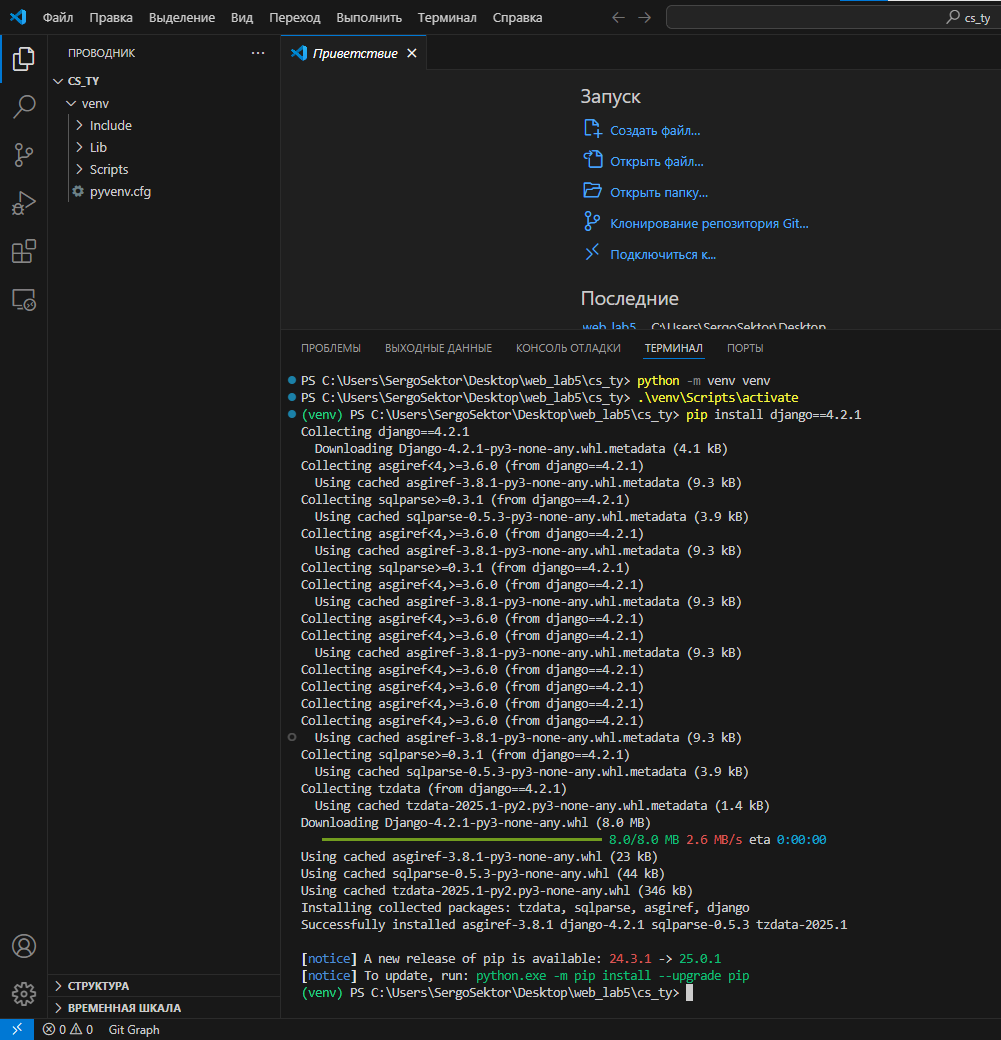
Новосибирск 2025

**Цель:** Установить фреймворк Django, изучить принципы его использования, разработать структуру проекта, добавить первые приложения.

