

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

«До захисту допущено»  
Завідувач кафедри  
\_\_\_\_\_ Віталій РОМАНКЕВИЧ  
(підпис)  
“ \_\_\_\_ ” червня 2025 р.

**Дипломний проект  
на здобуття ступеня бакалавра  
за освітньо-професійною програмою  
«Системне програмування та спеціалізовані комп’ютерні системи»  
123 «Комп’ютерна інженерія»**

на тему: Комп’ютерна система управління мережею торгових точок  
Виконав: студент IV курсу, групи КВ-11  
Чебан Максим Дмитрович

\_\_\_\_\_  
(підпис)

Керівник доц.каф. СПСКС, к.т.н.  
Павловський Володимир Ілліч

\_\_\_\_\_  
(підпис)

Консультант з нормоконтролю доц.каф. СПСКС, к.т.н  
Клятченко Ярослав Михайлович

\_\_\_\_\_  
(підпис)

Рецензент доц.каф. ОТ ФІВТ, к.т.н.  
Верба Олександр Андрійович

\_\_\_\_\_  
(підпис)

Засвідчую, що у цьому дипломному  
проекті немає запозичень з праць інших  
авторів без відповідних посилань.

Студент

\_\_\_\_\_  
(підпис)

Київ – 2025 рік

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

Кафедра системного програмування і спеціалізованих комп’ютерних  
систем

Рівень вищої освіти – перший (бакалаврський)

**Освітньо-професійною програма  
«Системне програмування та спеціалізовані  
комп’ютерні системи»  
Спеціальність 123 «Комп’ютерна інженерія»**

ЗАТВЕРДЖУЮ  
Завідувач кафедри  
Віталій РОМАНКЕВИЧ  
(підпис)  
“  ” червня 2025 р.

**ЗАВДАННЯ  
на дипломний проект студента  
Чебана Максима Дмитровича**

1. Тема проекту «Комп’ютерна система управління мережею торгових точок»,  
керівник проекту асистент каф. СПСКС, д.ф.

Павловський Володимир Ілліч,  
 затверджені наказом по університету від «30» квітня

2. Термін подання студентом проекту \_\_\_\_\_

3. Вихідні дані до проекту див. Технічне завдання.

4. Зміст пояснівальної записки:

- 1) Аналіз існуючих рішень
- 2) Розробка серверного додатку
- 3) Розробка клієнтського додатку

4) Тестування обох програм

5) Шляхи поліпшення та перспективи розвитку

5. Перелік графічного матеріалу:

- 1) Схема алгоритму роботи серверного застосунку
- 2) Схема алгоритму роботи клієнтського застосунку
- 3) Алгоритм роботи системи
- 4) Діаграми класів серверного та клієнтського додатків

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доц.каф. СПСКС, к.т.н		

7. Дата видачі завдання «31» квітня 2025 р.

Календарний план

з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою роботи	17.04.2025	Виконано
2	Розроблення та узгодження технічного завдання	30.04.2025	Виконано
3	Аналіз існуючих рішень	05.05.2025	Виконано
4	Розробка структури додатку	11.05.2025	Виконано
5	Програмна реалізація додатку	13.05.2025	Виконано
6	Тестування додатку	15.05.2025	Виконано
7	Підготовка матеріалів первого розділу дипломного проєкту	16.05.2025	Виконано
8	Підготовка матеріалів другого розділу дипломного проєкту	17.05.2025	Виконано
9	Підготовка матеріалів третього розділу дипломного проєкту	20.05.2025	Виконано
10	Підготовка матеріалів третього розділу дипломного проєкту	22.05.2025	Виконано
11	Підготовка матеріалів п'ятого розділу дипломного проєкту	24.05.2025	Виконано
12	Оформлення технічної документації проєкту	28.05.2025	Виконано

Студент

Максим ЧЕБАН

Керівник проєкту

Володимир ПАВЛОВСЬКИЙ

## АНОТАЦІЯ

Кваліфікаційна робота включає пояснівальну записку (приблизно 70 с., 54 рис., 4 додатки).

### КОМП'ЮТЕРНА СИСТЕМА УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК.

Об'єкт розробки — комп'ютерна система для централізованого управління мережею торгових точок.

Мета розробки — створити комп'ютерну систему, яка забезпечує управління товарами, персоналом, обліком продажів, аналізом залишків та автоматизує аналітичну діяльність у розрізі кожної торгової точки.

У ході розробки:

- Сформульовані функціональні та нефункціональні вимоги до системи;
- Спроектована архітектура багатокомпонентної системи;
- Розроблена структура бази даних з підтримкою зв'язків між об'єктами;
- Реалізований додаток на базі .NET Windows Forms;
- Впроваджено інтерфейс користувача з підтримкою ролей (адміністратор, менеджер, касир).

Основні характеристики та можливості системи:

- Управління товарами (каталог, категорії, залишки, ціни);
- Ведення обліку продажів по кожній торговій точці;
- Аналітика у вигляді таблиць та графіків;
- Система ролей та авторизації;
- Управління персоналом та призначення по магазинах;
- Внутрішні переміщення товарів між точками (опціонально);

У процесі розробки використано такі технології: мова програмування C#, фреймворк .NET Windows Forms, ASP.NET Core, система управління базами даних PostgreSQL, мова структурованих запитів SQL, HTML для створення шаблонів звітів.

Можливе подальше вдосконалення системи за рахунок інтеграції зі сторонніми сервісами аналітики, підключенням мобільного додатку або хмарного зберігання.

Ключові слова: комп'ютерна система, торгова точка, управління мережею, клієнт-серверна архітектура, C#, ASP.NET Core, Entity Framework Core, PostgreSQL, WinForms, RESTful API, ORM, інтерфейс користувача, категорії товарів, продажі, звітність, DataGridView, авторизація користувачів, система ролей, збереження даних, Excel-звіт.

## **ANNOTATION**

The qualification work includes an explanatory note (approximately 70 p., 54 fig., 4 appendices).

### **COMPUTER MANAGEMENT SYSTEM FOR A NETWORK OF RETAIL POINTS.**

The object of development is a computer system for centralized management of a network of retail points.

The purpose of the development is to create a computer system that provides management of goods, personnel, sales accounting, balance analysis and automates analytical activities in the context of each retail point.

During the development:

- functional and non-functional requirements for the system were formulated;
- The architecture of a multi-component system was designed;
- a database structure was developed with support for relationships between objects;
- an application based on .NET Windows Forms was implemented;
- a user interface with role support (administrator, manager, cashier) was implemented.

Main features and capabilities of the system:

- Product management (catalog, categories, balances, prices);
- Sales accounting for each point of sale;
- Analytics in the form of tables and graphs;
- Role and authorization system;
- Personnel management and store assignments;
- Internal movement of goods between points (optional);

The following technologies are used in the development process: C# programming language, .NET Windows Forms framework, ASP.NET Core, PostgreSQL database management system, SQL structured query language, HTML for creating report templates.

The system can be further improved by integrating with third-party analytics services, connecting a mobile application or cloud storage.

Keywords: computer system, point of sale, network management, client-server architecture, C#, ASP.NET Core, Entity Framework Core, PostgreSQL, WinForms, RESTful API, ORM, user interface, Product categories, sales, reporting, DataGridView, user authorization, role system, data storage, Excel report.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітк и
	A4	ІАЛЦ. 045440.002 Т3	Комп'ютерна система управління мережею торгових точок Технічне завдання	4		
	A4	ІАЛЦ. 045440.003 П3	Комп'ютерна система управління мережею торгових точок Пояснювальна записка	70		
	A4	ІАЛЦ. 045440.004 Д1	Схема алгоритму роботи серверного застосунку	1		
	A4	ІАЛЦ. 045440.005 Д2	Схема алгоритму роботи клієнтського застосунку	1		
	A4	ІАЛЦ. 045440.006 Д3	Алгоритм роботи системи	1		
	A1	ІАЛЦ. 045440.007 Д4	Діаграма класів клієнтського та серверного	1		

Змін.	Арк.	№ докум.	Підпис	Дата
Розробив	Чебан М. Д.			
Перевірив	Павловський В. І			
Консульт.				
Н. контролю				
Зав. каф.	Романкевич В. О.			

**ІАЛЦ.045440.001 ОА**

Комп'ютерна система управління  
мережею торгових точок

**Опис альбому**

Літ.	Аркуш	Аркушів
	1	2
КПІ ім. Ігоря Сікорського, ФПМ КВ-11		

Змін.	Арк.	№ докум.	Підпис	Дата
<i>Розробив</i>		<i>Чебан М. Д.</i>		
<i>Перевірив</i>		<i>Павловський В. І</i>		
<i>Консульт.</i>				
<i>Н. контролюв.</i>				
<i>Зав. каф.</i>		<i>Романкевич В.О.</i>		

# *ІАЛЦ.045440.001 OA*

## Комп'ютерна система управління мережею торгових точок

Літ.	Аркуш	Аркушів
	2	2
КПІ		
ім. Ігоря Сікорського,		
ФПМ КВ-11		

## ЗМІСТ

<u>1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ</u>	2
<u>2. ПІДСТАВА ДЛЯ РОЗРОБКИ</u>	2
<u>3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ</u>	2
<u>4. ДЖЕРЕЛА РОЗРОБКИ</u>	2
<u>5. ТЕХНІЧНІ ВИМОГИ</u>	3
<u>5.1. Вимоги до програмного продукту, що розробляється</u>	3
<u>5.2. Вимоги до апаратного забезпечення</u>	3
<u>5.3. Вимоги до програмного та апаратного забезпечення користувача</u>	3
<u>6. ЕТАПИ РОЗРОБКИ</u>	4

Зм.	Лист	№ докум.	Підп.	Дата	ІАЛЦ. 045440.002 ТЗ		
Розроб.	Чебан М.Д.				Комп'ютерна система управління мережею торгових точок	Літ.	Лист
Перев.	Павловський В.І.				Технічне завдання	1	Листів
Н. контр.						102	
Затв.	Романкевич В.О.				НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-11		

## 1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Комп'ютерна система управління мережею торгових точок».

Галузь застосування: Автоматизація обліку товарів, продажів та управління торговими точками малого та середнього бізнесу.

## 2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проєктування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

## 3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення застосунку для централізованого управління мережею торгових точок, що дозволяє вести облік товарів, продажів, категорій, ролей користувачів, а також здійснювати базовий аналіз діяльності.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом Джерелами інформації для розробки є:

- науково-технічна та навчальна література з програмування, баз даних, архітектури програмного забезпечення;
- документація до технологій C#, .NET, Windows Forms, ASP.NET CORE, PostgreSQL;
- сучасні практики у сфері розробки облікових систем для торгівлі;
- відкриті джерела, онлайн-статті та офіційна документація..

## 5. ТЕХНІЧНІ ВИМОГИ

### 5.1. Вимоги до програмного продукту, що розробляється:

- Віконний інтерфейс користувача на базі .NET Windows Forms;
- Робота з базою даних PostgreSQL;

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.002 ТЗ**

Лис

2

- Можливість створення, редагування та перегляду:
  - Товарів (Products)
  - Категорій товарів (Categories)
  - Торгових точок (Stores)
  - Користувачів і ролей (Users, Roles)
  - Продажів (Sales)
- Підтримка обліку залишків товару;
- Аутентифікація користувачів;
- Фільтрація, сортування та пошук даних;
- Створення базових звітів.

### 5.2. Вимоги до апаратного забезпечення

- Процесор: Intel Core i5 або аналогічний AMD;
- Оперативна пам'ять: від 8 ГБ;
- Місце на диску: мінімум 500 МБ вільного простору;
- \*\*Підтримка запуску PostgreSQL-сервера або підключення до зовнішнього сервера.

### 5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система: Windows 10/11, macOS або Android (для мобільної версії);
- Встановлене .NET середовище виконання;
- Доступ до локального або хмарного серверу PostgreSQL;
- Права на запис/читання локальних файлів (для експорту/імпорту даних).

