

Міністерство освіти і науки України

Національний технічний університет України „КПІ

імені Ігоря Сікорського ”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**ЗВІТ**

лабораторної работи №4

курсу «Інфраструктура Програмного забезпечення WEB - застосувань »

Тема: «Створення і розгортання програмної інфраструктури на основі docker-compose»

Перевірив: Виконали студенти:

Орленко С. П. Гр. ІП-01

Смислов Даніл

Хернуф Валід

Пасальський Олександр

Київ 2023

**ЛАБОРАТОРНА РОБОТА № 4**

**Мета роботи**: полягає у дослідженні процесу автоматичного розгортання відносно складної програмної інфраструктури розподіленого веб-застосунку за обраним напрямом технології. Зважаючи на те, що сучасні РПС являють собою систему програмних модулів, що взаємодіють між собою і реалізовані на різних технологіях, їх автоматичне розгортання потребує додаткових програмних механізмів, що спрощують процес розгортання і розробки.

Одним з таких механізмів є docker-compose. У цій ЛР відбувається розгортання РПС з різних модулів відповідно до наданого завдання, вивчення синтаксисту файлів docker-compose і тестування отриманих результатів.

Лабораторну роботу можна умовно розподілити на три частини:

* вивчення і тестування складових частин відносно складної РПС, дозволяється використовувати власні напрацювання і досвід роботи за фахом студентів, що навчаються;
* підготовка первинного файлу docker-compose, вивчення елементів синтаксису і формування стилю цього файлу;
* розгортання РПС з використанням отриманого docker-compose, тестування роботи складної програмної системи, виправлення помилок, що були виявлені.

**Вхідні дані ЛР4**

У якості вхідних даних для ЛР4 є:

– 2 бази даних (SQL та NoSQL), які містяться у відповідних стандартних

контейнерах, які можуть бути активовані з docker-compose;

– два або три контейнера з базовим веб-застосунком, що побудовані на

основі типового веб-фреймворку і взаємодіють з базами даних.

**Вихідні дані ЛР4**

У якості вихідних даних для ЛР4 є: система каталогів з файлом docker-compose.yml, звіт.

**Завдання**

1. У якості індивідуального завдання, на першому етапі слід вивчити переваги і недоліки баз даних (є надані у переліку варіантів, але ж дозволяються і власні варіанти), за необхідності побудувати файли docker-compose для баз даних і протестувати роботу цього файлу на реальній системі, використовуючи доступні засоби тестування БД.

Вивчити веб-застосунок, що побудований на основі фреймворку з наданих варіантів модифікувати його для роботи з вашими БД, або одразу працювати з обраним самостійно варіантом. При необхідності провести його дослідження з docker-compose, аналогічно БД.

2. Підготувати файл docker-compose, що дозволяє побудувати узагальнену систему з парою контейнерів з БД і двома-трьома контейнерами з веб-застосунку (для front/back end), які взаємодіють відповідно до функціоналу з БД.

3. Провести розгортання РПС з використанням отриманого docker-compose. Продумати послідовність запуску контейнерів. Після розгортання провести тестування роботи складної програмної системи, що запущена з використанням результатів п.2. У випадку наявності, провести виправлення помилок. При неможливості поєднати БД і фреймворк веб-застосунку, провести аналіз помилок та обрати іншу БД або фреймворк.

**Стек технологій**, обирається студентами за власними вподобаннями.

Для SQL бази даних можна обрати postgres:14.1-alpine, mysql або іншу.

