Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное   
образовательное учреждение высшего образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №1

Дисциплина: «Основы сетевого программирования»

Тема «Разработка базы данных»

Выполнили:

Воронин Павел  
Максимович,

Фомин Максим Андреевич.

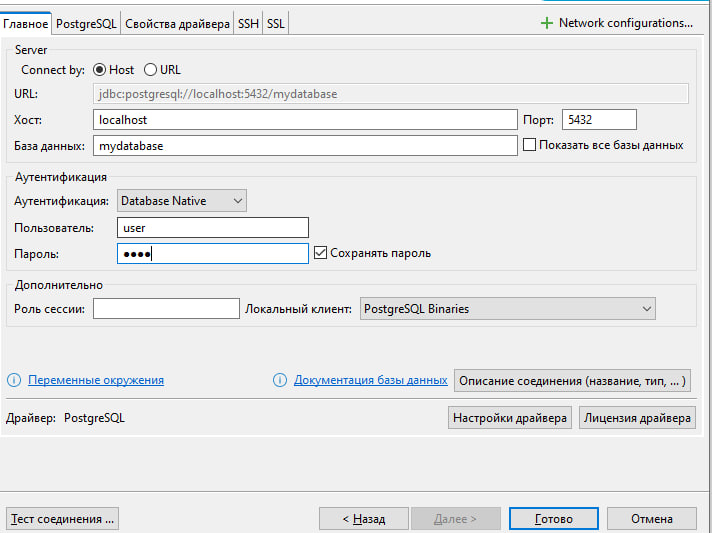
Группа: 6304-010302D

Самара 2025

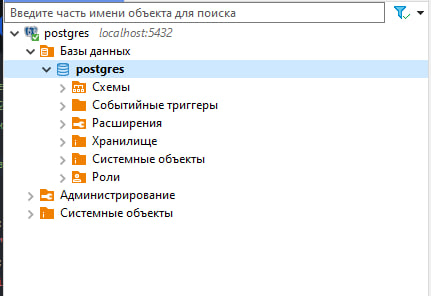
Основные задачи

1. **Настройка базы данных:**
   * Развернуть PostgreSQL в Docker;
   * Подключиться к БД через DBeaver;
2. **Разработка ORM-моделей**:
   * Определить сущности и связи между ними;
   * Реализовать модели с использованием SQLAlchemy/Django ORM.
   * Настроить миграции.
3. **Разработка модели пользователя**:
   * Определить структуру модели (email, пароль, имя и т. д.);
   * Настроить хеширование паролей.
4. **Функционал для работы с моделями**:
   * Написать скрипты для заполнения базы данных тестовыми данными.
   * Реализация функционала для работы с данными в соответствии с тематикой выбранного приложения.
5. **Оформление отчета**
   * Описать структуру базы данных
   * Подготовить скрипты создания таблиц и миграции
   * Создать презентацию с результатами работы

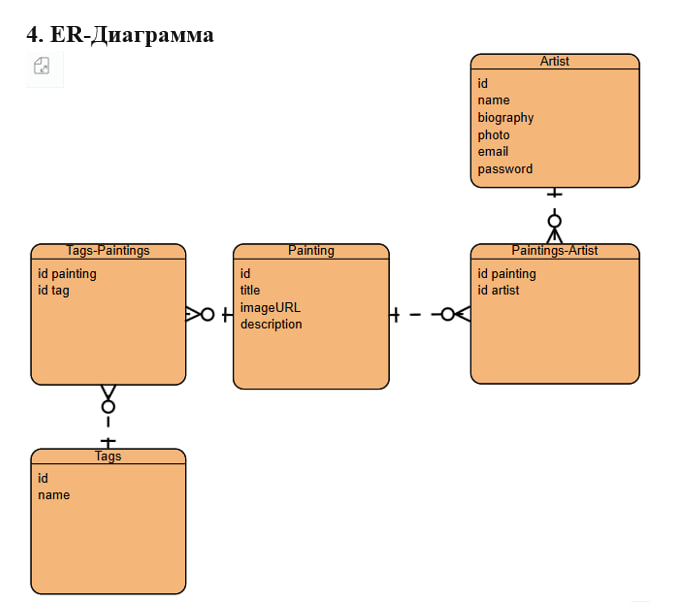
Приложение “Art Gallery” / “Галерея художников и картин”



Сначала по заданию необходимо было развернуть PostgreSQL в Docker. Для этого скачиваем с оф. сайта Docker, далее в терминале вводим docker run --name postgres-db -e POSTGRES\_PASSWORD=mysecretpassword -e POSTGRES\_USER=postgres -p 5432:5432 -d postgres и таким образом создаем БД, к которой подключаемся через DBeaver.