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.002 Т3**

Лис  
3

## 6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	17.04.2025
2.	Розроблення та узгодження технічного завдання	30.04.2025
3.	Аналіз існуючих рішень	05.05.2025
4.	Підготовка матеріалів первого розділу дипломного проекту	10.05.2025
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2025
6.	Підготовка серверної частини дипломного проекту	19.05.2025
7.	Підготовка графічної та клієнтської частини дипломного проекту	20.05.2025
8.	Оформлення документації дипломного проекту	25.05.2025
9.	Попередній огляд матеріалів диплому на кафедрі	30.05.2025

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.002 ТЗ**

Лис

4

**Пояснювальна записка  
до дипломного проекту**

на тему: «Комп'ютерна система управління мережею торгових точок»

Київ - 2025

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

**Лис  
5**

## ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ .....	8
ВСТУП.....	9
1. АНАЛІЗ СУЧАСНИХ КОМП'ЮТЕРНИХ СИСТЕМ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК.....	10
1.1. Галузь застосування комп'ютерних систем управління торговими точками .....	10
1.2. Огляд існуючих рішень та їх недоліки .....	10
1.3. Постановка задачі на розробку КС УМТТ .....	13
2. ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК.....	15
2.1     Основні вимоги до системи .....	15
2.2     Архітектура комп'ютерної системи .....	16
2.3     Проектування бази даних .....	19
2.4     Проектування інтерфейсів користувача .....	23
2.5     Вибір технологій реалізації .....	25
2.5.1     Вибір мови програмування та фреймворків .....	25
2.5.2     Вибір СУБД .....	25
2.5.3     Вибір інструментів створення клієнтського додатку .....	25
2.5.4     Вибір технологій створення серверної частини .....	26
3. РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК.....	28
3.1     Реалізація серверної частини засобами ASP.NET .....	28
3.1.1     Архітектура API та структура .....	28
3.1.2     Реалізація контролерів та обробка запитів .....	32
3.1.3     Реалізація доступу до бази даних засобами EF Core.....	37
3.2     Реалізація клієнтського застосунку засобами .NET WinForms.	41
3.2.1     Головне меню та структура інтерфейсу .....	41
3.2.2     Робота з API .....	45

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
6

3.2.3	Основні екрани та форми: товари, продажі, персонал, аналітика .....	47
4.	АНАЛІЗ ТА ТЕСТУВАННЯ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК.....	57
4.1	Тестування функціоналу клієнтської частини .....	57
4.2	Тестування API та запитів до БД .....	64
5.	ШЛЯХИ ПОЛІПШЕННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ .....	71
5.1.	Можливості розширення функціоналу .....	71
5.2.	Інтеграція з хмарними сервісами .....	72
	ВИСНОВКИ .....	74
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	77
	ДОДАТКИ	

Додаток 1. ІАЛЦ. 045440.004 Д1 Схема алгоритму роботи серверного застосунку.

Додаток 2. ІАЛЦ. 045440.005 Д2 Схема алгоритму роботи клієнтського застосунку.

Додаток 3. ІАЛЦ. 045440.006 Д3 Алгоритм роботи системи.

Додаток 4. ІАЛЦ. 045440.007 Д4 Діаграма класів клієнтського та серверного додатків.

Додаток 5. Презентація.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

7

## ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Фреймворк (від англійського "framework" - каркас, платформа, структура) в програмуванні - це готові інструменти, бібліотеки та шаблони, які допомагають розробникам прискорити та спростити процес розробки програмного забезпечення.

RESTful API — це архітектурний стиль створення інтерфейсів прикладного програмування, який базується на використанні HTTP-запитів для взаємодії з ресурсами. Основні типи запитів — GET, POST, PUT і DELETE — відповідають основним операціям: отримання, створення, оновлення та видалення даних.

СУБД (система управління базами даних) - це програмне забезпечення, що дозволяє створювати, зберігати, оновлювати та витягувати дані з бази даних. Це своєрідний "інструмент" для роботи з базами даних, який забезпечує їх організацію, безпеку та зручність використання.

ORM (Object-Relational Mapping) — це технологія, яка забезпечує зв'язок між реляційною базою даних і об'єктно-орієнтованим кодом, дозволяючи працювати з даними як з об'єктами без необхідності написання SQL-запитів вручну. Вона спрощує доступ до даних, уніфікує інтерфейси взаємодії та зменшує обсяг повторюваного коду, що сприяє пришвидшенню розробки.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

8

## ВСТУП

У сучасних умовах активного розвитку торгівлі та цифрових технологій ефективне управління мережею торгових точок стає важливою складовою успішної діяльності підприємств. Зростаючий обсяг даних, необхідність швидкого прийняття рішень та контроль за процесами продажу вимагають впровадження сучасних комп'ютерних систем обліку та управління.

Комп'ютерні системи автоматизації дають змогу не лише централізовано керувати окремими торговими точками, але й забезпечують інтеграцію різних підрозділів підприємства, оптимізують бізнес-процеси, зменшують вплив людського фактору та підвищують якість обслуговування клієнтів.

Розвиток сучасних мов програмування, зокрема C# і платформ типу ASP .NET, а також потужних систем керування базами даних, таких як PostgreSQL, відкриває нові можливості для створення гнучких, масштабованих та зручних у використанні програмних рішень.

Даний дипломний проект присвячено розробці комп'ютерної системи управління мережею торгових точок. У проекті реалізовано облік товарів, управління користувачами, відстеження залишків продукції та роботу з категоріями товарів, що дозволяє забезпечити ефективну та надійну підтримку процесів торгівлі, а також отримання аналітичний даних цих процесів.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
9

## 1. АНАЛІЗ СУЧАСНИХ КОМП'ЮТЕРНИХ СИСТЕМ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК

### 1.1. Галузь застосування комп'ютерних систем управління торговими точками

Комп'ютерні системи управління торговими точками — це комплексні програмно-апаратні засоби, призначені для автоматизації процесів продажу та управління торгівельною діяльністю у магазинах, ресторанах та інших комерційних закладах. Вони забезпечують оперативну обробку транзакцій, ведення товарного обліку, моніторинг роботи персоналу та надання аналітичних даних для підвищення ефективності бізнес-процесів.

Комп'ютерні системи цього типу застосовуються у сфері роздрібної та оптової торгівлі для автоматизації процесів обліку товарів, продажів, аналітики, управління персоналом та взаємодії між окремими торговими об'єктами. Їхнє основне призначення — забезпечити ефективне функціонування магазинів, супермаркетів, аптек, складів, торгових мереж тощо.

Такі системи охоплюють широкий спектр завдань:

- контроль та реєстрація продажів;
- облік і управління персоналом;
- формування звітності для керівництва;
- аналітика щодо прибутковості, асортименту та попиту;
- синхронізація даних між центральним офісом і віддаленими магазинами.
- облік залишків на складах і в торгових точках;

Особливої актуальності такі системи набувають у великих торгових мережах, де необхідна координація десятків або навіть сотень магазинів, а також у підприємствах, які прагнуть підвищити ефективність бізнес-процесів за рахунок цифровізації.

### 1.2. Огляд існуючих рішень та їх недоліки

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

10

На сучасному ринку існує велика кількість програмних рішень для автоматизації управління торговими точками. Серед найпопулярніших можна виділити такі системи, як Poster POS, 1C: Підприємство, RetailCRM, iiko (для ресторанного бізнесу), Square POS, Lightspeed, Shopify POS та інші. Ці системи надають широкий набір функцій — від обліку товарів до аналітики продажів, керування персоналом та клієнтськими програмами.

На сучасному ринку існує велика кількість програмних рішень для автоматизації управління торговими точками. Серед найпопулярніших можна виділити такі системи, як Poster POS, 1C: Підприємство, RetailCRM, iiko (для ресторанного бізнесу), Square POS, Lightspeed, Shopify POS та інші. Ці системи надають широкий набір функцій — від обліку товарів до аналітики продажів, керування персоналом та клієнтськими програмами.

### Poster POS[1]

Poster POS — це хмарна Point of sale (POS) система, орієнтована переважно на кафе, бари та ресторани. Вона дозволяє керувати замовленнями, обліком товарів, автоматизувати розрахунки та друк чеків, а також контролювати залишки на складі. Система має простий інтерфейс та мобільний застосунок. Основним недоліком є обмеження у кастомізації під нестандартні процеси, а також платна підписка з обмеженнями за кількістю точок або кас.

### 1C:Підприємство[2]

Це одна з найпоширеніших систем у країнах пострадянського простору. Вона забезпечує широкий функціонал: облік товарів, бухгалтерію, зарплати, звітність тощо. Платформа є дуже гнучкою, але складною в освоєнні. Серед недоліків — застарілий інтерфейс, складність налаштування та високі витрати на впровадження й технічну підтримку.

### RetailCRM[3]

RetailCRM орієнтована на інтернет-магазини та омніканальні продажі. Основна перевага — глибока інтеграція з e-commerce платформами, аналітика, маркетинг і автоматизація взаємодії з клієнтами. Проте вона не є повноцінною

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
11

POS-системою, а більше CRM-системою з окремими модулями для роздрібної торгівлі. Не всім підходить як рішення «все в одному».

#### Iiko[4]

iiko — це спеціалізоване рішення для ресторанного бізнесу. Воно охоплює всі аспекти: від автоматизації замовлень до складського обліку, HR та логістики. Працює у хмарі та на локальних серверах. Сильними сторонами є гнучкість та глибокий функціонал для фуд-сервісу. Недолік — відносно висока вартість та складність впровадження.

#### Square POS[5]

Square POS — популярна система в США та інших країнах, орієнтована на малий бізнес. Вона проста у використанні, підтримує облік продажів, інвентаризацію, аналітику та інтеграцію з банківськими терміналами. Має хорошу підтримку мобільних платформ. Основним недоліком для нашого ринку є відсутність локалізації та прив'язка до платіжної системи Square, яка не підтримується в Україні.

#### Lightspeed[6]

Lightspeed — потужна POS-система, що підходить для рітейлу, готельного та ресторанного бізнесу. Підтримує хмарне зберігання, інвентаризацію, управління персоналом, CRM-функції. Сильна сторона — високий рівень кастомізації. Проте система є дороговартісною, і не всі функції адаптовані для невеликих бізнесів.

#### Shopify POS[7]

Shopify POS — розширення популярної e-commerce платформи Shopify. Ідеально підходить для бізнесів, які поєднують онлайн- та офлайн-продажі. Має зручний інтерфейс, синхронізується з онлайн-магазином. Однак, її використання можливе лише в екосистемі Shopify, що обмежує масштабування або інтеграцію з іншими рішеннями.

Попри розмаїття функціоналу, більшість із них мають ряд недоліків:

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

12

- висока вартість ліцензій або щомісячної абонплати, що є суттєвим фактором для малого бізнесу.
- наявність зайвих функцій, які ускладнюють інтерфейс та створюють перевантаження для користувача.
- недостатня гнучкість та адаптивність під конкретні потреби бізнесу, зокрема — обмежені можливості кастомізації.
- прив’язаність до конкретної платформи або хмарного сервісу, що може викликати труднощі з перенесенням даних чи автономною роботою.
- низька швидкодія на слабкому обладнанні, що актуально для невеликих торгових точок, які не мають ресурсів для оновлення техніки.
- складність у налаштуванні та інтеграції з іншими системами підприємства.

У зв’язку з цим виникає потреба у створенні гнучкої, зручної та економічно ефективної комп’ютерної системи управління мережею торгових точок, яка відповідатиме реальним потребам малого та середнього бізнесу, забезпечуючи при цьому стабільність, швидкодію та легкість в обслуговуванні. Також важливо, щоб була доступна перспектива для розвитку системи на різних платформах: як у вигляді комп’ютерної програми, так і мобільного застосунку.

### 1.3. Постановка задачі на розробку КС УМТТ

Метою розробки є створення комп’ютерної системи, яка забезпечить ефективне управління мережею торгових точок, автоматизуючи основні бізнес-процеси: облік товарів, реєстрацію продажів, управління персоналом та формування звітності для аналітики.

Основні задачі, які має вирішувати система:

- реєстрація продажів із фіксацією дати, часу, товарних позицій, суми та відповідального працівника;

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис
13

- управління користувачами — створення облікових записів, призначення ролей (адміністратор, касир, менеджер), обмеження доступу до певних функцій;
- ведення обліку персоналу, зберігання їхніх даних, посад і прив'язаних до них торгових точок;
- облік товарів з функціями перегляду, додавання, редагування, категоризації та фільтрації;
- отримання статистики та формування звітів у зручному для аналізу форматі (HTML, .XLSX), з можливістю фільтрації за товарами, періодами, працівниками тощо;
- захист і цілісність даних — реалізація механізмів авторизації та розмежування прав доступу для запобігання несанкціонованим діям.
- результатом проекту має стати надійна та зручна у використанні комп’ютерна система, орієнтована на потреби малого та середнього бізнесу, що дозволить централізовано керувати мережею торгових точок, контролювати діяльність персоналу та приймати зважені управлінські рішення на основі звітів і аналітики.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис
14

## 2. ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК

### 2.2. Основні вимоги до системи

Комп'ютерна система управління мережею торгових точок повинна забезпечувати автоматизацію ключових бізнес-процесів, пов'язаних з торгівлею, обліком товарів, роботою персоналу та аналітикою діяльності. Для цього до системи висуваються як функціональні, так і нефункціональні вимоги.

Функціональні вимоги:

- облік товарів: додавання, редагування, видалення товарів, прив'язка до категорій, пошук та фільтрація;
- управління продажами: фіксація операцій продажу з відображенням дати, часу, продавця, товарів та суми;
- управління персоналом: додавання та редагування інформації про співробітників, призначення ролей (адміністратор, продавець тощо);
- авторизація та контроль доступу: підтримка ролей користувачів із різним рівнем доступу;
- статистика та звітність: формування звітів про продажі, товари, активність персоналу, експорт у форматах HTML або XLSX;
- синхронізація з сервером: забезпечення коректної взаємодії між клієнтською частиною та сервером через API.

