

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни **Бази даних і засоби управління**

*на тему: “***Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL***”*

Виконав:

студент ІII курсу

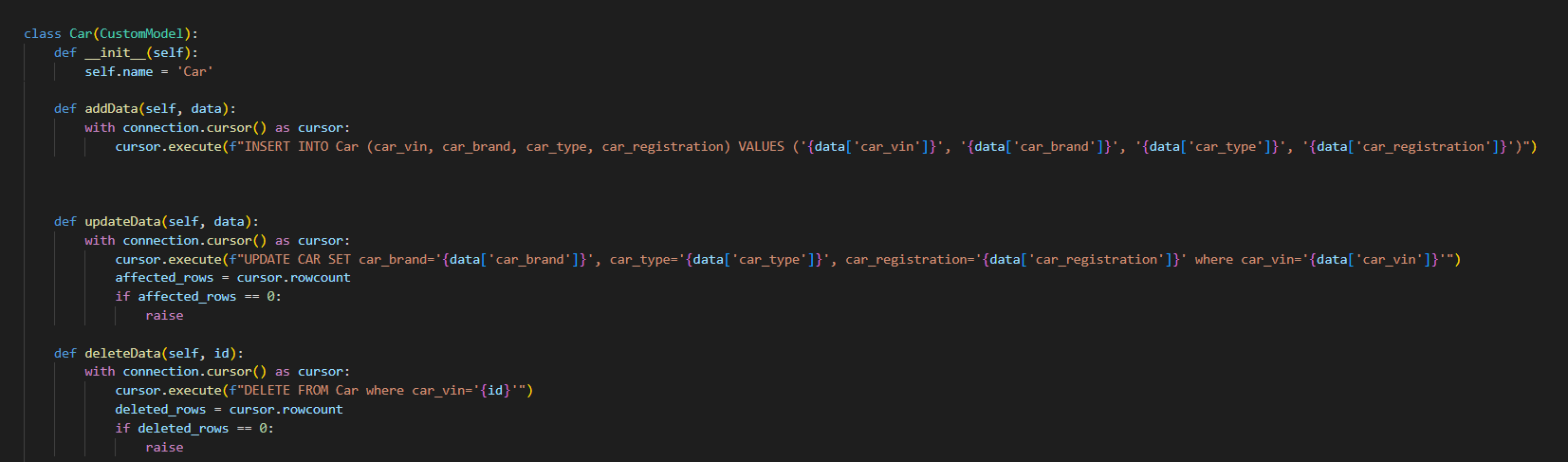
групи КВ-01

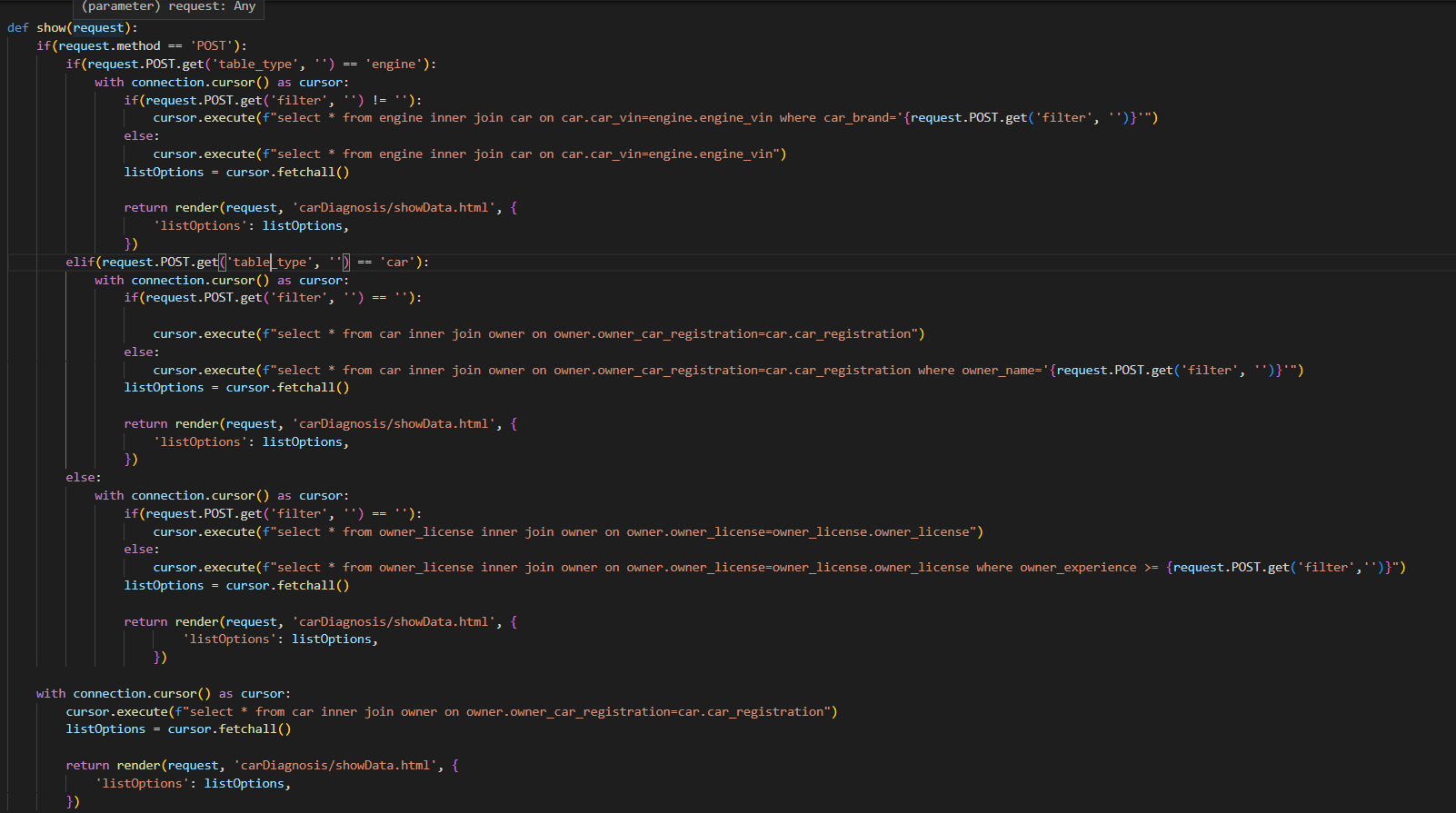
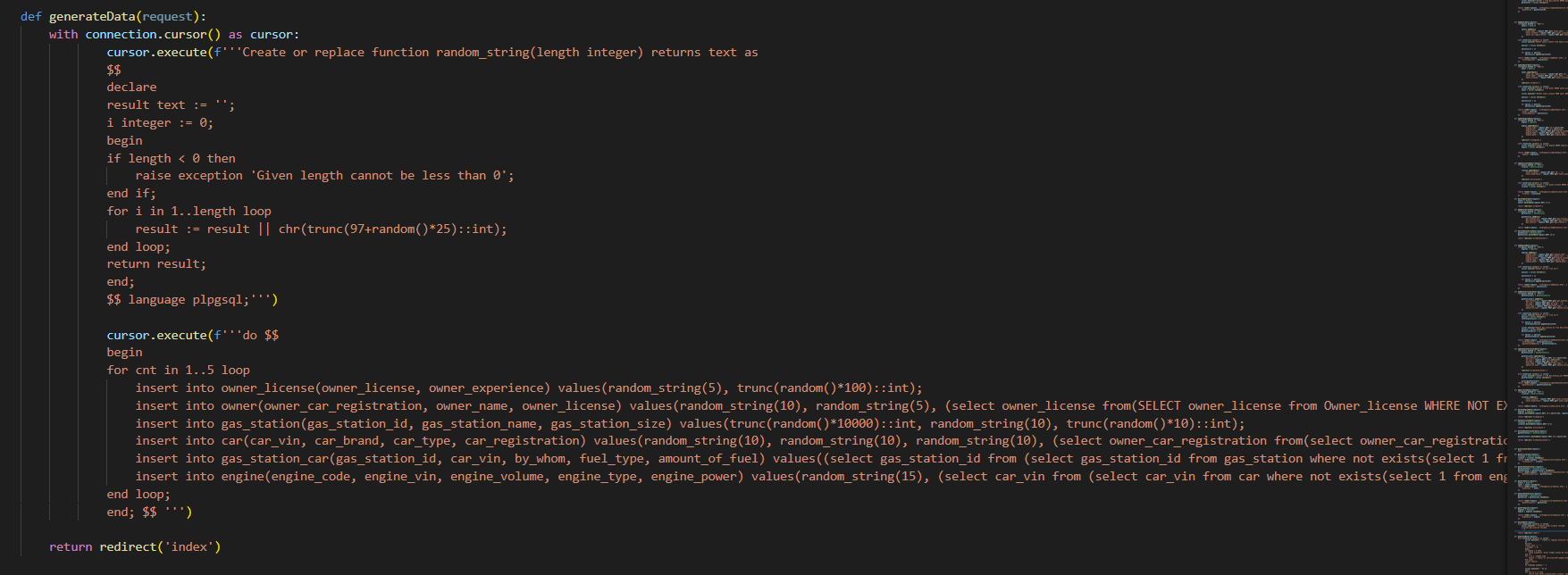
Шкільнюк В.О.

