



Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра інформаційних систем та технологій

**Лабораторна робота №1**  
**З дисципліни «Основи Back-end технологій»**

Виконав:

студент групи ІІІ-13  
Карамян В.С.

Перевірив:

викладач  
Зубко Р.А.

Київ 2024

# Лабораторна робота №1

## Тема:

Python&PostgreSQL. Установку та налаштування фреймворку Django. Створення бази даних MySQL. Просте виведення даних. Обмін даними між сторінками, використовуючи гіперпосилання.

## Завдання:

1. Виконати установку та налаштування фреймворку Django
2. Налаштування PostgreSQL. Засобами psql створити нову базу даних PostgreSQL .
3. Написати скрипт, який додає нового користувача в БД.
4. Підключення PostgreSQL до Django проекту.
5. За допомогою Django створити таблиці в БД та зв'язки між ними.
6. Засобами Django і SQL виконати виведення даних з таблиці бази даних на WEB-сторінку.
7. Створити ще одну сторінку сайту і здійснити обмін даними між сторінками, використовуючи гіперпосилання.

## Варіант 2

Спроектувати базу даних про автомобілі: номер, рік випуску, марка, колір, стан, прізвище власника, адреса.

## Хід роботи

1. Налаштування Django

Створюємо новий каталог car-app-api та активуємо нове віртуальне середовище Python за допомогою команд нижче:

```
PS D:\ProjectsPy\kpi-projects> mkdir car-app-api

Каталог: D:\ProjectsPy\kpi-projects

Mode                LastWriteTime         Length Name
----                -
d-----          27.02.2024    16:11             car-app-api

PS D:\ProjectsPy\kpi-projects> cd .\car-app-api\
PS D:\ProjectsPy\kpi-projects\car-app-api> python -m venv venv
PS D:\ProjectsPy\kpi-projects\car-app-api> .\venv\Scripts\activate
(venv) PS D:\ProjectsPy\kpi-projects\car-app-api> |
```

Рис1 – Віртуальне середовище

За допомогою `pip` завантажуюмо потрібні пакети:

```
pip install django
```

```
pip install django-environ
```

```
pip install psycopg2
```

Створюємо новий Django проект та додаток `car`

```
django-admin startproject app
```

```
cd app
```

```
python manage.py startapp car
```

Запускаємо проект

```
python runserver
```

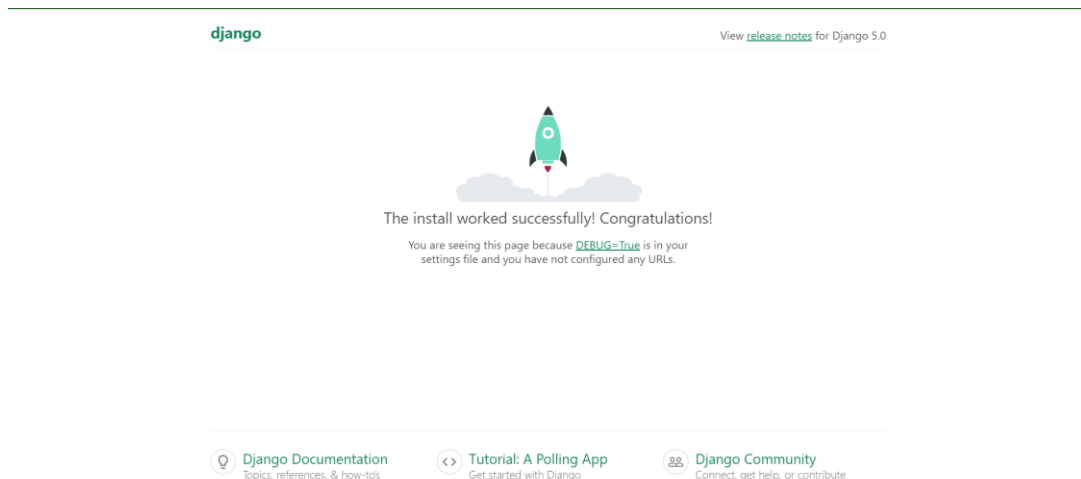


Рис 2 – Запущений проект

## 2. Налаштування PostgreSQL

Використовуючи `psql` підключаємося до існуючої бази даних та створюємо нову базу даних `car_db`:

```
(venv) PS D:\ProjectsPy\kpi-projects\backend-labs\car-app-api> psql -d postgres -U postgres
Password for user postgres:
psql (16.2)
WARNING: Console code page (866) differs from Windows code page (1251)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

postgres=# CREATE DATABASE car_db;
CREATE DATABASE
postgres=# \c car_db
You are now connected to database "car_db" as user "postgres".
car_db=#
```

