МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
Факультет механіки, енергетики та інформаційних технологій
Кафедра інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему:

«РОЗРОБКА КЛІЄНТСЬКОГО ЗАСТОСУНКУ ДЛЯ ОБЛІКУ ТОВАРІВ НА СКЛАДІ»

Виконав: здобувач групи ІТ-41
спеціальності 126 «Інформаційні системи та технології»

Дорощук В.О.
(прізвище та ініціали)

Керівник: Смолінський В.Б.
(прізвище та ініціали)

Дубляни 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____202 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дорощук Вадим Олександрович
(прізвище, ім’я, по батькові)

1. Тема роботи «Розробка клієнтського застосунку для обліку товарів на складі»

керівник роботи *к. е. н., доцент., Смолінський В. Б.*
(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП *від 27.11.2023 року № 641/к-с*
2. Строк подання студентом роботи *10 червня 2024 року*
3. Вихідні дані до роботи: *характеристика предметної сфери; вихідні дані та вимоги до роботи додатку, опис використаних технологій та мов програмування, науково-технічна і довідкова література.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ
Аналіз вимог
Відомості про використані технології
Проектування та реалізація застосунку
Охорона праці та безпека в надзвичайних ситуаціях
Висновки
Список використаних джерел
5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3	*Смолінський В. Б., к.е.н., доцент*			
4	*Городецький І. М., к.т.н., доцент*			

7. Дата видачі завдання *28 листопада 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Аналіз вимог*	*28.11.2023 – 31.12.2023*	
2	*Відомості про використані технології*	*01.01.2024 – 28.02.2024*	
3	*Проектування та реалізація застосунку*	*01.03.2024 – 30.04.2024*	
4	*Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях*	*01.05.2024 – 14.05.2024*	
5	*Завершення оформлення розрахунково-пояснювальної записки та презентаційного матеріалу*	*15.05.2024 – 31.05.2024*	
6	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*01.06.2024 – 10.06.2024*	

Здобувач _____ *Дорощук В. О.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Смолінський В. Б.*
(підпис) (прізвище та ініціали)

УДК 004.78

Розробка клієнтського застосунку для обліку товарів на складі. Кваліфікаційна робота. Дорощук Вадим Олександрович, кафедра інформаційних технологій, Дубляни, Львівський НУП, 2024р.

43 с. текст. част.,5 табл., 3 рис., 15 джерел.

У кваліфікаційній роботі описано розроблений клієнтський застосунок для обліку товарів на складі. Для цього були використані такі технології як мова програмування C#, .NET (Entity Framework і WPF), а також СУБД PostgreSQL.

Впровадження розробленого програмного забезпечення дозволить значно покращити ефективність управління запасами, зменшити час на їх облік та оптимізувати процеси замовлення та постачання.

Ключові слова: управління запасами, бази даних, PostgreSQL, Windows Presentation Foundation, Entity Framework

Keywords: inventory management, databases, PostgreSQL, Windows Presentation Foundation, Entity Framework.

Зміст

ВСТУП

Обґрунтування важливості теми

Розробка програмного забезпечення обліку товарів є дуже важливим елементом для кожного підприємства. Важливість має різні аспекти, а саме:

Підвищення ефективності бізнесу: Доцільність у використанні програмного забезпечення для обліку товарних запасів полягає у здатності оптимізувати процеси управління запасами, що може призвести до підвищення ефективності ведення бізнесу. Автоматизовані системи обліку дозволяють підприємцям швидше та точніше контролювати запаси, уникнути надлишків або нестачі та мінімізувати втрати.

Забезпечення точності та надійності даних: Ручний облік товарних запасів може призвести до помилок та неточностей. Розробка програмного забезпечення дозволить автоматизувати процес обліку, що сприятиме підвищенню точності та надійності даних про запаси.

Покращення прийняття управлінських рішень: Інтеграція аналітичних звітів та показників ефективності в програмне забезпечення дозволить підприємцям отримувати достовірну та зручну інформацію для прийняття управлінських рішень щодо оптимізації запасів, розробки маркетингових стратегій тощо.

Економія часу та ресурсів: Автоматизація процесів обліку товарних запасів зменшить ручні операції та ризик помилок, що дозволить підприємцям зосередитися на стратегічних завданнях розвитку бізнесу та зекономити час та ресурси.

Конкурентна перевага: В сучасному бізнес-середовищі, де швидкість та точність прийняття рішень є ключовими факторами успіху, наявність ефективної системи обліку товарних запасів може стати конкурентною перевагою для підприємства.

Отже, розробка програмного забезпечення для обліку товарних запасів для підприємств є важливою, оскільки вона сприяє покращенню управління бізнесом, оптимізації процесів та забезпеченню конкурентних переваг на ринку.

Опис об'єкта та предмета дослідження.

Об'єкт дослідження: Програмне забезпечення для обліку товарних запасів для підприємств.

Предмет дослідження: Методи, техніки та інструменти, що використовуються для розробки програмного забезпечення для обліку товарних запасів, а також його вплив на ефективність управління запасами та результативність підприємства.

Об'єкт дослідження - це саме програмне забезпечення, яке розробляється та впроваджується для вирішення питань обліку товарних запасів у підприємстві. Предмет дослідження охоплює більш широкий аспект, включаючи методи та інструменти розробки програмного забезпечення, вплив його використання на процеси управління запасами та підприємницьку діяльність загалом.

У рамках дослідження були розглянуті такі аспекти:

1. Аналіз поточних методів обліку товарних запасів на підприємстві.

2. Вивчення потреб підприємства в програмному забезпеченні для обліку товарних запасів.

3. Дослідження технологій та інструментів розробки програмного забезпечення.

4. Розробка та впровадження програмного забезпечення для обліку товарних запасів.

Таким чином, об'єкт та предмет дослідження визначають область дослідження та основні аспекти, які розглянуті у курсовій роботі.

Формулювання мети та завдань дослідження.

Мета дослідження є вивчення можливостей розробки та впровадження програмного ресурсу для обліку товарів у підприємствах з метою покращення ефективності управління запасами та підвищення конкурентоспроможності.

Завдання дослідження:

1. Проаналізувати поточні методи обліку товарних запасів: Визначити переваги та недоліки поточних підходів до обліку запасів у підприємстві.

2. Вивчити потреби та вимоги підприємства щодо програмного забезпечення для обліку товарних запасів: З'ясувати вимоги та очікування підприємства від програмного забезпечення для управління запасами.

3. Дослідити технології та інструменти розробки програмного забезпечення: Оцінити доступні технології та інструменти для розробки програмного забезпечення, які найбільше відповідають потребам підприємства.

4. Розробити програмне забезпечення для обліку товарних запасів: Розробити програмне забезпечення, яке відповідає вимогам та потребам підприємства з урахуванням зібраних даних та досліджень.

5. Оцінити результати та зробити висновки: Провести аналіз результатів роботи програмного забезпечення, визначити переваги та недоліки, зробити висновки щодо його впливу на управління запасами та ефективність діяльності підприємства.

Ці завдання дозволять досягти мети дослідження та відповідати на питання, пов'язані з розробкою та впровадженням програмного забезпечення для обліку товарних запасів у підприємствах.

РОЗДІЛ 1.
АНАЛІЗ ВИМОГ

1.1. Дослідження потреб цільової аудиторії (підприємців)

При вивченні потреб цільової аудиторії, особливо підприємців, можна додати наступні елементи:

Сегментація аудиторії: Розбиття цільової аудиторії на підгрупи за такими критеріями, як розмір бізнесу, галузь, регіон, потреби тощо. Це допоможе краще розуміти різноманітність потреб підприємців.

Аналіз поточних проблем і викликів: Визначення основних проблем, з якими зіштовхуються підприємці в управлінні своїми товарними запасами. Це може включати проблеми з контролем залишків, ефективністю замовлень, нестачею інформації тощо.

Аналіз конкурентного середовища: Вивчення того, які інструменти або програмні рішення використовують конкуренти для обліку товарних запасів. Це допоможе зрозуміти, як можна вдосконалити власну програму для привертання та задоволення потреб підприємців.

Збір відгуків і пропозицій: Організація опитувань, інтерв'ю або фокус-груп серед підприємців для отримання їхніх прямих відгуків щодо потреб у програмі для обліку товарних запасів. Це може розкрити додаткові вимоги або функціональні можливості, які можна врахувати під час розробки програмного забезпечення.

Аналіз тенденцій індустрії: Врахування тенденцій та інновацій у галузі управління запасами, таких як автоматизація процесів, використання штучного інтелекту або аналітики даних. Це допоможе створити програмне забезпечення, яке буде відповідати сучасним вимогам та потребам підприємців.

Ці елементи допоможуть зібрати повну інформацію про потреби та вимоги цільової аудиторії, що є ключовим для успішного розроблення програмного забезпечення.

Але це не є завдання в моїй курсовій роботі, я зосередився на програмній частині, а економічну можливо колись зроблю.

1.2. Формулювання функціональних та нефункціональних вимог до програми

Застосунок дозволить користувачам вести детальний облік товарів, слідкувати за їхнім рухом, контролювати залишки, робити замовлення та забезпечувати легкий доступ до інформації про товари.

Основні функціональні вимоги

- Авторизація та аутентифікація користувачів

- Забезпечення можливості реєстрації нових користувачів.

- Вхід до системи з використанням імені користувача та пароля.

- Можливість вибору ролі (адміністратор або оператор складу).

- Інтерфейс користувача

- Вікно реєстрації/авторизації.

- Головне вікно адміністратора - вікно управління користувачами.

- Вікно додавання/редагування товару адміністратора з можливістю фільтрації.

- Головне вікно оператора складу - можливість перегляду та оновлення інформації про кількість товарів після поставок або продажу з можливістю фільтрації.

- Можливість переходу між вікнами та використання "випадаючого" меню.

- Управління товарами

- Додавання, редагування та видалення товарів.

- Можливість додавати фотографії товарів.

- Можливість створення QR-коду для кожного товару.

- Категоризація товарів для зручного пошуку.

- Відстеження руху товарів

- Внесення інформації про прихід і відвантаження товарів.

- Фіксація дати та кількості товарів.

- Планування замовлень

- Відображення рівня запасів товарів.

- Рекомендації щодо потреби у нових замовленнях.

- Документація і підтримка

- Створення інструкції користувача та технічної документації.

- Надання технічної підтримки користувачам.

Основні нефункціональні вимоги

Ось деякі загальні приклади нефункціональних вимог:

Ефективність: Система повинна працювати ефективно, забезпечуючи швидкий доступ до даних та оперативну відповідь на запити користувачів.

Надійність: Система повинна бути надійною і стабільною, мінімізуючи можливість виникнення помилок та відновлюючи роботу після аварій.

Безпека: Система повинна забезпечувати захист конфіденційної інформації та запобігати несанкціонованому доступу до неї.

Масштабованість: Система повинна бути легко масштабована для відповіді на зростаючі потреби користувачів та обсяги даних.

Зручність використання: Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним у використанні, щоб користувачі могли легко оволодіти програмою.

А це вже мої конкретні нефункціональні вимоги до застосунку:

- Мова інтерфейсу: українська.

- Вимоги до ПЗ: Підтримка операційних систем Windows 10 і вище, наявність .NET Framework для запуску програми.

- База даних: Використання SQL бази даних для зберігання даних.

- Вимоги по безпеці: Авторизація користувачів перед доступом до функціоналу, можливість встановлення паролів для користувачів та різні рівні доступу (адміністратор, оператор), запобігання втрати даних через регулярне резервне копіювання бази даних і можливість відновлення даних у випадку втрати або пошкодження.

РОЗДІЛ 2.
ВІДОМОСТІ ПРО ВИКОРИСТАНІ ТЕХНОЛОГІЇ

Для створення даного застосунку було використано мову програмування C# та пов’язану з нею таку технології, як .NET (Entity Framework і WPF), а також СУБД PostgreSQL.