Перевірив:

Павловський В. І.

Київ – 2022

****

****

**Завдання роботи полягає у наступному**:

* + - 1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
      2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
      3. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
      4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

**Деталізоване завдання:**

Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати вилучення рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні внесення нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.

Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими не мовою програмування, а відповідним SQL-запитом!

Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

Програмний код організувати згідно шаблону Model-View-Controller(MVC). Приклад організації коду згідно шаблону доступний за даним посиланням. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати лише мову SQL.

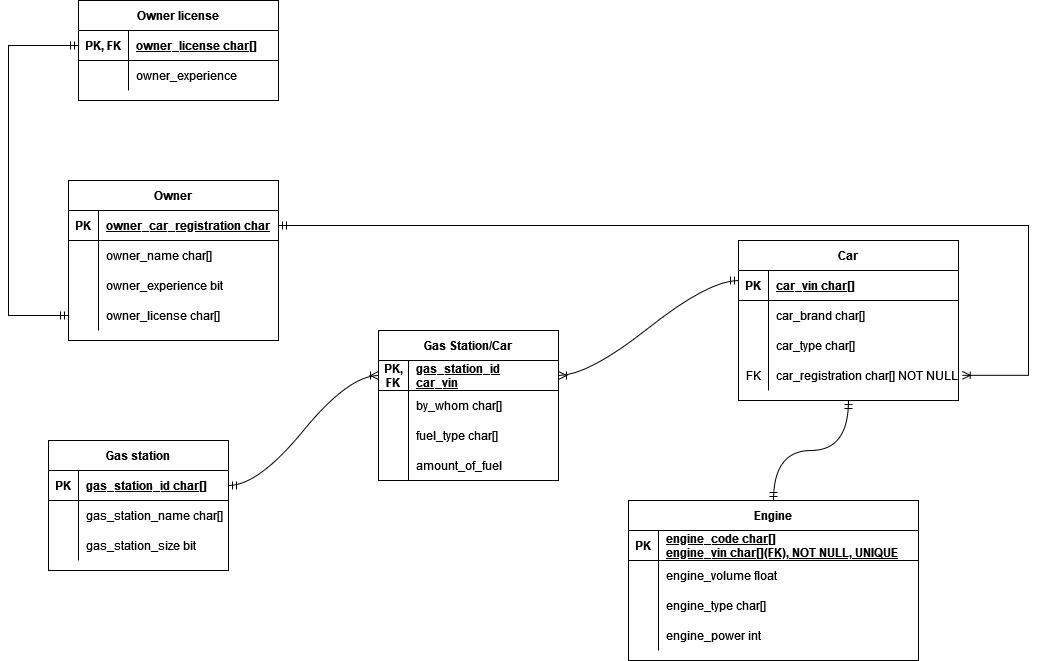
**Логічна модель предметної області «Сервісний центр з обслуговування»**

Рисунок 1. Схема бази даних

**Середовище та компоненти розробки**

Для розробки використовувалась мова програмування Python, фреймворк Django, а також стороння бібліотека, що надає API для доступу до PostgreSQL – psycopg2.

**Шаблон проектування**

MVC - Шаблон проектування, який використаний у програмі.

Django фреймворк надає чудовий набір інструментів для застосування шаблону MVC. Моделі описані у файлі models.py, відображення у фалі view.py, а контролери у urls.py.