Рис 3 – Створення бази даних

### 3. Створення користувача

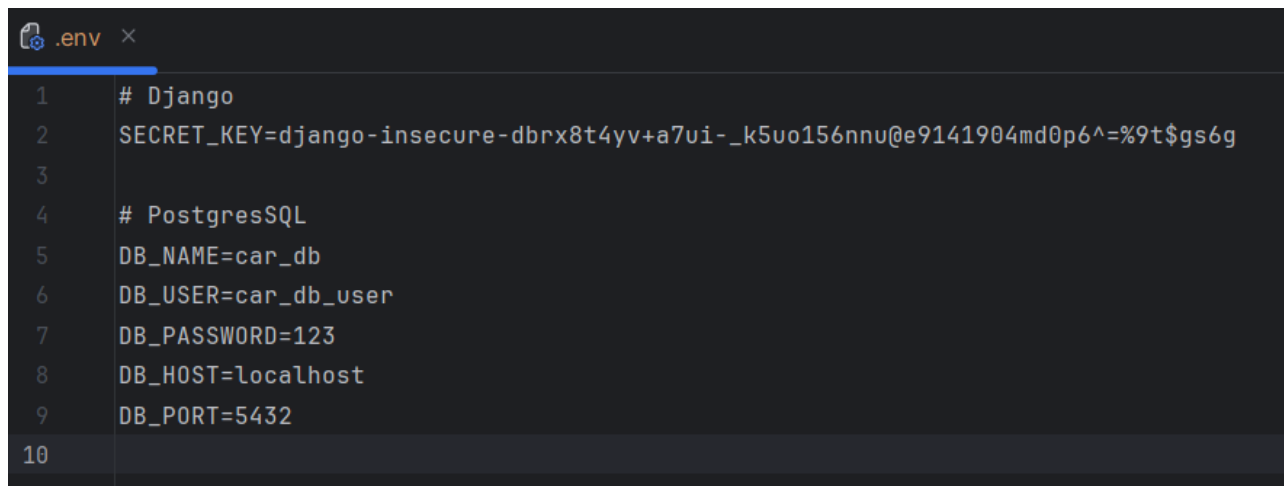
Створюємо нового користувача та надаємо йому права для роботи з базою даних:

```
car_db=# CREATE ROLE car_db_user WITH PASSWORD '123';
CREATE ROLE
car_db=# ALTER ROLE car_db_user WITH LOGIN;
ALTER ROLE
car_db=# GRANT ALL PRIVILEGES ON DATABASE car_db TO car_db_user;
GRANT
car_db=# GRANT ALL ON SCHEMA public TO car_db_user;
GRANT
car_db=#
```

Рис 4 – Створення нового користувача

### 4. Підключення PostgreSQL до Django проекту.

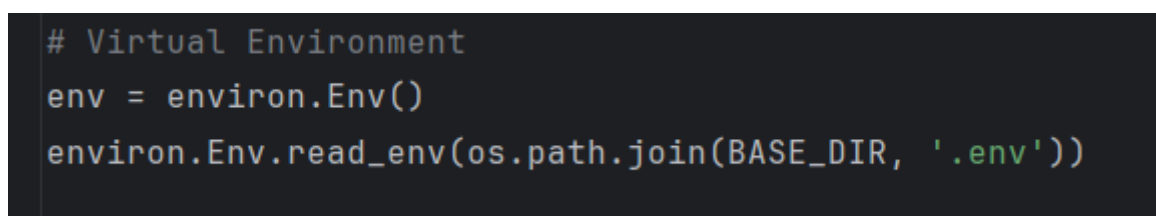
Додаємо налаштування бази даних у змінні середовища за допомогою модуля `django-environ`. Для цього створюємо `.env` файл:



```
.env x
1 # Django
2 SECRET_KEY=django-insecure-dbrx8t4yv+a7ui-_k5uo156nnu@e9141904md0p6^=%9t$gs6g
3
4 # PostgreSQL
5 DB_NAME=car_db
6 DB_USER=car_db_user
7 DB_PASSWORD=123
8 DB_HOST=localhost
9 DB_PORT=5432
10
```

