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 III 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:

1. 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.

2. 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!

3. 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.

4. 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 «Магазин»

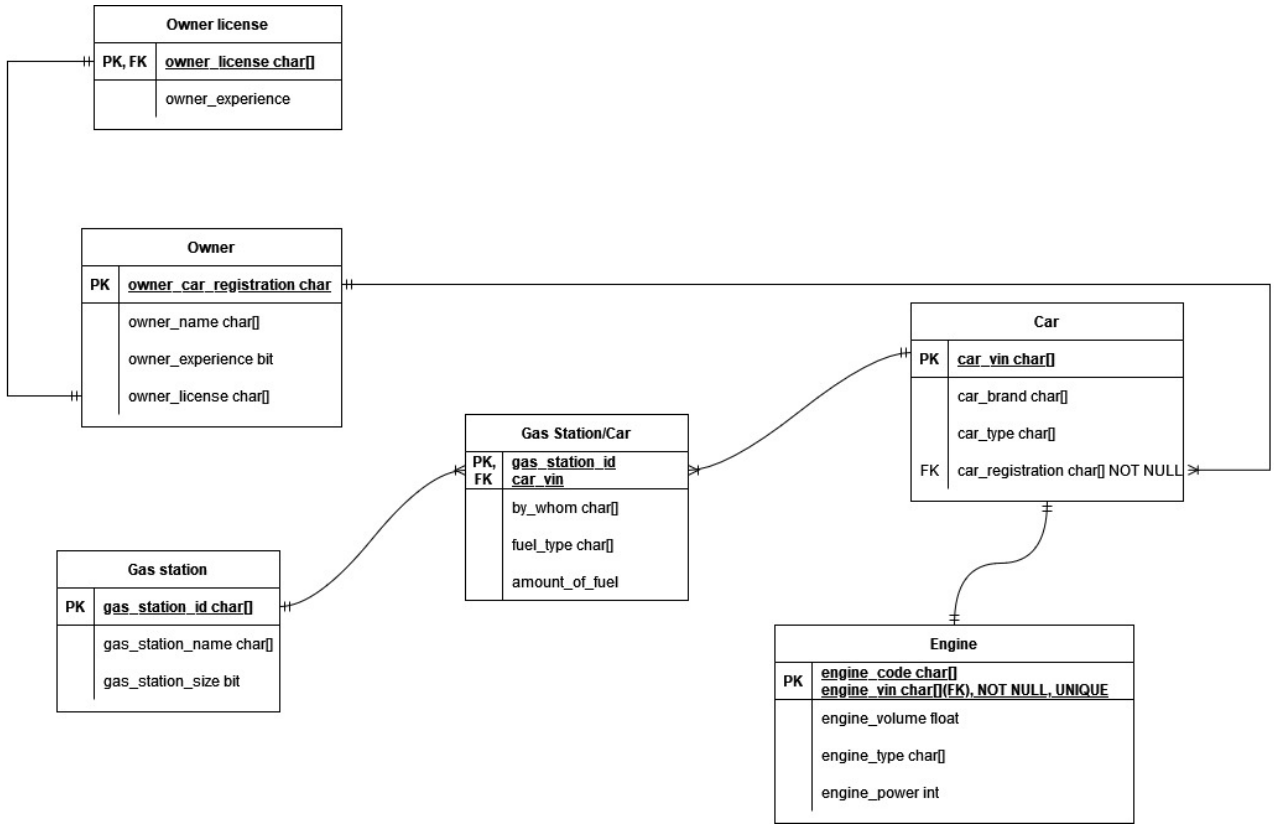


Рисунок 1. Схема бази даних

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.

Пункт 1.

('firstOwnerLicense', 1) delete update
('SecondLicense', 2) delete update
('ThirdLicense', 3) delete update
Back

Табл. Owner_license

Таблиця з ліцензіями – батьківська для власника.