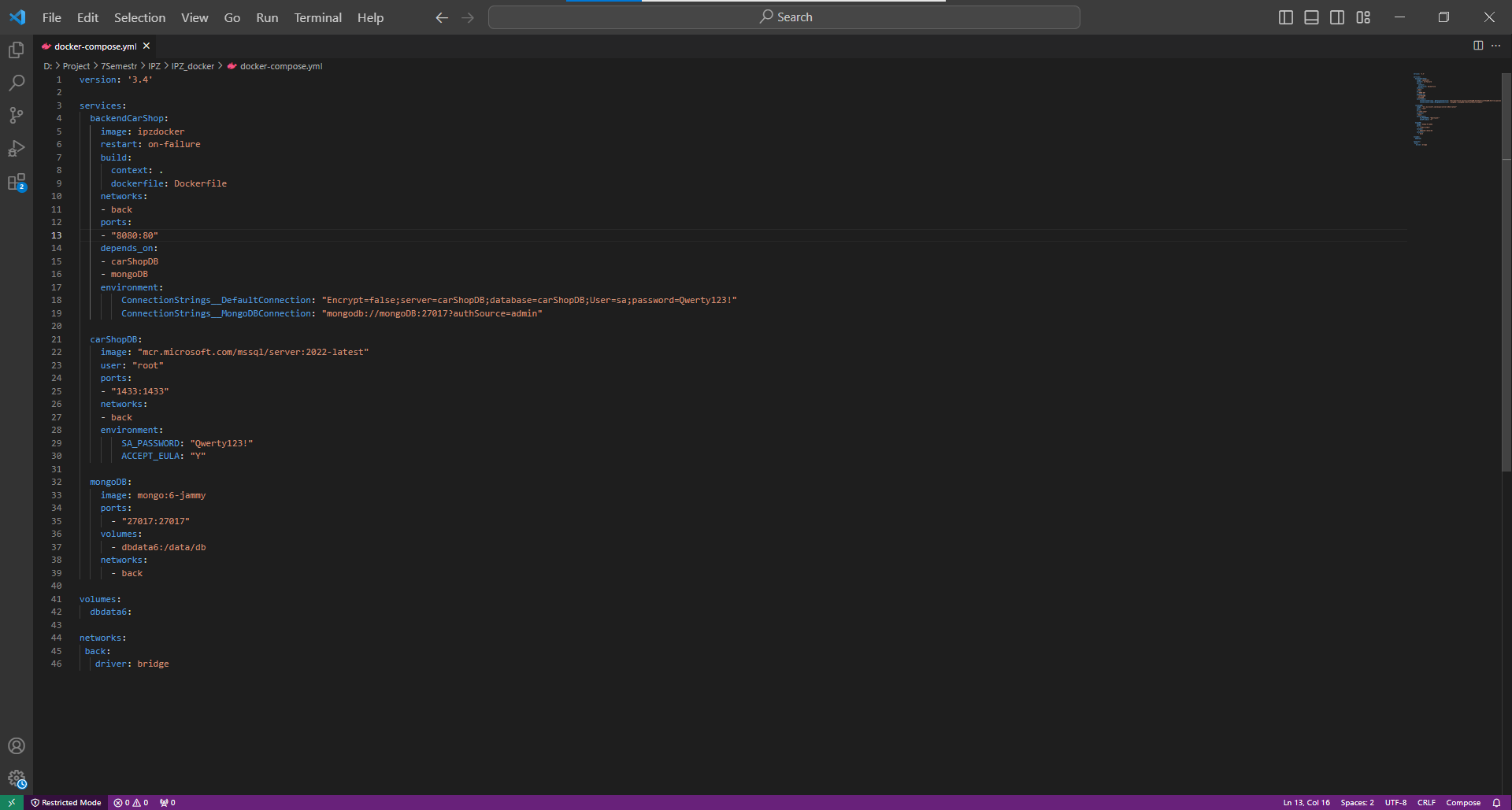
Для NoSQL БД гарно підійде MongoDB, але вибір теж не обмежений.