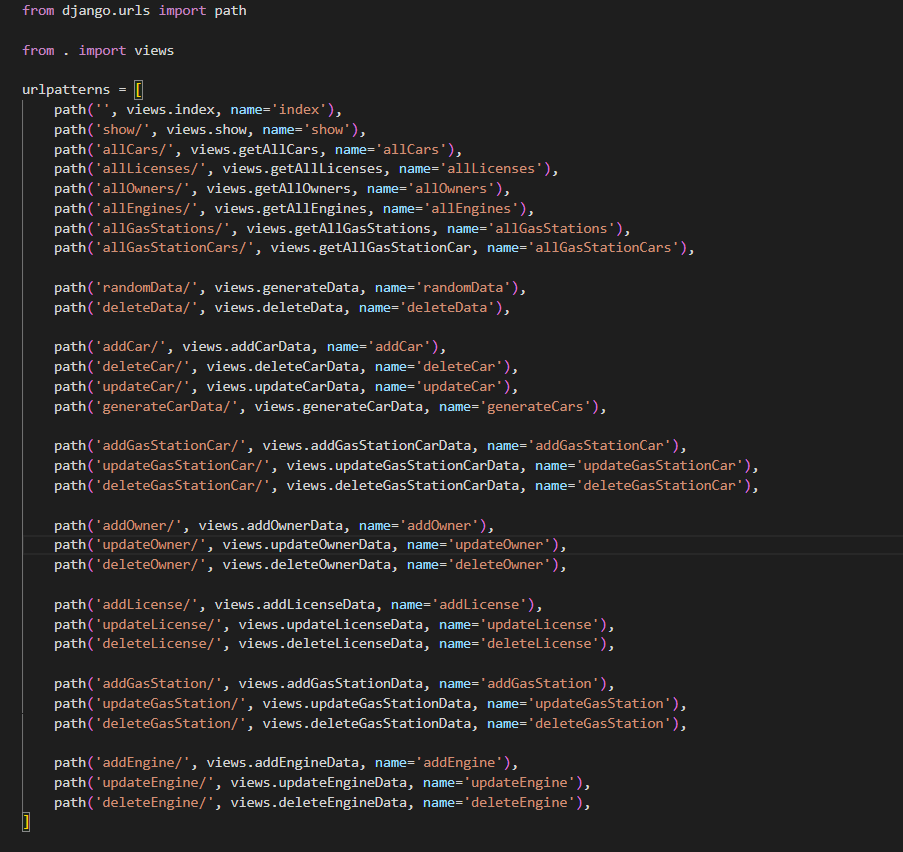
****

Рисунок 2. urls.py

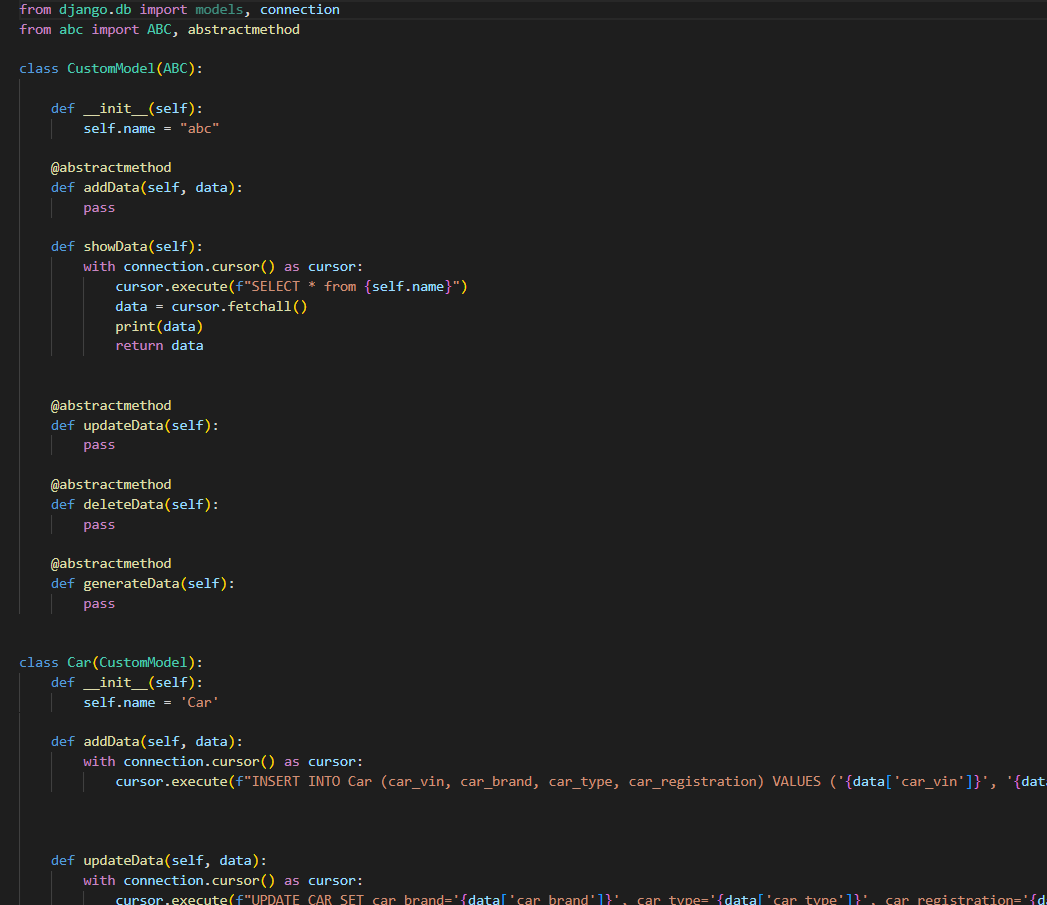
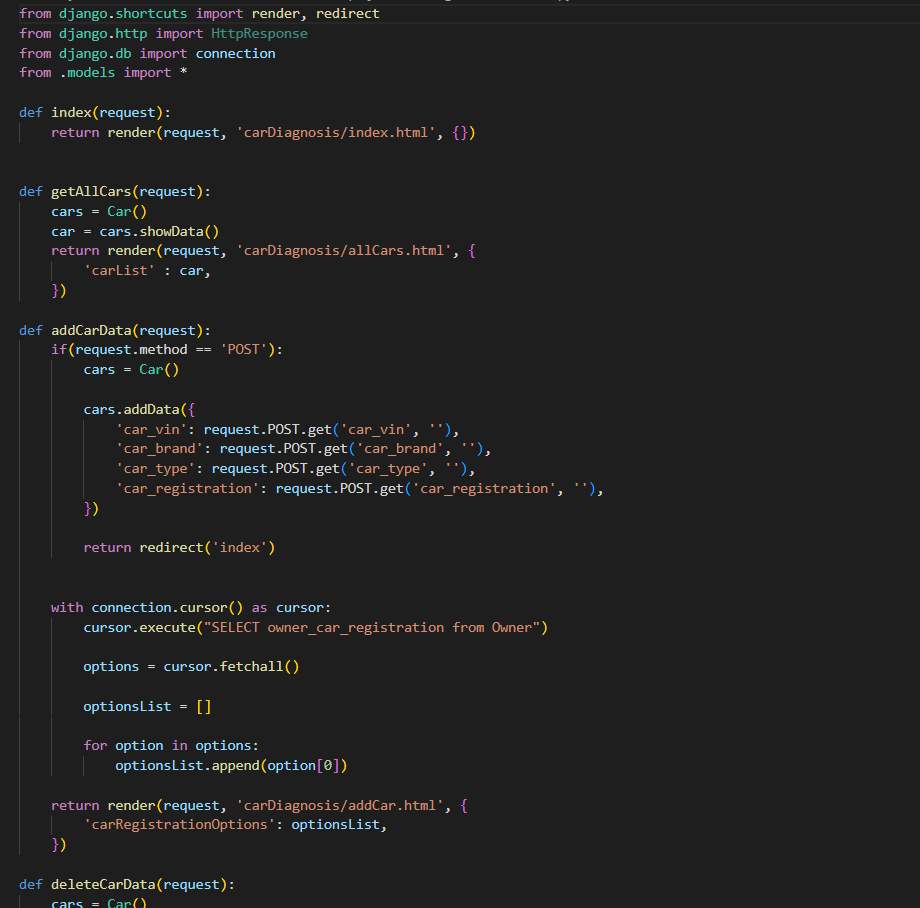