('firstOwnerCarRegistr', 'FirstOwner', 'firstOwnerLicense') delete update
('SecondOwnerRegistration', 'SecondOwner', 'SecondLicense') delete update
('ThirdOwnerCarRegistration', 'ThirdOwner', 'ThirdLicense') delete update
Back

Табл. Owner

Після видалення маємо:

('firstOwnerLicense', 1) delete update
('SecondLicense', 2) delete update
('ThirdLicense', 3) delete update
Back

Табл. Owner_license

('SecondLicense', 2) delete update
('ThirdLicense', 3) delete update
('NewLicense', 33) delete update
Back

('SecondOwnerRegistration', 'SecondOwner', 'SecondLicense') delete update
('ThirdOwnerCarRegistration', 'ThirdOwner', 'ThirdLicense') delete update
Back

Табл. Owner

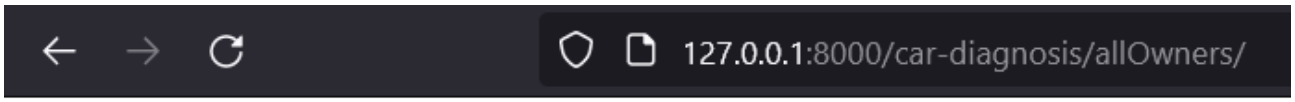
Owner name: NewOwner Owner license: NewLicense Car registration: newRegistration submit

Back

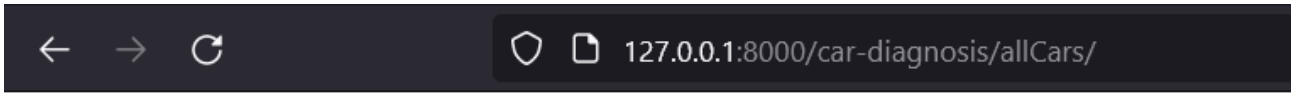
('SecondOwnerRegistration', 'SecondOwner', 'SecondLicense') delete update
('ThirdOwnerCarRegistration', 'ThirdOwner', 'ThirdLicense') delete update
('newRegistration', 'NewOwner', 'NewLicense') delete update
Back

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

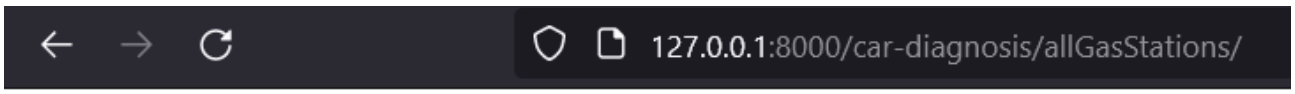
Пункт 1.



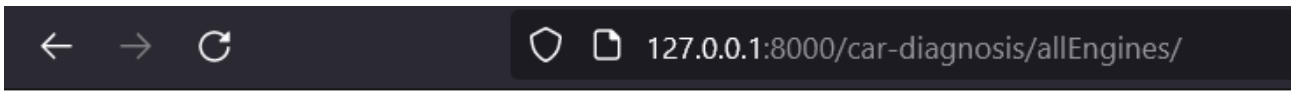
('SecondOwnerRegistration', 'SecondOwner', 'SecondLicense') delete update
('ThirdOwnerCarRegistration', 'ThirdOwner', 'ThirdLicense') delete update
('newRegistration', 'NewOwner', 'NewLicense') delete update
('jrnqywhdrp', 'mcjda', 'cpdfe') delete update
('xucwifipbt', 'sckpu', 'uqrmr') delete update
('gockhnyluh', 'mffab', 'hdcbt') delete update
('asuvyhlcts', 'dtdgc', 'fqvps') delete update
('ehnaqmlief', 'ckrff', 'jvjpr') delete update
Back



('vmfcajdikl', 'howeamwybc', 'xujwgeppfe', 'SecondOwnerRegistration') delete update
('uvlffkairc', 'enybhmmvsl', 'eusaqmbdbx', 'xucwifipbt') delete update
('stpepounmt', 'kglqjrqbep', 'inuthhtfls', 'newRegistration') delete update
('slbpmgtclc', 'frtmcwpali', 'aglthhjlfl', 'asuvyhlcts') delete update
('gwioguoonb', 'jjtwuenpql', 'wapnnjhxpr', 'ThirdOwnerCarRegistration') delete update
Back