У якості фреймворку можна обрати Django або Spring, хоча можна створити застосунок і на свій смак

**Хід роботи**

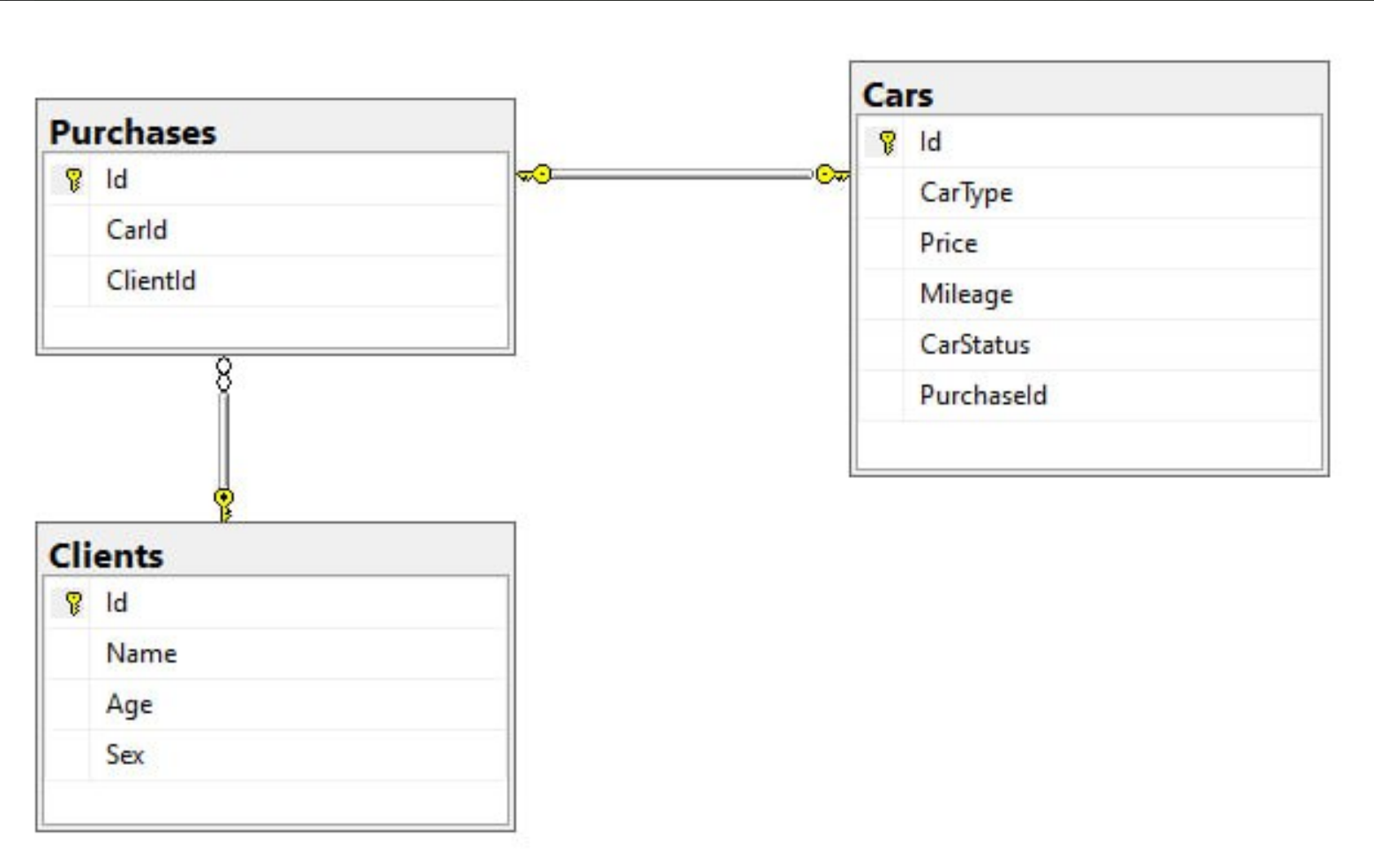
Docker-Compose піднімає три контейнери:

* backendCarShop – бекенд застосунку
* carShopDB – SQL база даних
* mongoDB – NoSQL база даних

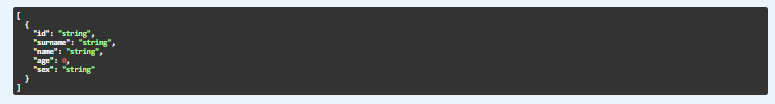


В SQL базі даних зберігається 3 типи сутностей Cars(Машини), Clients(Клієнти), Purchases(Покупці). В NoSQL зберігається 1 тип сутностей Sellers (Продавці).