**Задание 1:**

*1. Установить Django. Ознакомиться с теоретическим материалом и разобраться с принципами работы фреймворка.*

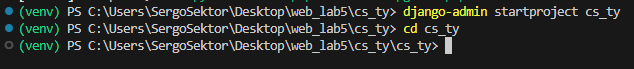
С помощью команды «python -m venv venv» создадим в vs code виртуальное окружение, активируем его командой «.\venv\Scripts\activate», установим нужную версию фреймворка Django с помощью команды «pip install django==4.2.1»



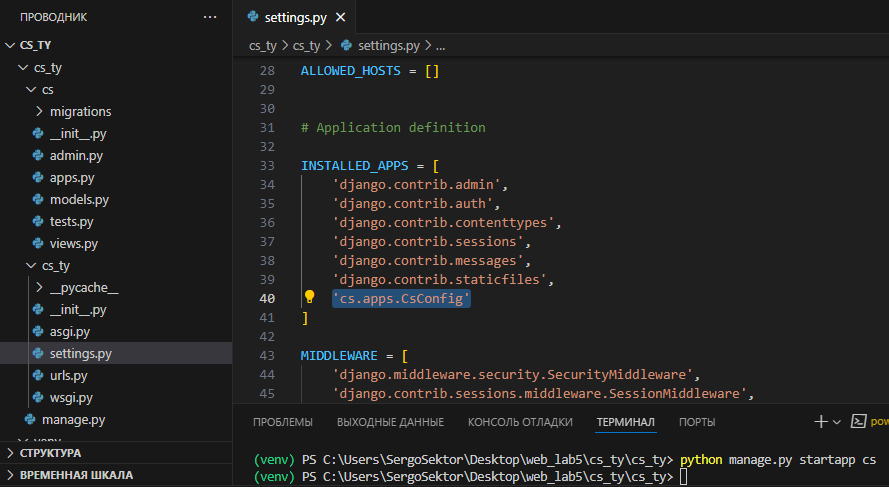
**Задание 2:**

*2. Разработать структуру собственного сайта, создать проект и добавить первые приложения.*

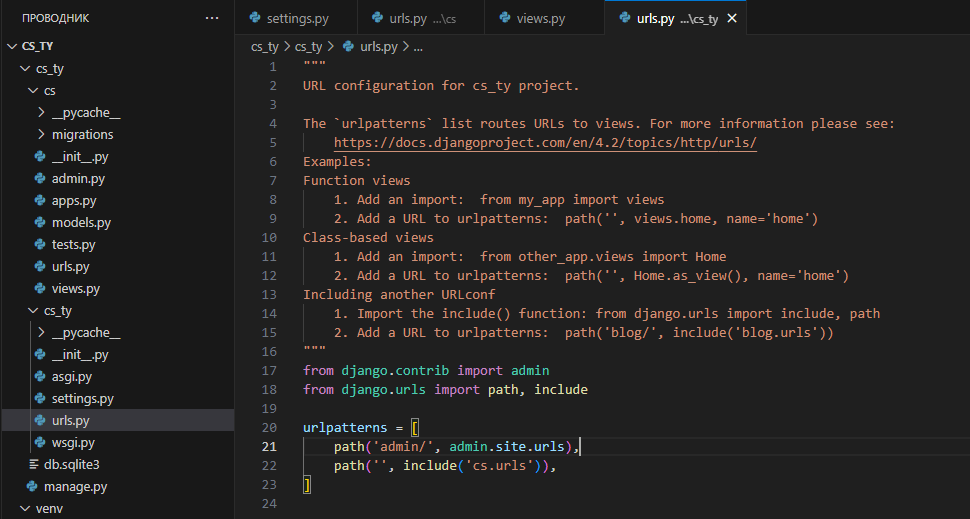
Создадим проект с помощью консольных команд. С помощью команды «cd» выберем папку проекта



