МІНІСТЕРСТВО ОСВІТИ И НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І. МЕЧНИКОВА

Факультет математики, фізики та інформаційних технологій

Кафедра комп'ютерних систем та технологій

**КУРСОВА РОБОТА**

з дисципліни

**«Організація баз даних»**

на тему:

« Торгівельна організація. »

Виконав:

студент ІІ курсу

групи \_\_\_\_\_2\_\_\_\_\_\_

спеціальності «Комп’ютерні науки»

Юзбеков Віктор Сергійович

Керівник:

к.т.н., доц., Камєнєва А.В.

Захищено « »\_ 2025 р. з оцінкою \_ Комісія:

\_

(ПIБ) (Підпис)

\_

(ПIБ) (Підпис)

\_

(ПIБ) (Підпис)

**Одеса – 2025**

**АНОТАЦІЯ**

Дана курсова робота присвячена комплексному дослідженню та практичній реалізації сучасного користувацького інтерфейсу для ефективного управління базою даних торгівельної організації. У роботі детально розглянуто весь процес створення багатофункціонального додатку, від початкового аналізу вимог до фінального тестування готового продукту.

Технологічну основу проекту складає потужна платформа .NET 8.0 в поєднанні з WPF (Windows Presentation Foundation), що забезпечує створення інтуїтивного та адаптивного користувацького інтерфейсу з сучасним дизайном. Для організації взаємодії з базою даних застосовано передову ORM-систему Entity Framework Core, яка гарантує надійне та безпечне управління даними, а також спрощує процес розробки завдяки об'єктно-орієнтованому підходу до роботи з реляційними базами даних.

Розроблений додаток реалізує повний спектр CRUD-операцій (Create, Read, Update, Delete), включаючи інтуїтивний перегляд інформації, зручне додавання нових записів через перевірені форми, гнучке редагування існуючих даних з контролем цілісності, безпечне видалення з підтвердженням операцій, а також потужну систему пошуку з підтримкою різних критеріїв та швидкої фільтрації великих обсягів даних.

Особливу увагу приділено архітектурним рішенням, що забезпечують масштабованість додатку, його подальший розвиток та легкість супроводу. Результатом роботи є повнофункціональний додаток, який демонструє сучасні підходи до розробки бізнес-додатків та може служити основою для створення комерційних систем управління торгівельними підприємствами.

**ANNOTATION**

This coursework is dedicated to comprehensive research and practical implementation of a modern user interface for efficient database management of a trading organization. The work provides a detailed examination of the entire process of creating a multifunctional application, from initial requirements analysis to final testing of the completed product.

The technological foundation of the project consists of the powerful .NET 8.0 platform combined with WPF (Windows Presentation Foundation), which ensures the creation of an intuitive and responsive user interface with modern design. For organizing database interaction, the advanced ORM system Entity Framework Core is employed, which guarantees reliable and secure data management while simplifying the development process through an object-oriented approach to working with relational databases.

The developed application implements a complete spectrum of CRUD operations (Create, Read, Update, Delete), including intuitive information viewing with sorting and filtering capabilities, convenient addition of new records through validated forms, flexible editing of existing data with integrity control, secure deletion with operation confirmation, as well as a powerful search system supporting various criteria and rapid filtering of large data volumes.

Special attention is given to architectural solutions that ensure application scalability, its future development, and ease of maintenance. The result of the work is a fully functional application that demonstrates modern approaches to business application development and can serve as a foundation for creating commercial management systems for trading enterprises.

**ЗМІСТ**

[**ВСТУП** 5](#_Toc200631616)

[**РОЗДІЛ 1** 7](#_Toc200631617)

[**ПРОЕКТУВАННЯ СТРУКТУРИ ПРОЕКТУ** 7](#_Toc200631618)

[**1.1 Специфікація технічного завдання** 7](#_Toc200631619)

[**1.2 Технічні специфікації та обмеження** 9](#_Toc200631620)

[**РОЗДІЛ 2** 10](#_Toc200631621)

[**ПРОГРАМНА МОДЕЛЬ ЗАСТОСУНКУ** 10](#_Toc200631622)

[**2.1 Архітектурна концепція програмного продукту** 10](#_Toc200631623)

[**2.2 Концептуальне моделювання інформаційної структури** 12](#_Toc200631624)

[**2.3 Створення бази даних** 13](#_Toc200631625)

[**2.4 Запити до бази даних** 17](#_Toc200631626)

# **ВСТУП**

В епоху стрімкого розвитку інформаційних технологій та повсюдної цифрової трансформації бізнес-процесів, комп'ютеризація операційної діяльності торгових підприємств стає невід'ємною складовою їх конкурентоспроможності. Програмні рішення для управління корпоративними даними дозволяють систематизувати інформацію про асортимент, замовлення, партнерів, споживачів та інші критично важливі бізнес-об'єкти.

Створення ергономічних та функціональних інтерфейсів користувача для роботи з корпоративними базами даних набуває стратегічного значення. Якісно спроектований інтерфейс забезпечує не тільки оптимізацію робочих процесів, але й мінімізує ризики операційних помилок, пов'язаних з людським фактором.

Метою даної курсової роботи є розробка програмного Desktop-продукту для управління інформаційними ресурсами торгівельного підприємства. Технічну основу проекту становить архітектурний патерн MVVM (Model-View-ViewModel), який вважається оптимальним рішенням для додатків, створених на базі WPF (Windows Presentation Foundation).

Технологічний стек проекту включає:

* Платформу .NET 8.0 як сучасне середовище для кросплатформенної розробки;
* Фреймворк WPF для створення візуального інтерфейсу користувача;
* ORM-систему Entity Framework Core для організації взаємодії з реляційними базами даних;
* СКБД SQL Server як основу для реалізації структури даних.

Ключовий функціонал, що був впроваджений у рамках проекту:

* представлення інформації з бази даних у вигляді структурованих таблиць;
* створення нових записів через спеціалізовані діалогові форми;
* модифікація існуючих даних у відібраних записах;
* операції видалення інформаційних елементів;
* система фільтрації та пошуку за визначеними критеріями (зокрема, за найменуванням продукції або категоріальною приналежністю);
* синхронізація змін з базою даних через ORM-механізми.

Отже, дана курсова робота спрямована не лише на практичну реалізацію специфічного функціоналу для торгівельних систем, але й на демонстрацію компетенцій у сфері сучасної архітектури програмного забезпечення, застосування проектних шаблонів та ефективних методів роботи з інформацією.

