**Звіт до лабораторної роботи №7**

**Тема**: Створення WPF-клієнта для WCF-сервісу.

**Мета**: Навчитися зв’язувати елементи WPF з WCF-сервісом.

**Хід виконання:**

**Етап 1:**

Створення сервіс-проекту створили новий ASP.NET проект під назвою AdventureWorksService, використовуючи шаблон Empty. Цей етап пройшов без особливостей.

**Етап 2:**

Створення FDM-моделі Замість створення традиційної EDM моделі, було вирішено використати Code First підхід з Entity Framework для більшої гнучкості та контролю над процесом розробки. Були створені класи сутностей відповідно до предметної області "Резервування білетів", а контекст бази даних був налаштований на роботу з локальною базою SQL Server.

Себто там треба провести міграцію з контексту і все буде ок, той самий результат + -

**Етап 3:**

Реалізація WCF-сервісу Замість WCF Data Service було рішено створити ASP.NET Core Web API сервіс, що пропонує більше можливостей та простіше інтегрується з сучасними веб-додатками та сервісами.

**Етап 4:**

Створення WPF-клієнта WPF-клієнт під назвою AdventureWorksClient був створений для взаємодії з розробленим Web API. Усі необхідні елементи користувацького інтерфейсу були реалізовані та зв'язані з джерелами даних через патерн MVVM для ефективної роботи з даними.

**Проблеми та їх вирішення:** Під час роботи стикнувся з наступними проблемами:

1. Складності з налаштуванням зв’язку між WPF-клієнтом і Web API. Вирішено використанням HttpClient та асинхронного програмування. Через це кількість необхідного коду збільшилась в рази, але особисто цей метод мені більше підходить чим качати різні версії .NET і експериментувати для якої версії був гайд в лабораторній.
2. Проблеми з серіалізацією даних. Вирішено застосуванням правильного форматування JSON і парсення даних через простенькі DTO щоб контроллери їх правильно обробляли.
3. Ускладнення з динамічним оновленням списків у ComboBox. Вирішено імплементацією події DropDownOpened. Підвантажую ІД елементів кожного разу як відкривається дропбокс список.
4. Проблеми з обробкою винятків при роботі з API. В ході розробки стикнувся з викликами винятків, пов'язаними з відсутністю даних або помилками доступу до сервера. Для вирішення цієї проблеми були розроблені додаткові блоки try-catch, які дозволили коректно обробляти помилки і надавати зрозумілі повідомлення користувачам.

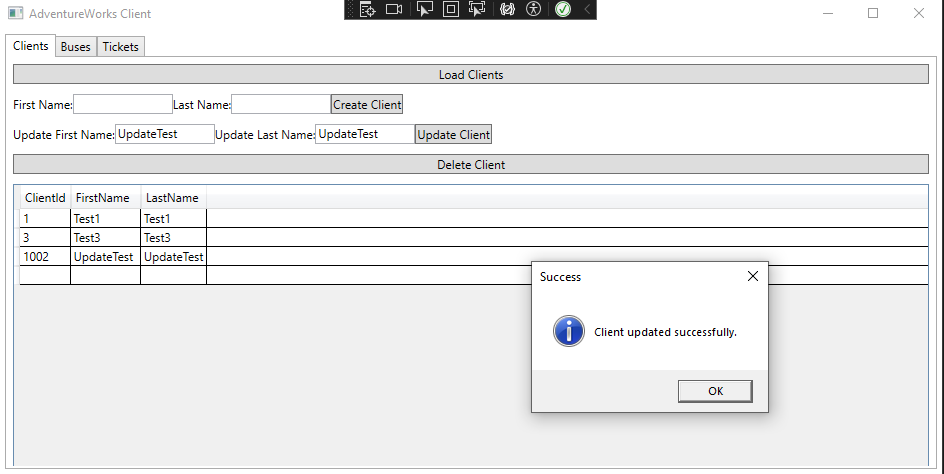
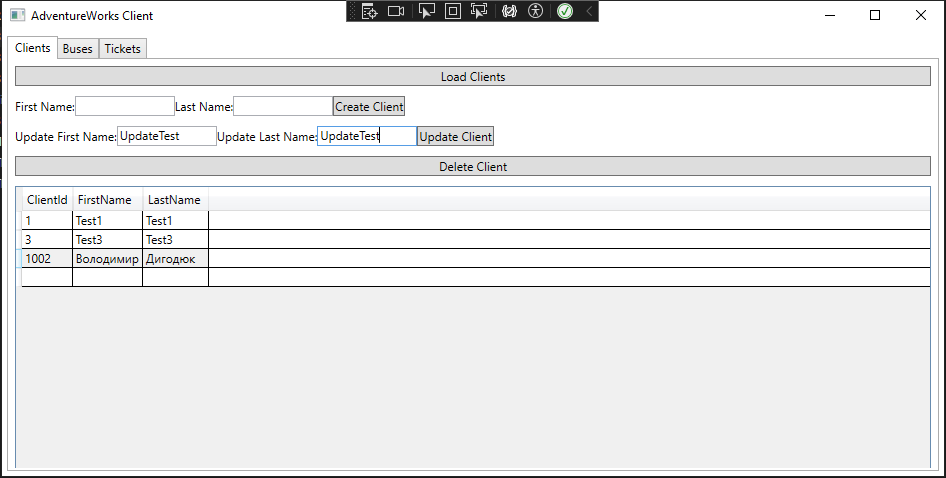
Фото звіт роботи лабораторної:  
  