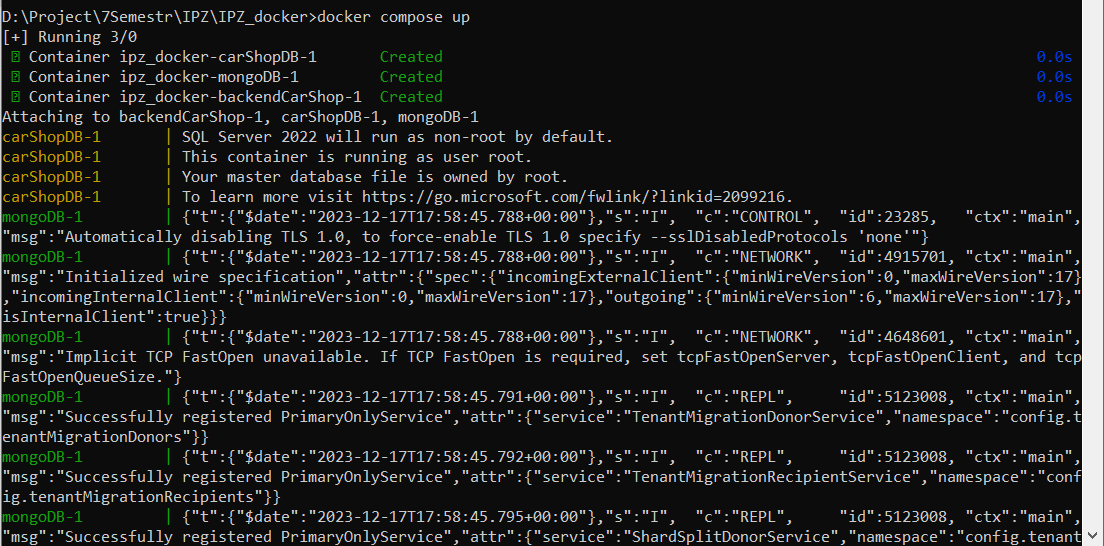
Схема SQL бази даних.



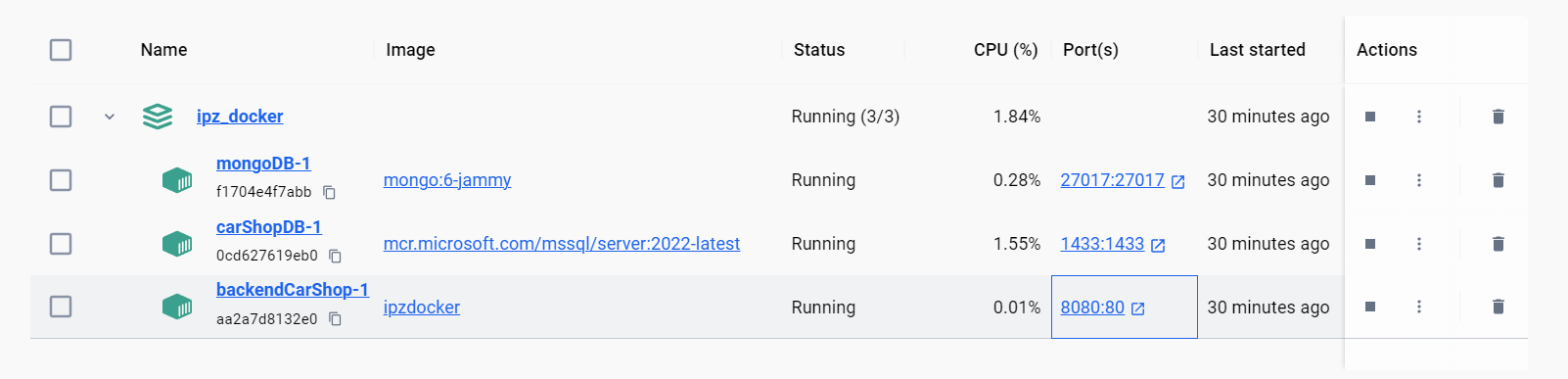
Структура типів в NoSQL базі даних.