Рис 5 – .env файл

Вносимо зміни у файл settings.py:



```
# Virtual Environment
env = environ.Env()
environ.Env.read_env(os.path.join(BASE_DIR, '.env'))
```

Рис 6 – Віртуальне середовище



```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': env.str('DB_NAME'),
        'USER': env.str('DB_USER'),
        'PASSWORD': env.str('DB_PASSWORD'),
        'HOST': env.str('DB_HOST'),
        'PORT': env.str('DB_PORT'),
    }
}
```

Рис 7 – Підключення PostgreSQL

5. За допомогою Django створити таблиці в БД та зв'язки між ними.

Розглянемо структуру бази даних за допомогою ER-моделі:

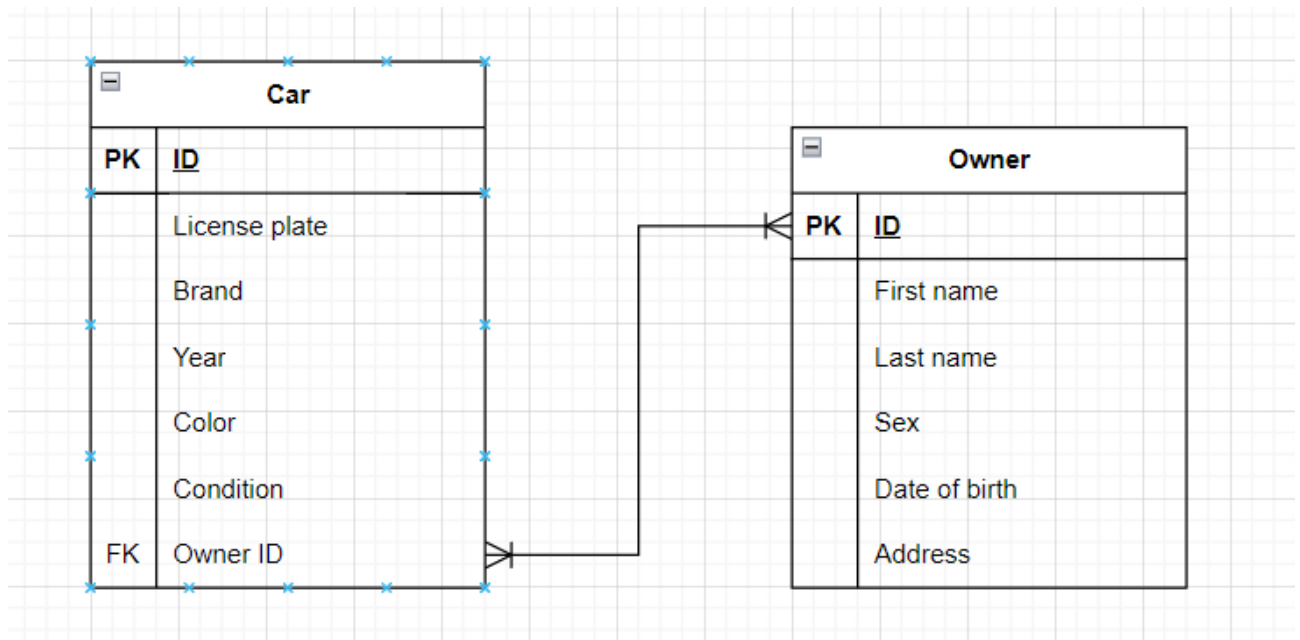


Рис 8 – ER-модель бази даних

За допомогою DjangoORM створюємо таблиці та зв'язки між ними:

```
.env settings.py models.py x
1 usage
4 class Owner(models.Model):
5     """Car owner model. Represents an owner of the car"""
6     GENDER_CHOICES = [
7         ('M', 'Male'),
8         ('F', 'Female'),
9     ]
10
11     first_name = models.CharField(max_length=255)
12     last_name = models.CharField(max_length=255)
13     sex = models.CharField(max_length=1, choices=GENDER_CHOICES)
14     date_of_birth = models.DateField()
15     address = models.CharField(max_length=255)
16
17     def __str__(self):
18         return f"Owner #{self.pk}. {self.first_name} {self.last_name}"
19
20
21 class Car(models.Model):
22     """Car Model"""
23     license_plate = models.CharField(max_length=10, unique=True)
24     brand = models.CharField(max_length=255)
25     year = models.IntegerField()
26     color = models.CharField(max_length=255)
27     condition = models.CharField(max_length=255)
28     owner = models.ForeignKey(Owner, on_delete=models.CASCADE)
29
30     def __str__(self):
31         return f"{self.color} {self.brand}"
32
```