2.1. Мова C# та .NET

C# та .NET є потужними технологіями, які часто використовуються разом для розробки програмного забезпечення. Ось детальний опис кожної з них:

C# (вимовляється як "сі шарп") - це сучасна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина платформи .NET. Вона була створена у 2000 році і має наступні основні особливості:

Основні Особливості C#

1. Об'єктно-орієнтоване програмування (ООП): Підтримка таких концепцій ООП, як наслідування, поліморфізм, інкапсуляція і абстракція.

2. Статична типізація: Всі змінні та об'єкти повинні мати визначений тип на момент компіляції, що допомагає виявляти помилки раніше.

3. Управління пам'яттю: Використовує автоматичний збирач сміття (Garbage Collector), що спрощує управління пам'яттю.

4. Розширюваність: Підтримка розширених методів (extension methods) та властивостей (properties), що дозволяє розширювати можливості класів без їх модифікації.

5. LINQ (Language Integrated Query): Мова запитів, інтегрована в C#, яка дозволяє працювати з даними в зручному та декларативному стилі.

6. Асинхронне програмування: Підтримка асинхронних методів через ключові слова `async` та `await`.

7. Безпека типів: Високий рівень контролю за типами даних, що зменшує ризик помилок типу і підвищує надійність коду.

Основні Конструкції

1. Класи та об'єкти:

```csharp
public class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

    public void SayHello()
    {
        Console.WriteLine($"Hello, my name is {Name}");
    }
}
```

2. Наслідування:

```csharp
public class Employee : Person
{
    public string Position { get; set; }
}
```

3. Інтерфейси:

```csharp
public interface IWorker
{
    void Work();
}
public class Manager : Person, IWorker
{
    public void Work()
    {
        Console.WriteLine("Managing work...");
```

```
    }
}
```

4. Асинхронні методи:

```csharp
public async Task<int> CalculateAsync()
{
    await Task.Delay(1000); // Симуляція асинхронної операції
    return 42;
}
```

.NET

.NET (вимовляється як "дот нет") - це кросплатформна, відкрита платформа для розробки програмного забезпечення, яка підтримує різні мови програмування, серед яких C#, VB.NET, F, та інші. Платформа .NET включає в себе кілька компонентів:

Основні Компоненти .NET

1. Common Language Runtime (CLR): Віртуальна машина, яка виконує код і надає такі послуги, як управління пам'яттю, безпека, управління потоками та обробка винятків.

2. .NET Class Library: Бібліотека класів, яка надає набір багаторазово використовуваних типів і API для розробки програм.

3. .NET Core / .NET 5+: Кросплатформна версія .NET, яка дозволяє створювати додатки для Windows, macOS і Linux. Починаючи з .NET 5, платформа об'єднала .NET Framework і .NET Core.

4. ASP.NET: Фреймворк для розробки веб-додатків і веб-сервісів.

5. Entity Framework (EF): ORM (Object-Relational Mapping) фреймворк для роботи з базами даних.

6. Xamarin: Інструменти для розробки мобільних додатків під Android та iOS.

7. ML.NET: Фреймворк для машинного навчання.

Переваги .NET

1. Кросплатформність: Можливість створення додатків для різних операційних систем.

2. Висока продуктивність: Оптимізація виконання коду та управління ресурсами.

3. Розширюваність та модульність: Підтримка пакунків NuGet для легкої інтеграції сторонніх бібліотек.

4. Підтримка різних мов програмування: Зокрема C#, F, VB.NET.

5. Безпека та стабільність: Засоби для захисту додатків та надійна обробка винятків.

Типовий Приклад Програми на C# в .NET

```csharp
using System;
using System.Threading.Tasks;

namespace HelloWorldApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            Console.WriteLine("Hello, World!");

            int result = await CalculateAsync();
            Console.WriteLine($"Result: {result}");
        }
        static async Task<int> CalculateAsync()
        {
            await Task.Delay(1000); // Симуляція асинхронної операції
```

```
            return 42;
        }
    }
}
```

Екосистема

1. Visual Studio: Потужне IDE для розробки на .NET.

2. Visual Studio Code: Легкий та кросплатформний редактор коду з підтримкою .NET.

3. NuGet: Система управління пакетами для .NET, яка дозволяє легко додавати сторонні бібліотеки в проекти.

.NET і C# - це потужна комбінація для створення різноманітного програмного забезпечення, від веб-додатків і мобільних додатків до десктопних і серверних рішень.

2.2. Entity Framework

Entity Framework (EF) — це об'єктно-реляційний маппер (ORM) для .NET, який дозволяє розробникам працювати з базами даних, використовуючи .NET об'єкти. Він позбавляє від необхідності писати більшість коду для доступу до даних, що спрощує та прискорює розробку застосунків.

Основні компоненти Entity Framework

1. Модель (Model):

- Концептуальна модель (CSDL): Модель даних, описана на концептуальному рівні з використанням сутностей та їхніх відносин.

- Модель сховища (SSDL): Модель даних, описана на рівні сховища, що представляє фізичну структуру бази даних.

- Модель мапування (MSL): Модель, що описує відповідність між концептуальною та фізичною моделлю.

2. DbContext:

- Основний клас для взаємодії з базою даних. Він управляє об'єктами сутностей, які витягуються з бази даних, та їхніми змінами.

3. Сутності (Entities):

- Класи, що представляють таблиці у базі даних. Кожен клас сутності відповідає таблиці, а його властивості відповідають стовпцям таблиці.

Основні концепції та можливості Entity Framework

1. Code First, Database First, Model First:

- Code First: Спочатку створюються класи у коді, а потім EF генерує схему бази даних.

- Database First: Спочатку створюється база даних, а потім EF генерує класи сутностей на основі існуючої бази даних.

- Model First: Спочатку створюється модель у дизайнері, а потім EF генерує схему бази даних та класи сутностей.

2. Мапування:

- Data Annotations: Атрибути у коді, що визначають мапування класів та властивостей до бази даних.

- Fluent API: Код, який дозволяє конфігурувати мапування більш детально, ніж за допомогою анотацій даних.

3. Запити LINQ (Language Integrated Query):

- EF дозволяє використовувати LINQ для написання запитів до бази даних у стилі C#. Це забезпечує безпечні типи та інтуїтивно зрозумілий спосіб написання запитів.

4. Зміни та збереження даних:

- Change Tracking: EF автоматично відслідковує зміни в об'єктах сутностей і дозволяє зберігати ці зміни в базі даних за допомогою методу `SaveChanges`.

5. Міграції (Migrations):

- Інструмент для керування змінами у схемі бази даних. Дозволяє створювати, застосовувати та відміняти зміни у схемі бази даних без втрати даних.

Приклад використання

```csharp
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
    public List<Post> Posts { get; set; }
}

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }
    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}

public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
}
```

Переваги використання Entity Framework:

- Продуктивність: Зменшує кількість коду для доступу до даних.

- Безпека: Використання LINQ забезпечує безпечний тип доступу до даних.

- Гнучкість: Підтримка різних підходів до розробки (Code First, Database First, Model First).

- Масштабованість: Підходить як для малих, так і для великих проєктів.

Entity Framework є потужним інструментом, що значно полегшує роботу з базами даних у .NET, забезпечуючи високу продуктивність та зручність розробки.

Entity Framework (EF) - це частина .NET, яка надає ORM (Object-Relational Mapping) для роботи з базами даних. EF дозволяє взаємодіяти з базою даних через об'єкти високого рівня, а не безпосередньо через SQL-запити. Завдяки цьому, можна зосередитися на логіці додатку, а система автоматично вирішує завдання зв'язку з базою даних.

Entity Framework підтримує різні провайдери баз даних, що означає, що його можна використовувати з різними СКБД, такими як Microsoft SQL Server, MySQL, PostgreSQL та іншими. Це дозволяє вибирати найбільш слушне для їх потреб джерело даних.

Entity Framework також має можливість використовувати мову LINQ (Language-Integrated Query) для написання запитів до бази даних. Це забезпечує читабельний та декларативний підхід до формулювання запитів, що полегшує роботу з даними та зменшує ймовірність помилок.

Загалом, Entity Framework дозволяє працювати з базою даних на вищому рівні абстракції, що сприяє збереженню часу та ефективній роботі з інформацією у .NET-додатках.

2.3. Windows Presentation Foundation

WPF (Windows Presentation Foundation) є сучасною технологією для створення графічних інтерфейсів користувача в десктопних додатках, спеціально призначеною для операційних систем Windows. Однією з її ключових особливостей є використання векторної графіки, що дозволяє створювати

інтерфейси, які адаптуються до різних роздільних здатностей екранів без втрати якості.

WPF — це графічна підсистема Windows для створення користувацьких інтерфейсів у додатках Windows. Вона входить до складу .NET Framework і .NET Core та дозволяє розробляти настільні додатки з багатою графікою, мультимедіа та складною композицією. Ось детальний опис основних аспектів WPF:

Основні особливості WPF:

1. Мова розмітки XAML:

- WPF використовує Extensible Application Markup Language (XAML) для опису візуального інтерфейсу.

- XAML дозволяє відокремити логіку бізнес-процесів (C# або VB.NET) від опису інтерфейсу, що сприяє кращому розподілу обов'язків і спрощує розробку та підтримку додатків.

2. Програмна модель:

- WPF базується на .NET, і її елементи можуть бути програмно створені та керовані за допомогою коду на C# або VB.NET.

- Програмна модель дозволяє легко створювати, змінювати та керувати елементами інтерфейсу.

3. Графічні можливості:

- Векторна графіка: В WPF використовуються векторні зображення, що дозволяє масштабувати графіку без втрати якості.

- 3D-графіка: WPF підтримує створення та маніпуляцію 3D-об'єктами, надаючи можливості для складних візуальних ефектів.

- Анімація: Широкий набір інструментів для створення анімацій та переходів, що дозволяє створювати плавні та інтерактивні інтерфейси.

4. Прив'язка даних (Data Binding):

- Підтримка двостороннього зв'язку даних, що дозволяє синхронізувати користувацький інтерфейс із джерелами даних. Це значно спрощує розробку динамічних додатків.

- Підтримка різних джерел даних, включаючи колекції, бази даних та веб-служби.

5. Шаблони та стилі:

- WPF підтримує шаблони (templates) для створення багаторазових стилів та контролів, що дозволяє легко змінювати зовнішній вигляд додатків.

- Стилі (styles) дозволяють централізовано визначати вигляд і поведінку елементів інтерфейсу, що сприяє узгодженому дизайну.

6. Мультимедіа:

- Підтримка мультимедіа, включаючи відео та аудіо, дозволяє створювати багатофункціональні додатки.

- Інтеграція з DirectX забезпечує високу продуктивність графічних обчислень.

7. Текст і типографіка:

- Підтримка складного тексту і типографіки, включаючи прозорість, перетворення та анімацію тексту.

8. Події та команди:

- Система подій (events) дозволяє обробляти користувацькі дії, такі як натискання кнопок або зміна тексту.

- Команди (commands) спрощують обробку введення користувача і сприяють кращій організації коду.

Переваги використання WPF:

- Гнучкість і потужність: Можливість створення складних і інтерактивних інтерфейсів.

- Відокремлення логіки і дизайну: Розробники і дизайнери можуть працювати незалежно один від одного.

- Повторне використання коду: Шаблони та стилі дозволяють використовувати одні й ті ж елементи в різних частинах додатка.

WPF є потужним інструментом для створення сучасних настільних додатків, надаючи розробникам велику кількість можливостей для реалізації

своїх ідей. Ще однією суттєвою перевагою є підтримка динамічних змін в інтерфейсі, таких як анімації та ефекти. Це допомагає легко додавати плавні переходи, виблискування та інші візуальні ефекти, покращуючи користувацький досвід.