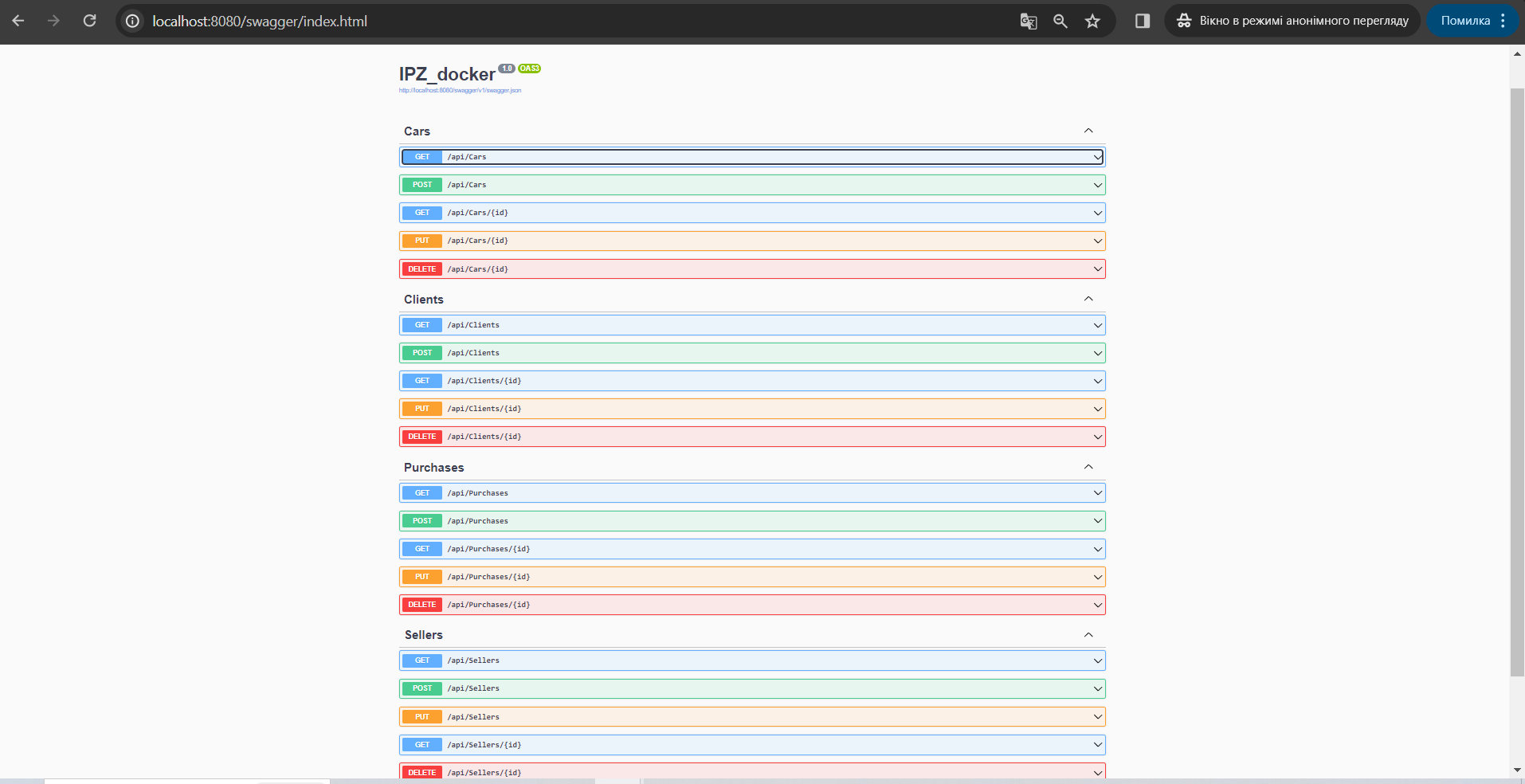
Запуск Docker-Compose.