(337, 'vsmfsslukg', 7) delete update
(3821, 'qxdbwqhqge', 7) delete update
(9774, 'fxyihkynpa', 2) delete update
(2623, 'qrjaabtqrj', 9) delete update
(7871, 'leesywiivb', 7) delete update
Back



('wnknfkgrxydkclq', 'vmfcajdikl', 0.0, 'jujhyxkkqy', 37) delete update
('qoowkuvftxegjml', 'uvlffkairc', 1.7456786255017251, 'eaepfrvxaj', 50) delete update
('totxjmjgjhjywht', 'stpepounmt', 4.4259569375556, 'xfltcsxudf', 43) delete update
('dcufprgaxgrdhyg', 'slbpmgtclc', 3.3217128531598807, 'hpmnpwwxvt', 52) delete update
('pkltifsxfdexobj', 'gwioguoonb', 4.109804259071354, 'tftbyltiqv', 13) delete update
Back

```
def generateData(request):
    with connection.cursor() as cursor:
            cursor.execute(f'''Create or replace function random_string(length in-
teger) 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) val-
ues(random_string(5), trunc(random()*100)::int);
                insert into owner(owner_car_registration, owner_name, own-
er_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 own-
er_car_registration from(select owner_car_registration from owner where not ex-
ists(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 ex-
ists(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, en-
gine_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')
```

('vmfcajdikl', 'howeamwybc', 'xujwgeppfe', 'SecondOwnerRegistration', 'SecondOwnerRegistration', 'SecondOwner', 'SecondLicense')

('uvlffkairc', 'enybhmmvsl', 'eusaqmbdbx', 'xucwifipbt', 'xucwifipbt', 'sckpu', 'uqrmr')

('stpepounmt', 'kglqjrqbep', 'inuthhtfls', 'newRegistration', 'newRegistration', 'NewOwner', 'NewLicense')

('slbpmgtclc', 'frtmcwpali', 'aglthhjlfl', 'asuvyhlcts', 'asuvyhlcts', 'dtdgc', 'fqvps')

('gwioguoonb', 'jjtwuenpql', 'wapnnjhxpr', 'ThirdOwnerCarRegistration', 'ThirdOwnerCarRegistration', 'ThirdOwner', 'ThirdLicense')

Filter by: 3 Owner info submit

Back

Фільтр за кількістю років стажу

('ThirdLicense', 3, 'ThirdOwnerCarRegistration', 'ThirdOwner', 'ThirdLicense')

('NewLicense', 33, 'newRegistration', 'NewOwner', 'NewLicense')

('cpdfe', 99, 'jrnqywhdrp', 'mcjda', 'cpdfe')

('uqrmr', 28, 'xucwifipbt', 'sckpu', 'uqrmr')

('hdcbt', 17, 'gockhnyluh', 'mffab', 'hdcbt')

('fqvps', 71, 'asuvyhlcts', 'dtdgc', 'fqvps')

('jvjpr', 93, 'ehnaqmlief', 'ckrff', 'jvjpr')

Filter by: Owner info submit

Back

```
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', '') == ''):
```