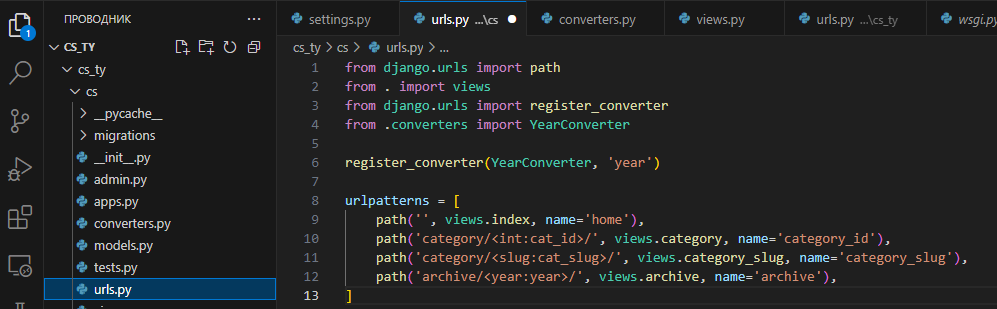
Создадим приложение и зарегистрируем в settings.py нашего проекта



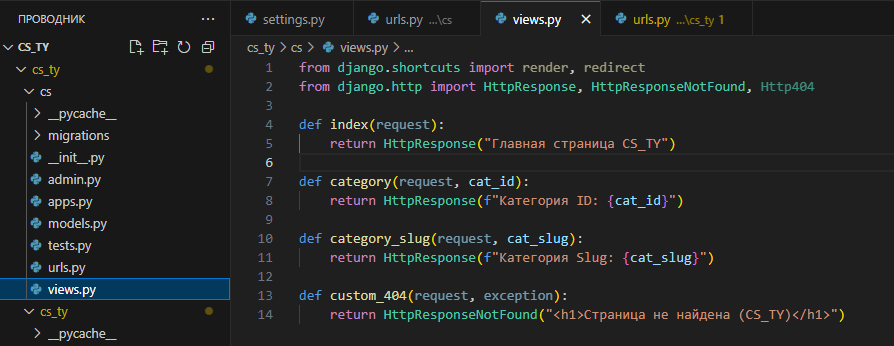
Добавим маршрутизацию в приложение,в файле urls.py нашего проекта, чтобы оно корректно работало



Создадим файл urls.py и добавим динамические маршруты на новые страницы category и archive



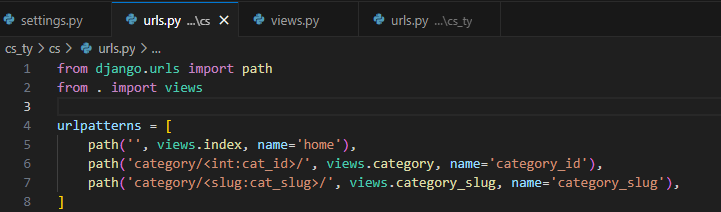
Реализуем ответы на запросы



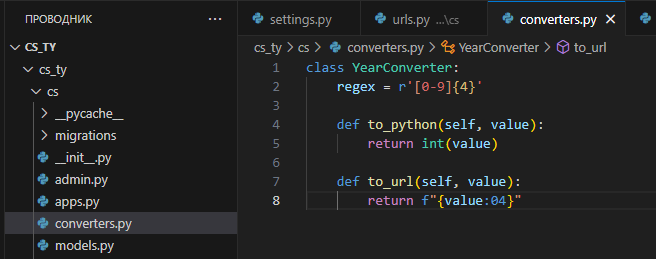
**Задание 3:**

*3. Поэкспериментировать с проектом: использовать динамические URL, собственные конвертеры, перенаправление, запросы, обработку исключений при запросах к серверу.*

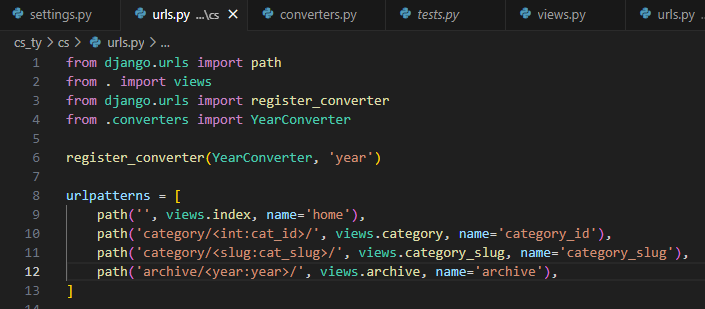
Динамические URL, path('category/<int:cat\_id>/', views.category) — динамический URL с параметром cat\_id (целое число). <int:cat\_id> — конвертер Django, преобразующий часть URL в целочисленный аргумент для функции category