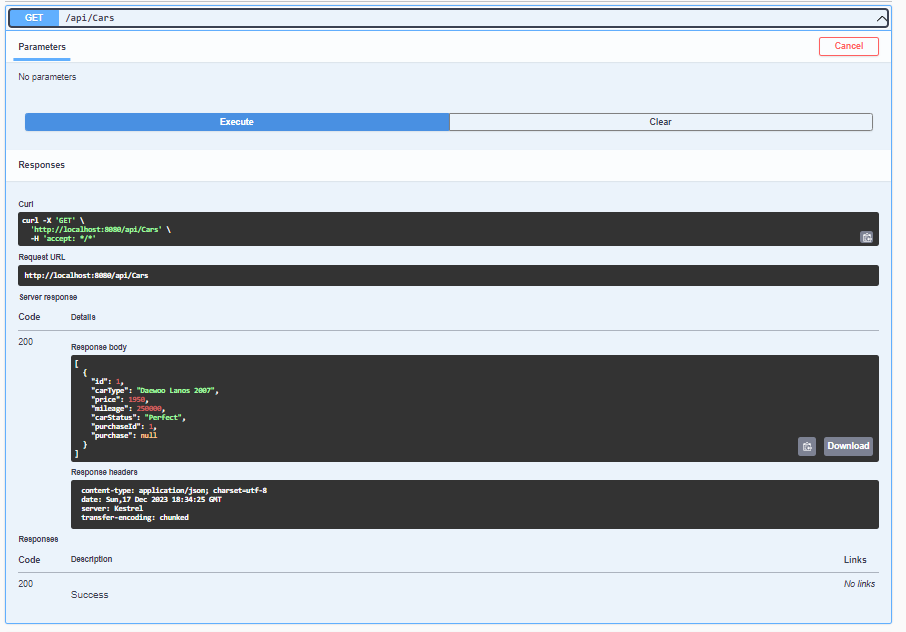
Підняті контейнери.



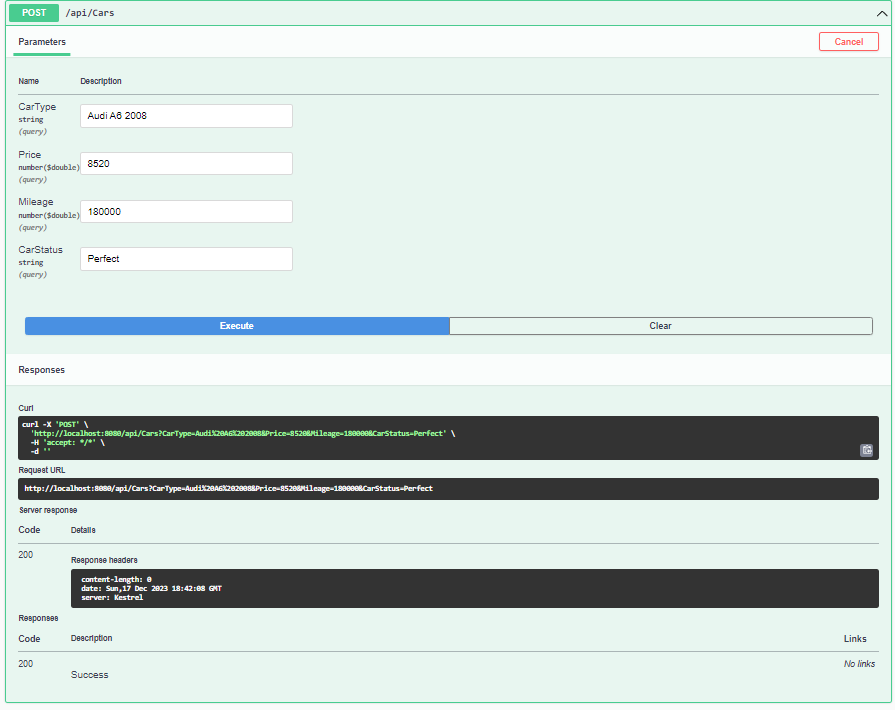
До всіх цих 4 сутностей можна застосовувати стандартні запити: Get, Post, Get by id, Put by id, Delete by id.