Результатом виконання роботи стане комплексний програмний продукт, що ілюструє практичне застосування концепцій об'єктно-орієнтованого програмування, технологій роботи з реляційними СКБД та методів розробки графічних інтерфейсів у екосистемі .NET.

# **РОЗДІЛ 1**

# **ПРОЕКТУВАННЯ СТРУКТУРИ ПРОЕКТУ**

Предметною сферою для створення програмного рішення в межах даної курсової роботи виступає торгівельна організація, операційна діяльність якого охоплює управління товарними позиціями, їх систематизацією за групами, координацією з бізнес-партнерами та обробкою клієнтських запитів. Пріоритетним завданням є побудова ергономічного та багатофункціонального графічного середовища, що забезпечує інтерактивну роботу користувача з інформаційними ресурсами в онлайн-режимі.

## **1.1 Специфікація технічного завдання**

Потрібно створити Desktop-ну програмну систему з наступними характеристиками:

* Візуалізація корпоративних даних**.** Система повинна включати основний робочий модуль, де структурована інформація з реляційних таблиць представлена у форматі сіток даних. Завантаження контенту має відбуватися автоматично при ініціалізації програми.
* Функціонал створення записів. Має бути реалізована можливість генерації нових елементів продукції, товарних груп, бізнес-партнерів та інших об'єктів предметної області. Інтерфейси введення повинні включати перевірочні механізми.
* Модифікація існуючої інформації. Користувач повинен мати можливість оновлювати характеристики будь-якого об'єкта через спеціалізовані форми коригування. При внесенні змін необхідно підтримувати реферативну цілісність між взаємопов'язаними сутностями (наприклад, перепризначення групи для товарної позиції).
* Операції видалення об'єктів. Система має підтримувати можливість елімінації обраних записів з обов'язковим діалогом підтвердження для запобігання ненавмисній втраті інформації. Обробка пов'язаних даних повинна здійснюватися через блокування операції або каскадне видалення (згідно з логікою СКБД).
* Система пошуку та селекції. Програмний продукт повинен надавати користувачам інструменти для текстового пошуку за ключовими атрибутами (назвами продукції) та застосування критеріїв відбору (за товарними групами, постачальниками або ціновими діапазонами).
* Відмово-стійкість системи. Поведінка додатку при обробці некоректних або неповних вхідних даних має бути контрольованою та прогнозованою. Обов'язковою є обробка винятків, що виникають при взаємодії зі СКБД, та надання інформативних повідомлень користувачеві.
* Ергономіка користувацького досвіду. Інтерфейс має характеризуватися інтуїтивністю, логічною організацією та чітким функціональним розподілом (перегляд, модифікація, пошук тощо). Критично важливо дотримуватися UX-принципів: простоти, зрозумілості та раціонального розміщення контрольних елементів.

## **1.2 Технічні специфікації та обмеження**

Серед обмежень хотілось виділите даний список:

* Операційне середовище: Windows 10+ з підтримкою .NET 8.0
* Мова локалізації: українська мова інтерфейсу
* Стабільність роботи: гарантування безперебійного функціонування при пролонгованому використанні
* Архітектурна гнучкість: можливість розширення функціональності без кардинальних змін в існуючій структурі

Також технічні специфікації даного застосунку:

Таблиця 1.1 – Технічні специфікації

|  |  |  |
| --- | --- | --- |
| № | Компонент | Характеристика |
| 1 | Презентація даних | Табличне представлення товарів з навігаційними можливостями |
| 2 | Генерація записів | Створення нових продуктів, категорій, постачальників |
| 3 | Коригування | Можливість модифікації параметрів наявних об'єктів |
| 4 | Елімінація | Видалення записів з користувацьким підтвердженням |
| 5 | Пошукова система | Пошук товарів за назвою або альтернативними параметрами |
| 6 | Обробка винятків | Управління помилками доступу до СКБД |
| 7 | ORM-інтеграція | Взаємодія з базою даних через Entity Framework Core |
| 8 | MVVM-архітектура | Розмежування презентаційної логіки, даних і обробки подій |

# **РОЗДІЛ 2**

# **ПРОГРАМНА МОДЕЛЬ ЗАСТОСУНКУ**

Застосунок, розроблений у межах цієї курсової роботи, реалізований як настільна програма з графічним інтерфейсом, побудована на основі технологій WPF (.NET 8.0) із використанням архітектурного шаблону MVVM (Model–View–ViewModel). Такий підхід забезпечує чітке відокремлення шарів відповідальності в системі: логіки обробки даних, користувацького інтерфейсу та зберігання інформації.

## **2.1 Архітектурна концепція програмного продукту**

Компонент Model відповідає за об'єктне представлення предметної області та охоплює:

* Сутності, що корелюють з таблицями реляційної бази даних
* Навігаційні атрибути для відображення між-сутнісних взаємозв'язків
* DTO-структури для передачі обмежених наборів даних між компонентами системи.

Усі моделі представлені у вигляді C#-класів та інтегровані з ORM Entity Framework Core, що забезпечує конвертацію LINQ-запитів у SQL-команди.

ViewModel функціонує як посередницький шар між Model і View, забезпечуючи:

* Властивості для зв'язування з UI-компонентами
* Команди для обробки користувацьких дій – створення, модифікації, видалення
* Механізм верифікації вхідних даних
* Взаємодію з контекстом реляційної бази даних

Кожний інтерфейсний модуль має власну ViewModel, яка функціонує незалежно від графічного шару, оперуючи виключно з даними.

Компонент View базується на XAML-розмітці та включає:

* Графічні елементи: ListView, ComboBox, TextBox, Button
* Зв'язування з ViewModel через механізм data binding;

Це забезпечує повну автономність між зовнішнім виглядом системи та внутрішньою бізнес-логікою.

Алгоритм типової взаємодії:

1. При ініціалізації системи MainViewModel встановлює з'єднання з СКБД.
2. Інформація завантажується через DbContext.
3. Дані транслюються до View через механізм прив'язування.
4. Користувач ініціює команду.
5. Відповідна команда у ViewModel модифікує модель та синхронізує контекст СКБД.
6. Інтерфейс оновлюється автоматично завдяки системі прив'язування.

