Звіт з лабораторної роботи №3

З дисципліни **Технології на платформі .NET**

Виконав студент групи 301-ТК Антропов В. О.

Мета роботи: Ознайомитися з принципами побудови REST API. Навчитися створювати Web API з використанням ASP.NET Core, реалізовувати маршрутизацію, використовувати Entity Framework для доступу до бази даних та виконання CRUD-операцій.

Хід роботи:

Створено два проєкти:
- `Lab3BusDatabase.Infrastructure` — для моделей, контексту БД, репозиторіїв
- `Lab3BusDatabase.App` — для запуску CRUD логіки

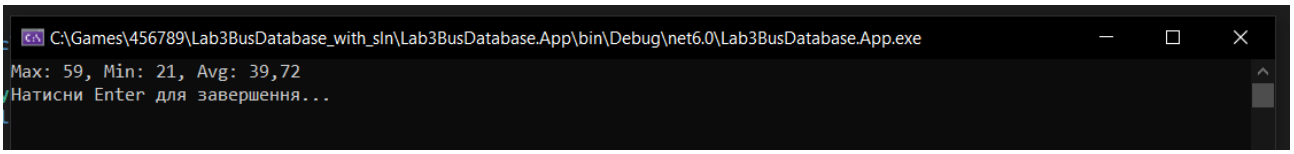
Використано SQLite як СУБД

Налаштовано зв'язок: **Driver 1:N Bus**

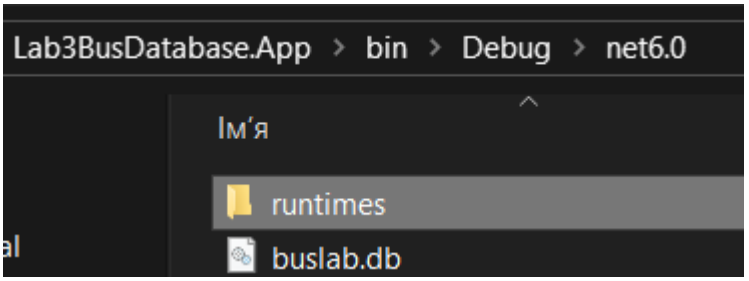
Створено 100 Bus-об'єктів для одного водія
Збереження здійснюється через EF Core

Обчислено `Max`, `Min`, `Average` місткість автобусів через LINQ

Виведено результати у консоль



Папка з файлом `buslab.db`



Висновок

Ознайомився з основами ORM.

Застосував репозиторій для роботи з БД.

Навчився створювати зв’язки між сутностями та зберігати дані через EF Core.

Підкріпив знання LINQ, async/await і структури проєкту в .NET.

Контрольні запитання

1. Що таке інтерфейс `ICrudServiceAsync<T>`?

➡ Generic-інтерфейс з асинхронними методами CRUD (створення, читання, оновлення, видалення).

2. Яке його призначення?

➡ Уніфікує логіку CRUD для будь-якого типу T.

3. Чому доцільно використовувати generic тип T?

➡ Щоб один сервіс працював із різними типами (Bus, Driver тощо) без дублювання коду.

4. Що таке процес та потік?

➡ Процес — окрема програма, потік — одиниця виконання в процесі.

5. Різниця між асинхронністю та багатопотоковістю?

➡ Асинхронність — не блокує потік, багатопотоковість — виконує паралельно.

6. Для чого async/await?

➡ Щоб не блокувати потік і писати зрозумілий асинхронний код.

7. Чим відрізняється `Task` від `ValueTask`?

➡ `Task` завжди створює об’єкт, `ValueTask` ефективніший при швидких результатах.

8. Що таке thread-safe колекція?

➡ Колекція, яку можна використовувати з кількох потоків без помилок (наприклад, `ConcurrentDictionary`).

9. Для чого lock, Semaphore, AutoResetEvent?

➡ Для синхронізації потоків і захисту спільних ресурсів.

10. Як забезпечити безпеку при одночасному зверненні до ресурсу?

➡ Через `lock`, `Monitor`, `Semaphore`, або thread-safe колекції.

11. Як з допомогою LINQ отримати min, max, avg?

➡ Використовуються методи `.Min()`, `.Max()`, `.Average()`.

12. Різниця між Select, Where, Aggregate, OrderBy?

➡

- `Select` — витягує поля
- `Where` — фільтрує
- `Aggregate` — обчислення з накопиченням
- `OrderBy` — сортування

13. Переваги LINQ перед циклами?

➡ Коротше, зрозуміліше, менше помилок.

14. Що буде, якщо два потоки одночасно збережуть у файл?

➡ Може виникнути помилка доступу або пошкодження файлу. Потрібна синхронізація.

15. Як працює `Parallel.For` і коли його використовувати?

➡ Запускає ітерації паралельно. Використовується для обчислень, генерації даних тощо.

16. Що таке пагінація і як вона реалізована?

➡ Виведення частинами (сторінками):