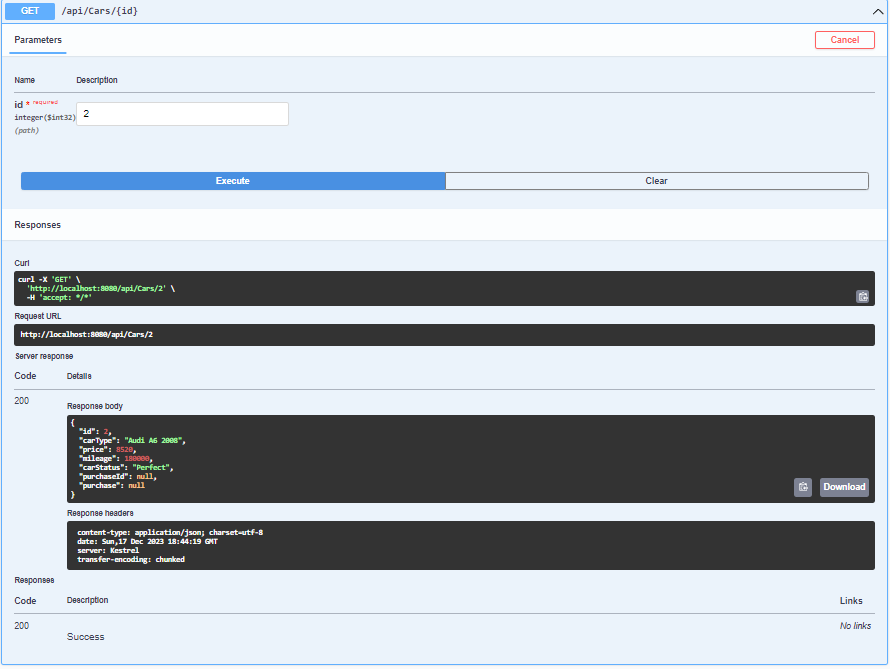
Запит Get для таблиці Cars повертає список всіх Car в SQL базі даних.



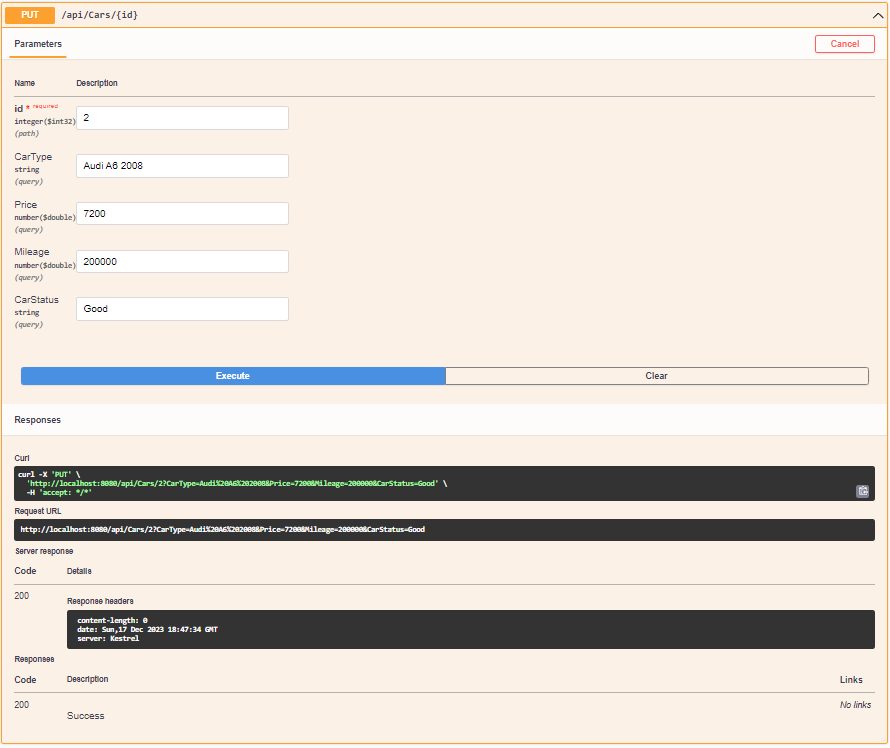
Запит Post для таблиці Cars створює нову сутність Car в SQL базі даних, з переданими даними в тілі запита.