Рис 9 – Створення моделей

Проводимо міграції для застосування змін:

```
(venv) PS D:\ProjectsPy\kpi-projects\backend-labs\car-app-api\app> python .\manage.py makemigrations
Migrations for 'car':
  car\migrations\0001_initial.py
    - Create model Owner
    - Create model Car
(venv) PS D:\ProjectsPy\kpi-projects\backend-labs\car-app-api\app> python .\manage.py makemigrations
No changes detected
(venv) PS D:\ProjectsPy\kpi-projects\backend-labs\car-app-api\app> python .\manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, car, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
```

Рис 10 – Міграції

Перевіряємо чи з'явилися таблиці у базі даних:

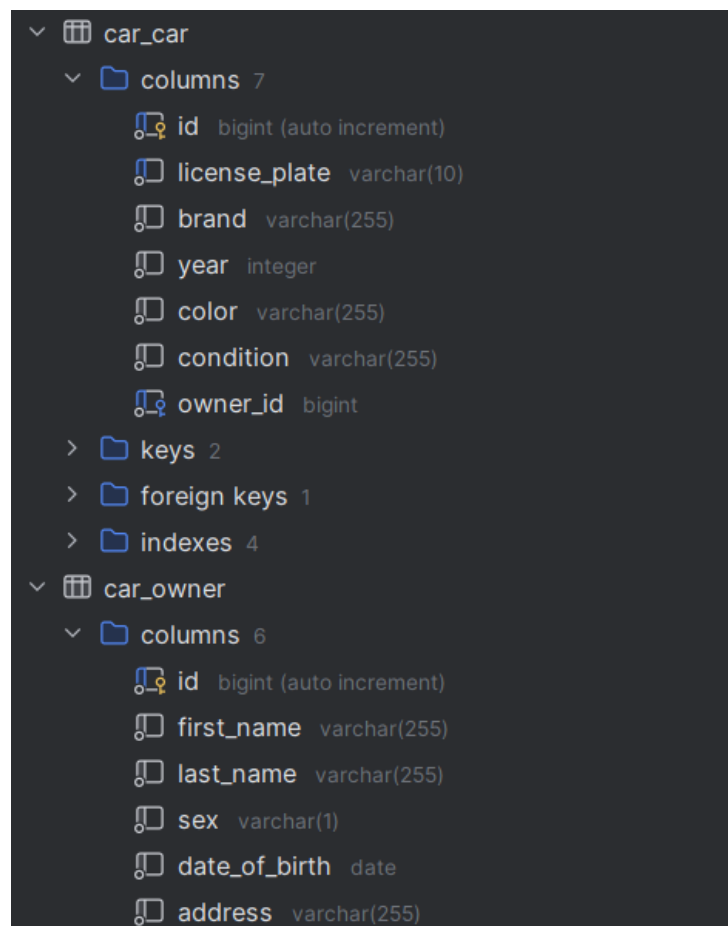


Рис 11 – Нові таблиці у базі даних

6. Засобами Django і SQL виконати виведення даних з таблиці бази даних на WEB-сторінку.