Архітектурні переваги MVVM-підходу:

* Адаптивність: можливість модифікації UI без впливу на бізнес-логіку
* Тесто-придатність: ViewModel функціонує незалежно від UI
* Розширюваність: нові функціональні модулі або View легко інтегруються в існуючу архітектуру
* Супроводжуваність: чітке розмежування відповідальностей спрощує підтримку кодової бази

## **2.2 Концептуальне моделювання інформаційної структури**

Побудова концептуальної моделі даних становить фундаментальний компонент у створенні програмних систем, забезпечуючи адекватне представлення бізнес-процесів у формі структурованих інформаційних конструкцій. Для даного курсового проекту об'єктом дослідження обрана торгівельна організація, діяльність якої охоплює управління товарними ресурсами, їх класифікацію за групами, взаємодію з постачальниками та обслуговування споживачів.

З метою реалізації комплексної логіки товарного обліку було ідентифіковано наступні центральні сутності:

Таблиця 2.2 – Основні сутності бази даних

|  |  |
| --- | --- |
| Сутність | Опис |
| Order | Замовлення – документ, який фіксує продаж товарів. |
| Organization | Організація – володіє торгівельними точками |
| Outlet | Торгова точка – місце продажу товару |
| Product | Товар – основна одиниця обліку. |
| Seller | Продавець – співробітник. |
| Supplier | Постачальник – юридична або фізична особа яка постачає товари. |

Кожна сутність у базі даних описується через набір атрибутів — характеристик, які зберігають необхідну інформацію про об’єкт предметної області. Атрибути визначають структуру таблиць, їх типи даних, а також зв’язки між сутностями.

У межах цієї роботи було визначено шість основних сутностей: Product, Organization, Outlets, Supplier, Order, Sellers (рисунок 2.1).

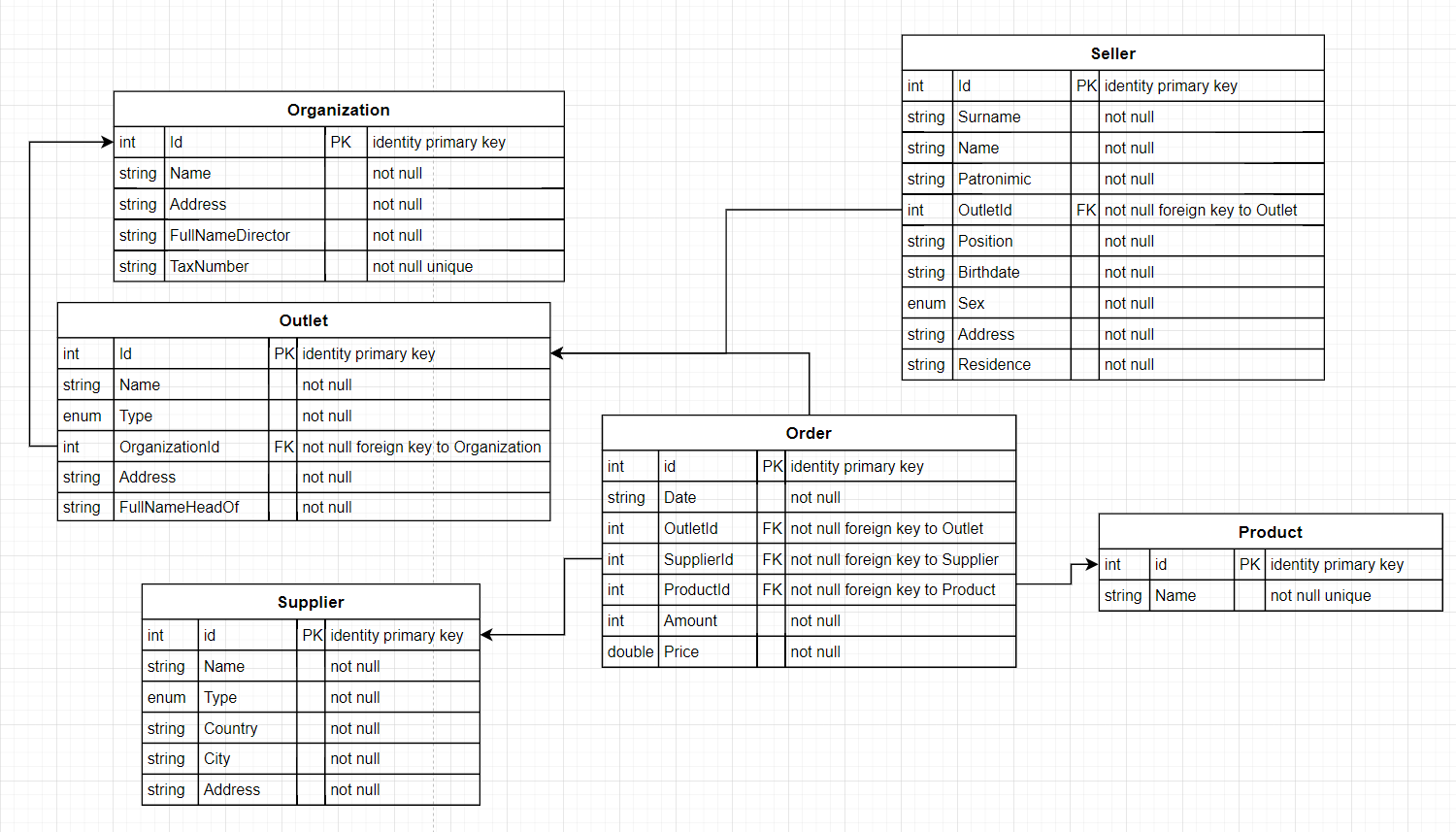


Рисунок 2.1 – Схема бази даних

## **2.3 Створення бази даних**

Для реалізації збереження, обробки та взаємодії з інформацією, пов’язаною з торгівельною організацією, було створено реляційну базу даних. Базу даних розроблено за допомогою ORM-фреймворку Entity Framework Core, який забезпечує зручне об’єктно-реляційне відображення та дозволяє працювати з даними на рівні моделей C#.