Рисунок 3. models.py

Рисунок 4. views.py

**Демонстрація коректної роботи видалення з таблиць зі зв’язком, у якому видаляється батько.**

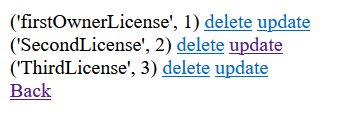


Табл. Owner\_license

Таблиця з ліцензіями – батьківська для власника.

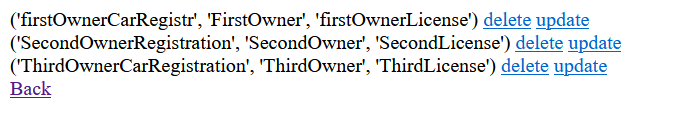


Табл. Owner

Після видалення маємо:

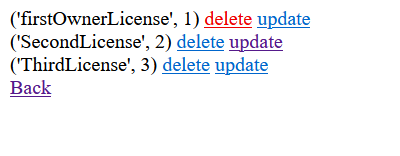
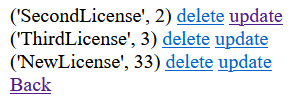


Табл. Owner\_license



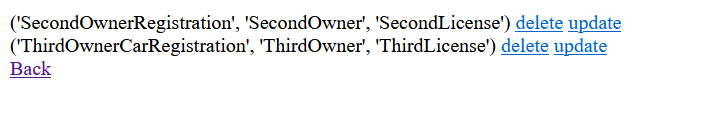
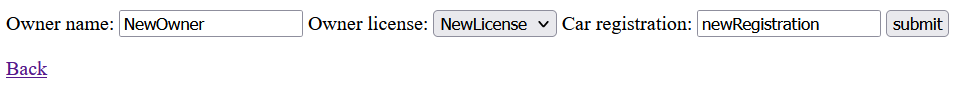
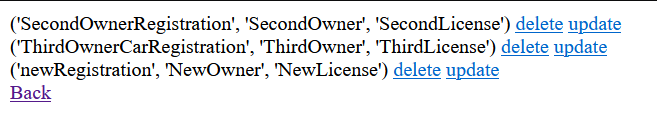