Додаємо декілька об'єктів у базу даних, для цього скористаємося Django administration:

Створимо двох власників автомобілів у таблиці owner:

The screenshot shows the 'Change owner' form in the Django Admin interface. The form is for 'Owner №1 | Vartan Karamian'. It includes fields for 'First name' (Vartan), 'Last name' (Karamian), 'Sex' (Male), 'Date of birth' (2004-05-27), and 'Address' (Kyiv, Ukraine). There are three buttons at the bottom: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Рис 12 – Створення власника автомобіля

The screenshot shows the 'owner' table view in the Django Admin interface. It displays a table with 7 columns: id, first\_name, last\_name, sex, date\_of\_birth, and address. There are two rows of data.

	id	first_name	last_name	sex	date_of_birth	address
1	1	Vartan	Karamian	M	2004-05-27	Kyiv, Ukraine
2	2	John	Smith	M	1992-12-01	New-York, USA

Рис 13 – Таблиця owner

Також додаємо декілька машин у таблицю car:

The screenshot shows the 'car' table view in the Django Admin interface. It displays a table with 8 columns: id, license\_plate, brand, year, color, condition, and owner\_id. There are five rows of data.

	id	license_plate	brand	year	color	condition	owner_id
1	1	KA2804EM	Skoda	2006	Black	Used	1
2	2	AB1784HT	Volkswagen	2004	Gray	Used	1
3	3	A9L881	Tesla	2023	Gray	New	2
4	4	7J2398	Ferrari	2017	Red	Crashed	2
5	5	782H6A	Toyota	2003	Black	Used	2

Рис 14 – Таблиця car

Виведемо вміст таблиці car на веб-сторінку:

The screenshot shows a code editor with Python code for rendering the car table. The code imports 'render' from 'django.shortcuts' and 'Car' from 'car.models'. It defines a function 'car\_table(request)' that retrieves all car objects and renders them using the 'car\_table.html' template, passing the cars as context.

```
1 from django.shortcuts import render
2
3 from car.models import Car
4
5
6 2 usages
7 def car_table(request):
8     cars = Car.objects.all()
9     return render(request, template_name='car_table.html', context={'cars': cars})
```

Рис 15 – Представлення для cars

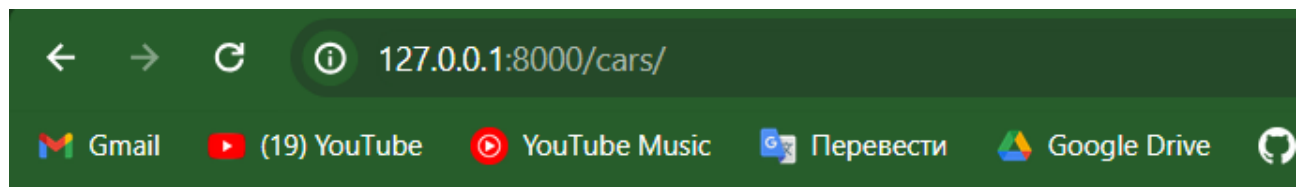


```

<table>
  <thead>
    <tr>
      <th>License Plate</th>
      <th>Brand</th>
      <th>Year</th>
      <th>Color</th>
      <th>Condition</th>
      <th>Owner</th>
    </tr>
  </thead>
  <tbody>
    {% for car in cars %}
    <tr>
      <td>{{ car.license_plate }}</td>
      <td>{{ car.brand }}</td>
      <td>{{ car.year }}</td>
      <td>{{ car.color }}</td>
      <td>{{ car.condition }}</td>
      <td>{{ car.owner }}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>

```

Рис 16 – Структура car\_table.html



## Car List

License Plate	Brand	Year	Color	Condition	Owner
KA2804EM	Skoda	2006	Black	Used	№1   Vartan Karamian
AB1784HT	Volkswagen	2004	Gray	Used	№1   Vartan Karamian
A9L881	Tesla	2023	Gray	New	№2   John Smith
7J2398	Ferarri	2017	Red	Crashed	№2   John Smith
782H6A	Toyota	2003	Black	Used	№2   John Smith

Рис 17 – Веб-сторінка /cars

7. Створити ще одну сторінку сайту і здійснити обмін даними між сторінками, використовуючи гіперпосилання.