Розробка бази даних здійснювалась у таких середовищах і з використанням таких технологій:

* .NET 8.0 + Entity Framework Core – для створення моделей сутностей і керування схемою БД
* SQL Server – як реляційна СКБД
* Міграції EF Core – для створення, оновлення та контролю версій бази даних
* Visual Studio + Package Manager Console / CLI – для запуску міграційних команд

На першому етапі були створені класи-сутності у вигляді C#-моделей, які описують структуру таблиць (рисунок 2.2 – рисунок 2.7).

|  |
| --- |
| public class Order  {  public int Id { get; set; }  [Required]  public DateOnly Date { get; set; }  [Required]  [ForeignKey("Outlet")]  public int OutletId { get; set; }  public Outlet Outlet { get; set; }  [Required]  [ForeignKey("Supplier")]  public int SupplierId { get; set; }  public Supplier Supplier { get; set; }  [Required]  [ForeignKey("Product")]  public int ProductId { get; set; }  public Product Product { get; set; }  [Required]  public int Amount { get; set; }  [Required]  public double Price { get; set; }  } |

Рисунок 2.2 – Модель сутності Order

|  |
| --- |
| public class Organization  {  public int Id { get; set; }  [Required]  public string Name { get; set; }  [Required]  public string Address { get; set; }  [Required]  public string FullNameDirector { get; set; }  [Required]  public string TaxNumber { get; set; }  } |

Рисунок 2.3 – Модель сутності Organization

|  |
| --- |
| public class Outlet  {  public int Id { get; set; }  [Required]  public string Name { get; set; }  [Required]  public OutletType Type { get; set; }  [Required]  [ForeignKey("Organization")]  public int OrganizationId { get; set; }  public Organization Organization { get; set; }  [Required]  public string Address { get; set; }  [Required]  public string FullNameHeadOf { get; set; }  } |

Рисунок 2.4 – Модель сутності Outlet

|  |
| --- |
| public class Product  {  public int Id { get; set; }  [Required]  public string Name { get; set; }  } |

Рисунок 2.5 – Модель сутності Product

|  |
| --- |
| public class Seller  {public int Id { get; set; }  [Required]  public string Surname { get; set; }  [Required]  public string Name { get; set; }  [Required]  public string Patronymic { get; set; }  [Required] [ForeignKey("Outlet")]  public int OutletId { get; set; }  public Outlet Outlet { get; set; }  [Required]  public string Position { get; set; }  [Required]  public DateOnly BirthDate { get; set; }  [Required]  public SexType Sex { get; set; }  [Required]  public string Address { get; set; }  [Required]  public string Residence { get; set; }  } |

Рисунок 2.6 – Модель сутності Seller

|  |
| --- |
| public class Supplier  {  public int Id { get; set; }  [Required]  public string Name { get; set; }  [Required]  public SupplierType Type { get; set; }  [Required]  public string Country { get; set; }  [Required]  public string City { get; set; }  [Required]  public string Address { get; set; }  } |

Рисунок 2.7 – Модель сутності Supplier

Було створено клас DatabaseContext, який успадковується від базового класу DbContext. У цьому класі визначено набори таблиць (DbSet<T>) і налаштовано підключення до СКБД (рисунок 2.8).

|  |
| --- |
| public class DatabaseContext : DbContext  {  public DatabaseContext(DbContextOptions<DatabaseContext> options) : base(options)  {  Database.EnsureCreated();  }  public DbSet<Organization> Organizations { get; set; }  public DbSet<Product> Products { get; set; }  public DbSet<Supplier> Suppliers { get; set; }  public DbSet<Order> Orders { get; set; }  public DbSet<Outlet> Outlets { get; set; }  public DbSet<Seller> Sellers { get; set; }  protected override void OnModelCreating(ModelBuilder builder)  {  builder.Entity<Organization>()  .HasIndex(o => o.TaxNumber)  .IsUnique();  builder.Entity<Product>()  .HasIndex(p => p.Name)  .IsUnique();  builder.UsePropertyAccessMode(PropertyAccessMode.Property);  }  } |

Рисунок 2.8 – Клас контексту

## **2.4 Запити до бази даних**

Механізми доступу до даних становлять центральний інструмент взаємодії між користувачем та інформаційними ресурсами, що зберігаються в системі. У межах реалізації даного курсового проекту взаємодія з базою даних здійснюється через ORM-платформу Entity Framework Core, що надає можливість формулювання запитів засобами LINQ замість прямого використання SQL. Такий підхід суттєво підвищує захищеність, зрозумілість та супроводжуваність програмного коду.

Операції з даними класифікуються за наступними категоріями:

* Селекція інформації
* Генерація нових записів
* Модифікація наявних записів
* Елімінація даних
* Пошукові операції та селективна фільтрація

Для представлення даних у візуальному інтерфейсі (зокрема, у компонентах ListView) зазвичай застосовується вибірка повного набору записів або їх підмножини з урахуванням асоційованих сутностей (рисунок 2.9).

|  |
| --- |
| public List<Product> GetAll()  {  return \_context.Products.ToList();  } |

Рисунок 2.9 – Селекція інформації

Процес створення нового товарного запису реалізується наступним чином (рисунок 2.10):

|  |
| --- |
| public void Add(Product product)  {if (product == null) {  throw new ArgumentNullException(nameof(product), "Product cannot be null");  }  \_context.Products.Add(product);  \_context.SaveChanges();  } |

Рисунок 2.10 – Генерація нових записів

Коригування параметрів товару (наприклад, оновлення складських залишків) виконується таким способом (рисунок 2.11):

|  |
| --- |
| public void Edit(Product product)  {  if (product == null)  {  throw new ArgumentNullException(nameof(product), "Product cannot be null");  }  var existingProduct = \_context.Products  .FirstOrDefault(p => p.Id == product.Id);  if (existingProduct == null)  {  throw new KeyNotFoundException($"Product with ID {product.Id} not found");  }  existingProduct.Name = product.Name;  \_context.SaveChanges();  } |

Рисунок 2.11 – Модифікація наявних записів

Операція видалення товарного об'єкта реалізується наступним чином (рисунок 2.12):

|  |
| --- |
| public void Delete(int id)  {  var product = \_context.Products  .FirstOrDefault(p => p.Id == id);  if (product == null)  {  throw new KeyNotFoundException($"Product with ID {id} not found");  }  \_context.Products.Remove(product);  \_context.SaveChanges();  } |

Рисунок 2.12 – Елімінація даних

У програмному продукті реалізовано функціонал пошуку товарів за текстовими критеріями (рисунок 2.13 – рисунок 2.14):

|  |
| --- |
| public List<Product> SearchByName(string? namePart)  {  if (string.IsNullOrWhiteSpace(namePart))  return \_context.Products.ToList();  string lowered = namePart.ToLower();  return \_context.Products  .Where(p => p.Name.ToLower().Contains(lowered))  .ToList();  } |

Рисунок 2.13 - Пошукові операції та селективна фільтрація

|  |
| --- |
| public string GetProductNameById(int id)  {  var product = \_context.Products  .FirstOrDefault(p => p.Id == id);  return product?.Name ?? string.Empty;  }  public int GetProductIdByName(string name)  {  var product = \_context.Products  .FirstOrDefault(p => p.Name == name);  return product?.Id ?? 0;  } |

Рисунок 2.14 - Пошукові операції та селективна фільтрація

Переваги використання LINQ та EF Core:

* Висока читабельність коду — логіка запитів легко зчитується і схожа на природну мову
* Безпека типів — компілятор перевіряє правильність структури запиту
* Інтеграція з інтерфейсом — запити можна напряму прив’язувати до властивостей ViewModel
* Можливість побудови складних запитів без написання SQL
* Підтримка асинхронності — для роботи з великими обсягами даних без блокування інтерфейсу

# **РОЗДІЛ 3**

# **ВЗАЄМОДІЯ З ПРОГРАМНИМИ КОМПОНЕНТАМИ**

Спочатку треба запустити програму, після чого зустрічаємось з розділом взаємодії з «Продукти» (рисунок 3.1).