Нефункціональні вимоги:

- зручний графічний інтерфейс, адаптований під різні розміри екранів;
- безпечно зберігання та передача даних;
- можливість масштабування системи при збільшенні кількості точок продажу;
- чітка документація коду, що забезпечує простоту супроводу та модифікації.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

15

Розробка системи має ґрунтуватися на сучасних технологіях програмування, зокрема використанні C#, ASP.NET Core для серверної частини, .NET Windows Forms для клієнтського застосунку та PostgreSQL як СУБД. Система повинна бути адаптована для локального та мережевого використання у середовищах малого та середнього бізнесу.

### 2.3. Архітектура комп’ютерної системи

Архітектура комп’ютерної системи управління мережею торгових точок має модульну, багаторівневу структуру, що забезпечує розділення відповідальностей, гнучкість у розробці та легкість масштабування. Система реалізується за клієнт-серверною моделлю з чітким поділом на клієнтську, серверну та базову (даних) частини.

Основні компоненти архітектури:

#### 1. Клієнтський застосунок (Front-end)

Реалізується за допомогою фреймворку, що дозволяє створити кросплатформений інтерфейс користувача з підтримкою десктопних та мобільних платформ. Клієнт відповідає за взаємодію з користувачем, надсилає запитів до API та відображення отриманих даних. Клієнтський застосунок забезпечує представлення аналітики та звітності. В додаток вбудований механізм формування звітів, який дозволяє створювати HTML- та xlsx-звіти на основі даних з бази, забезпечуючи керівництво актуальною інформацією про стан продажів і діяльність торгових точок.

#### 2. Серверна частина (Back-end)

Розробляється з використанням RESTful API[8], що обумовлюється використанням HTTP-запитів для взаємодії з ресурсами. Основні типи цих запитів відповідають операціям отримання, створення, оновлення та видалення даних. Сервер виконує бізнес-логіку, обробляє запити клієнта, перевіряє права доступу, виконує операції над базою даних та повертає результати. Також на сервер покладена відповідальність за виконання валідації даних, отриманих від

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

16

користувачя. Уся взаємодія з клієнтом відбувається через HTTP-запити у форматі JSON.

### 3. База даних (Data Layer)

У системі використовується СУБД для надійного зберігання інформації про товари, продажі, працівників, користувачів та інші сутності. Робота з БД на сервері реалізована через ORM-технологію, що спрощує створення, оновлення та взаємодію з таблицями. Доступ до бази даних реалізований через API.

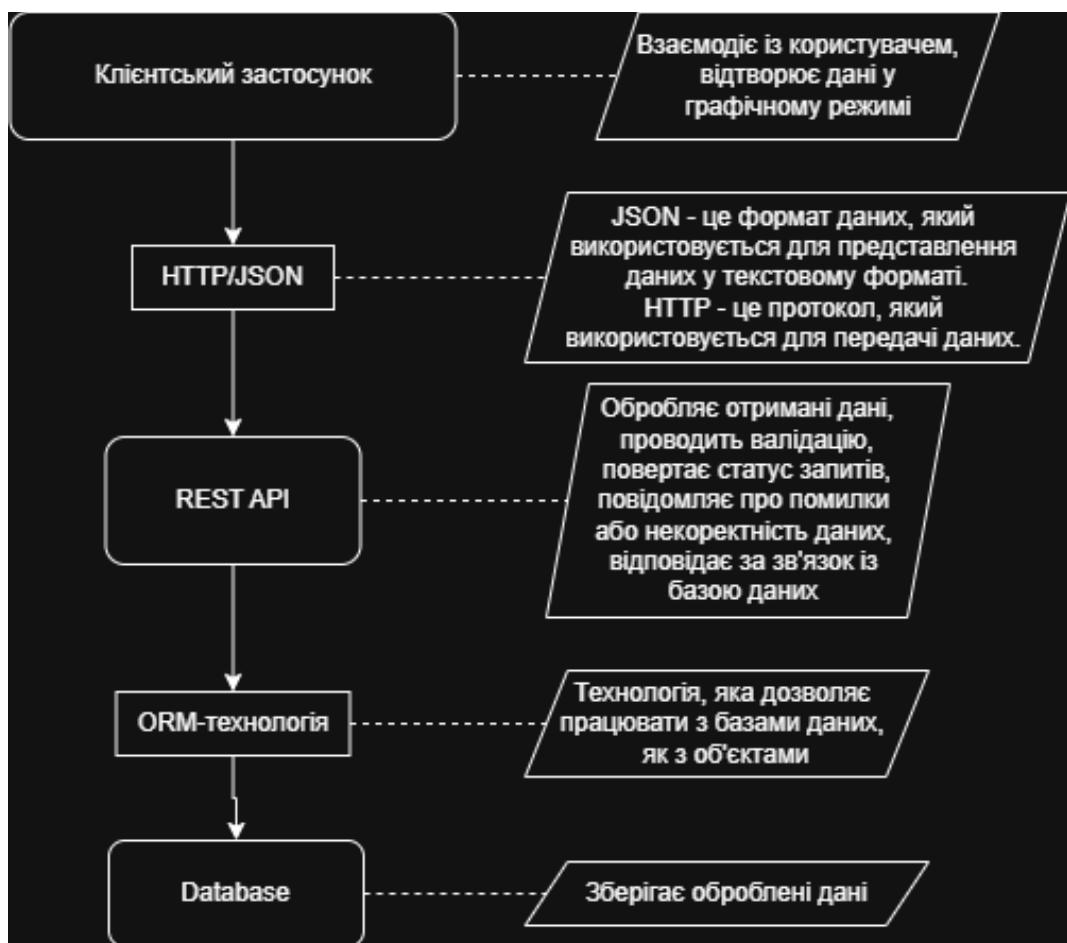


Рисунок 2.1 – Схема взаємодії компонентів комп’ютерної системи

На рисунку 2.1 зображено архітектуру[9] системи, яка реалізує тришаровий підхід: клієнт, сервер та база даних. Усі компоненти взаємодіють між собою через стандартизовані інтерфейси (REST API та ORM), що забезпечує масштабованість, модульальність та гнучкість системи.

Зм	Лист	№ докум.	Підп.	Дата

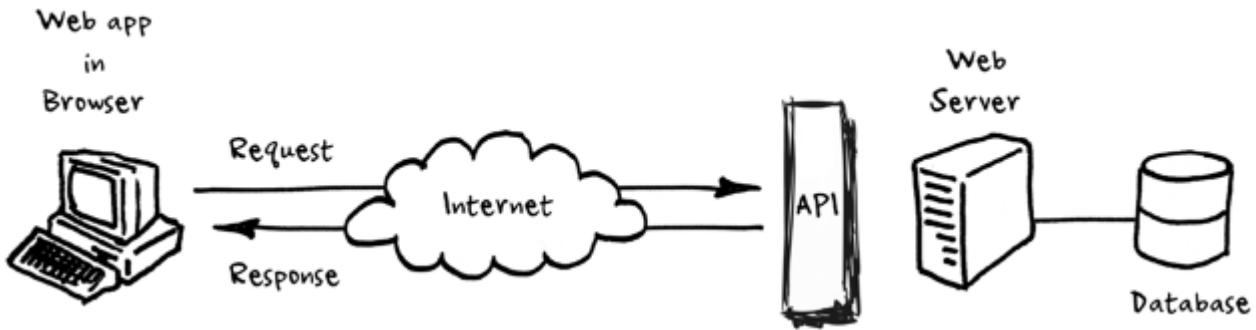


Рисунок 2.2 – Принципи роботи WEB-API

На рисунку 2.2 зображене схему взаємодії RESTful API між клієнтом і сервером. У випадку комп’ютерної системи, розробленої вданому проєкті, актуальнішим є Desktop App або Mobile App замість Web App, як позначено на схемі, проте API передбачає майбутню перспективу розробки і Web-додатку для доступу до системи через браузер.

У системах з RESTful API, які обслуговують велику кількість клієнтів, асинхронність є критично важливою для забезпечення масштабованості, відповідності сучасним стандартам і ефективного використання ресурсів сервера.

У зв’язку з тим, що система може обслуговувати одночасно декілька клієнтів (наприклад, різні торгові точки чи користувачів, які працюють із застосунком паралельно), серверна частина реалізована з використанням асинхронних контролерів. Це означає, що всі запити до REST API обробляються у неблокуючому режимі з використанням ключових слів `async` та `await`. Такий підхід дозволяє:

- ефективно масштабувати систему при великій кількості запитів;
- уникати блокування потоків виконання;
- зменшити час відповіді сервера;
- забезпечити кращу продуктивність при роботі з базою даних через ORM.

Завдяки асинхронності система залишається стабільною і швидкодійною навіть при високому навантаженні, гарантуючи, що кожен клієнт отримає відповідь без затримок.

Зм	Лист	№ докум.	Підп.	Дата

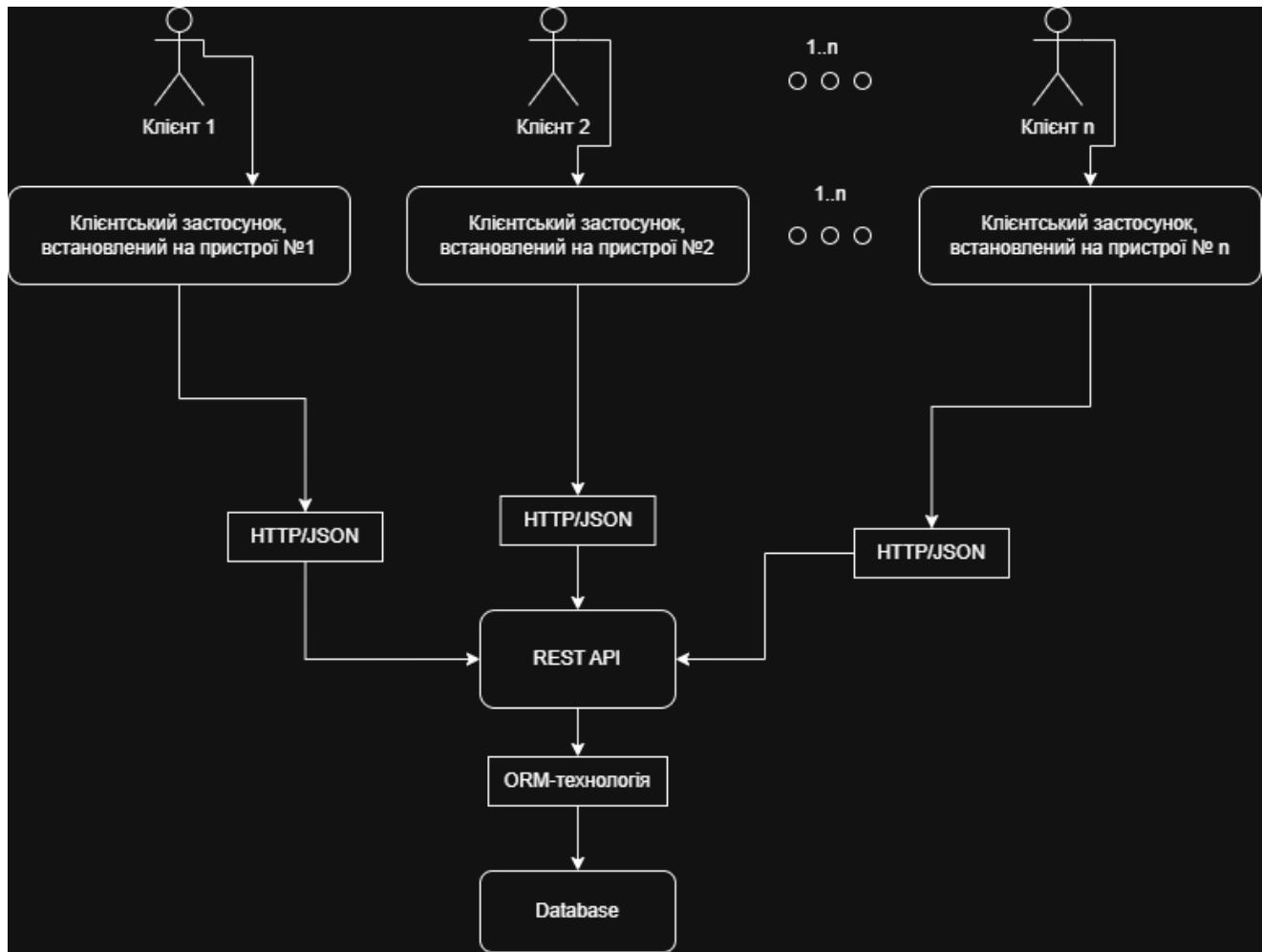


Рисунок 2.3 – Схема доступу багатьох клієнтів до бази даних

На рисунку 2.3 зображено схему доступу багатьох клієнтів до одного серверу одночасно. Саме за цим принципом працюють всі сучасні системи подібного типу.

#### 2.4. Проектування бази даних

У системі управління мережею торгових точок використовується реляційна база даних, спроектована відповідно до вимог нормалізації даних, модульності та розширеності. Всі таблиці пов'язані між собою через зовнішні ключі, що забезпечує цілісність і узгодженість даних. Опис сутностей та їх атрибутив наведено в таблиці 2.1.