Рисунок 1. Робота апдейту

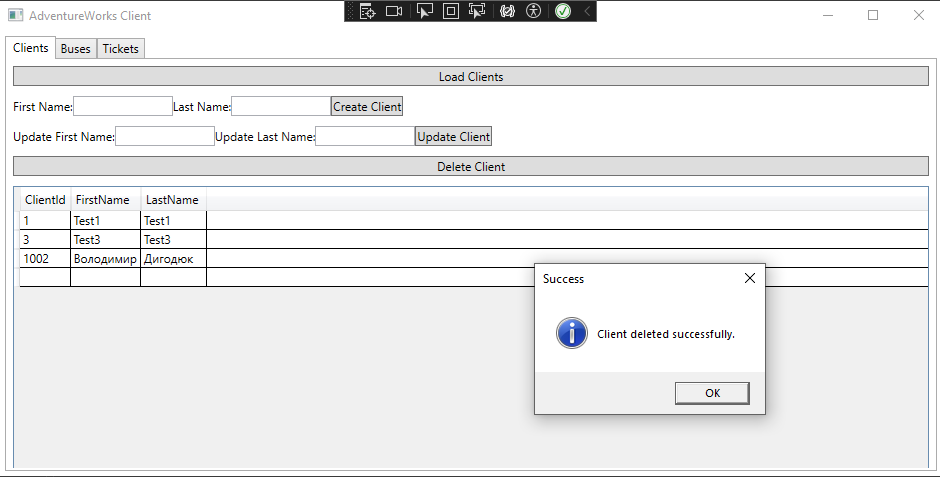
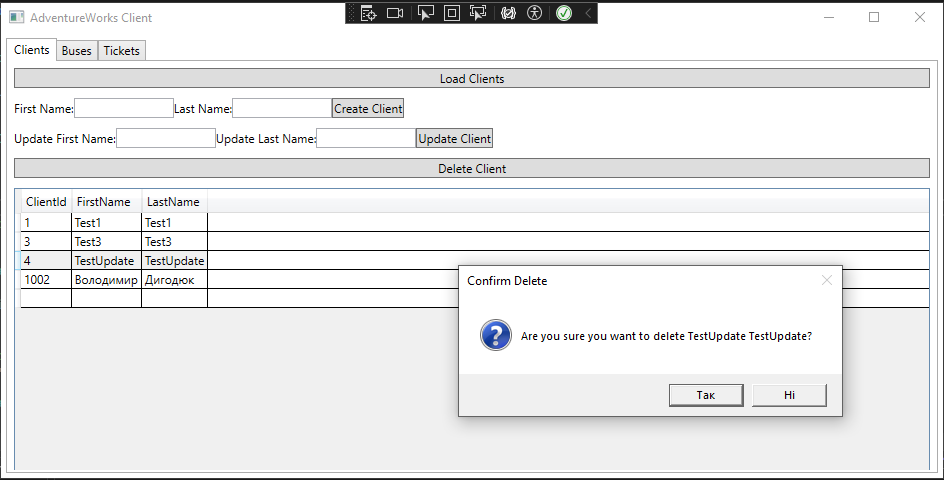
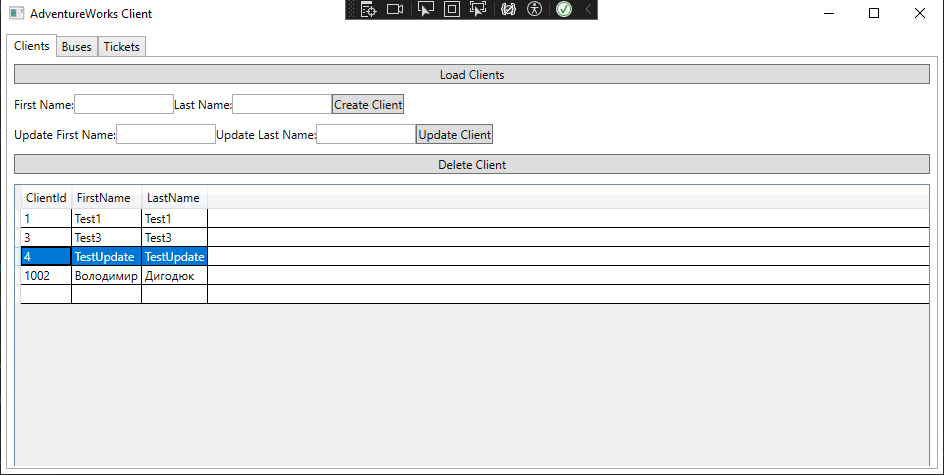