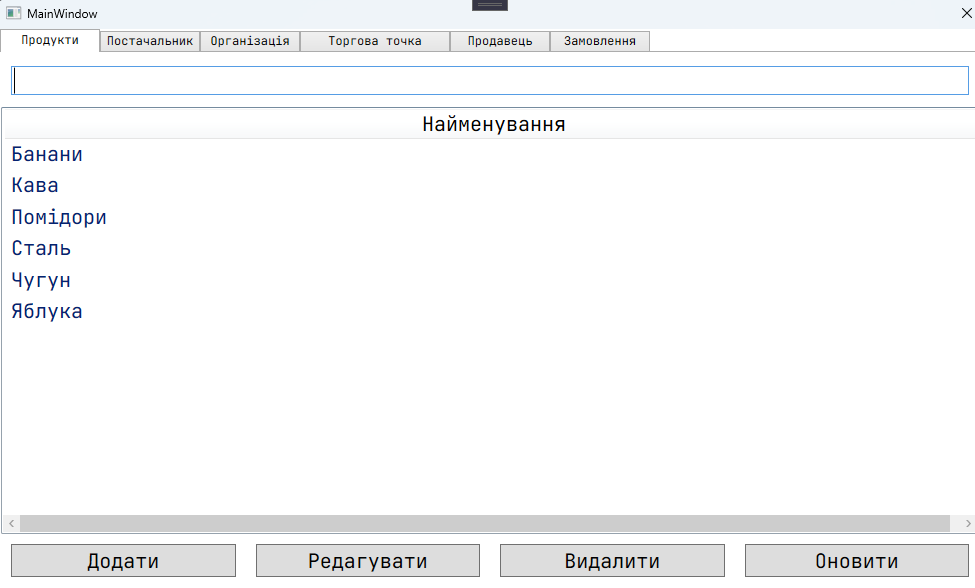


Рисунок 3.1 – Меню взаємодії з «Продуктами»

Для додавання даних до «Продукти» треба натиснути кнопку «Додати», після у вікно яке відкриється ввести назву та натиснути «Додати продукт» (рисунок 3.2).

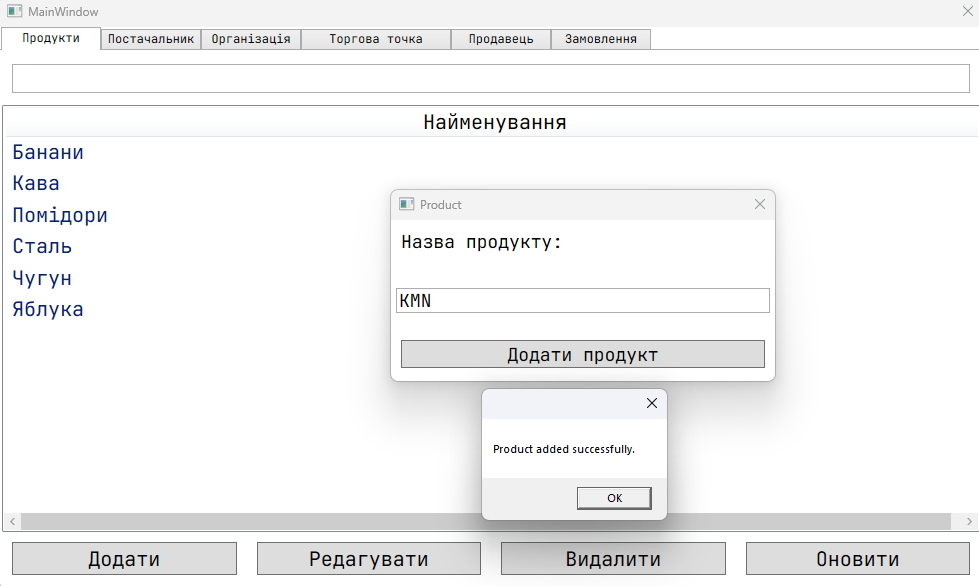


Рисунок 3.2 – Додавання продукту до бази

Для редагування даних треба натиснути кнопку «Редагувати», після чого змінити дані котрі не подобаються та натиснути на «Зберегти продукт» (рисунок 3.3 – рисунок 3.4).