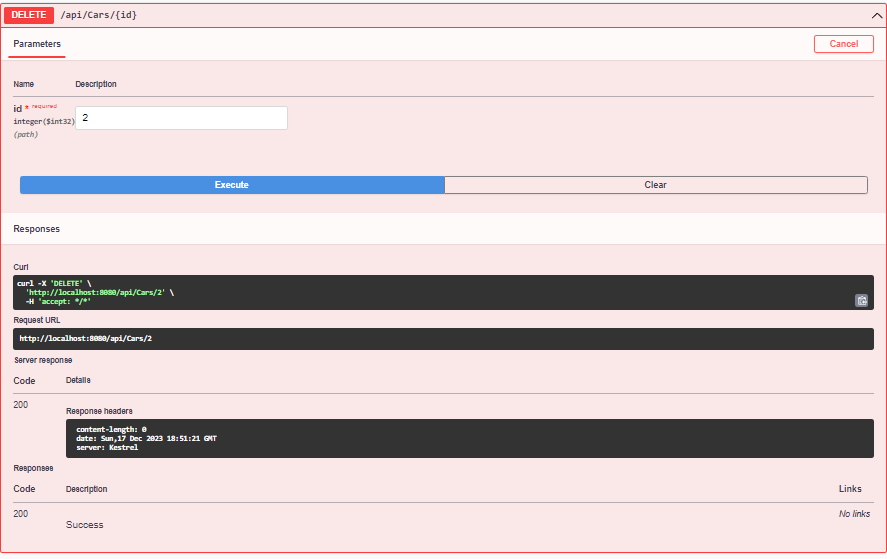
Запит Get by id для таблиці Cars повертає сутність Car з переданим id в запиті, що зберігається в SQL базі даних.



Запит Put by id для таблиці Cars змінює існуючу в SQL базі даних сутність Car з переданим id в запиті.

****

Запит Delete by id для таблиці Cars видаляє існуючу в SQL базі даних сутність Car з переданим id в запиті.

****

**Висновок:**

У ході виконання лабораторної роботи було вивчено та досліджено процес автоматичного розгортання відносно складної програмної інфраструктури розподіленого веб-застосунку за допомогою технології docker-compose.

Лабораторна робота була розділена на три основні частини. Перша частина включала вивчення і тестування складових частин розподіленої програмної системи. Друга частина передбачала підготовку файлу docker-compose, вивчення синтаксису та формування структури цього файлу. Нарешті, третя частина включала розгортання розподіленої системи з використанням отриманого файлу docker-compose, тестування роботи системи та виправлення виявлених помилок.

У якості вхідних даних були надані дві бази даних (SQL та NoSQL) та один контейнер з базовим веб-застосунком, побудованим на основі ASP.NET. Основним завданням було підготувати файл docker-compose, щоб забезпечити побудову узагальненої системи з контейнерів баз даних та веб-застосунка, який взаємодіє із базами даних.

В результаті виконаної роботи був розроблений та опрацьований файл docker-compose.yml, який успішно використовувався для розгортання програмної інфраструктури.