Технологія також підтримує різноманітні шрифти та стилі, що робить можливим створення стильних інтерфейсів, адаптованих до корпоративного бренду або дизайнерських концепцій.

Узагальнюючи, завдяки WPF з’являється можливість створювати не лише функціонально ефективні, але й естетично привабливі додатки для операційних систем Windows, забезпечуючи високий рівень користувацької зручності та задоволення від використання програмного продукту.

2.4. PostgreSQL

PostgreSQL — це потужна, відкрита об'єктно-реляційна система управління базами даних (СУБД), яка широко використовується для зберігання, управління і обробки даних. Вона відома своєю надійністю, високою продуктивністю і підтримкою стандарту SQL. Ось детальний опис основних характеристик PostgreSQL:

Основні характеристики PostgreSQL:

1. Відкрите програмне забезпечення:

- PostgreSQL є безкоштовним та має відкритий вихідний код, що дозволяє користувачам змінювати та поширювати його без обмежень.

- Активна спільнота розробників постійно працює над покращенням і додаванням нових функцій.

2. Підтримка SQL:

- PostgreSQL повністю відповідає стандартам SQL, включаючи підтримку основних і розширених можливостей мови запитів.

- Підтримка складних запитів, підзапитів, агрегатних функцій і користувацьких функцій.

3. Розширюваність:

- Можливість створення нових типів даних, операторів, функцій, індексів та мов програмування, що дозволяє розширювати функціональність бази даних відповідно до специфічних потреб.

- Підтримка розширень (extensions), таких як PostGIS для геопросторових даних, яка додає додаткову функціональність.

4. Типи даних:

- Широкий набір вбудованих типів даних, включаючи числові, символьні, часові, булеві та інші.

- Підтримка складних типів даних, таких як масиви, JSON/JSONB, XML, hstore, і спеціалізовані типи, такі як UUID і IP-адреси.

5. Транзакції і цілісність даних:

- Підтримка ACID-транзакцій (Atomicity, Consistency, Isolation, Durability) забезпечує надійність і цілісність даних.

- Механізм багатовіртуального доступу (MVCC) дозволяє одночасно читати і змінювати дані без блокувань.

6. Індекси:

- Підтримка різних типів індексів, таких як B-tree, Hash, GiST, SP-GiST, GIN і BRIN, що дозволяє оптимізувати запити.

- Можливість створення користувацьких індексів і підтримка паралельного індексування для покращення продуктивності.

7. Висока доступність і реплікація:

- Вбудована підтримка стрімінгової реплікації і логічної реплікації, що дозволяє створювати високопродуктивні і відмовостійкі системи.

- Підтримка асинхронної, синхронної і каскадної реплікації.

8. Безпека:

- Підтримка аутентифікації користувачів за допомогою методів LDAP, SCRAM-SHA-256, GSSAPI, SSPI, і сертифікатів SSL.

- Гнучка система управління правами доступу на рівні користувачів, ролей і об'єктів бази даних.

- Підтримка шифрування з'єднань за допомогою SSL/TLS.

9. Розподілені обчислення і шардінг:

Розподілені обчислення і шардінг в PostgreSQL забезпечують можливість горизонтального масштабування бази даних для обробки великих обсягів даних і підвищення продуктивності. Розглянемо ці концепції в деталях.

Розподілені обчислення

Розподілені обчислення в PostgreSQL дозволяють виконувати запити і обробляти дані, розподілені між декількома серверами або вузлами. Це дозволяє забезпечити більшу обчислювальну потужність і швидкість обробки даних.

Основні аспекти розподілених обчислень:

1. Федеративні системи:

- Використання Foreign Data Wrappers (FDW) для доступу до зовнішніх баз даних і таблиць, що дозволяє виконувати запити до різних джерел даних.

- Системи, такі як PostgreSQL FDW, дозволяють інтегрувати PostgreSQL з іншими базами даних, включаючи інші екземпляри PostgreSQL, MySQL, Oracle і інші.

2. Паралельне виконання запитів:

- Підтримка паралельного виконання запитів, що дозволяє розподіляти обчислення між декількома процесорами або ядрами для підвищення продуктивності.

- Паралельні індексні скани, паралельні об’єднання (joins) та агрегації забезпечують швидшу обробку складних запитів.

3. Розширення для розподілених обчислень:

- Citus: Розширення для PostgreSQL, яке перетворює базу даних у розподілену систему, дозволяючи горизонтально масштабувати таблиці шляхом шардінгу.

- Postgres-XL: Мульти-майстер кластер з горизонтальним масштабуванням, який дозволяє виконувати запити на декількох вузлах одночасно.

Шардінг

Шардінг (sharding) — це техніка горизонтального поділу даних, при якій великі таблиці розділяються на менші, керовані сегменти (шарди), які розподіляються між різними вузлами.

Основні аспекти шардінгу:

1. Методи шардінгу:

- Розділ на основі діапазону (Range-based Sharding): Дані розподіляються на основі діапазонів значень ключа. Наприклад, записи з датами до 2020 року зберігаються на одному шарді, а записи після 2020 року — на іншому.

- Хешування (Hash-based Sharding): Дані розподіляються на основі хеш-функції, яка обчислює хеш ключа і розподіляє дані між шардами рівномірно.

- Список (List-based Sharding): Дані розподіляються на основі переліку значень ключа. Наприклад, записи з певними категоріями зберігаються на окремих шардах.

2. Переваги шардінгу:

- Горизонтальне масштабування: Додаючи нові вузли, можна легко збільшувати обсяг даних і потужність обробки.

- Підвищена доступність: Дані розподіляються між кількома вузлами, що зменшує ризик повної відмови системи при виході з ладу одного з вузлів.

- Оптимізація продуктивності: Запити можуть бути розподілені між різними вузлами, що знижує навантаження на кожен окремий вузол і підвищує швидкість обробки.

3. Приклади використання шардінгу:

- Citus: Це розширення автоматизує шардінг і надає інструменти для управління розподіленими таблицями. Citus використовує хешування для розподілу даних і підтримує паралельне виконання запитів.

- Postgres-XL: Здійснює горизонтальний шардінг даних і підтримує паралельні запити, забезпечуючи масштабованість і продуктивність.

Використання Citus для шардінгу в PostgreSQL

Citus — це одне з найбільш популярних розширень для шардінгу в PostgreSQL. Ось як Citus працює:

1. Розподіл даних:

- Citus використовує хешування або розподіл на основі діапазону для розподілу даних між нодами.

- Кожна таблиця може бути розбита на кілька шардів, які розподіляються між різними вузлами кластера.

2. Управління шардами:

- Адміністрування кластерів Citus здійснюється за допомогою команд SQL, що дозволяє легко додавати нові вузли, переміщати дані і балансувати навантаження.

- Citus підтримує автоматичне балансування шардів для рівномірного розподілу даних і навантаження.

3. Виконання запитів:

- Citus дозволяє виконувати як глобальні (які обробляються на всіх шардах), так і локальні (які обробляються на одному шарді) запити.

- Оптимізатор Citus автоматично розподіляє запити між вузлами для максимальної продуктивності.

Використання розподілених обчислень і шардінгу в PostgreSQL дозволяє створювати масштабовані, високопродуктивні та відмовостійкі бази даних, здатні обробляти великі обсяги даних і складні запити.

10. Інструменти і інтерфейси:

- Підтримка різних інструментів і інтерфейсів для управління базами даних, таких як pgAdmin, psql та інші.

- Інтерфейси для програмування на різних мовах, включаючи C/C++, Python, Java, PHP, Ruby, Perl, і багато інших.

11. Підтримка повнотекстового пошуку:

- Вбудовані можливості повнотекстового пошуку, що дозволяють ефективно виконувати запити по текстових даних.

- Підтримка різних мов і морфологічного аналізу для покращення пошукових запитів.

Переваги використання PostgreSQL:

- Надійність: Високий рівень надійності та цілісності даних.

- Гнучкість: Можливість налаштування і розширення під специфічні потреби користувача.

- Висока продуктивність: Підтримка паралельних запитів, індексації і оптимізації виконання запитів.

- Спільнота: Активна спільнота, яка сприяє розвитку, підтримці і обміну знаннями.

PostgreSQL є потужним і гнучким інструментом для роботи з базами даних, що задовольняє вимоги як малих проектів, так і великих корпоративних систем.

Загальна схема застосунку:

РОЗДІЛ 3.
ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1. Розробка структури бази даних.

На рисунку нижче подана структура бази даних з чотирма таблицями "товари", "склади", "оператори" і "адміністратори":

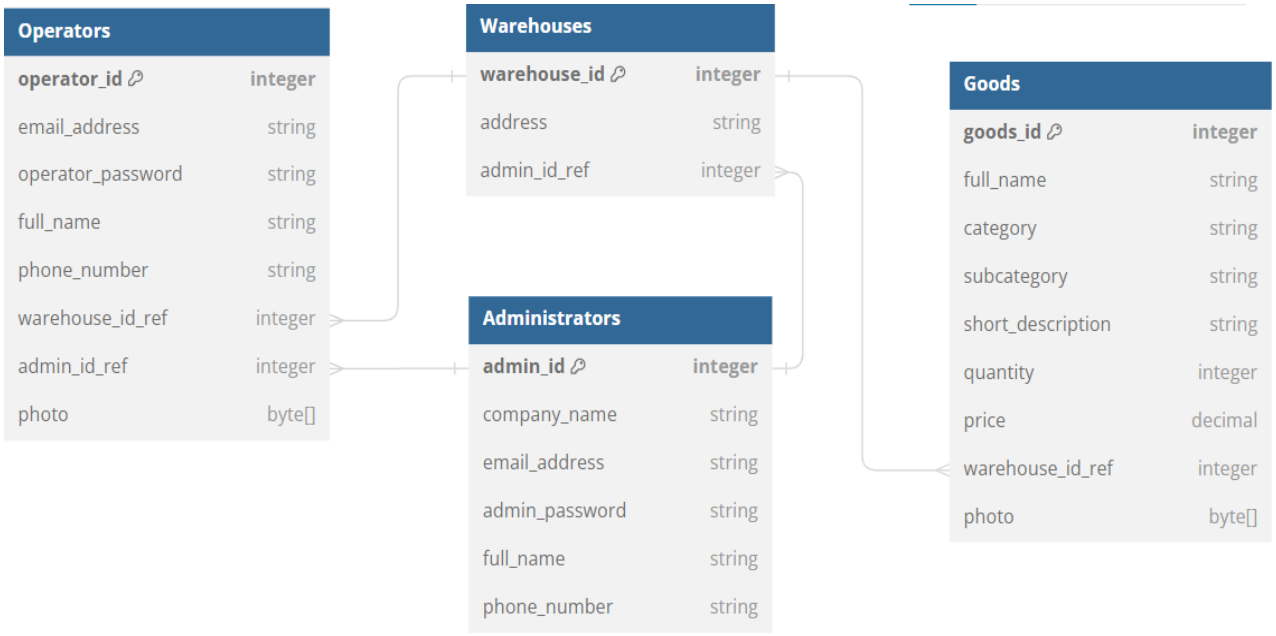

Рис. 1. Схема бази даних

Наведемо опис сутностей цих таблиць.