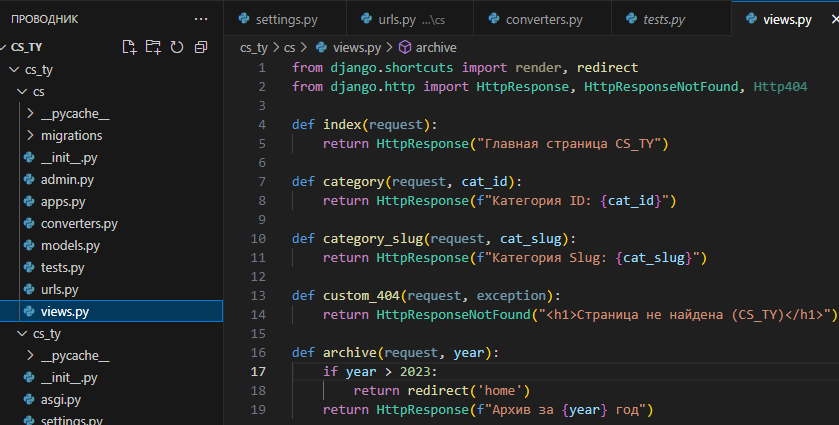
Создадим конвертер. Конвертер проверяет, что год состоит из 4 цифр (regex = r'[0-9]{4}'). to\_python преобразует строку в целое число. to\_url форматирует число для вставки в URL (например, 2023 → 2023).



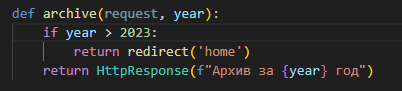
Зарегистрируем конвертер в приложении, добавим URL



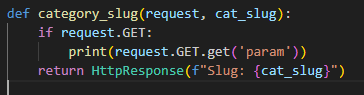
Реализуем представление archive, чтобы при переадресации открывалась страница с архивом за конкретный год