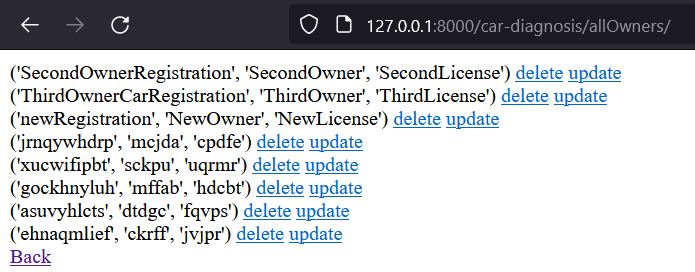
Табл. Owner

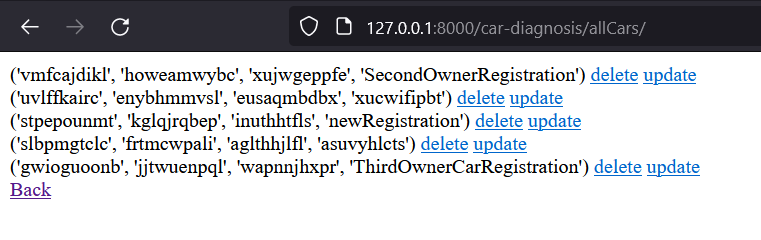


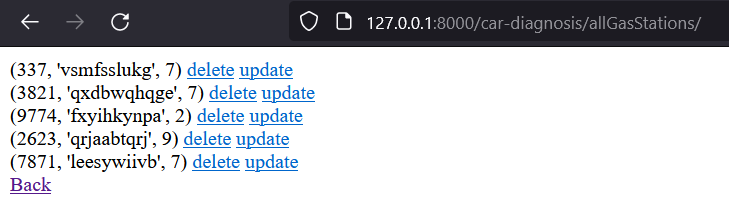


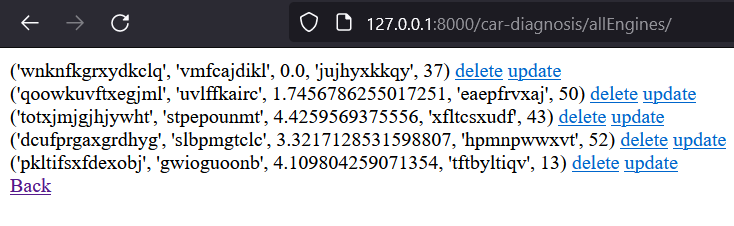
Вставка реалізована таким чином, що пов’язати можна лише з ліцензією, яка ще не пов’язана, тому помилка виключається.

Демонстрація згенерованих даних відповідними функціями.









def generateData(request):

    with connection.cursor() as cursor:

            cursor.execute(f'''Create or replace function random\_string(length integer) returns text as

            $$

            declare

            result text := '';

            i integer := 0;

            begin

            if length < 0 then

                raise exception 'Given length cannot be less than 0';

            end if;

            for i in 1..length loop

                result := result || chr(trunc(97+random()\*25)::int);

            end loop;

            return result;

            end;

            $$ language plpgsql;''')

            cursor.execute(f'''do $$

            begin

            for cnt in 1..5 loop

                insert into owner\_license(owner\_license, owner\_experience) values(random\_string(5), trunc(random()\*100)::int);

                insert into owner(owner\_car\_registration, owner\_name, owner\_license) values(random\_string(10), random\_string(5), (select owner\_license from(SELECT owner\_license from Owner\_license WHERE NOT EXISTS(SELECT 1 FROM Owner WHERE Owner.owner\_license = owner\_license.owner\_license)) as foo ORDER BY random() limit 1));

                insert into gas\_station(gas\_station\_id, gas\_station\_name, gas\_station\_size) values(trunc(random()\*10000)::int, random\_string(10), trunc(random()\*10)::int);

                insert into car(car\_vin, car\_brand, car\_type, car\_registration) values(random\_string(10), random\_string(10), random\_string(10), (select owner\_car\_registration from(select owner\_car\_registration from owner where not exists(select 1 from car where car\_registration = owner\_car\_registration)) as foo order by random() limit 1 ));