Таблиця	Колонка	Опис
Goods (Товари)	Goods_id	(PRIMARY KEY): унікальний ідентифікатор товару
	Full_name	назва товару
	Category	категорія товару
	Subcategory	підкатегорія товару
	shortDescription	короткий опис
	Price	ціна товару
	Quantity	кількість товару на складі
	Photo	фото
	Warehouse_id_ref	посилання на склад
Warehouses (Склади)	Warehouse_id	(PRIMARY KEY): унікальний ідентифікатор складу
	Admin_id_ref	Посилання на адміна складу

	address	місцезнаходження складу
Operators (Оператори)	Operator_id	PRIMARY KEY): унікальний ідентифікатор оператора
	Full_name	ім'я оператора
	Email_address	Ел.пошта оператора
	Warehouse_id_ref	(FOREIGN KEY): ідентифікатор складу, на якому працює оператор
	Operator_password	пароль оператора
	photo	фото
	Admin_id_ref	Посилання на адміна складу
	Phone_number	телефон
Admins (Адміністратори)	Admin_id	
	Full_name	
	E_mail_address	
	password	
	Phone_number	телефон

У даній структурі кожна таблиця має свій унікальний ідентифікатор (PRIMARY KEY), що дозволяє однозначно ідентифікувати записи. Також використовуються зовнішні ключі (FOREIGN KEY), щоб встановити зв'язок між таблицями, наприклад, оператор працює на певному складі.

3.2. Проектування інтерфейсу користувача.

При запуску програми відкривається вікно автентифікації. В полях можна ввести дані та залогуватись, якщо вже існує профіль користувача, а також перейти на сторінку реєстрації для створення власного Інвентаріуму.

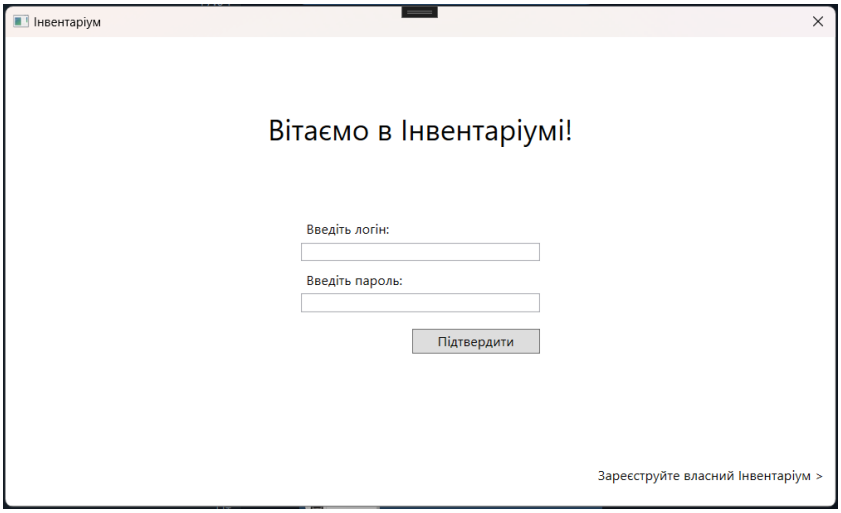

Рис.3.2. Вікно автентифікації

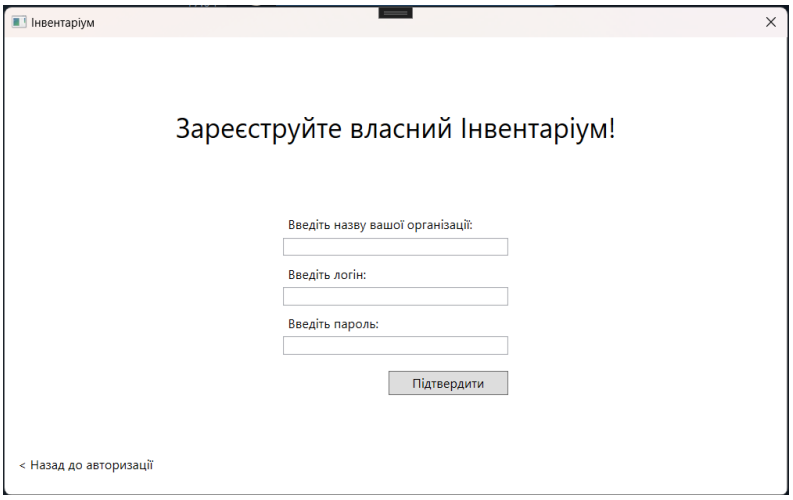

Рис.3.3. Вікно реєстрації власного застосунку для підприємства

У даному застосунку є два типи користувачів: адміністратори та оператори. Спочатку розглянемо функціонал для адміністратора.

Повернувшись на сторінку автентифікації та увійшовши в свій профіль, користувач опиняється на головній сторінці з товарами.

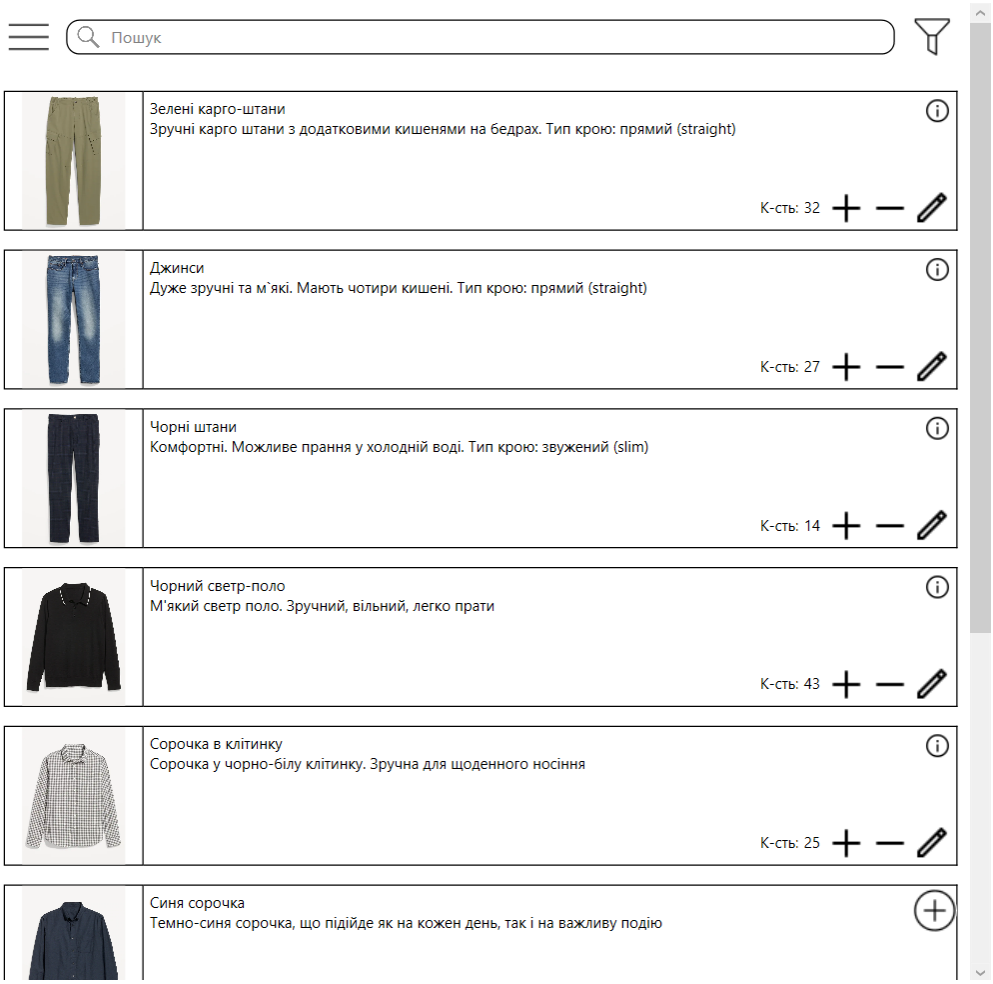

Рис. 3.4. Вікно Головна сторінка з товарами.

3.3. Визначення функціональних модулів програми.

Оскільки користувач є адміністратором, то в нього є можливість додавати, редагувати та видаляти товари, змінювати їхню кількість. Для створення потрібно натиснути на плюсик в правому нижньому куті вікна. Після цього відкривається модальне вікно для створення самого товару.

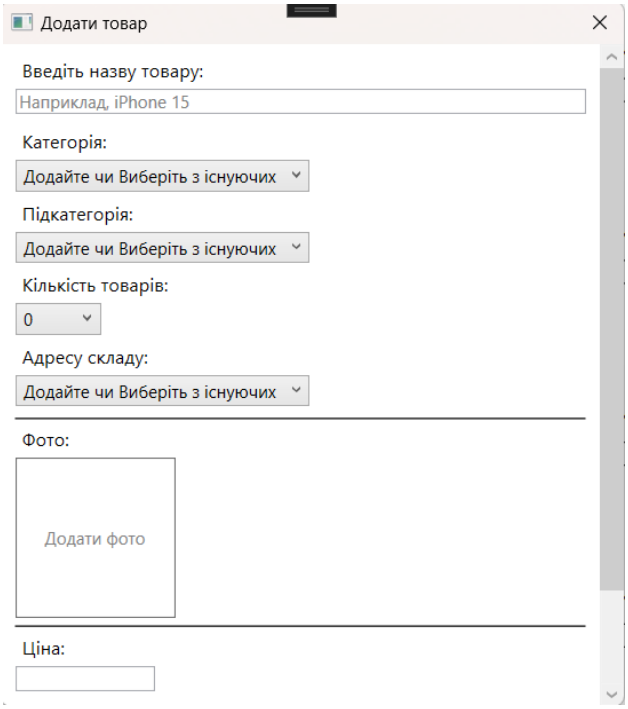

Рис. 3.5. Вікно додавання товару

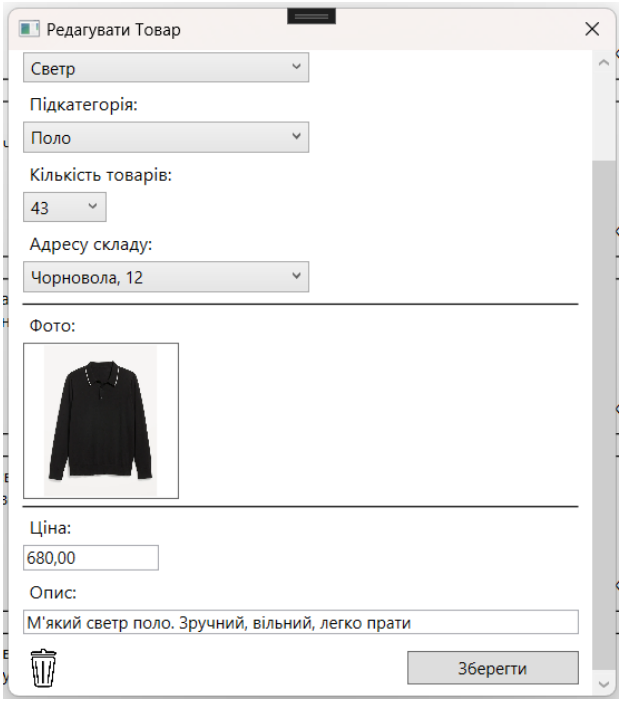

Рис. 3.6. Вікно редагування товару

Після введення необхідних даних можна зберегти товар натиснувши на кнопку Зберегти.

При натисканні на кнопку в лівому верхньому куті відкривається меню.

Рис.3.7. Вікно відкриття меню

Перше - це домашня сторінка, тобто сторінка з товарами. Друга - це сторінка профілю адміністратора. Третя - це сторінка для керування операторами та іншими адміністраторами.