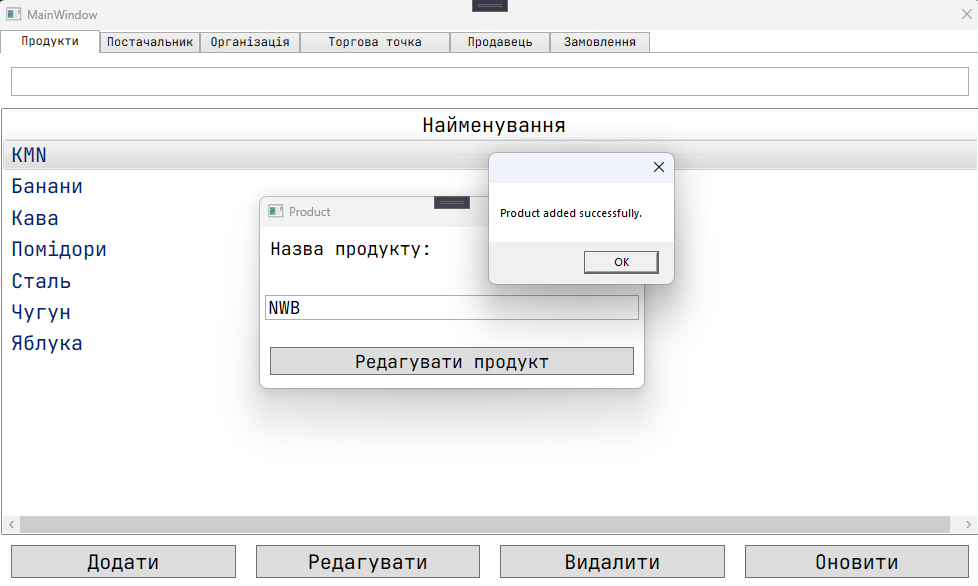


Рисунок 3.3 – Зміна назви продукту

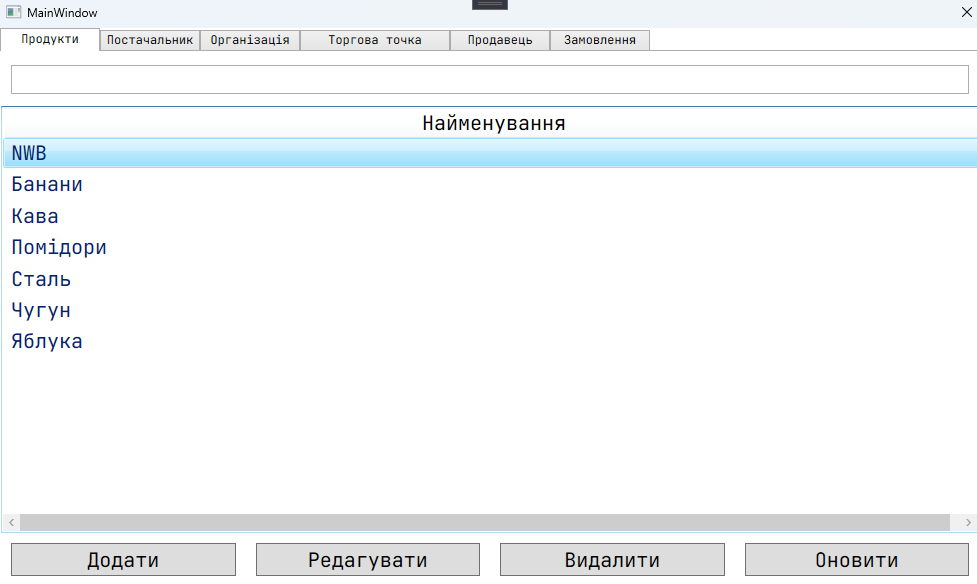


Рисунок 3.4 – Відредагований запис продукту

# **ВИСНОВКИ**

Протягом виконання курсового проекту було успішно створено комплексне програмне рішення для управління базою даних торговельного підприємства. Розроблена система надає користувачам зручні можливості взаємодії з інформаційними ресурсами через сучасний графічний інтерфейс, побудований із застосуванням технологій WPF та архітектурної концепції MVVM.

Ключові результати проекту включають:

* Здійснено детальне дослідження предметної сфери та ідентифіковано головні об'єкти системи, їхні характеристики та взаємозв'язки;
* Сформовано концептуальну модель інформаційної структури та втілено її у формі програмних моделей на мові C#;
* Розгорнуто реляційну систему збереження даних із залученням Entity Framework Core та методології Code First, що гарантує простоту модифікації та обслуговування архітектури;
* Впроваджено ключові функціональні можливості: перегляд, створення, оновлення, вилучення записів, а також засоби пошуку та відбору інформації;
* Розроблено ергономічний та логічно структурований користувацький інтерфейс із дотриманням принципу поділу відповідальності через застосування шаблону MVVM.

Досягнутий результат повною мірою задовольняє визначені цілі та завдання дослідження. Створений програмний комплекс демонструє ефективне використання сучасного інструментарію .NET-екосистеми для розробки прикладних рішень із можливістю інтеграції з системами управління базами даних.

Перспективи подальшого розвитку проекту охоплюють можливості розширення функціоналу: впровадження системи ідентифікації користувачів, модулів аналітичної звітності або переходу до хмарних рішень для зберігання даних.

Варто відзначити, що архітектурні рішення, прийняті в ході розробки, забезпечують високу масштабованість системи. Використання патерну MVVM дозволяє легко додавати нові функціональні модулі без порушення існуючої логіки, а застосування Dependency Injection створює передумови для ефективного тестування та підтримки коду.

Особливу увагу приділено забезпеченню надійності роботи з даними. Реалізовано механізми валідації введених користувачем значень, обробки виняткових ситуацій та контролю цілісності інформації в базі даних. Система міграцій Entity Framework Core гарантує безпечне оновлення структури бази даних у виробничому середовищі.

З точки зору користувацького досвіду, інтерфейс програми характеризується інтуїтивністю та функціональною повнотою. Впроваджено адаптивний дизайн, що забезпечує коректне відображення на різних роздільностях екрану, а також реалізовано систему швидких клавіш для підвищення продуктивності роботи досвідчених користувачів.

Результати курсової роботи мають практичну цінність та можуть бути використані як основа для створення реальних комерційних рішень у сфері автоматизації торговельних процесів. Застосовані технології та підходи відповідають сучасним стандартам індустрії розробки програмного забезпечення.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. *Entity Framework Core* – офіційна документація. [Електронний ресурс] – Режим доступу: <https://learn.microsoft.com/en-us/ef/core>
2. *Visual Studio 2022* – офіційна документація. [Електронний ресурс] – Режим доступу: <https://learn.microsoft.com/en-us/visualstudio>
3. *Git* – офіційна документація. [Електронний ресурс] – Режим доступу: <https://git-scm.com/doc>
4. *Code First* – офіційна документація Microsoft. [Електронний ресурс] – Режим доступу: <https://learn.microsoft.com/en-us/ef/core/modeling>
5. *LINQ (Language Integrated Query)* – офіційна документація. [Електронний ресурс] – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq>

# **ДОДАТОК А**

<Window x:Class="CourseWork\_DB\_ONU.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CourseWork\_DB\_ONU"

mc:Ignorable="d" Title="MainWindow" ResizeMode="NoResize"

Height="600" Width="1000" FontFamily="JetBrains Mono">

<Grid>

<Grid>

<TabControl>

<TabItem Header="Продукти" Width="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="8\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<TextBox x:Name="\_TextBoxSearchProduct" Grid.Column="0"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchProduct\_TextChanged"/>

</Grid>

<ListView x:Name="\_ListViewProducts" Grid.Row="1"

FontSize="20" Loaded="\_ListViewProducts\_Loaded">

<ListView.View>

<GridView>

<GridViewColumn Header="Найменування"

Width="{Binding ActualWidth, ElementName=\_ListViewProducts}"