Зм	Лист	№ докум.	Підп.	Дата

Таблиця 2.1 – Опис структури бази даних "Retail-Outlet-Network"

Відношення	Атрибут	Тип атрибуту
<b>Stores</b> - містить інформацію про всі торгові точки мережі.	<b>StoreID</b> – унікальний ідентифікатор торгової точки. <b>StoreName</b> – назва магазину. <b>Address</b> – адреса розташування. <b>CreatedAt</b> – дата й час створення торгової точки.	<b>uuid</b> (унікальний ідентифікатор) <b>text</b> (текст) <b>text</b> (текст)  <b>datetime</b> (дата й час)
<b>Users</b> - зберігає дані про зареєстрованих користувачів системи (адміністратори, касири тощо). Паролі зберігаються у вигляді хешу з міркувань безпеки. Ролі винесено в окрему таблицю для гнучкості управління доступом.	<b>UserID</b> – унікальний ідентифікатор користувача <b>UserFullName</b> – повне ім'я користувача. <b>UserName</b> – логін для входу. <b>PasswordHash</b> – хешований пароль. <b>RoleID</b> – зовнішній ключ на таблицю Roles.	<b>uuid</b> (унікальний ідентифікатор) <b>text</b> (текст)  <b>text</b> (текст)  <b>uuid</b> (унікальний ідентифікатор)
<b>Roles</b> – забезпечує розмежування доступу до функціоналу системи. Замість дублювання текстових значень у таблиці Users, використовується зовнішній ключ.	<b>RoleID</b> – унікальний ідентифікатор ролі. <b>RoleName</b> – назва ролі (наприклад, "admin", "cashier").	<b>uuid</b> (унікальний ідентифікатор) <b>text</b> (текст)
<b>Categories</b> – організація товарів за категоріями (наприклад, "Напої", "Молочні продукти", "Побутова хімія"). Це дає змогу фільтрувати та структурувати товари.	<b>CategoryID</b> – унікальний ідентифікатор категорії. <b>CategoryName</b> – назва категорії товарів.	<b>uuid</b> (унікальний ідентифікатор) <b>text</b> (текст)
<b>Products</b> – містить повну інформацію про всі товари. Тип Double використано для точної	<b>ProductID</b> – унікальний ідентифікатор товару. <b>ProductName</b> – назва товару. <b>CategoryID</b> – зовнішній ключ	<b>uuid</b> (унікальний ідентифікатор) <b>text</b> (текст) <b>uuid</b> (унікальний

роботи з десятковими значеннями цін. Зовнішній ключ CategoryID забезпечує логічну класифікацію товарів.	на Categories. <b>Price</b> – ціна товару  <b>SKU</b> – артикул/штрихкод. <b>CreatedAt</b> – дата додавання товару.	ідентифікатор) <b>double</b> (дробове число) <b>text</b> (текст) <b>datetime</b> (дата й час)
<b>Inventory</b> – зберігає інформацію про залишки товарів у конкретній торговій точці. Спільне використання StoreID і ProductID дозволяє контролювати запаси по кожному магазину.	<b>InventoryID</b> – унікальний ідентифікатор ролі. <b>StoreID</b> – зовнішній ключ на Stores. <b>ProductID</b> – зовнішній ключ на Products. <b>Quantity</b> – кількість товару в наявності.	<b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>integer</b> (числовий)
<b>Sales</b> – таблиця фіксує всі транзакції продажу. Додаткові поля забезпечують історію операцій і аналітику.	<b>SaleID</b> – унікальний ідентифікатор продажу. <b>StoreID</b> – магазин, у якому здійснено продаж. <b>UserID</b> – користувач, який оформив продаж. <b>Total</b> – загальна сума покупки.  <b>CreatedAt</b> – дата та час продажу.	<b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>double</b> (дробове число) <b>datetime</b> (дата й час)
<b>SaleProducts</b> – деталізує продаж – які саме товари були продані, в якій кількості та за якою ціною. Це дозволяє формувати аналітику та звіти.	<b>SaleProductID</b> – унікальний ідентифікатор запису. <b>SaleID</b> – зовнішній ключ на Sales. <b>ProductID</b> – товар, що був проданий. <b>Quantity</b> – кількість одиниць товару. <b>Price</b> – ціна на момент продажу.	<b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>integer</b> (числовий) <b>double</b> (дробове число)

Загальні принципи проектування:

- **типи даних uuid**[10] обрано для унікальності та захисту від зіткнень при  
розділеному доступі.

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 ПЗ**

Лис  
21

- **text** застосовується там, де немає обмеження на довжину або потрібно зберігати довільний рядок.
- **double precision** дозволяє уникати помилок округлення при грошових операціях (у майбутньому можна замінити на numeric(10, 2) для кращої точності).
- **використання зовнішніх ключів** гарантує цілісність даних.

Таким чином, спроектована база даних є логічною, масштабованою та повністю відповідає завданням автоматизації управління мережею торгових точок.

Після проєктування структури бази даних важливо перевірити її на відповідність нормальним формам[11], що дозволяють мінімізувати надмірність даних, уникнути логічних помилок та забезпечити коректність операцій над таблицями. Для цього проаналізуємо, чи відповідає створена база даних основним нормальним формам — від першої до третьої.

### 1. 1НФ — Перша нормальна форма

Виконується для всіх таблиць, бо:

- кожна колонка містить атомарне значення (немає списків або масивів).
- значення мають одинаковий тип у кожному стовпці.
- є унікальні первинні ключі (uuid), які однозначно ідентифікують кожен рядок.

### 2. 2НФ — Друга нормальна форма

Виконується, оскільки:

- у таблицях немає складених первинних ключів, тобто всім ключам — uuid, які самі по собі є повними ключами.
- жоден неключовий атрибут не залежить тільки від частини ключа, тому часткова залежність виключена.

### 3. 3НФ — Третя нормальна форма

Виконується, тому що:

- усі неключові поля залежать безпосередньо від первинного ключа.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

22

- наприклад, у таблиці Products, ProductName, Price, SKU — усі прямо залежать від ProductID.
- у Users, UserName, UserFullName, PasswordHash залежать від UserID. Поля типу RoleID, CategoryID, StoreID — зовнішні ключі, і логічно відокремлені в окремі таблиці (Roles, Categories, Stores) — немає транзитивних залежностей.

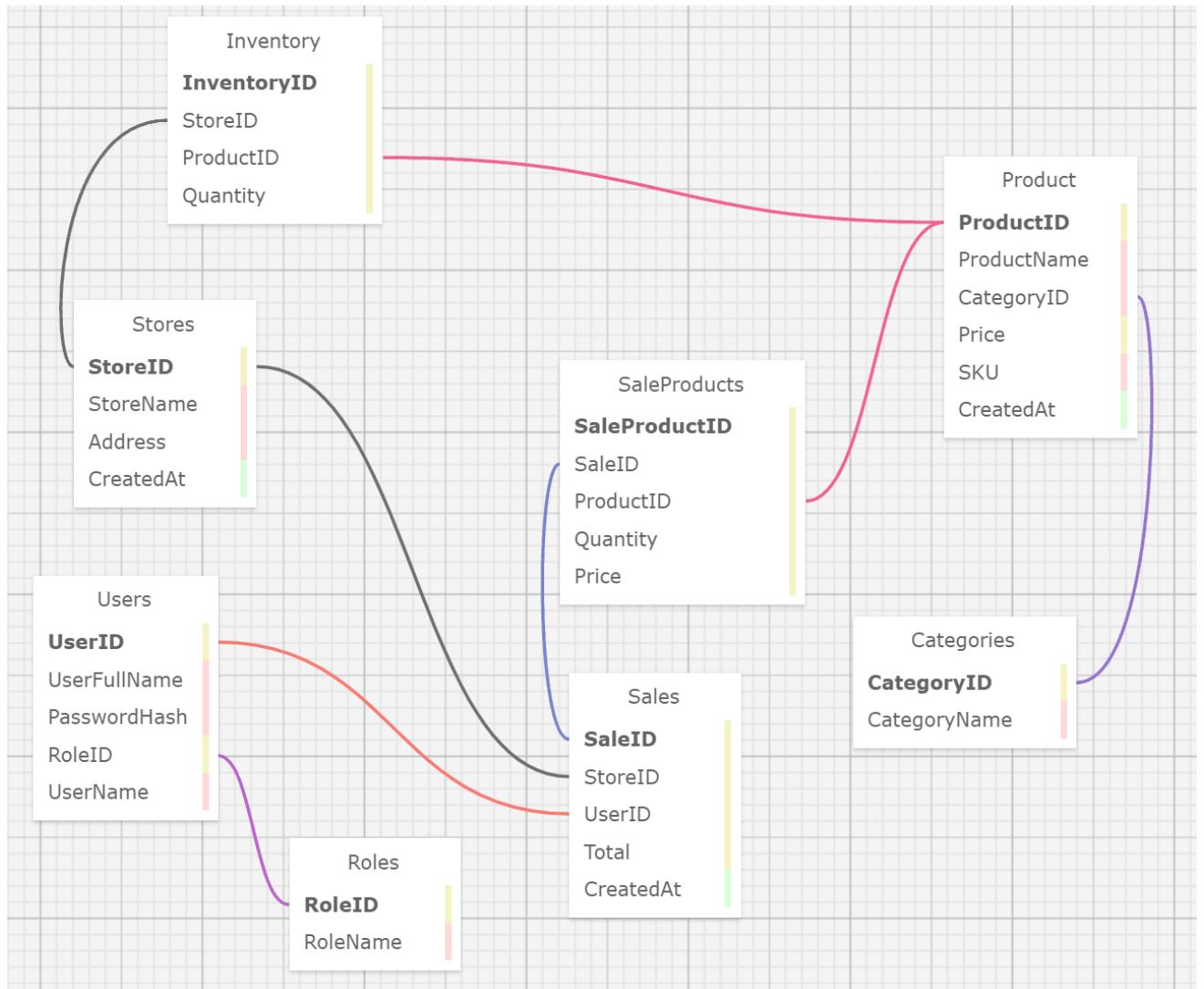


Рисунок 2.4 – Схема бази даних

## 2.5. Проектування інтерфейсів користувача

Інтерфейс користувача відіграє ключову роль у забезпеченні ефективної взаємодії між користувачем і комп’ютерною системою управління мережею торгових точок. У проектованій системі було прийнято рішення реалізувати

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
23

інтерфейс за допомогою фреймворку .NET Windows Forms, що дозволяє створювати Desktop застосунки для операційної системи Windows.

Основні вимоги до інтерфейсу:

1. Інтуїтивно зрозуміле розташування елементів;
2. Адаптивність до різних розмірів екранів;
3. Чіткий поділ на функціональні блоки;
4. Доступ до основних функцій у декілька кліків;
5. Відповідність ролям користувачів (адміністратор, касир, менеджер тощо).

Основні екрани застосунку:

1. Головне меню – стартовий екран після авторизації, що містить доступ до основних модулів системи (продажі, товари, працівники, аналітика, звіти тощо).
2. Сторінка продажів – дозволяє касиру швидко здійснювати продажі, обирати товари зі списку або через пошук/сканер, обробляти оплату.
3. Каталог товарів – дає змогу переглядати список товарів, додавати нові позиції, редагувати або видаляти наявні.
4. Керування категоріями – забезпечує організацію товарів за категоріями для зручності навігації та звітності.
5. Список користувачів і ролей – використовується адміністратором для створення, редагування, призначення ролей і контролю доступу.
6. Аналітика та звіти – дозволяє формувати динамічні таблиці та графіки, експортувати інформацію у форматах HTML або XLSX.
7. Форма авторизації – екран входу в систему з перевіркою облікових даних користувача.

Для покращення UX/UI були враховані загальноприйняті принципи дизайну: логічне групування елементів, використання іконок, контрастної палітри кольорів, а також мінімізація кількості кроків для виконання рутинних операцій.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

24

Важливо, що вся взаємодія з даними відбувається через API, що дозволяє зберігати інтерфейс максимально швидким і динамічним. Крім того, інтерфейс підтримує обробку повідомлень про помилки, підтвердження дій та відображення стану операцій (наприклад, спінер при завантаженні або повідомлення про успішну зміну даних).

## 2.6. Вибір технологій реалізації

На етапі розробки комп’ютерної системи управління мережею торгових точок було обрано сучасний стек технологій, який забезпечує продуктивність, масштабованість та зручність у розробці.

### 2.6.1. Вибір мови програмування та фреймворків

Основною мовою програмування в проекті обрано C#[12]. Це сучасна мова, яка поєднує об’єктно-орієнтований підхід, високу продуктивність та зручний синтаксис. Вона активно підтримується корпорацією Microsoft та має широку екосистему бібліотек, що робить її ідеальним вибором для розробки як клієнтської, так і серверної частини системи.