Рисунок 2. Робота видалення

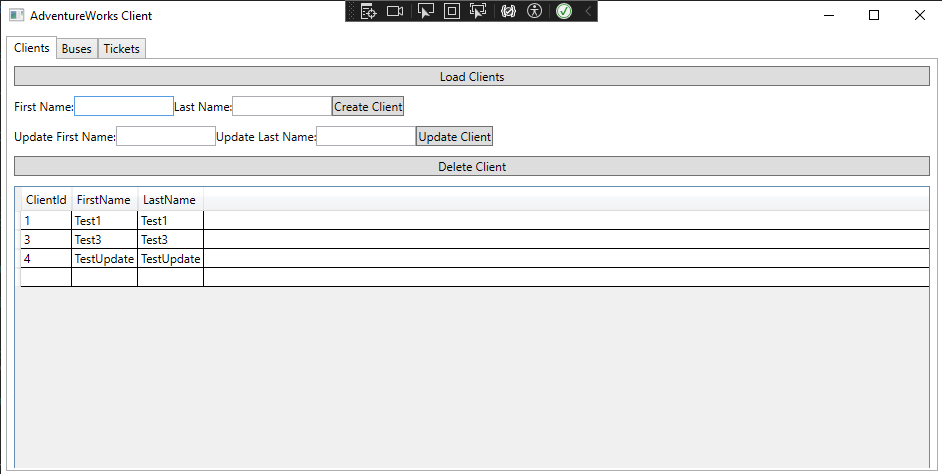


Рисунок 3. Відображення інформації з БД

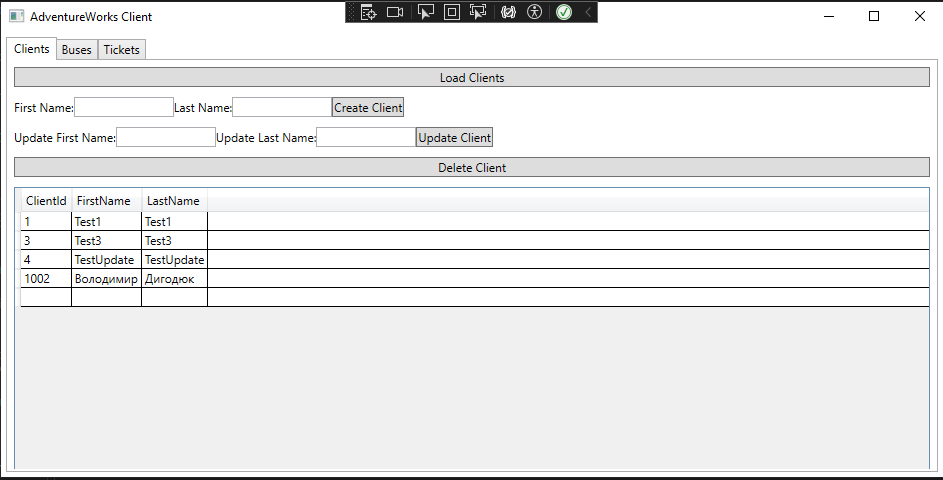
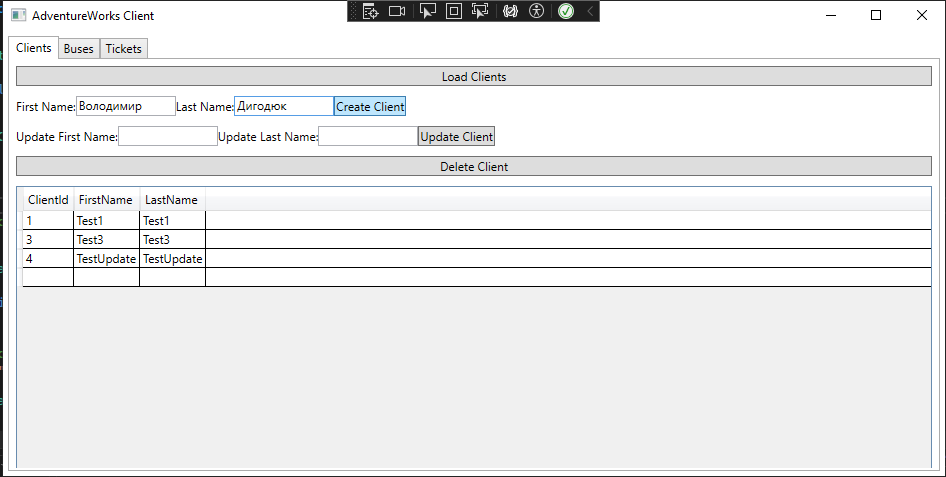