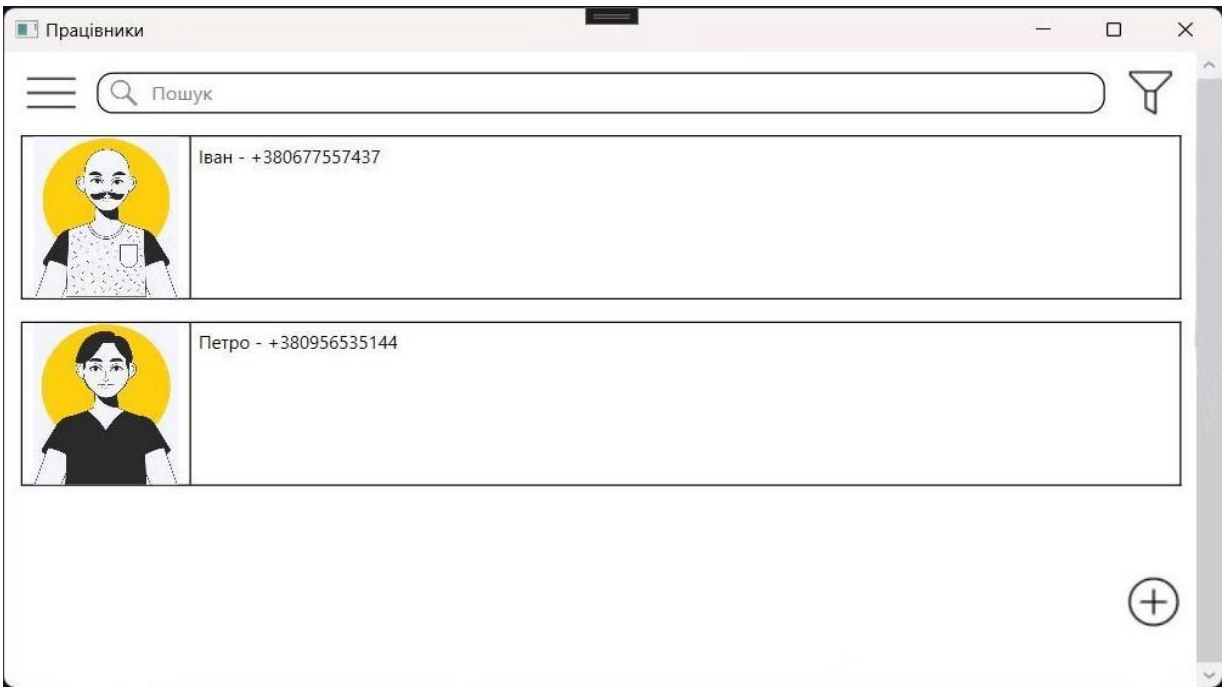

Рис.3.8. Вікно працівників

На даний момент, працівників лише два. Щоб додати інших потрібно натиснути на плюсик в правому нижньому кутку

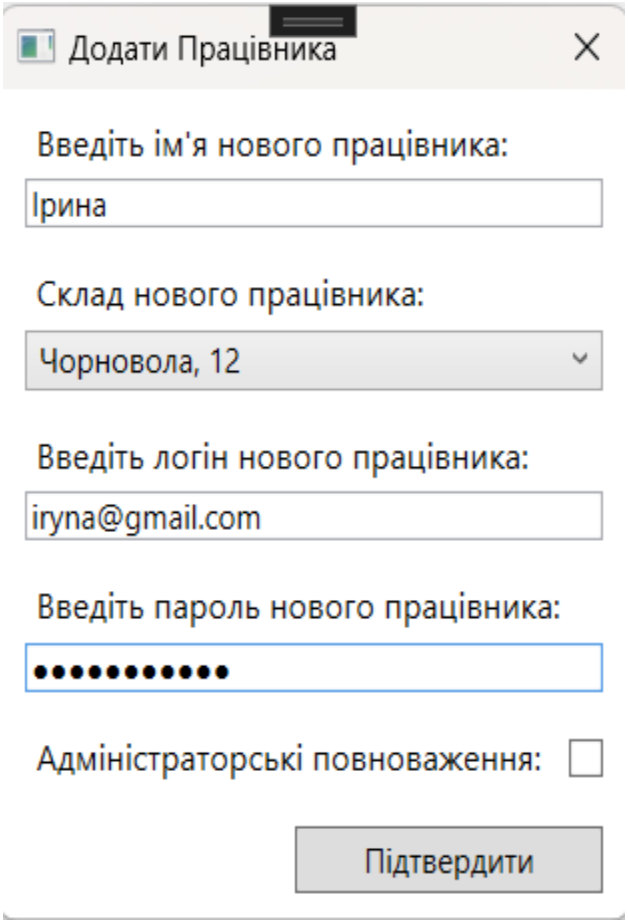

Рис.3.9. Вікно створення працівника

Ввівши всі потрібні дані, адміністратор може створити нового як оператора, так і адміністратора.

Тепер розглянемо функціонал для оператора. Знову ж таки проходимо автентифікацію. Домашня сторінка містить тільки ті товари, які знаходяться на тому ж складі, що і оператор. Оператор, як і адміністратор може змінювати кількість певного товару, але вже не може ні створювати, ні редагувати, ні видаляти їх.

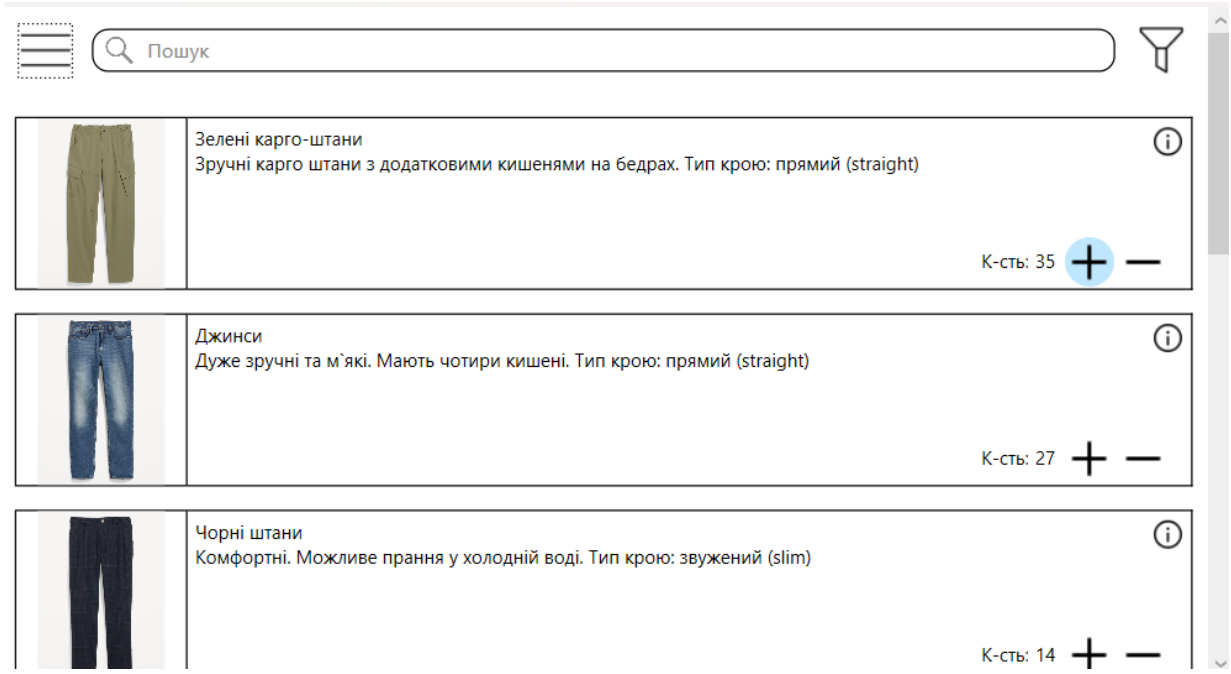

Рис.3.10. Вікно відповідальності працівника

А ще в оператора немає можливості переглядати інформацію щодо інших користувачів/працівників.

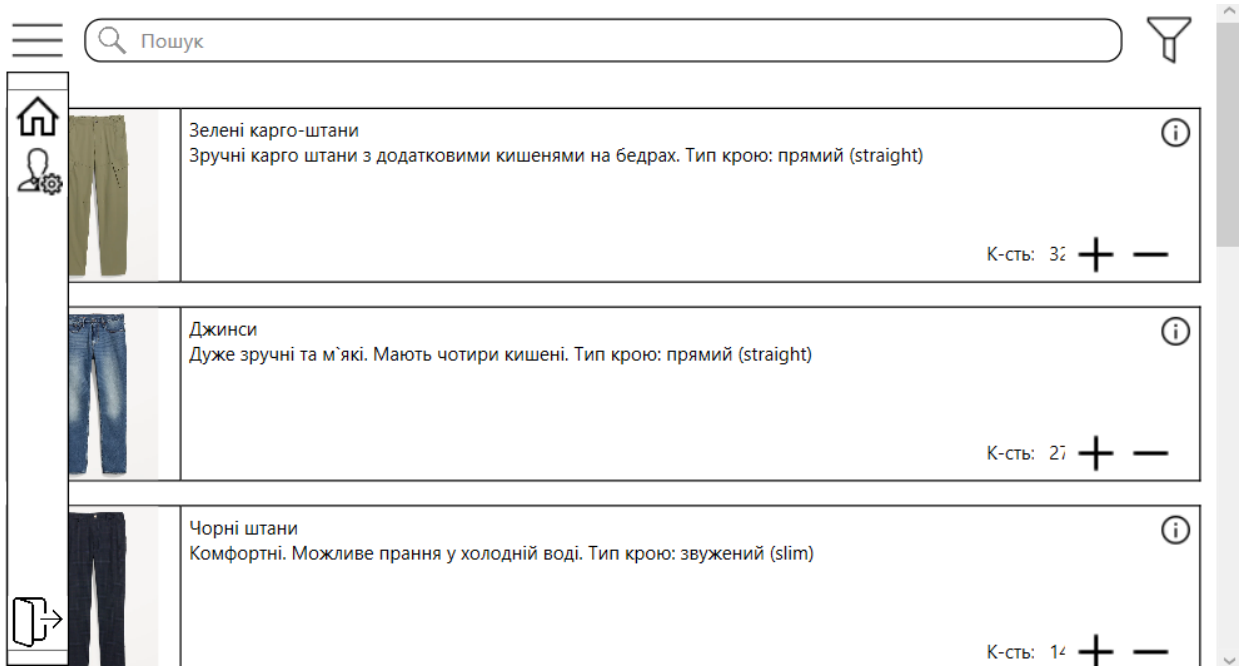

Рис.3.11. Вікно відповідальності працівника з меню

Він може редагувати свою інформацію, тобто змінювати номер телефону, фото та інші дані.

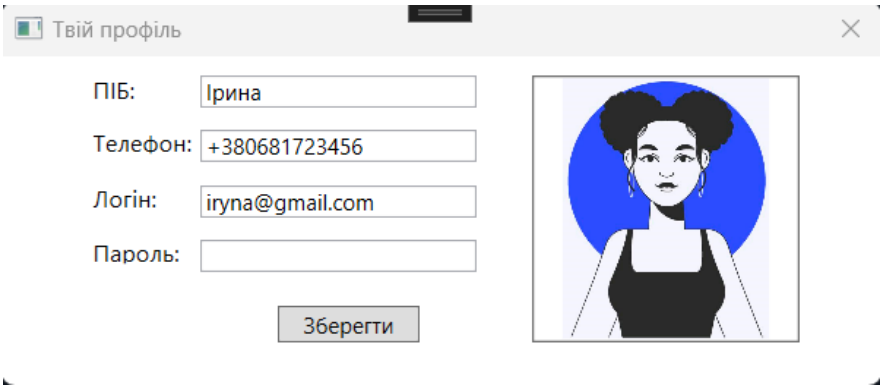

Рис.3.12. Вікно властивостей працівника

Повертаючись до головної сторінки, у користувача будь-якого типу є можливість сортувати товари за назвою та кількістю, натиснувши на кнопку фільтра, зображену у правому верхньому куті.