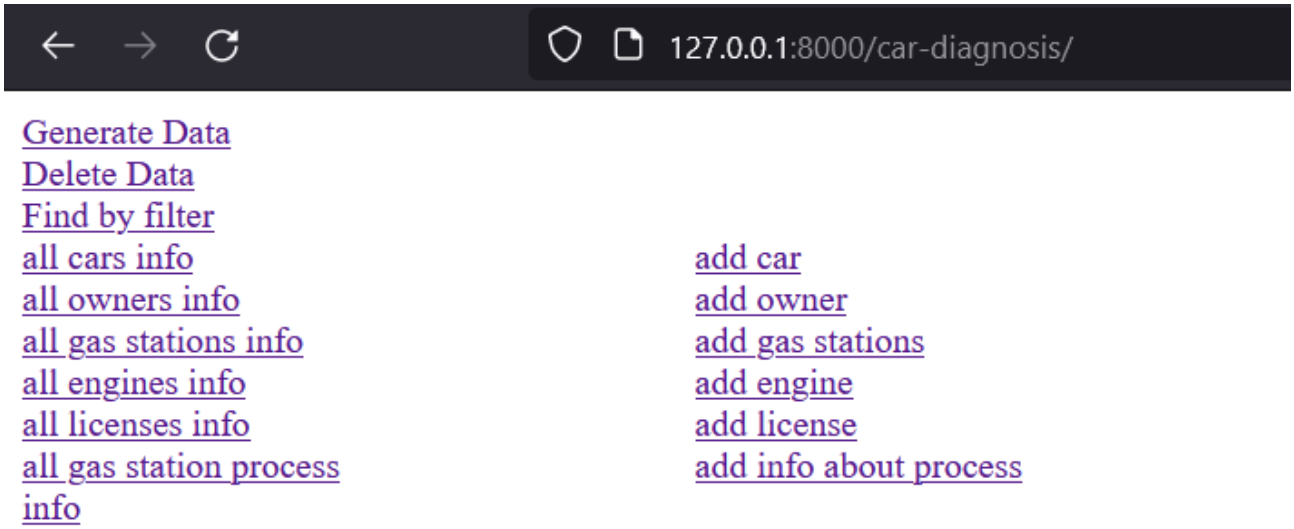
```
                    cursor.execute(f"select * from car inner join owner on own-
er.owner_car_registration=car.car_registration")
                else:
                    cursor.execute(f"select * from car inner join owner on own-
er.owner_car_registration=car.car_registration where own-
er_name='{request.POST.get('filter', '')}'")
                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 >= {re-
quest.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 own-
er.owner_car_registration=car.car_registration")
        listOptions = cursor.fetchall()

        return render(request, 'carDiagnosis/showData.html', {
            'listOptions': listOptions,
        })
```

__pycache__	04.12.2022 20:24	Папка с файлами	
migrations	16.11.2022 23:04	Папка с файлами	
templates	19.11.2022 15:50	Папка с файлами	
__init__.py	16.11.2022 22:23	Python Source File	0 КБ
admin.py	16.11.2022 22:23	Python Source File	1 КБ
apps.py	16.11.2022 22:23	Python Source File	1 КБ
models.py	04.12.2022 14:54	Python Source File	8 КБ
tests.py	16.11.2022 22:23	Python Source File	1 КБ
urls.py	04.12.2022 19:44	Python Source File	3 КБ
views.py	04.12.2022 20:24	Python Source File	17 КБ

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 відповідості 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.

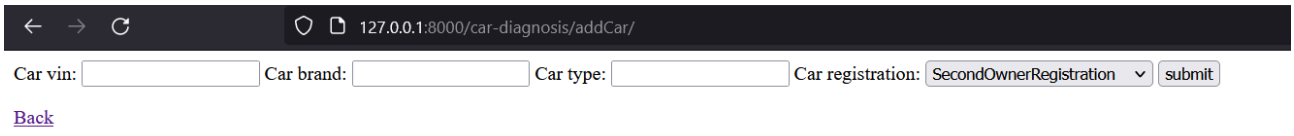


Головна сторінка для взаємодії

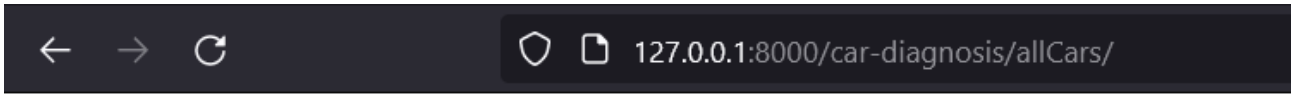
Generate Data – генерує рандомізовані дані.

Delete Data – видаляє данні з усіх таблиць.

Find by filter – базовий пошук по фільтру.



Шаблонна сторінка додавання



('vmfcajdikl', 'howeamwybc', 'xujwgeppfe', 'SecondOwnerRegistration') delete update
('uvlffkairc', 'enybhmmvsl', 'eusaqmbdbx', 'xucwifipbt') delete update
('stpepounmt', 'kglqjrqbep', 'inuthhtfls', 'newRegistration') delete update
('slbpmgtclc', 'frtmcwpali', 'aglthhjlfl', 'asuvyhlcts') delete update
('gwioguoonb', 'jjtwuenpql', 'wapnnjhxpr', 'ThirdOwnerCarRegistration') delete update
Back

Шаблонна сторіна виведення данних