### 2.6.2. Вибір СУБД

Як систему управління базами даних було обрано PostgreSQL[13] — одну з найпопулярніших реляційних СУБД з відкритим кодом. Причини такого вибору:

- підтримка складних запитів і транзакцій;
- високий рівень відповідності стандарту SQL;
- розшируваність (можливість створювати власні типи, функції тощо);
- стабільна робота при великих обсягах даних;
- добра інтеграція з .NET через Entity Framework Core та Npgsql-провайдер.

PostgreSQL є оптимальним рішенням для систем, де важливі цілісність, надійність та масштабованість даних.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

25

### 2.6.3. Вибір інструментів створення клієнтського додатку

Для створення клієнтського застосунку було обрано Windows Forms (WinForms)[14] — один із найпопулярніших фреймворків для створення настільних застосунків під Windows. Його переваги:

- простота розробки — інтуїтивно зрозумілий підхід до побудови інтерфейсів за допомогою drag-and-drop редактора;
- швидка побудова UI — підтримка візуального дизайнера у середовищі Visual Studio дозволяє швидко створювати інтерфейси. Ця властивість буде корисною в разі потреби оновлення додатку або розширення функціоналу.
- широка підтримка елементів управління — вбудовані компоненти, такі як DataGridView, TextBox, ComboBox, дозволяють реалізувати функціональний інтерфейс без додаткових бібліотек;
- глибока інтеграція з Windows — фреймворк дозволяє легко взаємодіяти з файлами, реєстром, системними діалогами тощо;
- стабільність і перевіреність часом — WinForms використовується в промислових рішеннях десятиліттями та добре себе зарекомендував.

### 2.6.4. Вибір технологій створення серверної частини

Серверна частина реалізована на базі ASP.NET Core[15], сучасного фреймворку для створення вебзастосунків і API. Його головні переваги — модульна архітектура, висока продуктивність, підтримка асинхронного програмування, кросплатформеність і зручна інтеграція з базами. Для розробки додатку використані технології:

- RESTful API — для забезпечення структурованого обміну даними між клієнтом і сервером через HTTP-запити (GET, POST, PUT, DELETE);
- Entity Framework Core (EF Core) — ORM для роботи з базою даних, що дозволяє працювати з об'єктами, а не з SQL-запитами напряму;

Зм	Лист	№ докум.	Підп.	Лата

- Npgsql — офіційний .NET-драйвер для PostgreSQL;

Також у серверній частині передбачена асинхронна обробка запитів, що дозволяє ефективно обробляти одночасні звернення від кількох клієнтів без блокування основного потоку.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
27

### 3. РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК

#### 3.2. Реалізація серверної частини засобами ASP.NET

Серверна частина програмного комплексу реалізована у вигляді вебзастосунку на базі ASP.NET Core Web API, що забезпечує зручний та ефективний спосіб обміну даними між клієнтською частиною та базою даних. Розробка велась у середовищі Visual Studio, а структура рішення відображає розділення відповідальностей за принципами чистої архітектури. На рисунку 3.1 зображене структуру проєкту серверної частини. Назвою серверного додатку є RON.WebAPI, де RON означає Retail Outlet Network – мережа торгових точок.

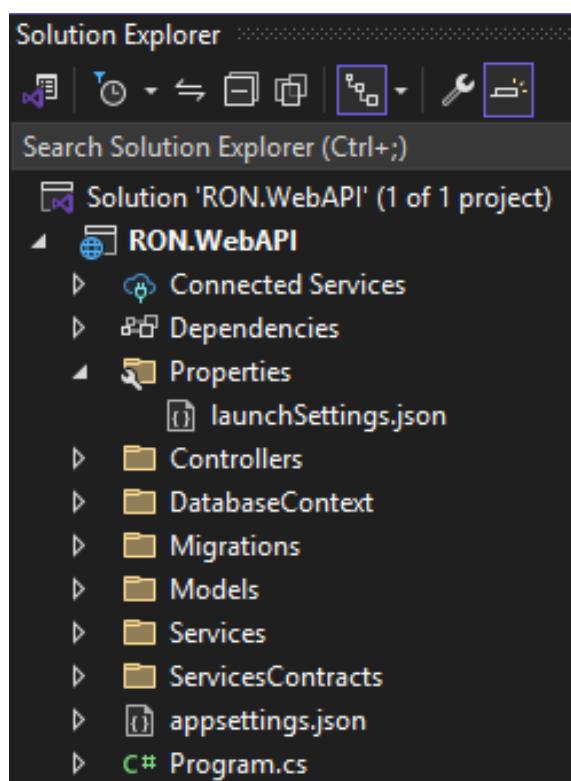


Рисунок 3.1 – Структура проєкту серверної частини RON.WebAPI у Visual Studio

Основними складовими структури є Controllers, DatabaseContext, Migrations, Models, Services, appsettings.json та Program.cs.

Controllers — каталог, що містить API-контролери, які обробляють HTTP-запити від клієнта. Кожен контролер відповідає за певну сутність (наприклад, ProductsController, UsersController) і реалізує відповідні методи для роботи з нею.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
28

(Get, Post, Put, Delete). На рисунку 3.2 зображено перелік всіх контролерів, що використовуються в даному проекті. Кожен контролер пов'язаний із окремою таблицею в базі даних.

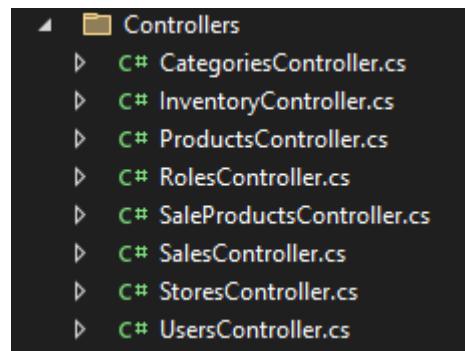


Рисунок 3.2 – Перелік всіх передбачених контролерів

DbContext — містить клас контексту бази даних (звичай ApplicatonDbContext), який є головною точкою взаємодії з базою через ORM Entity Framework Core. Тут визначено DbSet-и для кожної таблиці та методи конфігурації моделі. DbSet представляє собою набір сущностей певного типу, з якими можна працювати в контексті бази даних. Він дозволяє виконувати запити до таблиць, а також додавати, змінювати або видаляти записи. Об'єкти DbSet оголошуються у класі DbContext дляожної сущності, яку потрібно зберігати в базі даних. На рисунку 3.3 зображено складову DbContext.

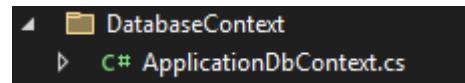


Рисунок 3.3 – Складова API DbContext

Migrations — каталог, що містить міграції бази даних. Міграції — це інструмент, який дає змогу керовано змінювати структуру бази даних відповідно до змін у коді програми. Вони дозволяють створювати таблиці, змінювати стовпці, ключі та інші об'єкти БД. Також їхня користь проявляється у випадку використання ORM, наприклад Entity Framework у C#. Завдяки міграціям можна:

- формувати структуру бази даних безпосередньо з коду (підхід code first),
- вносити зміни до схеми бази без втрати існуючих даних,
- підтримувати відповідність між моделями в коді та реальною базою даних.

Зм	Лист	№ докум.	Підп.	Лата

Файл міграції — це спеціальний програмний файл, який містить інструкції для зміни структури бази даних. Його створює на основі змін у моделях (класах). На рисунку 3.4 зображено перелік всіх файлів міграцій, створених у проєкті. Назви цих файлів складаються з мітки часу та імені міграції, яке розробник (або Visual Studio) задає при створенні.

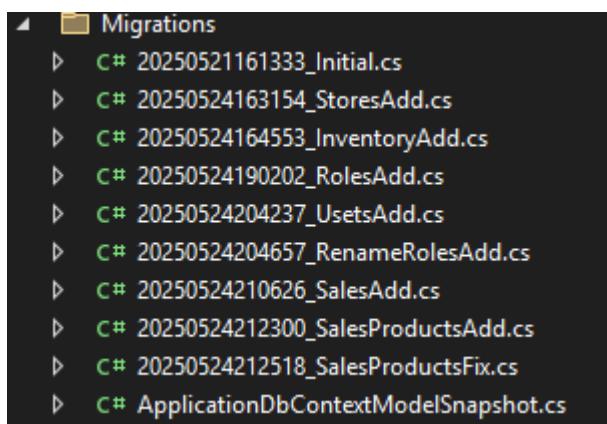


Рисунок 3.4 – Перелік файлів міграцій.

Models — містить опис сутностей (класи, що відображають таблиці в базі даних). Наприклад, Product, User, Store, кожна з яких відповідає відповідному набору полів у таблиці. На рисунку 3.5 зображено перелік всіх задіяних класів моделей.

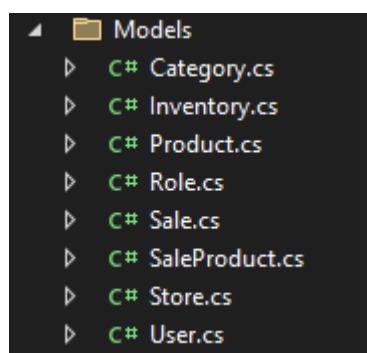


Рисунок 3.5 – Перелік класів моделей.

Services — реалізує бізнес-логіку системи. Сервіси виконують валідацію та обробку даних, взаємодію з базою даних, перевірку умов, обчислення тощо. На рисунку 3.6 зображено перелік класів сервісів, що були розроблені під час виконання роботи.

Зм	Лист	№ докум.	Підп.	Дата

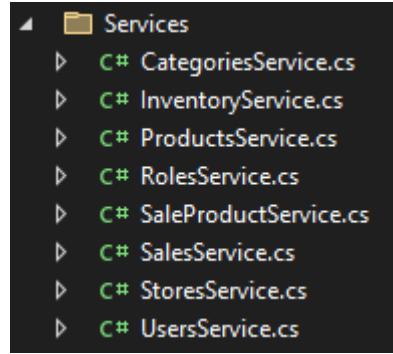


Рисунок 3.6 – Перелік класів сервісів.

ServicesContracts — інтерфейси до сервісів. Вони визначають контракти (угоди), які реалізуються в папці Services, що дозволяє легко змінювати реалізацію. Інтерфейси у високорівневих мовах програмування визначають, які методи та властивості повинні мати класи, що реалізують цей інтерфейс. Таким чином, вони забезпечують гнучкість та повторне використання коду, дозволяючи різним класам взаємодіяти через загальний стандарт. На рисунку 3.7 зображено перелік всіх необхідних інтерфейсів сервісів.

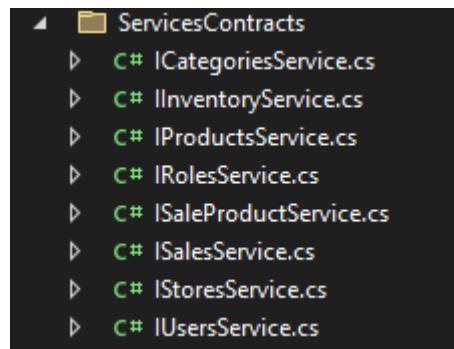


Рисунок 3.7 – Перелік інтерфейсів сервісів.

appsettings.json — файл конфігурації застосунку. Тут зберігаються параметри підключення до бази даних, конфігурації логування, шляхи до ресурсів тощо. На рисунку 3.8 зображено складову appsettings.json.

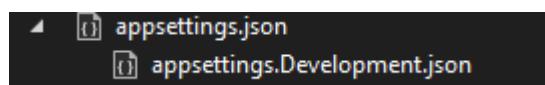


Рисунок 3.8 – Компонент appsettings.json.

Program.cs — головний вхідний файл застосунку, в якому налаштовується запуск вебсервера, реєструються сервіси в контейнері залежностей, додаються middleware-компоненти. Контейнер залежностей (Dependency Injection container)

Зм	Лист	№ докум.	Підп.	Дата

у Program.cs — це центральне місце, де реєструються сервіси і об'єкти, які потім використовуються в різних частинах програми (наприклад, у контролерах, формах, сервісах тощо). Middleware-компоненти — це проміжні програмні модулі, які виконуються послідовно під час обробки HTTP-запиту в ASP.NET Core. Вони можуть обробляти запити (наприклад, перевірка авторизації), змінювати запити чи відповіді, вирішувати, чи передавати запит далі по ланцюжку.

### 3.1.2. Реалізація контролерів та обробка запитів

Перед розглядом контролерів необхідно зрозуміти логіку побудови моделей, на основі яких і працює серверна частина. Усі моделі в системі репрезентовані в таблиці 3.1 і відповідають за структуру даних, які обробляються.