Рисунок 4. Робота додавання в таблицю

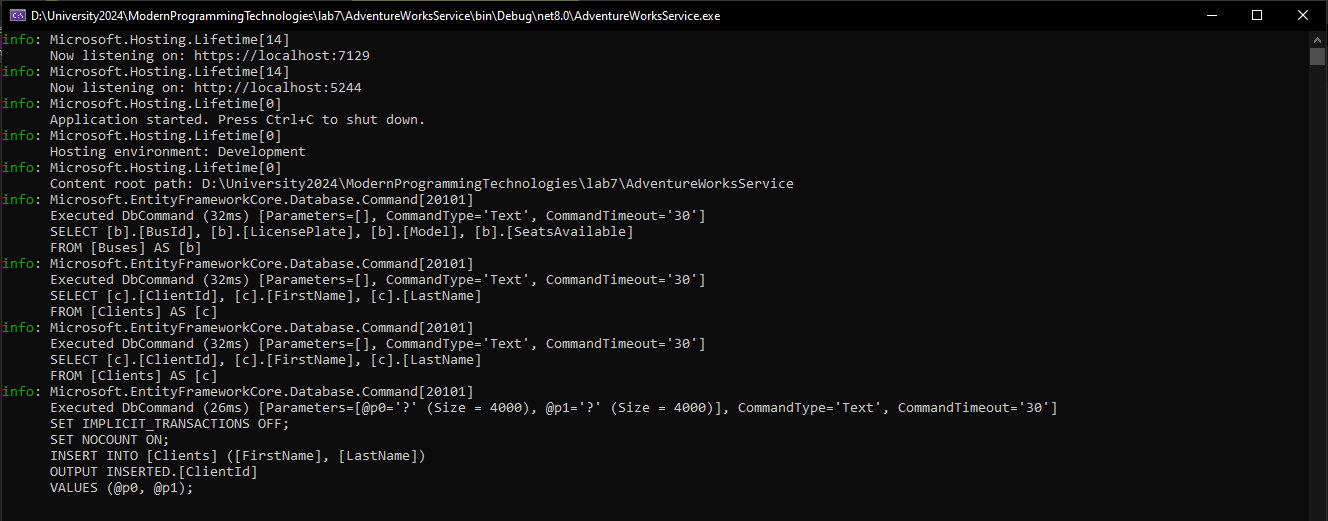


Рисунок 5. Логи команд до БД в консоль

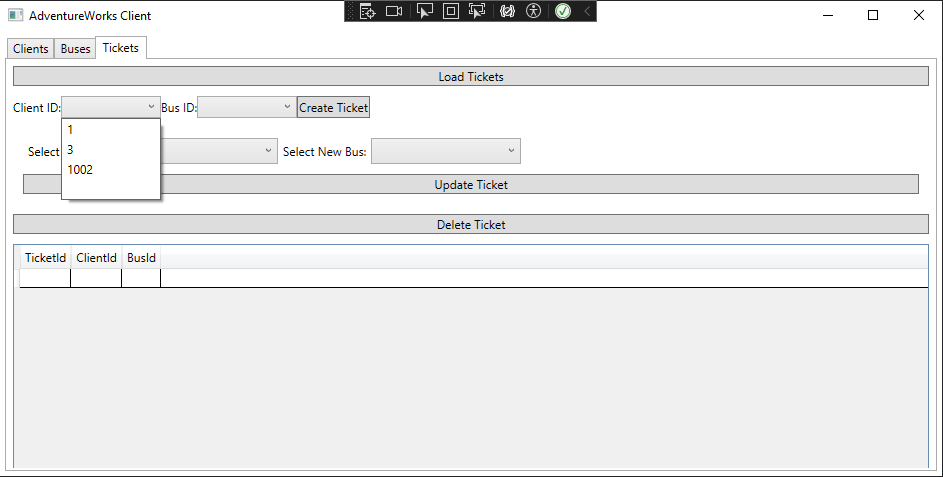
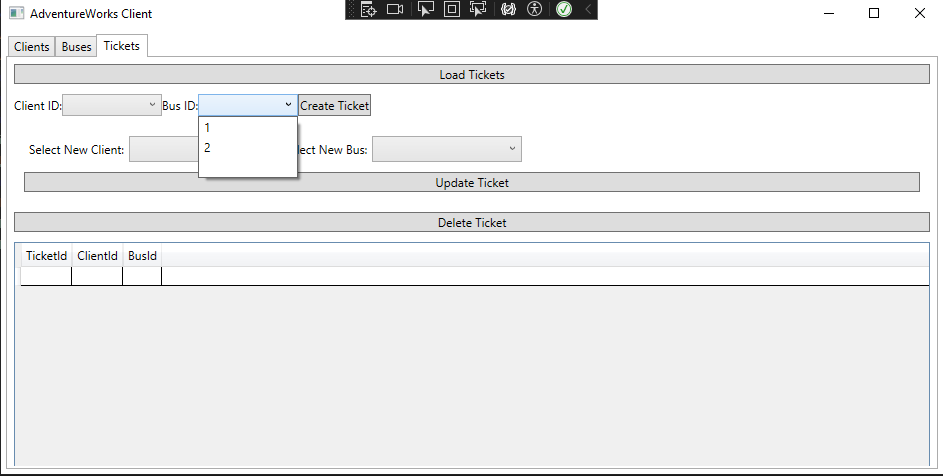


Рисунок 6. Дропбокси в меню Тікетів

В додатку пророблена вся логіка стосовно додавання і видалення пов’язаних сутностей, до прикладу програма не дасть видалити юзера, якщо в нього є заброньований білет, хоча видалення можливе якщо видалити спершу білет. Справедливо і у випадку з Автобусами. Також пророблена логіка з не можливістю додати більше білетів на рейс, чим параметр “кількість місць” у відповідного автобуса. Додаток поверне специфічну помилку при спробі.

Контрольні питання:

1. **Що таке EDM-модель?** EDM (Entity Data Model) - це концептуальна модель, яка описує структуру даних для додатків, які будуються за допомогою ADO.NET Entity Framework. Вона використовується для представлення об'єктів та їх відносин на високому рівні абстракції. В рамках даної лабораторної роботи, замість створення EDM за допомогою візуального дизайнера, було використано підхід Code First для створення моделей сутностей та їх контексту в коді, що дозволило мати більший контроль над моделлю даних та її поведінкою.
2. **Як відбувається поєднання Data Source і користувацького інтерфейсу WPF?** У WPF використовується патерн MVVM (Model-View-ViewModel) для поєднання джерела даних (Data Source) з користувацьким інтерфейсом. ViewModel виступає як посередник між View (інтерфейс користувача) і Model (джерело даних), де вона містить логіку презентації та обробляє взаємодію з даними. У цій лабораторній роботі, дані з сервісу були зв'язані з елементами інтерфейсу через ObservableCollection, який автоматично оновлює інтерфейс при зміні даних.
3. **Що таке Data Service?** Data Service – це сервіс, який надає доступ до даних через веб, використовуючи стандарти, такі як HTTP та JSON або XML для обміну даними. У цій лабораторній роботі було рішено використовувати Web API як Data Service, що дозволило взаємодіяти з клієнтом через стандартний HTTP протокол, використовуючи сучасні підходи RESTful API для CRUD операцій над даними.