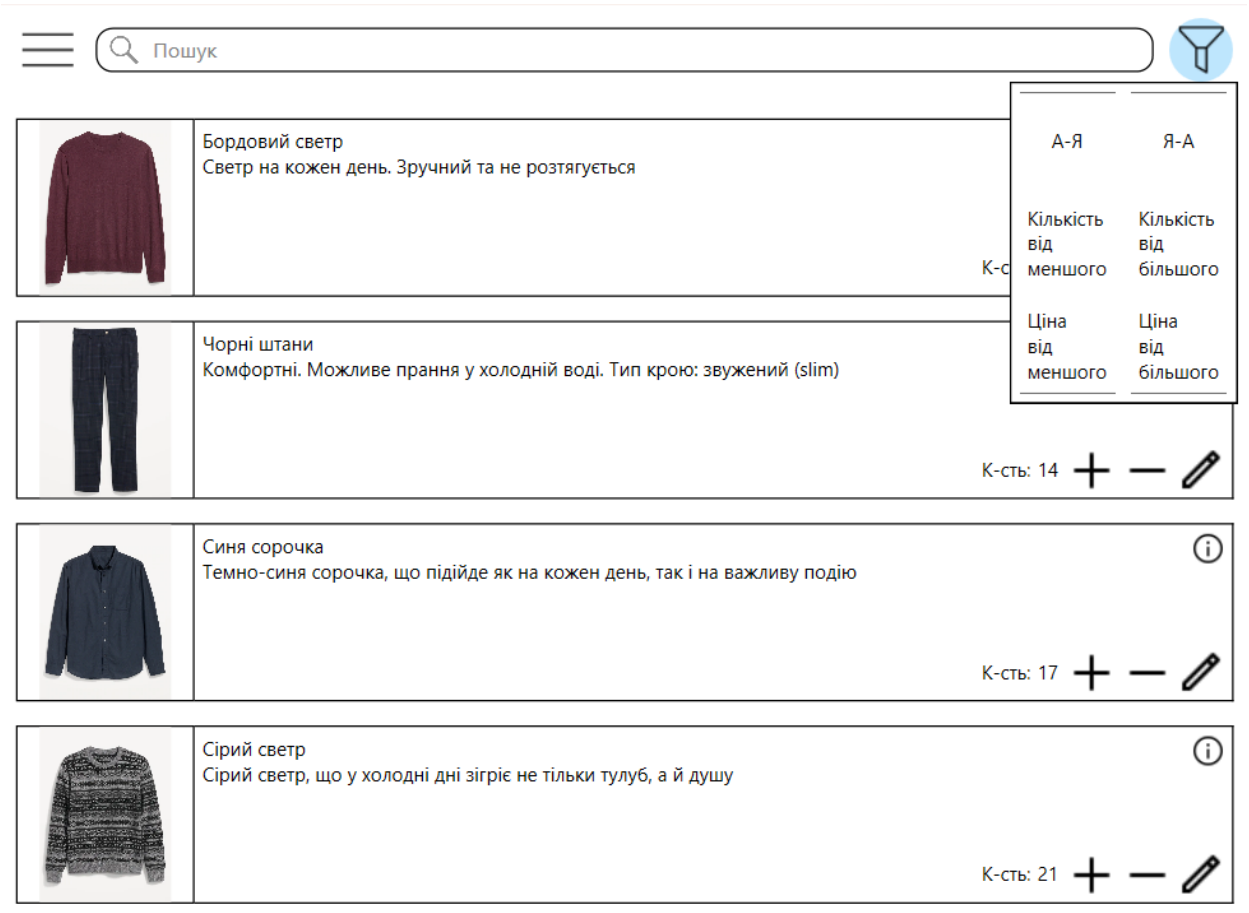

Рис.3.13. Прилад вікна сортування товарів

А також шукати товари в полі введення над списком товарів:

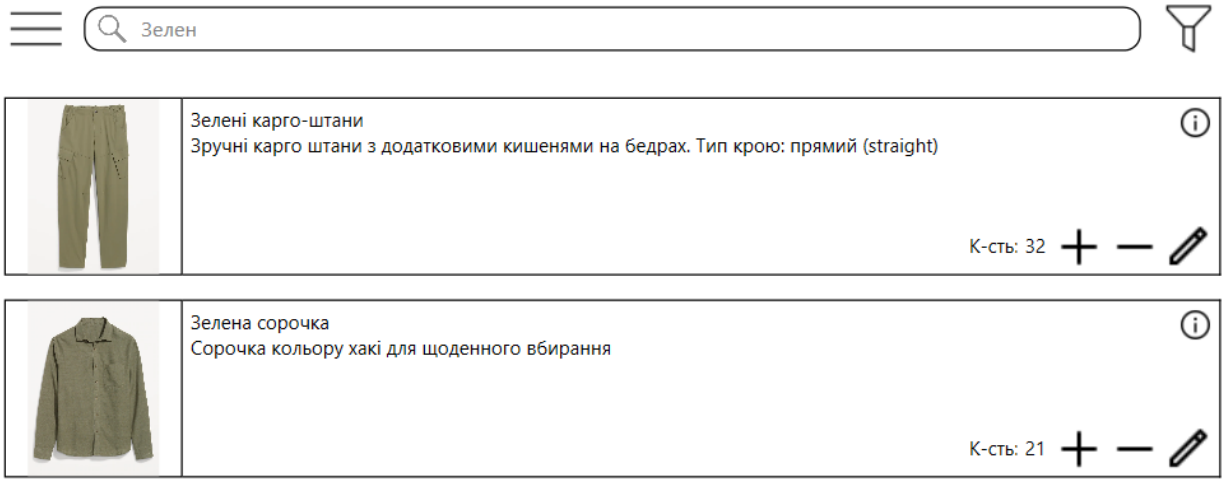

Рис.3.14. Прилад вікна пошуку товарів

Крім того, для кожного товару генерується власний QR-код, що містить всю необхідну інформацію. Створений QR-код можна зчитати за допомогою сторонньої програми під назвою QR Code & Barcode Scanner, що можна легко завантажити на будь-який мобільний пристрій.

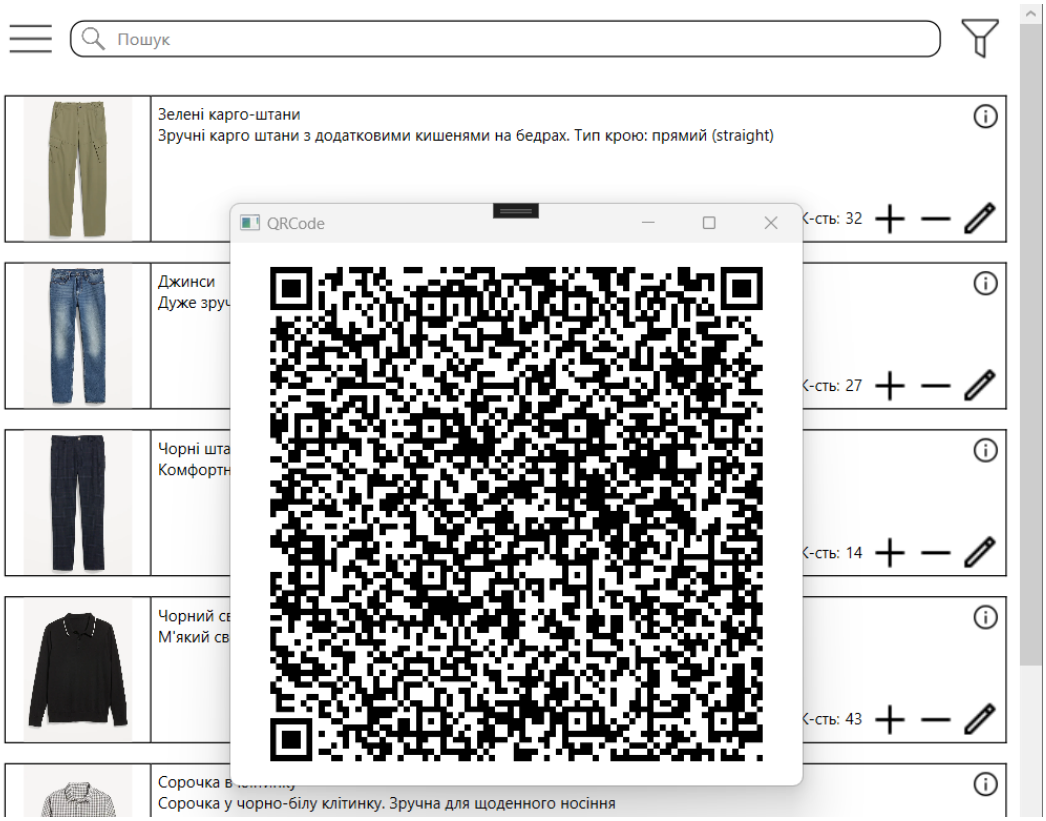

Рис.3.15. Прилад вікна з QR кодом для товару товарів

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Структурно-функціональний аналіз технологічного процесу

Розробка та вживання ефективних заходів запобігання аварійним і травмонебезпечних ситуаціям можливі лише при завчасному виявлення тих небезпек, з яких починаються процеси їх формування. Оскільки небезпечні умови не завжди завчасно можна виявити, а для вивчення небезпечних дій іноді потрібно багато часу, щоб зібрати статичний матеріал, то і методи виявлення цих небезпек повинні бути відповідно диференційовані (табл. 4.1)

Таблиця 4.1. Моделі формування та виникнення травмонебезпечних і аварійних ситуацій

Вид робіт, виробничий підрозділ, робоче місце, виробниче обладнання, склад агрегату	Виробнича небезпека			Можливі наслідки	Заходи запобігання небезпечним ситуаціям
	Небезпечна умова (НУ)	Небезпечна дія (НД)	Небезпечна ситуація (НС)		
Виконання робіт із електрообладнанням	Не вимкнено живлення. Відсутність заземлення.	Нехтування правилами ТБ	Ураження струмом	Травма (Т)	Проведення повторного інструктажу з ТБ. Розробка нових способів захисту. Встановлення заземлення.
НД ↓ НУ → НС → Т					

Відповідно до аналізу небезпечних умов, які існують у виробничому процесі виокремлено такі наступні за характером дії на працівника їх групи:

– характеризують стан або рівень небезпеки обладнання, які використовуються.

– сприяють виникненню технологічних помилок обслуговуючого персоналу впродовж виробничого процесу;

– створювати умови та можливість проникнення працівника в небезпечну зону;

– приводять до виникнення небезпечних дій (внаслідок низького рівня професійної підготовки працівників та організації навчання з охорони праці).

Моделі формування та виникнення травмонебезпечних і аварійних ситуацій в комп'ютерному кабінеті представлено у вигляді моделі формування та виникнення травмонебезпечних і аварійних ситуацій – табл. 4.1.

4.2. Розрахунок освітлення приміщення комп'ютерного кабінету

Освітленість виробничих приміщень може бути штучною і природною. Природне освітлення при правильному обладнанні найбільш сприятливе для людини. Основні вимоги для освітлення наступні:

– освітлення повинне бути достатнім для швидкого і легкого розпізнання об'єктів роботи;

– освітлення повинно бути рівномірне без різких тіней;

– джерело світла не повинно осліплювати працівника;

– рівень освітленості не повинен обмежуватись часом.

Природне освітлення забезпечується обладнанням вікон (бокове освітлення) фонарів і світильних покрить приміщень (верхнє освітлення). Природне освітлення нормується коефіцієнтом природної освітленості. Коефіцієнт природної освітленості – це процентне відношення фактичної освітленості $Fв$ в будь-якій точці приміщення до освітленості $Fн$ розсіяної

світлом небозводу точки, яка лежить на відкритій місцевості. Розрахунок природного освітлення через бокові вікна по нормам освітленості ведеться для самої дальньої від вікон точки, тобто знаходять мінімальне значення стик коефіцієнта природної освітленості:

$$e_{\min} = \frac{Fb}{Fн} \cdot 100. \qquad (4.1)$$

Значення коефіцієнта природної освітленості визначається не менше чим в п'яти точках. Значення коефіцієнта природної освітленості для сільськогосподарських виробничих приміщень в даному випадку ремонтній майстерні, беремо $e_{min} = 5\%$.