1. **Разработка ORM-моделей**

Согласно нашей ER-Диаграмме разработаем модели 

Из ER-диаграммы видны следующие сущности:

- Artist (Художник)

- Painting (Картина)

- Tags (Теги)

- Tags-Paintings (связь между тегами и картинами)

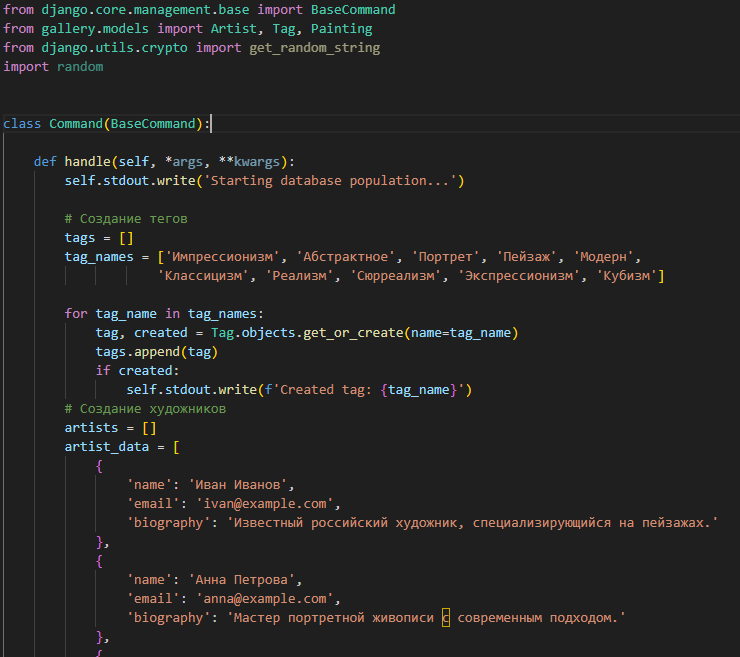
- Paintings-Artist (связь между картинами и художниками)

Для выполнения этого пункта в VSC терминале создаем новое приложение командой «python manage.py startapp gallery».

Создаем в папке проекта файл models.py и определяем внутри него наши сущности [1]

Так как мы используем Django, миграции были настроены при помощи терминальных команд python manage.py makemigrations и python manage.py migrate

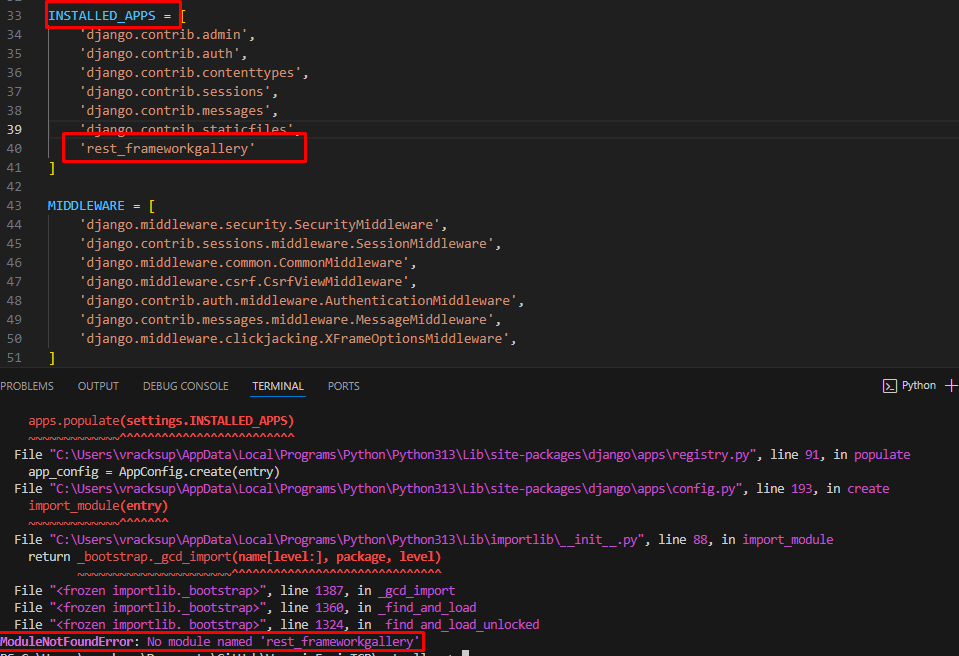
Далее была предпринята попытка заполнить базу данных скриптом.



После этого мы решили протестировать работоспособность всего и..



[2]



* **Backend**: Python, Django, PostgreSQL.
* **Дополнительные инструменты**: Docker DBeaver
* **GitHub**: <https://github.com/PeVANoD/VoroninFominTCP>

Итог.

В ходе этой лабораторной, мы попытались настроить базу данных, заполнить ее скриптом и выполнить миграции, составили примерную архитектуру и настроили стартовые инструменты и окружения.