Додаємо нове представлення, що прийматиме Id власника та буде виводити інформацію про нього:

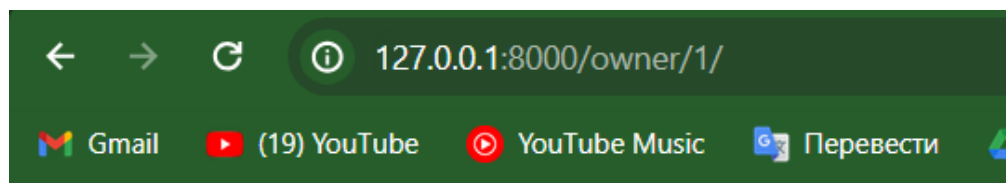
```
10 2 usages
11 def owner_detail(request, owner_id):
12     owner = Owner.objects.get(id=owner_id)
13     return render(request, template_name='owner_detail.html', context={'owner': owner})
14
```

Рис 18 – Представлення owner\_detail

Html-сторінка:

```
<h1>Owner Detail</h1>
<table>
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Sex</th>
      <th>Date of Birth</th>
      <th>Address</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>{{ owner.first_name }}</td>
      <td>{{ owner.last_name }}</td>
      <td>{{ owner.get_sex_display }}</td>
      <td>{{ owner.date_of_birth }}</td>
      <td>{{ owner.address }}</td>
    </tr>
  </tbody>
</table>
```

Рис 19 – Html-сторінка owner\_detail



## №1 | Vartan Karamian

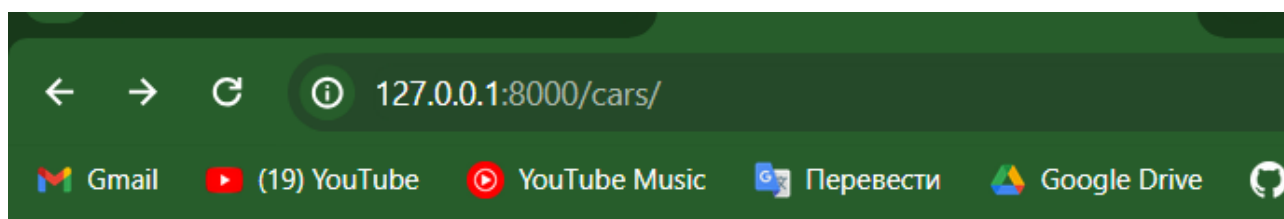
First Name	Last Name	Sex	Date of Birth	Address
Vartan	Karamian	Male	May 27, 2004	Kyiv, Ukraine

Рис 20 – Веб-сторінка owner/1

Додамо гіперпосилання у car\_table.html:

```
<tr>
  <td>{{ car.license_plate }}</td>
  <td>{{ car.brand }}</td>
  <td>{{ car.year }}</td>
  <td>{{ car.color }}</td>
  <td>{{ car.condition }}</td>
  <td><a href="{% url 'owner' car.owner.id %}">{{ car.owner }}</a></td>
</tr>
```

Рис 21 – Гіперпосилання



## Car List

License Plate	Brand	Year	Color	Condition	Owner
KA2804EM	Skoda	2006	Black	Used	<a href="#">№1</a>   <a href="#">Vartan Karamian</a>
AB1784HT	Volkswagen	2004	Gray	Used	<a href="#">№1</a>   <a href="#">Vartan Karamian</a>
A9L881	Tesla	2023	Gray	New	<a href="#">№2</a>   <a href="#">John Smith</a>
7J2398	Ferarri	2017	Red	Crashed	<a href="#">№2</a>   <a href="#">John Smith</a>
782H6A	Toyota	2003	Black	Used	<a href="#">№2</a>   <a href="#">John Smith</a>

Рис 22 – Веб-сторінка cars/ з гіперпосиланнями

### Код програми:

<https://github.com/Vartan14/car-app-api/>

### Висновок

У результаті виконання лабораторної роботи було успішно встановлено та налаштовано фреймворк Django та базу даних PostgreSQL. За допомогою Django ORM та SQL було створено необхідні таблиці в базі даних та встановлено зв'язки між ними. Для цього було використано можливості Django для опису моделей даних та створення міграцій для бази даних.

Виконано виведення даних з таблиці бази даних на веб-сторінку. Це було досягнуто шляхом створення відповідного представлення (view) у Django та шаблону, який відображає дані у вигляді таблиці. Також, була створена ще одна сторінка сайту, на якій реалізовано обмін даними між сторінками за допомогою гіперпосилань. Це дозволяє користувачам зручно переміщатися між різними частинами веб-додатку та отримувати необхідну інформацію.

Загалом, завдяки виконанню цих кроків, було успішно реалізовано основні функціональність та взаємодія між різними частинами веб-додатку, що дозволяє його ефективно використовувати та подальший розвиток.