DisplayMemberBinding="{Binding Name}">

</GridViewColumn>

</GridView>

</ListView.View>

</ListView>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="\_ButtonAddProduct" Grid.Column="0"

Content="Додати" Margin="10"

FontSize="20" Click="\_ButtonAddProduct\_Click"/>

<Button x:Name="\_ButtonEditProduct" Grid.Column="1"

Content="Редагувати" Margin="10"

FontSize="20" Click="\_ButtonEditProduct\_Click"/>

<Button x:Name="\_ButtonDeleteProduct" Grid.Column="2"

Content="Видалити" Margin="10"

FontSize="20" Click="\_ButtonDeleteProduct\_Click"/>

<Button x:Name="\_ButtonUpdateProduct" Grid.Column="3"

Content="Оновити" Margin="10"

FontSize="20" Click="\_ButtonUpdateProduct\_Click"/>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Постачальник" Width="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="8\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBox x:Name="\_TextBoxSearchSupplierByName" Grid.Column="0"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchSupplier\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSupplierByType" Grid.Column="1"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchSupplier\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSupplierByCountry" Grid.Column="2"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchSupplier\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSupplierByCity" Grid.Column="3"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchSupplier\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSupplierByAddress" Grid.Column="4"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchSupplier\_TextChanged"/>

</Grid>

<ListView x:Name="\_ListViewSupplier" Grid.Row="1"

FontSize="20" Loaded="\_ListViewSupplier\_Loaded">

<ListView.View>

<GridView>

<GridViewColumn Header="Назва" Width="200"

DisplayMemberBinding="{Binding Name}"/>

<GridViewColumn Header="Тип" Width="200"

DisplayMemberBinding="{Binding TypeDescription}"/>

<GridViewColumn Header="Країна" Width="150"

DisplayMemberBinding="{Binding Country}"/>

<GridViewColumn Header="Місто" Width="150"

DisplayMemberBinding="{Binding City}"/>

<GridViewColumn Header="Адреса" Width="150"

DisplayMemberBinding="{Binding Address}"/>

</GridView>

</ListView.View>

</ListView>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="\_ButtonAddSupplier" Grid.Column="0"

Content="Додати" Margin="10"

FontSize="20" Click="\_ButtonAddSupplier\_Click"/>

<Button x:Name="\_ButtonEditSupplier" Grid.Column="1"

Content="Редагувати" Margin="10"

FontSize="20" Click="\_ButtonEditSupplier\_Click"/>

<Button x:Name="\_ButtonDeleteSupplier" Grid.Column="2"

Content="Видалити" Margin="10"

FontSize="20" Click="\_ButtonDeleteSupplier\_Click"/>

<Button x:Name="\_ButtonUpdateSupplier" Grid.Column="3"

Content="Оновити" Margin="10"

FontSize="20" Click="\_ButtonUpdateSupplier\_Click"/>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Організація" Width="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="8\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBox x:Name="\_TextBoxSearchOrganizationByName" Grid.Column="0"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOrganization\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrganizationByAddress" Grid.Column="1"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOrganization\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrganizationByFullNameDirector" Grid.Column="2"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOrganization\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrganizationByTaxNumber" Grid.Column="3"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOrganization\_TextChanged"/>

</Grid>

<ListView x:Name="\_ListViewOrganization" Grid.Row="1"

FontSize="20" Loaded="\_ListViewOrganization\_Loaded">

<ListView.View>

<GridView>

<GridViewColumn Header="Назва" Width="200"

DisplayMemberBinding="{Binding Name}"/>

<GridViewColumn Header="Адреса" Width="200"

DisplayMemberBinding="{Binding Address}"/>

<GridViewColumn Header="ПІБ Директора" Width="250"

DisplayMemberBinding="{Binding FullNameDirector}"/>

<GridViewColumn Header="Номер" Width="150"

DisplayMemberBinding="{Binding TaxNumber}"/>

</GridView>

</ListView.View>

</ListView>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="\_ButtonAddSOrganization" Grid.Column="0"

Content="Додати" Margin="10"

FontSize="20" Click="\_ButtonAddOrganization\_Click"/>

<Button x:Name="\_ButtonEditOrganization" Grid.Column="1"

Content="Редагувати" Margin="10"

FontSize="20" Click="\_ButtonEditOrganization\_Click"/>

<Button x:Name="\_ButtonDeleteOrganization" Grid.Column="2"

Content="Видалити" Margin="10"

FontSize="20" Click="\_ButtonDeleteOrganization\_Click"/>

<Button x:Name="\_ButtonUpdateOrganization" Grid.Column="3"

Content="Оновити" Margin="10"

FontSize="20" Click="\_ButtonUpdateOrganization\_Click"/>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Торгова точка" Width="150">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="8\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBox x:Name="\_TextBoxSearchOutletByName" Grid.Column="0"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOutlet\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOutletByType" Grid.Column="1"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOutlet\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOutletByOrganization" Grid.Column="2"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOutlet\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOutletByAddress" Grid.Column="3"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOutlet\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOutletByFullNameHeadOf" Grid.Column="4"

Margin="10" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="20" TextChanged="\_TextBoxSearchOutlet\_TextChanged"/>

</Grid>

<ListView x:Name="\_ListViewOutlet" Grid.Row="1"

FontSize="20" Loaded="\_ListViewOutlet\_Loaded">

<ListView.View>

<GridView>

<GridViewColumn Header="Назва" Width="200"

DisplayMemberBinding="{Binding Name}"/>

<GridViewColumn Header="Тип" Width="200"

DisplayMemberBinding="{Binding TypeDescription}"/>

<GridViewColumn Header="Організація" Width="250"

DisplayMemberBinding="{Binding OrganizationName}"/>

<GridViewColumn Header="Адреса" Width="150"

DisplayMemberBinding="{Binding Address}"/>

<GridViewColumn Header="ПІБ керівника" Width="150"

DisplayMemberBinding="{Binding FullNameHeadOf}"/>

</GridView>

</ListView.View>

</ListView>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="\_ButtonAddOutlet" Grid.Column="0"

Content="Додати" Margin="10"

FontSize="20" Click="\_ButtonAddOutlet\_Click"/>

<Button x:Name="\_ButtonEditOutlet" Grid.Column="1"

Content="Редагувати" Margin="10"

FontSize="20" Click="\_ButtonEditOutlet\_Click"/>

<Button x:Name="\_ButtonDeleteOutlet" Grid.Column="2"

Content="Видалити" Margin="10"