Розрахунок природного освітлення зводиться до визначення площі світлових променів.

Сумарну площу світлових променів $\sum F_o \, (м^2)$ по коефіцієнту природної освітленості для бокових променів визначаємо по формулі:

$$\sum F_o = \frac{F_н \cdot e_{\min} \cdot r_o \cdot K}{100 \cdot \tau \cdot Г_1}, \qquad (4.2)$$

де $F_н$ – площа підлоги, м^2; e_{min} – величина мінімального коефіцієнта природного освітленості; τ – загальний коефіцієнт світловикористання віконного отвору із врахуванням його забруднення, τ = 0,25; r_o – світлова характеристика вікна, r_o – 9,5; $Г_1$ – коефіцієнт, який враховує підвищення освітленості за рахунок світла, яке відбивається від стін і стелі, $Г_1$ = 1,2; K – коефіцієнт, який враховує затінення вікон сусідніми приміщеннями і загорожею, K = 1.

$$\sum F_o = \frac{36 \cdot 0,5 \cdot 9,5 \cdot 1}{100 \cdot 0,25 \cdot 1,2} = 5,7 м^2.$$

Кількість світлових променів визначимо:

$$N = \frac{\sum F_o}{F_o}, \tag{4.3}$$

де F_o – площа вікна згідно стандарту, м^2.

$$N = \frac{5,7}{6} = 0,95.$$

Приймаємо кількість вікон – одне вікно.

При розрахунку природного освітлення найбільш поширеним і простим є метод світлового потоку. При цьому методі розраховуємо світловий потік $F_л$ (Лк), який повинна випромінювати кожна лампа (при заданій кількості ламп).

$$F_л = \frac{k \cdot S_п \cdot E}{n_л \cdot \eta \cdot r^2}, \tag{4.4}$$

де k – коефіцієнт запасу, $k = 1,3$; $S_п$ – площа підлоги, м^2; $S_п = 36$ м^2. E – нормативна освітленість, Е = 300 Лк; $n_л$ – кількість встановлених ламп, $n_л$=6 од; η – коефіцієнт використання світлового потоку, $\eta = 0,25$; r – коефіцієнт нерівномірності освітленості, $r = 0,545$.

Коефіцієнт запасу (K) враховує можливість забруднення світильників пилом, що залежить від характеру виробництва.

Розрахунок штучного освітлення починаємо із визначення висоти розташування світильника і їх кількості. Висоту $h_н$ (м) розташування світильників над робочим місцем знаходимо за формулою:

$$h_н = H - (h_1 + h_2), \tag{4.5}$$

де H – висота приміщення, м; h_1 – віддаль від підлоги до освітлювальної поверхні, м; h_2 – віддаль від стелі до світильника, м.

$$h_н = 4,5 - (2,2 + 1,5) = 0,8 \text{ м}.$$

При симетричному розміщенні світильників по вершинах квадратів їх кількість визначається за формулою:

$$n_c = \frac{S_n}{l^2}, \qquad (4.6)$$

де l – віддаль між світильниками, м.

Підставивши значення отримаємо:

$$n_c = \frac{36}{9} = 4 \text{ од.}$$

Тоді світловий потік буде становити

$$F_л = \frac{1,3 \cdot 36 \cdot 300}{4 \cdot 0,25 \cdot 0,545} = 2576,2 \text{ Лк.}$$

При світловому потоці 2576,2 Лк для заданої лампи вибираємо тип і потужність.

Вибираємо тип лампи – люмінесцентну, потужністю 40Вт.

4.3. Безпека в надзвичайних ситуаціях

Забезпечення захисту населення і території у разі загрози і виникнення надзвичайних ситуацій є одним з най важливіших завдань держави.

Захист населення є системою загально державних заходів, які реалізуються центральними і місцевими органами виконавчої влади, виконавчими органами влад, органами управління з питань надзвичайних ситуацій та цивільного захисту населення, підпорядкованими їм системами, та підприємств, що забезпечують виконання організаційних, інженерно – технічних, санітарно – гігієнічних, проти епідемічних та інших заходів у сфері запобігання та ліквідації наслідків надзвичайних ситуацій.

Загрози життєво важливих інтересів громадян, держави, суспільства поділяють на зовнішні та внутрішні, виконують під час надзвичайних ситуацій техногенного та природного характеру та воєнних конфліктах.

Принципи захисту випливають з основних положень Женевської конвенції щодо захисту жертв війни та додаткових протоколів до неї, можливого характеру воєнних дій, реальних можливостей держави щодо створення матеріальної бази захисту. З метою захисту населення, зменшення втрат та шкоди економіці в разі виникнення надзвичайних ситуацій має право проводитись спеціальний комплекс заходів.

Оповіщення та інформування, яке досягається завчасним створенням і підтримкою в постійній готовності загально державної,територіальних та об’єктивних систем оповіщення населення.

ВИСНОВКИ

Проведений аналіз поточних методів управління товарними запасами у різних підприємців підтвердив необхідність впровадження програмного забезпечення для автоматизації обліку та контролю за запасами."

Розробка програмного ресурсу виявилася критичною для оптимізації процесів управління запасами, забезпечення точності обліку та зменшення втрат.

Впровадження розробленого програмного забезпечення дозволить значно покращити ефективність управління запасами, зменшити час на їх облік та оптимізувати процеси замовлення та постачання."

В результаті впровадження програмного продукту має зрости продуктивність праці, зниження витрат на управління запасами та підвищення конкурентоспроможності конкретного підприємця.

Для подальшого розвитку системи рекомендується розширення функціональності, включення додаткових засобів аналізу даних та вдосконалення інтерфейсу користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adam Freeman. Pro Entity Framework Core 2 for ASP.NET Core MVC. Apress, 2018. 656p.
2. Alex Khang. Professional WPF and C# Programming: Practical Software Development Using WPF and C#. Kindle Edition, 2019. 405p
3. C# Guide - .NET managed language. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 14.04.2024).
4. dotnet/docs. URL: https://github.com/dotnet/docs (дата звернення: 8.04.2024).
5. Luca Ferrari, Enrico Pirozzi. Learn PostgreSQL: Use, manage and build secure and scalable databases with PostgreSQL 16. 2nd ed. Packt Publishing, 2023. 744p.
6. Programming Concepts (C#). URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/ (дата звернення: 7.04.2024).
7. Rahul Rajat Singh. Mastering Entity Framework. Packt Publishing, 2015. 304p.
8. Ricardo Peres. Entity Framework Core Cookbook – Second Edition, 2016. 324p.
9. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. Fourth Edition. The MIT Press, 2022. 1322p.
10. Windows Presentation Foundation for .NET 8 documentation. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view= netdesktop-8.0 (дата звернення: 18.04.2024).
11. Джон Шрайбфедер. Ефективне управління запасами. Альпіна Бізнес Букс, 2016. 304 с.
12. Крикавський Є., Похильченко О., Фертч М. Логістика та управління ланцюжками поставок. Львів : Вид-во НУ Львівська політехніка, 2019. 848с.
13. Стюарт Емметт, Мистецтво управління складом. Як зменшити витрати та підвищити ефективність, Фоліо, Харків, 2007. 320с.
14. Хом'як Р.Л., Лемешівський В.І. Бухгалтерський облік в Україні : навч. посібн. Вид. 2-ге, [перероб. та доп.]. Львів : Вид-во НУ "Львівська політехніка", "Інтелект-Захід", 2008. 1224 с.
15. .NET Framework documentation. URL: https://learn.microsoft.com/en-us/dotnet/framework/ (дата звернення: 20.04.2024).