Таблиця 3.1 – Опис моделей додатку " RON.WebAPI "

Клас моделі	Поля (властивості)	Тип поля
<b>Category</b> — модель для категорій товарів.	<b>CategoryID</b> — унікальний ідентифікатор категорії. <b>CategoryName</b> — назва категорії (наприклад, "Напої", "Овочі").	<b>Guid</b> (унікальний ідентифікатор) <b>string</b> (текст)
<b>Inventory</b> — облік товарів у магазині.	<b>InventoryID</b> — унікальний ідентифікатор запису. <b>StoreID</b> — зовнішній ключ, що посилається на магазин. <b>ProductID</b> — зовнішній ключ, що посилається на товар. <b>Quantity</b> — кількість одиниць даного товару в магазині. <b>Store</b> — навігаційна властивість до магазину. <b>Product</b> — навігаційна властивість до товару.	<b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>uuid</b> (унікальний ідентифікатор) <b>int</b> (ціле число)  <b>Store</b> (об'єкт класу Store) <b>Product</b> (об'єкт класу Product)
<b>Product</b> — описує товар у системі.	<b>ProductID</b> — унікальний ідентифікатор товару. <b>ProductName</b> — назва	<b>uuid</b> (унікальний ідентифікатор) <b>string</b> (текст)

	<p>товару.</p> <p><b>CategoryID</b> — зовнішній ключ до категорії.</p> <p><b>Price</b> — ціна товару.</p> <p><b>SKU</b> — артикул (унікальний код для товару).</p> <p><b>CreatedAt</b> — дата створення товару.</p> <p><b>Category</b> — навігаційна властивість до категорії.</p>	<p><b>uuid</b> (унікальний ідентифікатор)</p> <p><b>double</b> (дійсне число)</p> <p><b>string</b> (текст)</p> <p><b>DateTime</b> (дата та час)</p> <p><b>Category</b> (об'єкт класу Category)</p>
<b>Role</b> — ролі користувачів.	<p><b>RoleID</b> — унікальний ідентифікатор ролі.</p> <p><b>RoleName</b> — назва ролі (наприклад, "Адміністратор", "Касир").</p>	<p><b>Guid</b> (унікальний ідентифікатор)</p> <p><b>string</b> (текст)</p>
<b>Sale</b> — інформація про здійснений продаж.	<p><b>SaleID</b> — унікальний ідентифікатор продажу.</p> <p><b>StoreID</b> — зовнішній ключ до магазину.</p> <p><b>UserID</b> — зовнішній ключ до користувача, який здійснив продаж.</p> <p><b>Total</b> — загальна сума продажу.</p> <p><b>CreatedAt</b> — дата продажу.</p> <p><b>Store</b> — навігаційна властивість до магазину.</p> <p><b>User</b> — навігаційна властивість до користувача.</p>	<p><b>uuid</b> (унікальний ідентифікатор)</p> <p><b>uuid</b> (унікальний ідентифікатор)</p> <p><b>uuid</b> (унікальний ідентифікатор)</p> <p><b>double</b> (дійсне число)</p> <p><b>DateTime</b> (дата та час)</p> <p><b>Store</b> (об'єкт класу Store)</p> <p><b>User</b> (об'єкт класу User)</p>
<b>SaleProduct</b> — товари, що входять до конкретного продажу.	<p><b>SaleProductID</b> — унікальний ідентифікатор.</p> <p><b>SaleID</b> — зовнішній ключ до продажу.</p> <p><b>ProductID</b> — зовнішній ключ до товару.</p> <p><b>Quantity</b> — кількість одиниць товару в продажі.</p> <p><b>Price</b> — ціна на момент продажу.</p>	<p><b>uuid</b> (унікальний ідентифікатор)</p> <p><b>uuid</b> (унікальний ідентифікатор)</p> <p><b>uuid</b> (унікальний ідентифікатор)</p> <p><b>int</b> (ціле число)</p> <p><b>double</b> (дійсне число)</p>

	<b>Sale</b> — навігаційна властивість до продажу. <b>Product</b> — навігаційна властивість до товару.	<b>Sale</b> (об'єкт класу Sale) <b>Product</b> (об'єкт класу Product)
<b>Store</b> — магазини мережі.	<b>StoreID</b> — унікальний ідентифікатор магазину. <b>StoreName</b> — назва магазину. <b>Address</b> — адреса магазину. <b>CreatedAt</b> — дата створення магазину.	<b>uuid</b> (унікальний ідентифікатор) <b>string</b> (текст)  <b>string</b> (текст) <b>DateTime</b> (дата та час)
<b>User</b> — користувачі системи.	<b>UserID</b> — унікальний ідентифікатор користувача. <b>UserFullName</b> — повне ім'я користувача. <b>UserName</b> — логін. <b>PasswordHash</b> — захешований пароль. <b>RoleID</b> — зовнішній ключ до ролі. <b>Role</b> — навігаційна властивість до ролі.	<b>uuid</b> (унікальний ідентифікатор) <b>string</b> (текст)  <b>string</b> (текст)  <b>uuid</b> (унікальний ідентифікатор) <b>Role</b> (об'єкт класу Role)

Після створення класів моделей можна перейти до реалізації сервісів та контролерів. Для реалізації контролеру необхідно використання певних атрибутив для позначення методів класу. Атрибут у C# — це спеціальний тип метаданих, який додається до класів, методів, властивостей, параметрів або інших елементів програми для впливу на їх поведінку або для надання додаткової інформації. У контексті ASP.NET Core Web API атрибути широко використовуються для налаштування маршрутів, контролю доступу, валідації даних тощо. Основні атрибути, використані у контролерах цього проєкту:

- `[Route("api/[controller]")]` визначає шаблон маршруту для всіх методів контролера.
- `"[controller]"` є плейсхолдером, який автоматично підставляє назгу контролера без суфікса "Controller". Наприклад, для `CategoriesController` шлях буде `api/categories`.

- [ApiController] позначає клас як API-контролер. Цей атрибут автоматично забезпечує перевірку вхідних моделей (ModelState), спрощує обробку HTTP-запитів, дозволяє використовувати атрибути маршрутизації без додаткової конфігурації, генерує відповіді 400 Bad Request, якщо модель невалідна.
- [HttpGet] визначає, що метод обробляє GET-запити. Застосовується для отримання списків або окремих об'єктів з сервера.
- [HttpGet("{ id }")] визначає GET-запит із параметром id. Наприклад, GET /api/categories/5 — повертає категорію з вказаним ідентифікатором.
- [HttpPost] визначає, що метод обробляє POST-запити, які зазвичай використовуються для створення нових об'єктів у базі даних.

Приклад реалізації контролера для роботи з сущістю Category наведено на рисунку 3.9.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

35

```

14  [Route("api/[controller]")]
15  [ApiController]
16  public class CategoriesController : ControllerBase
17  {
18      private readonly ICategoriesService _categoriesService;
19
20      public CategoriesController(ICategoriesService categoriesInterface)
21      {
22          _categoriesService = categoriesInterface;
23      }
24      // GET: api/Categories
25      [HttpGet]
26      public async Task<ActionResult<IEnumerable<Category>>> GetCategories()
27      {
28          return await _categoriesService.GetCategories();
29      }
30      // GET: api/Categories/5
31      [HttpGet("{id}")]
32      public async Task<ActionResult<Category>> GetCategory(Guid id)
33      {
34          var category = await _categoriesService.GetCategory(id);
35
36          if (category == null)
37          {
38              return NotFound();
39          }
40
41          return category;
42      }
43      // PUT: api/Categories/5
44      [HttpPut("{id}")]
45      public async Task<IActionResult> PutCategory(Guid id, Category category)
46      {
47          bool result = await _categoriesService.UpdateCategory(id, category);
48          if (result)
49          {
50
51              return NoContent();
52          }
53          else
54          {
55              return BadRequest("This category already exists");
56          }
57      }
58      // POST: api/Categories
59      [HttpPost]
60      public async Task<ActionResult<Category>> PostCategory(Category category)
61      {
62          bool result = await _categoriesService.AddCategory(category);
63          if (result)
64          {
65              return CreatedAtAction("GetCategory", new { id = category.CategoryID }, category);
66          }
67          else
68          {
69              return BadRequest("This category already exists");
70          }
71      }
72      // DELETE: api/Categories/5
73      [HttpDelete("{id}")]
74      public async Task<IActionResult> DeleteCategory(Guid id)
75      {
76          bool result = await _categoriesService.DeleteCategory(id);
77          if (result)
78          {
79              return NoContent();
80          }
81          else
82          {
83              return BadRequest();
84          }
85      }
86  }

```

Рисунок 3.9 – Приклад реалізації класу CategoriesController.cs.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 П3**

Лис  
36

З прикладу видно, що контролери працюють як endpoints для запитів. Контролер звертається до сервісу, передаючи дані для обробки, валідації та передання в базу даних, або ж навпаки задля отримання даних від сервісу.

### 3.1.3. Реалізація доступу до бази даних засобами EF Core

Entity Framework Core (EF Core) — це сучасний об'єктно-реляційний диспетчер (ORM) з відкритим вихідним кодом, розроблений компанією Microsoft. Він дозволяє .NET-розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід замість прямого написання SQL-запитів. EF Core автоматично перетворює операції над об'єктами у відповідні SQL-запити до бази даних.

У використанні EF Core є вагома кількість переваг:

- зменшення кількості коду. Замість великої кількості SQL-запитів розробник працює з об'єктами та колекціями. CRUD-операції (створення, читання, оновлення, видалення) реалізуються просто і зрозуміло.
- підвищення читабельності та підтримуваності коду. Операції з базою даних виглядають як звичайні виклики методів, що полегшує супровід і розширення коду.
- безпечна робота з параметрами (захист від SQL-ін'єкцій). EF Core автоматично захищає від SQL-ін'єкцій, використовуючи параметризовані запити.
- підтримка міграцій. EF Core дозволяє керувати схемою бази даних безпосередньо з коду. Зміни у моделях зручно переносити в базу даних за допомогою механізму міграцій.
- кросплатформеність. EF Core підтримує роботу з різними СУБД (SQL Server, PostgreSQL, MySQL тощо) і може використовуватись у будь-яких .NET-застосунках, включно з десктопними, веб та мобільними.

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 ПЗ**

Лис  
37

Хоча використання прямих SQL-запитів (через ADO.NET або Dapper) є гнучким і може забезпечити дещо вищу продуктивність в окремих випадках, такий підхід має ряд недоліків:

- більша кількість шаблонного коду;
- вища ймовірність помилок при ручному формуванні запитів;
- складніше оновлювати та змінювати структуру таблиць;
- немає автоматичного відображення зв'язків між таблицями;
- більша складність у тестуванні та розширенні системи.

EF Core, натомість, дозволяє працювати з об'єктами, а не з таблицями напряму. Також обумовлена можливість зручно описувати зв'язки між сущностями (наприклад, один-до-багатьох або багато-до-багатьох), не думаючи про дрібниці, як-от відкриття/закриття з'єднань, ручну обробку результатів запитів тощо. Додатково можна відмітити легкість масштабувати коду, оскільки логіка не прив'язана до конкретної СУБД.

У підсумку можна зазначити, що застосування EF Core у проекті дозволяє реалізувати надійний, чистий та ефективний доступ до бази даних з мінімальними витратами часу. Незважаючи на те, що пряме використання SQL могло б забезпечити більше контролю над виконанням запитів, у більшості випадків EF Core є кращим вибором, особливо на етапах активної розробки та підтримки системи.

Реалізацію доступу до бази даних можна описати на прикладі коду одного з реалізованих сервісів. На рисунку 3.10 зображено конструктор, методи AddProduct(), DeleteProduct(), GetProduct(id) та GetProduct() з класу сервісу ProductsService. Представлено реалізацію чотирьох методів класу, наведено приклади алгоритмів валідації даних. Методи реалізують додавання, видалення та отримання продуктів відповідно. В коментарях покроково описані дії.

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 ПЗ**

Лис

38

```

8  public class ProductsService : IProductsService
9  {
10     private readonly ApplicationDbContext _context; // Ін'єкція класу DbContext зв'язку з БД
11     public ProductsService(ApplicationDbContext context) // Конструктор класу
12     {
13         _context = context;
14     }
15     public async Task<bool> AddProduct(Product product) // Метод додавання продукту
16     {
17         List<Product>? products = await GetProducts(); // Отримання всіх існуючих продуктів
18         if (products != null)
19         {
20             foreach (Product prod in products)
21             {
22                 if (prod.ProductName == product.ProductName && // Перевірка, чи не існує
23                     prod.Price == product.Price && // такого ж самого продукту
24                     prod.SKU == product.SKU && // в базі даних.
25                     prod.CategoryID == product.CategoryID)
26                 {
27                     return false;
28                 }
29                 if (product.Price <= 0) // Перевірка, чи не є ціна
30                     // продукту від'ємною
31                 return false;
32             }
33         }
34         _context.products.Add(product); // Додавання продукту
35         try // Спроба зберегти змінні в БД
36         {
37             await _context.SaveChangesAsync(); // Збереження змін
38             return true;
39         }
39         catch (DbUpdateConcurrencyException)
40         {
41             return false;
42         }
43     }
44 }
45 public async Task<bool> DeleteProduct(Guid id) // Метод видалення продукту
46 {
47     var product = await _context.products.FindAsync(id);
48     if (product == null) // Перевірка, чи існує в БД продукт,
49     // який треба видалити
50     {
51         return false;
52     }
53     _context.products.Remove(product); // Видалення продукту
54     await _context.SaveChangesAsync(); // Збереження змін
55     return true;
56 }
57 public async Task<Product?> GetProduct(Guid id) // Метод отримання продукту
58 // за ідентифікатором
59 {
60     var product = await _context.products.Include("Category") // Пошук продукту в БД
61     .FirstOrDefaultAsync(temp => temp.ProductID == id); // Перевірка на нуль
62     if (product == null)
63     {
64         return null;
65     }
66     return product; // Повернення результату
67 }
68 public async Task<List<Product>?> GetProducts() // Метод отримання всіх продуктів
69 {
70     List<Product> products = await _context.products // Отримання колекції всіх продуктів
71     .Include("Category").ToListAsync();
72     if (products == null) // Перевірка на нуль
73     {
74         return null;
75     }
76     else
77     {
78         return products; // Повернення результату
79     }
80 }

```

Рисунок 3.10 – Конструктор, AddProduct(), DeleteProduct(), GetProduct(id) та GetProduct() з класу сервісу ProductsService.cs.