FontSize="20" Click="\_ButtonDeleteOutlet\_Click"/>

<Button x:Name="\_ButtonUpdateOutlet" Grid.Column="3"

Content="Оновити" Margin="10"

FontSize="20" Click="\_ButtonUpdateOutlet\_Click"/>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Продавець" Width="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="8\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Прізвище"/>

<TextBlock Grid.Column="1" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Ім'я"/>

<TextBlock Grid.Column="2" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="По-батькові"/>

<TextBlock Grid.Column="3" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Точка"/>

<TextBlock Grid.Column="4" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Посада"/>

<TextBlock Grid.Column="5" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Стать"/>

<TextBlock Grid.Column="6" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Адреса"/>

<TextBlock Grid.Column="7" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Місто прож."/>

<TextBox x:Name="\_TextBoxSearchSellerBySurname" Grid.Column="0" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSellerByName" Grid.Column="1" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSellerByPatronymic" Grid.Column="2" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSellerByOutlet" Grid.Column="3" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSellerByPosition" Grid.Column="4" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSellerBySex" Grid.Column="5" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSellerByAddress" Grid.Column="6" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchSellerByResidence" Grid.Column="7" Grid.Row="1"

VerticalAlignment="Center" FontSize="16" Margin="5"

TextChanged="\_TextBoxSearchSeller\_TextChanged"/>

</Grid>

<ListView x:Name="\_ListViewSeller" Grid.Row="1"

FontSize="20" Loaded="\_ListViewSeller\_Loaded">

<ListView.View>

<GridView>

<GridViewColumn Header="Прізвище" Width="200"

DisplayMemberBinding="{Binding Surname}"/>

<GridViewColumn Header="Ім'я" Width="200"

DisplayMemberBinding="{Binding Name}"/>

<GridViewColumn Header="По-батькові" Width="250"

DisplayMemberBinding="{Binding Patronymic}"/>

<GridViewColumn Header="Назва точки" Width="150"

DisplayMemberBinding="{Binding OutletName}"/>

<GridViewColumn Header="Посада" Width="150"

DisplayMemberBinding="{Binding Position}"/>

<GridViewColumn Header="Дата народження" Width="150"

DisplayMemberBinding="{Binding BirthDate}"/>

<GridViewColumn Header="Стать" Width="150"

DisplayMemberBinding="{Binding TypeDescription}"/>

<GridViewColumn Header="Адреса" Width="150"

DisplayMemberBinding="{Binding Address}"/>

<GridViewColumn Header="Місто проживання" Width="150"

DisplayMemberBinding="{Binding Residence}"/>

</GridView>

</ListView.View>

</ListView>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="\_ButtonAddSeller" Grid.Column="0"

Content="Додати" Margin="10"

FontSize="20" Click="\_ButtonAddSeller\_Click"/>

<Button x:Name="\_ButtonEditSeller" Grid.Column="1"

Content="Редагувати" Margin="10"

FontSize="20" Click="\_ButtonEditSeller\_Click"/>

<Button x:Name="\_ButtonDeleteSeller" Grid.Column="2"

Content="Видалити" Margin="10"

FontSize="20" Click="\_ButtonDeleteSeller\_Click"/>

<Button x:Name="\_ButtonUpdateSeller" Grid.Column="3"

Content="Оновити" Margin="10"

FontSize="20" Click="\_ButtonUpdateSeller\_Click"/>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Замовлення" Width="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="8\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Дата"/>

<TextBlock Grid.Column="1" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Торгова точка"/>

<TextBlock Grid.Column="2" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Постачальник"/>

<TextBlock Grid.Column="3" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Товар"/>

<TextBlock Grid.Column="4" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Кількість"/>

<TextBlock Grid.Column="5" Margin="5" VerticalAlignment="Center"

FontSize="16" Text="Ціна"/>

<TextBox x:Name="\_TextBoxSearchOrderByDate" Grid.Column="0" Grid.Row="1"

Margin="5" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="16" TextChanged="\_TextBoxSearchOrder\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrderByOutlet" Grid.Column="1" Grid.Row="1"

Margin="5" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="16" TextChanged="\_TextBoxSearchOrder\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrderBySupplier" Grid.Column="2" Grid.Row="1"

Margin="5" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="16" TextChanged="\_TextBoxSearchOrder\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrderByProduct" Grid.Column="3" Grid.Row="1"

Margin="5" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="16" TextChanged="\_TextBoxSearchOrder\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrderByAmount" Grid.Column="4" Grid.Row="1"

Margin="5" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="16" TextChanged="\_TextBoxSearchOrder\_TextChanged"/>

<TextBox x:Name="\_TextBoxSearchOrderByPrice" Grid.Column="5" Grid.Row="1"

Margin="5" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

FontSize="16" TextChanged="\_TextBoxSearchOrder\_TextChanged"/>

</Grid>

<ListView x:Name="\_ListViewOrder" Grid.Row="1"

FontSize="20" Loaded="\_ListViewOrder\_Loaded">

<ListView.View>

<GridView>

<GridViewColumn Header="Дата" Width="200"

DisplayMemberBinding="{Binding Date}"/>

<GridViewColumn Header="Торгова точка" Width="200"

DisplayMemberBinding="{Binding OutletName}"/>

<GridViewColumn Header="Постачальник" Width="250"

DisplayMemberBinding="{Binding SupplierName}"/>

<GridViewColumn Header="Товар" Width="150"

DisplayMemberBinding="{Binding ProductName}"/>

<GridViewColumn Header="Кількість" Width="150"

DisplayMemberBinding="{Binding Amount}"/>

<GridViewColumn Header="Ціна" Width="150"

DisplayMemberBinding="{Binding Price}"/>

</GridView>

</ListView.View>

</ListView>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="\_ButtonAddOrder" Grid.Column="0"

Content="Додати" Margin="10"

FontSize="20" Click="\_ButtonAddOrder\_Click"/>

<Button x:Name="\_ButtonEditOrder" Grid.Column="1"

Content="Редагувати" Margin="10"

FontSize="20" Click="\_ButtonEditOrder\_Click"/>

<Button x:Name="\_ButtonDeleteOrder" Grid.Column="2"

Content="Видалити" Margin="10"

FontSize="20" Click="\_ButtonDeleteOrder\_Click"/>

<Button x:Name="\_ButtonUpdateOrder" Grid.Column="3"

Content="Оновити" Margin="10"

FontSize="20" Click="\_ButtonUpdateOrder\_Click"/>

</Grid>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Grid>

</Window>