В процессе выполнения программ мы столкнулись с чередой ошибок, которые, пока что, не знаем как исправить и правильно настроить.

Приложение

[1] from django.db import models

from django.contrib.auth.models import AbstractBaseUser, BaseUserManager, PermissionsMixin

class ArtistManager(BaseUserManager):

def create\_user(self, email, name, password=None, \*\*extra\_fields):

if not email:

raise ValueError('Email address is required')

email = self.normalize\_email(email)

user = self.model(email=email, name=name, \*\*extra\_fields)

user.set\_password(password)

user.save(using=self.\_db)

return user

def create\_superuser(self, email, name, password=None, \*\*extra\_fields):

extra\_fields.setdefault('is\_staff', True)

extra\_fields.setdefault('is\_superuser', True)

return self.create\_user(email, name, password, \*\*extra\_fields)

class Artist(AbstractBaseUser, PermissionsMixin):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=255)

biography = models.TextField(blank=True)

photo = models.ImageField(upload\_to='artists/', blank=True, null=True)

email = models.EmailField(unique=True)

password = models.CharField(max\_length=128) # Django автоматически хеширует пароли

is\_active = models.BooleanField(default=True)

is\_staff = models.BooleanField(default=False)

objects = ArtistManager()

USERNAME\_FIELD = 'email'

REQUIRED\_FIELDS = ['name']

def \_\_str\_\_(self):

return self.name

class Tag(models.Model):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=100, unique=True)

def \_\_str\_\_(self):

return self.name

class Painting(models.Model):

id = models.AutoField(primary\_key=True)

title = models.CharField(max\_length=255)

imageURL = models.URLField()

description = models.TextField(blank=True)

artist = models.ForeignKey('Artist', on\_delete=models.CASCADE, related\_name='paintings')

tags = models.ManyToManyField('Tag', related\_name='paintings')

def \_\_str\_\_(self):

return self.title

[2] Traceback (most recent call last):

File "C:\Users\vracksup\Documents\GitHub\VoroninFominTCP\artgallery\gallery\manage.py", line 22, in <module>

main()

~~~~^^

File "C:\Users\vracksup\Documents\GitHub\VoroninFominTCP\artgallery\gallery\manage.py", line 18, in main

execute\_from\_command\_line(sys.argv)

~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\core\management\\_\_init\_\_.py", line 442, in execute\_from\_command\_line

utility.execute()

~~~~~~~~~~~~~~~^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\core\management\\_\_init\_\_.py", line 436, in execute

self.fetch\_command(subcommand).run\_from\_argv(self.argv)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\core\management\commands\test.py", line 24, in run\_from\_argv

super().run\_from\_argv(argv)

~~~~~~~~~~~~~~~~~~~~~^^^^^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\core\management\base.py", line 405, in run\_from\_argv

parser = self.create\_parser(argv[0], argv[1])

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\core\management\base.py", line 368, in create\_parser

self.add\_arguments(parser)

~~~~~~~~~~~~~~~~~~^^^^^^^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\core\management\commands\test.py", line 49, in add\_arguments

test\_runner\_class = get\_runner(settings, self.test\_runner)

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\test\utils.py", line 373, in get\_runner

test\_runner\_class = test\_runner\_class or settings.TEST\_RUNNER

^^^^^^^^^^^^^^^^^^^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\conf\\_\_init\_\_.py", line 81, in \_\_getattr\_\_

self.\_setup(name)

~~~~~~~~~~~^^^^^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\conf\\_\_init\_\_.py", line 68, in \_setup

self.\_wrapped = Settings(settings\_module)

~~~~~~~~^^^^^^^^^^^^^^^^^

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\site-packages\django\conf\\_\_init\_\_.py", line 166, in \_\_init\_\_

mod = importlib.import\_module(self.SETTINGS\_MODULE)

File "C:\Users\vracksup\AppData\Local\Programs\Python\Python313\Lib\importlib\\_\_init\_\_.py", line 88, in import\_module

return \_bootstrap.\_gcd\_import(name[level:], package, level)

~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "<frozen importlib.\_bootstrap>", line 1387, in \_gcd\_import

File "<frozen importlib.\_bootstrap>", line 1360, in \_find\_and\_load

File "<frozen importlib.\_bootstrap>", line 1310, in \_find\_and\_load\_unlocked

File "<frozen importlib.\_bootstrap>", line 488, in \_call\_with\_frames\_removed

File "<frozen importlib.\_bootstrap>", line 1387, in \_gcd\_import

File "<frozen importlib.\_bootstrap>", line 1360, in \_find\_and\_load

File "<frozen importlib.\_bootstrap>", line 1324, in \_find\_and\_load\_unlocked

ModuleNotFoundError: No module named 'artgallery'