Під час розробки коду було підключено простір імен Microsoft.EntityFrameworkCore. Саме цей простір відповідає за ORM. Властивість

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 П3**

Лис  
39

класу `_context` містить в собі колекції об'єктів, які в свою чергу містять дані з таблиць бази даних. У коді продемонстровано можливість звертатись до таблиць як до звичайних типізованих колекцій мови C#, а також застосовувати стандартні функції для роботи з колекціями (`List<T>`), такі як `Add()`, `Remove()`, `FirstOrDefaultAsync()`, `Include()` та `ToListAsync()`.

У мові програмування C#, `List<T>` — це динамічна колекція, яка дозволяє зберігати об'єкти у вигляді списку. На відміну від масивів, які мають фіксовану довжину, список може змінювати свій розмір під час виконання програми. Одним із основних методів цього класу є `Add()`. Цей метод додає елемент до кінця списку `List<T>`.

Метод `Remove()` використовується для видалення певного елемента з колекції, зокрема зі списку (`List<T>`). Він шукає заданий об'єкт у колекції і, якщо знаходить його, видаляє. Повертає булеве значення, яке вказує, чи було видалення успішним.

Метод `FirstOrDefaultAsync()` використовується в Entity Framework Core для асинхронного отримання першого елемента з колекції, що відповідає умові (якщо така задана). Якщо жоден елемент не знайдено, повертає значення за замовчуванням (наприклад, `null` для класів). Працює асинхронно, тому не блокує потік виконання.

Метод `Include()` використовується в EF Core для жадного завантаження пов'язаних даних. Дозволяє явно вказати, які пов'язані сутності (через зовнішні ключі) потрібно завантажити разом із головною сутністю. Це зручно для уникнення додаткових запитів до бази даних.

Метод `ToListAsync()` перетворює результат запиту LINQ у Entity Framework Core на асинхронний список (`List<T>`). Завантажує усі елементи з бази даних у список і повертає його, не блокуючи основний потік завдяки асинхронності.

На рисунку 3.11 представлено методи `UpdateProduct()` та `ProductExists()`, які створені для оновлення та перевірки існування продукту відповідно.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
40

```

81     public async Task<bool> UpdateProduct(Guid id, Product product) // Метод оновлення
82     {
83         if (id != product.ProductID)                                // перевірка чи є передані
84         {                                                       // ідентифікатори однаковими
85             return false;
86         }
87         Product? existingProduct = await _context.products.FindAsync(id); // Пошук існуючого
88         if (existingProduct == null)                                     // продукту, дані
89         {                                                       // якого треба
90             return false;                                              // оновити
91         }
92         List<Product>? products = await GetProducts();           // отримання всіх продуктів
93         if (product.Price <= 0)                                     // Якщо в БД відсутні продукти,
94         {                                                       // то дані для оновлення відсутні
95             return false;
96         }
97         if (products != null)
98         {
99             foreach (Product prod in products)                      // перевірка, чи не описанеться
100            {                                                      // в базі даних два однакових
101                if (prod.ProductName == product.ProductName && // продуктів після оновлення
102                    prod.Price == product.Price &&
103                    prod.SKU == product.SKU &&
104                    prod.CategoryID == product.CategoryID)
105                {
106                    return false;
107                }
108            }
109        }
110    else
111    {
112        return false;
113    }
114    existingProduct.ProductName = product.ProductName;          // Оновлення даних
115    existingProduct.Price = product.Price;                         // існуючого запису таблиці
116    existingProduct.SKU = product.SKU;
117    existingProduct.CategoryID = product.CategoryID;
118
119    try                                         // спроба зберегти зміни
120    {
121        await _context.SaveChangesAsync();
122        return true;
123    }
124    catch (DbUpdateConcurrencyException) // Відловлювання виняткових ситуацій
125    {
126        if (!await ProductExists(id))
127        {
128            return false;
129        }
130        else
131        {
132            throw;
133        }
134    }
135 }
1 reference
136 private async Task<bool> ProductExists(Guid id) // Метод для перевірки снування продукту
137 {
138     return _context.products.Any(e => e.ProductID == id);
139 }

```

Рисунок 3.11 – Методи UpdateProduct() та ProductExists() з класу сервісу ProductsService.cs.

### 3.2. Реалізація клієнтського застосунку засобами .NET WinForms

#### 3.2.1. Головне меню та структура інтерфейсу

Windows Forms (WinForms) надає набір стандартних графічних елементів керування (control elements), які використовуються для створення зручного і функціонального інтерфейсу користувача. Нижче наведено основні з них:

1. Form (Форма).

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 ПЗ**

Лис  
41

- основне вікно застосунку або окреме вікно в межах програми. Може містити інші елементи інтерфейсу.
  - Має властивості для налаштування зовнішнього вигляду, розміру, заголовку тощо.
2. Label
- відображає текстову інформацію.
  - використовується для підписів до полів, заголовків, пояснень.
3. TextBox
- поле для введення тексту користувачем.
  - часто використовується для введення логіну, назви товару, кількості тощо.
4. Button
- кнопка, яка викликає певну дію при натисканні (наприклад, "Зберегти", "Додати", "Видалити").
  - обробка відбувається через подію Click.
5. ComboBox
- випадний список, з якого можна вибирати одне значення.
  - зручно використовувати для вибору категорії, магазину тощо.
6. CheckBox
- елемент для вибору або зняття вибору (галочки).
  - дає змогу вибирати кілька параметрів одночасно.
7. RadioButton
- схожий на CheckBox, але дозволяє вибрати лише один варіант із групи.
8. DataGridView
- таблиця для відображення та редагування колекцій даних (наприклад, списку товарів, користувачів, продажів).
  - підтримує пагінацію, сортування, вибір рядків, редагування клітинок.

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 ПЗ**

Лис  
42

## 9. MenuStrip / ToolStrip

- MenuStrip: панель меню у верхній частині форми (Файл, Довідка, Налаштування тощо).
- ToolStrip: панель із кнопками (іконками) для швидкого доступу до основних функцій (як у Word або Excel).

## 10. Panel / GroupBox

- Panel: контейнер для групування елементів.
- GroupBox: схожий на Panel, але має заголовок. Зручно об'єднувати пов'язані елементи.

## 11. Timer

- компонент для створення подій через певні проміжки часу (наприклад, автоматичне оновлення даних).

## 12. PictureBox

- використовується для відображення зображень.

Дані компоненти можна перетягувати на форму за допомогою візуального конструктора у Visual Studio або створювати вручну в коді. Їхня взаємодія організовується через події, наприклад Click,TextChanged, SelectedIndexChanged тощо. Всі елементи було додано на форму за допомогою візуального конструктора.

В даній роботі інтерфейс складається з наступних компонентів: Form, MenuStrip / ToolStrip, TextBox, ComboBox, Button, Label, DataGridView та PictureBox. До кожного елементу прив'язані певні події.

При натисканні на будь-який елемент MenuStrip на основній формі відображається DataGridView, в якому відображаються дані, відповідно до пункту меню. В цей же час у всіх кнопок, міток, текстбоксів та інших елементів, що не стосуються цього пункту меню, змінюється властивість visible на false – тобто всі ці елементи перестають відображатись на основній формі. Приклад функції, яка відображає всі необхідні компоненти для роботи зі списком продуктів після натискання на пункт меню Products наведено на рисунку 3.12.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
43

```

395     ^ reference
396
397     private async void productsToolStripMenuItem_Click(object sender, EventArgs e)
398     {
399         dataGridView1.Visible = true;      // Вимикаються всі інші кнопки,
400         AddCategButton.Visible = false;    // що не стосуються пункту меню
401         button2.Visible = false;          // Products.
402         btnUpdCateg.Visible = false;
403
404         btnStoreAdd.Visible = false;
405         btnStoreDelete.Visible = false;
406         btnStoreUpdate.Visible = false;
407         showAllProducts.Visible = false;
408         showAllSales.Visible = false;
409
410         btnRoleAdd.Visible = false;
411         btnRoleDelete.Visible = false;
412         btnRoleUpdate.Visible = false;
413
414         btnUserAdd.Visible = false;
415         btnUserDelete.Visible = false;
416         btnUserUpdate.Visible = false;
417
418         btnAddOperation.Visible = false;
419         btnDeleteOperation.Visible = false;
420
421         btnExportCategoriesExcl.Visible = false;
422         prodExpExc.Visible = false;
423         storesExpExc.Visible = false;
424         allSalesInfo.Visible = false;
425
426         pictureBox1.Visible = false;
427         pictureBox2.Visible = false;
428
429         prodhtml.Visible = false;
430         catehtml.Visible = false;
431         storesexp.Visible = false;
432         allsaleshtml.Visible = false;
433
434         numberOfCateg.Visible = false;
435         numberOfProd.Visible = false;
436         numberOfStores.Visible = false;
437         numberOfUsers.Visible = false;
438         numberOfRows.Visible = false;
439
440         btnProductAdd.Visible = true;      // Активуються потрібні елементи
441         btnProductDelete.Visible = true;
442         btnProductUpdate.Visible = true;   // Завантажуються потрібні дані та
443         await ReadShowProducts();          // подаються у вигляді таблиці.
}

```

Рисунок 3.12 – Функція активації необхідних компонентів для роботи зі списком продуктів.

Всього доступно 7 пунктів меню:

1. Stores – забезпечує активацію графічних елементів для подачі списку торгових точок. Також надає можливість провести CRUD операцій до кожного магазину, а також переглянути інвентар або продажі обраної торгової точки.
2. Operations – активує показ списку проведених операцій. Також надає можливість провести CRUD операцій.
3. Products – представляє список продуктів, дає доступ до CRUD операцій.
4. Categories – надає доступ до операцій над доступними категоріями

Зм	Лист	№ докум.	Підп.	Дата

продуктів.

5. Reports – надає доступ до кнопок, які дозволяють завантажити звіти у форматі HTML або XLSX. Передбачена можливість експортувати табличні дані категорій, продуктів, магазинів та повної інформації про всі операції продажів. Також після натискання на пункт меню Reports виводяться основні дані про систему: кількість категорій, продуктів, магазинів, користувачів та здійснених операцій.
6. Roles – представляє список ролей користувачів, дає доступ до CRUD операцій
7. Users – відображає список користувачів, дає можливість проводити дії над персональними даними робочого персоналу.

### 3.2.2. Робота з API

У клієнтській частині програми, розробленій з використанням Windows Forms, зв'язок із серверною частиною (яка побудована на ASP.NET Core) здійснюється через веб API. Для цього застосовується клас HttpClient, який дозволяє надсилюти HTTP-запити до серверу і обробляти відповіді.

Для зручності й повторного використання запитів створюються окремі класи-репозиторії, наприклад, CategoryRepository, які інкапсулюють логіку взаємодії з API. Ці класи виконують такі завдання:

1. Надсилання HTTP-запитів.

Кожен метод репозиторію відповідає певному типу запиту до API:

- GetAsync() — отримання даних (GET)
- PostAsync() — створення нових записів (POST)
- PutAsync() — оновлення існуючих записів (PUT)
- DeleteAsync() — видалення записів (DELETE)

2. Серіалізація / десеріалізація.

Дані, які надсилаються на сервер, перетворюються у формат JSON за допомогою JsonSerializer.Serialize(). Отримані від сервера відповіді у

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
45

вигляді JSON-рядків конвертуються назад у об'єкти моделі через JsonSerializer.Deserialize().