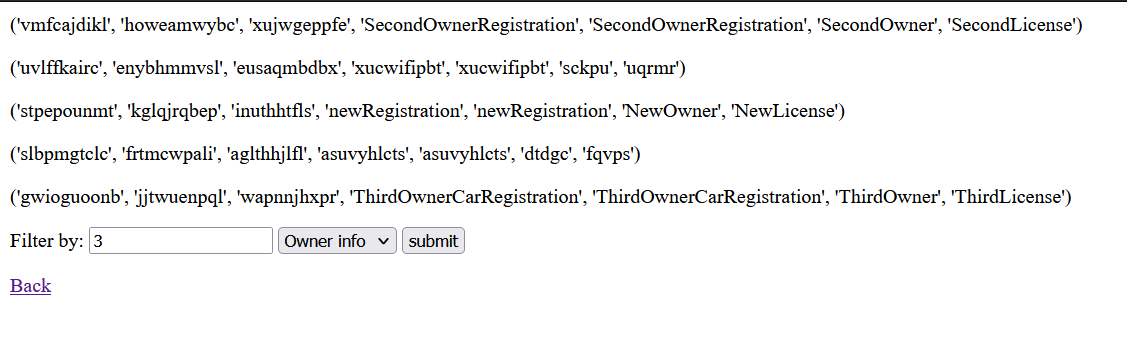
                insert into gas\_station\_car(gas\_station\_id, car\_vin, by\_whom, fuel\_type, amount\_of\_fuel) values((select gas\_station\_id from (select gas\_station\_id from gas\_station where not exists(select 1 from gas\_station\_car where gas\_station\_car.gas\_station\_id = gas\_station.gas\_station\_id)) as boo ORDER BY random() limit 1), (select car\_vin from (select car\_vin from car where not exists(select 1 from gas\_station\_car where gas\_station\_car.car\_vin = car.car\_vin)) as foo ORDER BY random() limit 1), random\_string(5), trunc(random()\*100)::int, trunc(random()\*100)::int);

                insert into engine(engine\_code, engine\_vin, engine\_volume, engine\_type, engine\_power) values(random\_string(15), (select car\_vin from (select car\_vin from car where not exists(select 1 from engine where engine.engine\_vin = car.car\_vin)) as foo ORDER BY random() limit 1), trunc(random()\*10)/random()/3::float, random\_string(10), trunc(random()\*100)::int);

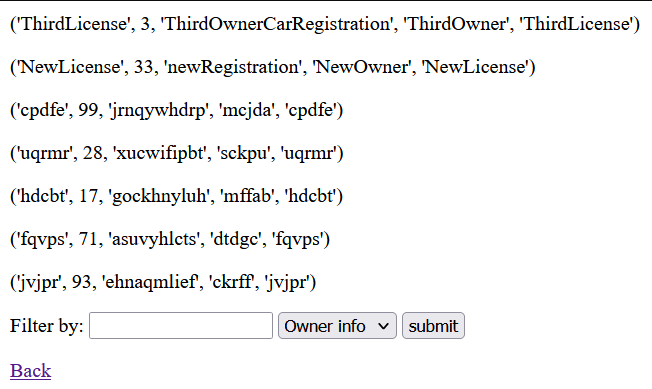
            end loop;

            end; $$ ''')

    return redirect('index')



**Фільтр за кількістю років стажу**



def show(request):

    if(request.method == 'POST'):

        if(request.POST.get('table\_type', '') == 'engine'):

            with connection.cursor() as cursor:

                if(request.POST.get('filter', '') != ''):

                    cursor.execute(f"select \* from engine inner join car on car.car\_vin=engine.engine\_vin where car\_brand='{request.POST.get('filter', '')}'")

                else:

                    cursor.execute(f"select \* from engine inner join car on car.car\_vin=engine.engine\_vin")

                listOptions = cursor.fetchall()

                return render(request, 'carDiagnosis/showData.html', {

                    'listOptions': listOptions,

                })

        elif(request.POST.get('table\_type', '') == 'car'):

            with connection.cursor() as cursor:

                if(request.POST.get('filter', '') == ''):

                    cursor.execute(f"select \* from car inner join owner on owner.owner\_car\_registration=car.car\_registration")

                else:

                    cursor.execute(f"select \* from car inner join owner on owner.owner\_car\_registration=car.car\_registration where owner\_name='{request.POST.get('filter', '')}'")

                listOptions = cursor.fetchall()

                return render(request, 'carDiagnosis/showData.html', {

                    'listOptions': listOptions,

                })

        else:

            with connection.cursor() as cursor:

                if(request.POST.get('filter', '') == ''):

                    cursor.execute(f"select \* from owner\_license inner join owner on owner.owner\_license=owner\_license.owner\_license")

                else:

                    cursor.execute(f"select \* from owner\_license inner join owner on owner.owner\_license=owner\_license.owner\_license where owner\_experience >= {request.POST.get('filter','')}")

                listOptions = cursor.fetchall()

                return render(request, 'carDiagnosis/showData.html', {

                        'listOptions': listOptions,

                    })

    with connection.cursor() as cursor:

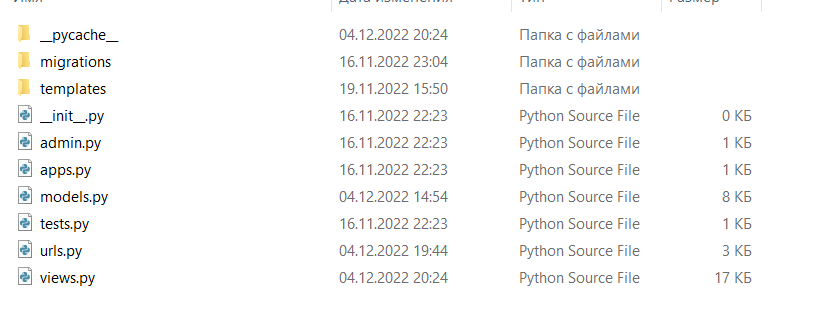
        cursor.execute(f"select \* from car inner join owner on owner.owner\_car\_registration=car.car\_registration")

        listOptions = cursor.fetchall()

        return render(request, 'carDiagnosis/showData.html', {

            'listOptions': listOptions,

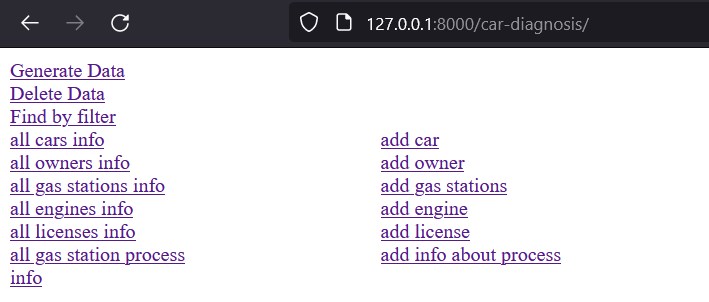
        })



У файлі models.py описані класи таблиць та базові дії з ними щодо видалення, редагування та вставки. Вся логіка взаємодії з базою даних переважно лежить у цьому файлі.

У файлі urls.py лежать відповідності між url адресами та відображеннями, які будуть підключатись.

У файлі views.py лежать безпосередньо відображення, які оброблюють запити та надсилають данні на html шаблони, які лежать у папці templates.

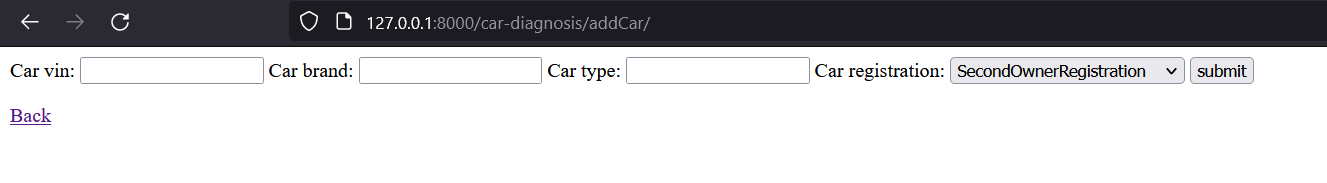


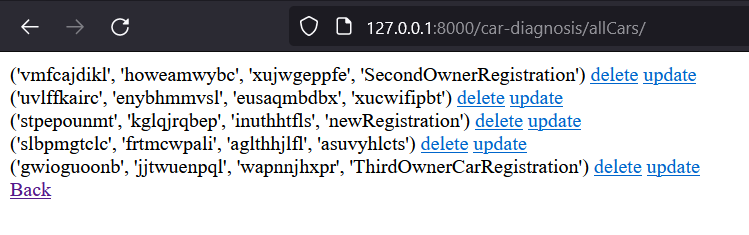
Головна сторінка для взаємодії

Generate Data – генерує рандомізовані дані.

Delete Data – видаляє данні з усіх таблиць.

Find by filter – базовий пошук по фільтру.

Шаблонна сторінка додавання



Шаблонна сторінка виведення даних