ДОДАТКИ

Додаток А

Створення вікна для товарів

```
using BusinessLogic;
using DB;
using DB_Setup;
using Inventory_Context;
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;

namespace Wpf_Inventarium
{
    /// <summary>
    /// Interaction logic for EditGoodsWindow.xaml
    /// </summary>
    public partial class EditGoodsWindow : Window
    {
        AdministratorRepository admin_repo = new AdministratorRepository();
        WarehouseRepository warehouse_repo = new WarehouseRepository();
        GoodsRepository goods_repo = new GoodsRepository();
        List<string> subcategories;
        string previousCategory;

        public MainWindowAdmin ParentMainWindowAdmin { get; set; }

        private Goods goods_to_edit;

        public EditGoodsWindow(Goods goods_to_edit_)
        {
            GoodsService goods_service = new GoodsService(goods_repo);
            AdministratorService admin_service = new
AdministratorService(admin_repo);
            WarehouseService warehouse_service = new
WarehouseService(warehouse_repo);

            goods_to_edit = goods_to_edit_;
            previousCategory = goods_to_edit_.category;
            InitializeComponent();

            ComboBoxAddress.SelectedItem =
warehouse_service.GetWarehouseById(goods_to_edit_.warehouse_id_ref).addres;
            ComboBoxCategory.SelectedItem = goods_to_edit_.category;
            ComboBoxSubcategory.SelectedItem = goods_to_edit_.subcategory;

ComboBoxCountGoods.Items.Add(goods_to_edit_.quantity.ToString());
            ComboBoxCountGoods.SelectedItem =
goods_to_edit_.quantity.ToString();
            TextBoxGoodsName.Text = goods_to_edit_.full_name;
            TextBoxDescription.Text = goods_to_edit_.short_description;
            TextBoxPrice.Text = goods_to_edit_.price.ToString();
            byte[] photoBytes = goods_to_edit.photo;
            BitmapImage bitmapImage =
ImageFormatter.ByteArrayToBitmapImage(photoBytes);
```

```
                Image photo = new Image();
                photo.Source = bitmapImage;

                AddPhoto.Content = photo;
                foreach (string category in
goods_service.GetCategoriesForAdministrator(admin_service.GetAdministratorByEmai
l(MainWindow.username).admin_id))
                {
                    ComboBoxCategory.Items.Add(category);
                }

                ComboBoxCategory.DisplayMemberPath = ".";
                foreach (Warehouse warehouse in
warehouse_service.GetWarehousesForAdministrator(admin_service.GetAdministratorBy
Email(MainWindow.username).admin_id))
                {
                    ComboBoxAddress.Items.Add(warehouse.addres);
                }

                ComboBoxAddress.DisplayMemberPath = ".";
                this.Closed += EditWindow_Closed;
            }

            private void EditWindow_Closed(object sender, EventArgs e)
            {
                this.Close();
            }

            private string GetSelectedItemComboBoxCountGoods()
            {
                return
ComboBoxCountGoods.SelectedItem.ToString().Replace("System.Windows.Controls.Comb
oBoxItem: ", "");
            }

            private string GetSelectedItemComboBoxAddress()
            {
                return
ComboBoxAddress.SelectedItem.ToString().Replace("System.Windows.Controls.ComboBo
xItem: ", "");
            }

            private string GetSelectedItemComboBoxCategory()
            {
                return
ComboBoxCategory.SelectedItem.ToString().Replace("System.Windows.Controls.ComboB
oxItem: ", "");
            }

            private string GetSelectedItemComboBoxSubcategory()
            {
                return
ComboBoxSubcategory.SelectedItem.ToString().Replace("System.Windows.Controls.Com
boBoxItem: ", "");
            }

            private void ExampleTextBox_GotFocus(object sender, RoutedEventArgs
e)
            {
                TextBox textBox = (TextBox)sender;
                if (textBox.Text == goods_to_edit.full_name)
                {
                    textBox.Text = string.Empty;
                    textBox.Foreground = Brushes.Black;
```

```
                }
            }

            private      void      ExampleTextBox_LostFocus(object      sender,
RoutedEventArgs e)
            {
                TextBox textBox = (TextBox)sender;
                if (string.IsNullOrWhiteSpace(textBox.Text))
                {
                    textBox.Text = goods_to_edit.full_name;
                    textBox.Foreground = Brushes.Gray;
                }
            }

            private      void      ComboBox_CategoryChanged(object      sender,
SelectionChangedEventArgs e)
            {
                GoodsService goods_service = new GoodsService(goods_repo);
                AdministratorService        admin_service        =        new
AdministratorService(admin_repo);
                string selectedCategory = GetSelectedItemComboBoxCategory();
                if (!string.IsNullOrEmpty(selectedCategory))
                {
                    if (selectedCategory == "Додати")
                    {
                        ComboBox_SelectionChanged(sender, e);
                    }
                    else
                    {
                        bool    categoryChanged    =    selectedCategory    !=
previousCategory;
                        if (categoryChanged)
                        {
                            categoryChanged = false;
                            subcategories = new List<string>();
                            previousCategory = selectedCategory;
                            subcategories                                    =
goods_service.GetSubCategoriesForAdministrator(admin_service.GetAdministratorByE
mail(MainWindow.username).admin_id, selectedCategory);
                            foreach (string sub_category in subcategories)
                            {
                                ComboBoxSubcategory.Items.Add(sub_category);
                            }
                            ComboBoxCategory.SelectedItem = selectedCategory;
                        }
                    }
                }
            }

            private     void     ComboBox_SubcategoryChanged(object     sender,
SelectionChangedEventArgs e)
            {
                string                   selectedSubcategory                  =
GetSelectedItemComboBoxSubcategory();
                if (selectedSubcategory == "Додати")
                {
                    ComboBox_SelectionChanged(sender, e);
                }
                else
                {
                    ComboBoxSubcategory.SelectedItem = selectedSubcategory;
                }
            }
```

```
            private      void      ComboBox_AddressChanged(object      sender,
SelectionChangedEventArgs e)
            {
                string selectedAddress = GetSelectedItemComboBoxAddress();
                if (selectedAddress == "Додати")
                {
                    ComboBox_SelectionChanged(sender, e);
                }
                else
                {
                    ComboBoxAddress.SelectedItem = selectedAddress;
                }
            }

            private     void     ComboBox_CountGoodsChanged(object     sender,
SelectionChangedEventArgs e)
            {
                string selectedCount = GetSelectedItemComboBoxCountGoods();
                if (selectedCount == "Додати")
                {
                    ComboBox_SelectionChanged(sender, e);
                }
                else
                {
                    ComboBoxCountGoods.SelectedItem = selectedCount;
                }
            }

            private     void     ComboBox_SelectionChanged(object     sender,
SelectionChangedEventArgs e)
            {
                ComboBox comboBox = (ComboBox)sender;
                ComboBoxItem                  selectedItem                   =
(ComboBoxItem)comboBox.SelectedItem;

                if (selectedItem.Content.ToString() == "Додати")
                {
                    TextBox textBox = new TextBox();
                    textBox.Width = 200;
                    textBox.VerticalAlignment = VerticalAlignment.Center;
                    textBox.HorizontalAlignment = HorizontalAlignment.Center;
                    textBox.Margin = new Thickness(0, 0, 0, 100);

                    this.ContentPanel.Children.Add(textBox);

                    textBox.KeyUp += (s, args) =>
                    {
                        if (args.Key == Key.Enter)
                        {
                            string newItem = textBox.Text.Trim();
                            if        (!string.IsNullOrEmpty(newItem)        &&
!comboBox.Items.OfType<ComboBoxItem>().Any(item  =>  item.Content.ToString()  ==
newItem))
                            {
                                comboBox.Items.Insert(comboBox.Items.Count    -
1, new ComboBoxItem { Content = newItem });
                                comboBox.SelectedItem                          =
comboBox.Items[comboBox.Items.Count - 2];
                            }

                            this.ContentPanel.Children.Remove(textBox);
                        }
                    };
                }
```

```
            }

            private void AddPhoto_Click(object sender, RoutedEventArgs e)
            {
                GoodsService goods_service = new GoodsService(goods_repo);
                OpenFileDialog openFileDialog = new OpenFileDialog();
                openFileDialog.Filter                                          =
"Зображення|*.jpg;*.png;*.bmp;*.gif|Усі файли|*.*";

                byte[] imageData = goods_to_edit.photo;

                BitmapImage imageSource = new BitmapImage();
                using (MemoryStream stream = new MemoryStream(imageData))
                {
                    stream.Position = 0;
                    imageSource.BeginInit();
                    imageSource.CreateOptions                                  =
BitmapCreateOptions.PreservePixelFormat;
                    imageSource.CacheOption = BitmapCacheOption.OnLoad;
                    imageSource.UriCachePolicy = null;
                    imageSource.StreamSource = stream;
                    imageSource.EndInit();
                }

                Image image = new Image();
                image.Source = imageSource;

                if (openFileDialog.ShowDialog() == true)
                {
                    string imagePath = openFileDialog.FileName;

                    Button addButton = (Button)sender;
                    string                  fileExtension                      =
System.IO.Path.GetExtension(imagePath).ToLower();

                    ImageFormat imageFormat = ImageFormat.Jpeg;

                    if (fileExtension == ".jpg" || fileExtension == ".jpeg")
                    {
                        imageFormat = ImageFormat.Jpeg;
                    }
                    else if (fileExtension == ".png")
                    {
                        imageFormat = ImageFormat.Png;
                    }
                    else if (fileExtension == ".bmp")
                    {
                        imageFormat = ImageFormat.Bmp;
                    }
                    else if (fileExtension == ".gif")
                    {
                        imageFormat = ImageFormat.Gif;
                    }

                    image.Source = new BitmapImage(new Uri(imagePath));
                    goods_to_edit.photo                                        =
ImageConverter.ConvertImageToByteArray(imagePath, imageFormat);

                    Grid grid = new Grid();
                    grid.Children.Add(image);

                    grid.HorizontalAlignment = HorizontalAlignment.Center;
                    grid.VerticalAlignment = VerticalAlignment.Center;
```

```
                    addButton.Content = grid;
                }
            }

            private void TextBoxPrice_TextChanged(object sender,
TextChangedEventArgs e)
            {
                if (decimal.TryParse(TextBoxPrice.Text, out decimal result))
                {
                    goods_to_edit.price = result;
                }
                else
                {
                    MessageBox.Show("Некоректне значення ціни.", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Warning);
                    TextBoxPrice.Text = "";
                }
            }

            private void TextBoxDescription_TextChanged(object sender,
TextChangedEventArgs e)
            {
                goods_to_edit.short_description = TextBoxDescription.Text;
            }

            private void Save_Click(object sender, object e)
            {
                AdministratorService admin_service = new
AdministratorService(admin_repo);

                string selectedQuantity = GetSelectedItemComboBoxCountGoods();
                if (selectedQuantity != null)
                {
                    if (int.TryParse(selectedQuantity, out int quantity))
                    {
                        goods_to_edit.quantity = quantity;
                    }
                }

                string selectedCategory = GetSelectedItemComboBoxCategory();
                if (selectedCategory != null)
                {
                    goods_to_edit.category = selectedCategory;
                }

                string selectedSubcategory =
GetSelectedItemComboBoxSubcategory();
                if (selectedSubcategory != null)
                {
                    goods_to_edit.subcategory = selectedSubcategory;
                }

                string selectedAddress = GetSelectedItemComboBoxAddress();
                if (selectedAddress != null)
                {
                    WarehouseService warehouse_service = new
WarehouseService(warehouse_repo);
                    if
(warehouse_service.GetWarehouseByAddress(GetSelectedItemComboBoxAddress()) ==
null)
                    {
                        Warehouse warehouse = new Warehouse { addres =
selectedAddress, admin_id_ref =
admin_service.GetAdministratorByEmail(MainWindow.username).admin_id };
```

```
                        warehouse_service.CreateWarehouse(warehouse);
                        goods_to_edit.warehouse_id_ref                          =
warehouse.warehouse_id;
                    }
                    else
                    {
                        goods_to_edit.warehouse_id_ref                          =
warehouse_service.GetWarehouseByAddress(GetSelectedItemComboBoxAddress()).wareho
use_id;
                    }
                }
                else
                {
                    MessageBox.Show("Поле не заповнено. Спробуйте ще раз.",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Warning);
                }

                GoodsService goods_service = new GoodsService(goods_repo);
                goods_service.UpdateGoods(goods_to_edit);

                if (ParentMainWindowAdmin != null)
                {
                    MainWindowAdmin     refreshedMainWindowAdmin     =     new
MainWindowAdmin();
                    refreshedMainWindowAdmin.Width                              =
ParentMainWindowAdmin.ActualWidth;
                    refreshedMainWindowAdmin.Height                             =
ParentMainWindowAdmin.ActualHeight;
                    refreshedMainWindowAdmin.Show();
                    ParentMainWindowAdmin.Close();
                }

                Close();
            }

            private void buttonTrash_Click(object sender, RoutedEventArgs e)
            {
                GoodsService goods_service = new GoodsService(goods_repo);
                MessageBoxResult result = MessageBox.Show("Ви справді хочете
видалити товар?", "Запитання", MessageBoxButton.YesNo, MessageBoxImage.Question);

                if (result == MessageBoxResult.Yes)
                {
                    goods_service.DeleteGoods(goods_to_edit.goods_id);
                    Close();
                }
            }
        }
    }
```

Додаток Б

Створення вікна логування

```
using BusinessLogic;
using DB;
using System.Windows;
using Serilog;

namespace Wpf_Inventarium
{
    /// <summary>
    /// Interaction logic for Create_window.xaml
    /// </summary>
    public partial class CreateWindow : Window
    {
        AdministratorRepository admin_repo = new
AdministratorRepository();
        ILogger _logger = LoggerManager.Instance.Logger;

        public CreateWindow()
        {
            InitializeComponent();
        }

        private void ButtonAuthorization_Click(object sender,
RoutedEventArgs e)
        {
            MainWindow win = new MainWindow();
            win.Show();
            Close();
        }

        private void Window_Closed(object sender, System.EventArgs
e)
        {
            Close();
        }

        private void Confirm_Registration_Button_Click(object
sender, RoutedEventArgs e)
        {
            _logger.Information("Спроба створення користувача");
            string companyName = textBoxOrganization.Text;
            string username = textBoxLogin.Text;
            string password = passwordBox.Password;
            AdministratorService admin_service = new
AdministratorService(admin_repo);
            if (!InputValidator.IsCompanyNameValid(companyName))
            {
                _logger.Error("Помилка створення користувача");
                MessageBox.Show("Назва компанії повинна мати довжину
від 2 до 18 символів." +
                "\nНе може містити спецільних символів." +
                "\nНе може містити цифр.", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
            else if (!InputValidator.IsEmailValid(username))
```

```
                {
                    _logger.Error("Помилка створення користувача");
                    MessageBox.Show("Ім'я користувача повинне бути у
формі електронної пошти (example@example.com).", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
                }
                else if (!InputValidator.IsPasswordValid(password))
                {
                    _logger.Error("Помилка створення користувача");
                    MessageBox.Show("Пароль повинен мати довжину від 8
до 20 символів." +
                        "\nПовинен містити як мінімум 1 велику літеру."
+
                        "\nПовинен містити як мінімум 1 малу літеру." +
                        "\nПовинен містити як мінімум 1 цифру.",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
                }
                else                                                if
(admin_service.RegisterAdministrator(companyName, username, password) !=
null)
                {
                    MainWindow.username = username;
                    MessageBox.Show("Вхід в систему успішний!");
                    MainWindowAdmin win = new MainWindowAdmin();
                    Close();
                    win.Show();
                }
                else
                {
                    _logger.Error("Помилка створення користувача");
                    MessageBox.Show("Спробуйте ще раз.", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
        }
    }
```