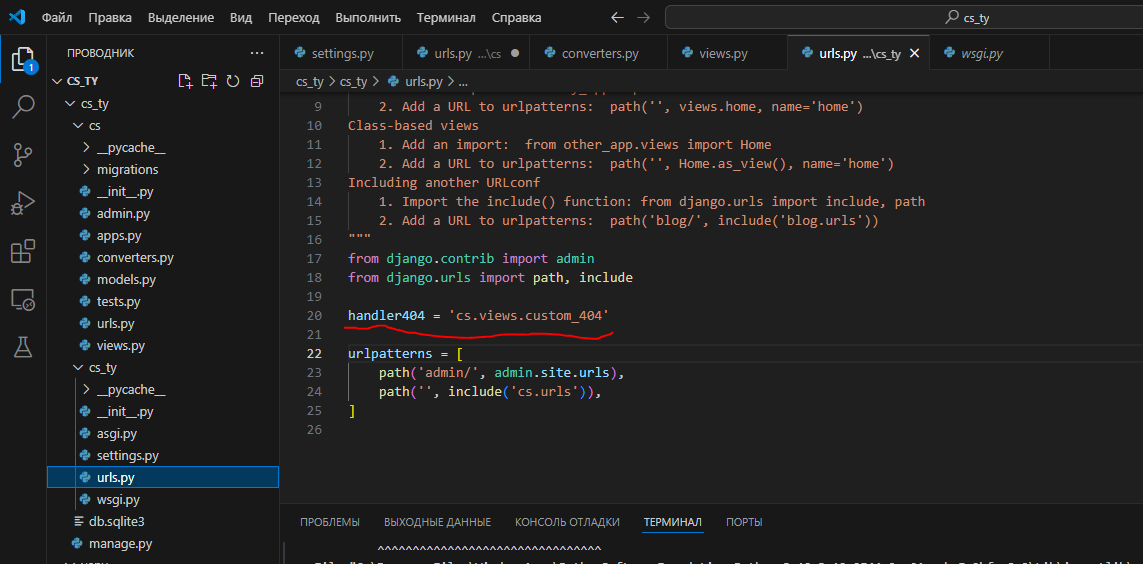
Перенаправление на главную страницу



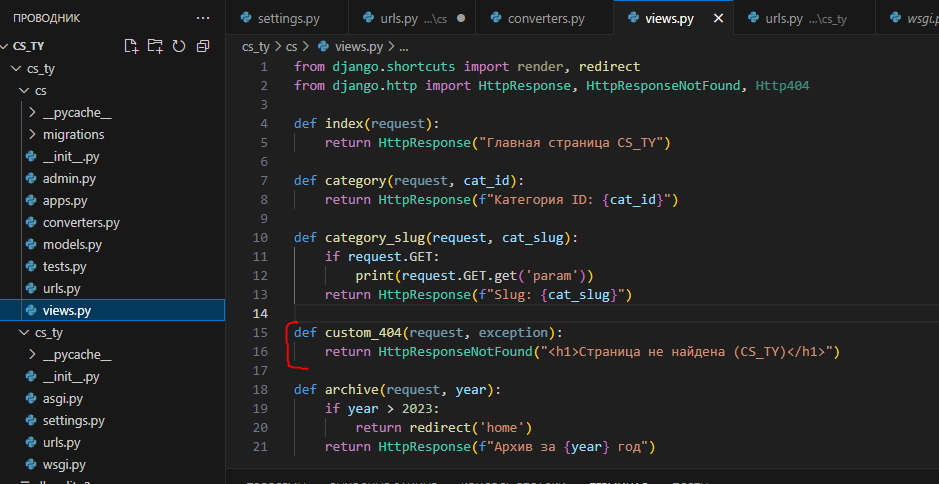
Обработка GET-запросов, то есть запросов, которые содержат определенный параметр, в данном случае cat\_slug



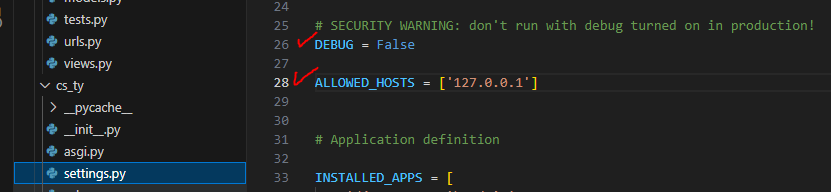
Обработка исключений. При открытии несуществующей страницы нас переадресует на специальную страницу. Укажем это в константе нашего проекта



Добавим функцию переадресации в приложении при открытии несуществующей страницы



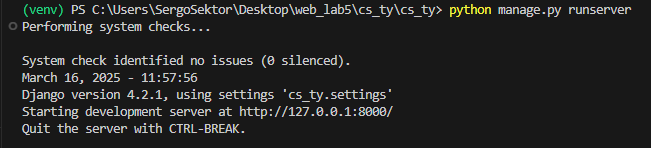
Изменение настроек приложения, чтобы работала отладка



**Задание 4:**

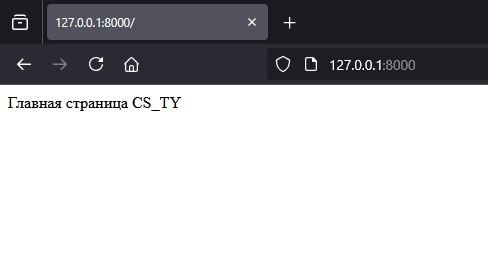
*4. Произвести отладку проекта, используя командную строку и PyCharm или VS Code.*

Запускаем локальный сервер для отладки. Сервер запускается на http://127.0.0.1:8000. В режиме отладки (DEBUG = True) Django показывает детали ошибок. В нашем случае DEBUG = False

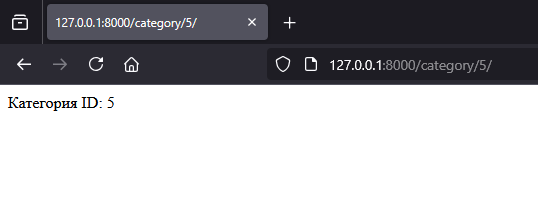


Проверяем маршруты:

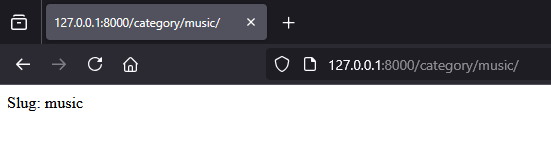
Главная страница



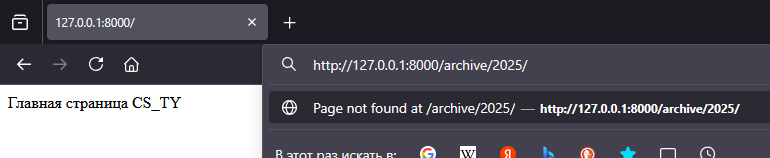
Категория ID

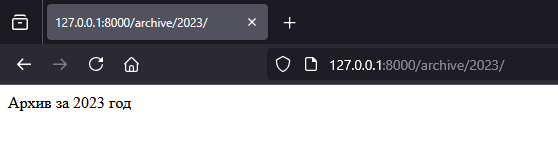


Категория Slug: music

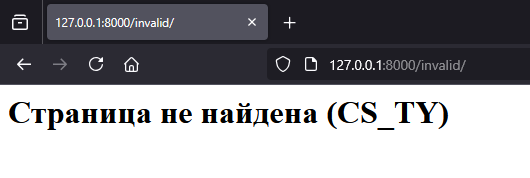


Редирект на главную страницу





Несуществующие маршруты



**Вывод:** Проект демонстрирует полноценную реализацию Django-функционала, включая динамические маршруты с параметрами, кастомные конвертеры для валидации данных, гибкие перенаправления, обработку HTTP-запросов и исключений, а также пользовательскую страницу 404, обеспечивая основу для масштабируемого веб-приложения.

**Листинг:**

***cs/converters.py***

from django.urls.converters import IntConverter

class YearConverter:

regex = r'[0-9]{4}'

def to\_python(self, value):

return int(value)

def to\_url(self, value):

return f"{value:04}"

***cs/urls.py***

from django.urls import path, register\_converter

from . import views

from .converters import YearConverter

register\_converter(YearConverter, 'year')

urlpatterns = [

path('', views.index, name='home'),

path('category/<int:cat\_id>/', views.category, name='category\_id'),

path('category/<slug:cat\_slug>/', views.category\_slug, name='category\_slug'),

path('archive/<year:year>/', views.archive, name='archive'),

]

***cs/views.py***

from django.shortcuts import render, redirect

from django.http import HttpResponse, HttpResponseNotFound, Http404

def index(request):

return HttpResponse("Главная страница CS\_TY")

def category(request, cat\_id):

return HttpResponse(f"Категория ID: {cat\_id}")

def category\_slug(request, cat\_slug):

if request.GET:

print(f"GET-параметры: {request.GET}")

return HttpResponse(f"Категория Slug: {cat\_slug}")

def archive(request, year):

if year > 2023:

return redirect('home', permanent=True)

return HttpResponse(f"Архив за {year} год")

def custom\_404(request, exception):

return HttpResponseNotFound("<h1>404: Страница не найдена</h1>")

***cs\_ty/urls.py***

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('cs.urls')),

]

handler404 = 'cs.views.custom\_404'

***cs\_ty/settings.py***

DEBUG = False # Для активации кастомной 404

ALLOWED\_HOSTS = ['127.0.0.1']