На рисунку 3.13 зображене приклад коду класу-репозиторію CategoryRepository.

```
11  public class CategoryRepository
12  {
13      private readonly HttpClient _httpClient; // Ін'єкція класу HttpClient
14      public CategoryRepository() // Конструктор
15      {
16          _httpClient = new HttpClient();
17      }
18      public async Task<List<Category>> GetCategories() // Метод для отримання всіх
19      { // категорій з бази даних
20          var categories = new List<Category>();
21
22          var response = await _httpClient // Надсилання запиту
23              .GetAsync("https://localhost:7015/api/Categories"); // Посилання на API
24          response.EnsureSuccessStatusCode();
25
26          var json = await response.Content.ReadAsStringAsync(); // Отримання результату
27          categories = JsonSerializer // Десеріалізація отриманих даних
28              .Deserialize<List<Category>>(json, new JsonSerializerOptions
29              {
30                  PropertyNameCaseInsensitive = true
31              });
32          return categories; // Поверення колекції класів Category
33      }
34
35      public async Task CreateCategory(Category category) // Метод для створення категорії
36      {
37          var json = JsonSerializer.Serialize(category); // Сєrialізація даних
38          var content = new StringContent(json, Encoding.UTF8, "application/json");
39
40          var response = await _httpClient // Відсылання запиту із серіалізованими даними
41              .PostAsync("https://localhost:7015/api/Categories", content);
42          response.EnsureSuccessStatusCode(); // Перевірка запиту на успіх
43      }
44      public async Task UpdateCategory(Category category) // Метод для оновлення категорії
45      {
46          var json = JsonSerializer.Serialize(category); // Сєrialізація даних
47          var content = new StringContent(json, Encoding.UTF8, "application/json");
48
49          var response = await _httpClient // Відсылання запиту із серіалізованими даними
50              .PutAsync($"https://localhost:7015/api/Categories/{category.CategoryID}", content);
51          response.EnsureSuccessStatusCode(); // Перевірка запиту на успіх
52      }
53      public async Task DeleteCategory(Guid guid) // Метод для видалення категорії
54      {
55          var response = await _httpClient // Відсылання запиту разом з Guid значенням
56              .DeleteAsync($"https://localhost:7015/api/Categories/{guid}"); // об'єкту для видалення
57      }
}
```

Рисунок 3.13 – Приклад коду класу-репозиторію CategoryRepository.

Клас CategoryRepository у проекті відповідає за взаємодію з API для роботи з категоріями. Він використовує об'єкт HttpClient для надсилання HTTP-запитів до серверної частини застосунку. У класі реалізовано методи для отримання

Зм	Лист	№ докум.	Підп.	Дата

списку категорій, створення нової, оновлення наявної та видалення існуючої категорії. Всі ці методи працюють асинхронно, що дозволяє не блокувати основний потік виконання програми. Дані передаються у форматі JSON: перед відправленням вони серіалізуються, а після отримання — десеріалізуються. Таким чином, CategoryRepository забезпечує зручний спосіб обміну даними між клієнтською частиною (інтерфейсом користувача) та API. За аналогією розроблено ще 7 інших репозиторіїв: InventoryRepository(), ProductRepository(), RoleRepository(), SaleProductRepository(), SalesRepository() та StoreRepository(), UsersRepository().

Програма звертається до подібних класів кожний раз, коли потрібно отримати дані з БД або відправити нові. Використання класу репозиторію має низку переваг, головною з яких є відокремлення логіки доступу до даних від іншої частини програми. Це забезпечує кращу структурованість, читабельність та підтримуваність коду. Репозиторій інкапсулює всі запити до API, що дозволяє централізовано змінювати логіку доступу до даних без потреби вносити зміни по всьому застосунку. Крім того, використання репозиторіїв спрощує тестування і дозволяє легко реалізовувати заміщення залежностей, що особливо корисно при модульному тестуванні. Це також сприяє реалізації принципів SOLID, зокрема принципу єдиної відповідальності та інверсії залежностей.

### 3.2.3. Основні екрани та форми: товари, продажі, персонал, аналітика

Робота програми починається із запиту на введення логіну та паролю. Відповідно до ролі користувача буде надано доступ до основних пунктів меню, які на початку роботи програми недоступні. На рисунку 3.14 зображено інтерфейс для введення логіну та паролю.

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 ПЗ**

Лис  
47

The image shows a login form with a light gray background. At the top, there is a horizontal menu bar with the following items: Stores, Operations, Products, Categories, Reports, Roles, and Users. Below the menu, there are two input fields: 'Username:' followed by a text input box, and 'Password:' followed by another text input box. Below these fields is a blue rectangular button labeled 'Log In'.

Рисунок 3.14 – Інтерфейс для введення логіну та паролю

Після успішного введення необхідних даних користувачеві надається доступ до пунктів меню. Якщо користувач має роль адміністратора, то йому доступні всі пункти меню. Також надається можливість додавати нових користувачів, надаючи їм певну роль. На рисунку 3.15 зображені доступні пункти меню відповідно до ролі користувача.

The image contains three separate screenshots of the application's main menu, each corresponding to a different user role:

- Administrator:** Shows a top navigation bar with 'Stores', 'Operations', 'Products', 'Categories', 'Reports', 'Roles', and 'Users'. To the right, a user profile box displays 'Gabe Newell' and 'Administrator', with a 'Log Out' button below it.
- Manager:** Shows the same top navigation bar. To the right, a user profile box displays 'Jim Taylor' and 'Manager', with a 'Log Out' button below it.
- Cashier:** Shows the same top navigation bar. To the right, a user profile box displays 'Dan Brown' and 'Cashier', with a 'Log Out' button below it.

3.15 – Доступні пункти меню відповідно до ролі користувача.

Якщо користувач має роль менеджера, то він позбавляється ролі додавати інших користувачів та редагувати ролі, проте користувач все одно може оперувати всіма іншими пунктами меню, зокрема Stores, Operations, Products, Categories та Reports. Роль касир є найбільш обмеженою в цій системі, адже доступ надається тільки над керуванням магазинами, продажами, операціями та

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

48

продуктами, що є логічним, адже це основні пункти, необхідні для використання системи та забезпечення діяльності торгової точки.

На рисунку 3.16 зображене вікно для перегляду, додавання, редагування та видалення категорій продуктів. Аналогічний вигляд мають інші пункти меню, крім Stores та Reports.

Category ID		Category Name
1		Food
2		Sport
3		Fishing
4		Groceries
5		Drinks
6		School
7		Tools
8		Rest
9		Computer Parts
10		Auto parts
11		Toys
12		Garden
13		Kids
14		Kitchen
15		Houshold equipment
16		Electronics
17		Healthcare and beauty
18		Another
19		Tourism
20		Shoes
21		Books

3.16 – Інтерфейс CRUD-операцій над категоріями продуктів.

При натисканні на кнопку «Add Category» посередині екрану з'являється нова форма для додавання або редагування категорії. Форму для додавання зображенено на рисунку 3.17.

Stores		Operations		Products		Categories		Reports		Roles		Users	
Add Category		Delete Category		Update Category		Gabe Newell Administrator		Log Out					
Category ID		Category name											
1	Food												
2	Sport												
3	Fishing												
4													
5													
6													
7													
8													
9													
10													
11													
12													
13	Kids												
14	Kitchen												
15	Household equipment												
16	Electronics												
17	Healthcare and beauty												
18	Another												
19	Tourism												
20	Shoes												
21	Books												

Рисунок 3.17 – Форма для додавання категорії.

При натисканні на кнопку «Update Product» запускається аналогічне вікно, як у випадку із додаванням, проте змінюється текст міток, а також підтягується значення обраного користувачем запису в таблиці, який необхідно редагувати. Приклад форми для редагування зображенено на рисунку 3.18.

Stores		Operations		Products		Categories		Reports		Roles		Users	
Add Category		Delete Category		Update Category		Gabe Newell Administrator		Log Out					
Category ID		Category name											
9	Computer Parts												
10	Auto parts												
11	Toys												
12													
13													
14													
15													
16	Bakery												
17													
18													
19													
20													
21	Books												
22	Subscribe services												
23	Software												
24	Animals												
25	Clothes												
26	Organic food												
27	Building Materials												
28	Bakery												
29	Medicines												

Рисунок 3.18 – Форма для редагування категорії.

У випадку, коли користувачеві необхідно видалити певний рядок, то цей рядок треба виділити та натиснути на кнопку «Delete». Після цього з'явиться діалогове вікно, в якому буде запрошено підтвердження на видалення. Діалогове вікно для видалення запису зображене на рисунку 3.19.

The screenshot shows a web-based application interface for managing categories. At the top, there is a navigation bar with links: Stores, Operations, Products, Categories, Reports, Roles, and Users. On the far right of the header, it says "Gabe Newell Administrator" and has a "Log Out" button. Below the header is a toolbar with three buttons: "Add Category", "Delete Category" (which is highlighted in blue), and "Update Category". The main content area is a table listing categories. The table has two columns: "Category ID" and "Category name". The rows contain the following data:

Category ID	Category name	
12	Garden	
13	Kids	
14	Kitchen	
15	Household equipment	
16	Electronics	
17	Healthcare and beauty	
18	Delete Category	
19	Are you sure that you want to delete this category?	
20	<input type="button" value="Take"/> <input type="button" value="Hi"/>	
21	Subscribe services	
22	Software	
23	Animals	
24	Clothes	
25	Organic food	
26	Building Materials	
27	Bakery	
28	Medicines	
29	Office Equipment	
30	Video Games	
31		
32	TEST CATEGORY	

A modal dialog box is overlaid on the table, centered over row 18. It has a purple header "Delete Category" and a white body containing the message "Are you sure that you want to delete this category?". At the bottom of the dialog are two buttons: "Take" (highlighted in blue) and "Hi".

Рисунок 3.19 – Діалогове вікно для видалення запису.

Варто також відмітити, що в деяких пунктах меню, під час додавання певних сущностей в таблицю може виникнути необхідність обрати дані з вже існуючого переліку даних. Наприклад, при створенні продукту треба обрати категорію. Даний вибір забезпечений через використання елементу ComboBox, який надає перелік можливих опцій. Даний перелік заповнюється програмно під час запуску потрібного вікна. На рисунку 3.20 зображене інтерфейс для додавання продукту.

Pencil	Garden	PNU990-
Ball	Kids	BL0998E
Keyboard	Kitchen	KBHPR>
Sausages	Houshold equipment	88WWS
Microsoft Office	Electronics	9900OC
Cucumbers	Healthcare and beauty	FOODO
Car Toy	Another	978575E
The Little	Tourism	A88798E
T-shirt po	Shoes	9898737
Shirt Red	Books	SHT331
Nike Air N	Subscribe services	77833AI
Bread Sa	Software	BRD334
Paracetom	Animals	PRCTM:
Fishing R	Clothes	SKU-12
Wireless	Organic food	SKU-20
Garden S	Building Materials	SKU-34
Baby Dia	Bakery	SKU-98
Yoga Mat	Medicines	SKU-78
Mozzarell	Office Equipment	SKU-90
Wall Paint Roller	Video Games	SKU-39
Bluetooth Keyboard		SKU-23
Men's Jeans		SKU-39
Dog Leash		SKU-93

Add Product

Create

Product Name

Category Name

Price

SKU

Save

Рисунок 3.20 – Інтерфейс для додавання продукту.

На рисунку 3.21 представлено основне вікно для роботи з торговими точками. В цьому пункті меню також можна переглянути, додати, редагувати або видалити продукти або продажі, що пов’язані з певною торговою точкою.

Stores Operations Products Categories Reports Roles Users					
		Add Store	Delete Store	Update Store	Show All Products
	Store ID	Store Name	Address	Created At	
1	Badiory	Kov. Lane 4	27.05.2025 18:38:03		
2	Pidvalchyk	Metalystiv 23	27.05.2025 18:47:36		
3	Novus	Boryspilska 43	27.05.2025 19:13:42		
4	Food Market	Chavdar 5	27.05.2025 19:13:56		
5	Novus	Khrechatyk 5	27.05.2025 22:12:53		
6	Bdzhilka	Khmelnytskogo 53	28.05.2025 1:39:03		
7	Varus	Bazhana Ave 34	28.05.2025 1:39:37		
8	Silpo	Shulyavksa 78	24.05.2025 16:37:55		
9	Fora	Zhilyanska 8	24.05.2025 16:37:55		
10	ATB	Kozatska 17	28.05.2025 1:40:57		
11	Products	Heroiv Dnipro 44	28.05.2025 1:41:29		
12	IBook	Pravdy Ave 1	28.05.2025 1:44:34		
13	Ortas	Saksahanskogo 4	28.05.2025 1:45:01		
14	ISLANDSHOP	Zoologichna 5A	28.05.2025 1:45:37		
15	Master Zoo	Pryvatna 41	28.05.2025 1:46:12		
16	DNIPRO	Telihy 87	28.05.2025 1:46:45		
17	Sport Move	Chokolivsky Ave 44	28.05.2025 1:47:17		
18	Dance Foods	Bohatyrska 48	28.05.2025 1:47:37		
19	Allo	Pecherska 9	28.05.2025 1:47:59		
20	BRUSNYCHKA	Kozachok 766	26.05.2025 16:48:40		
21	Water Shop	Clean Str. 41	28.05.2025 20:19:14		

Рисунок 3.21 – Вікно для роботи з торговими точками

Перегляд всіх продуктів, що є наявними у певному магазині, буде доступним після вибору потрібної торгової точки та натискання кнопки «Show All

Products». Перегляд всіх продажів працює аналогічно. На рисунку 3.22 представлено меню перегляду всіх продуктів обраної торгової точки.

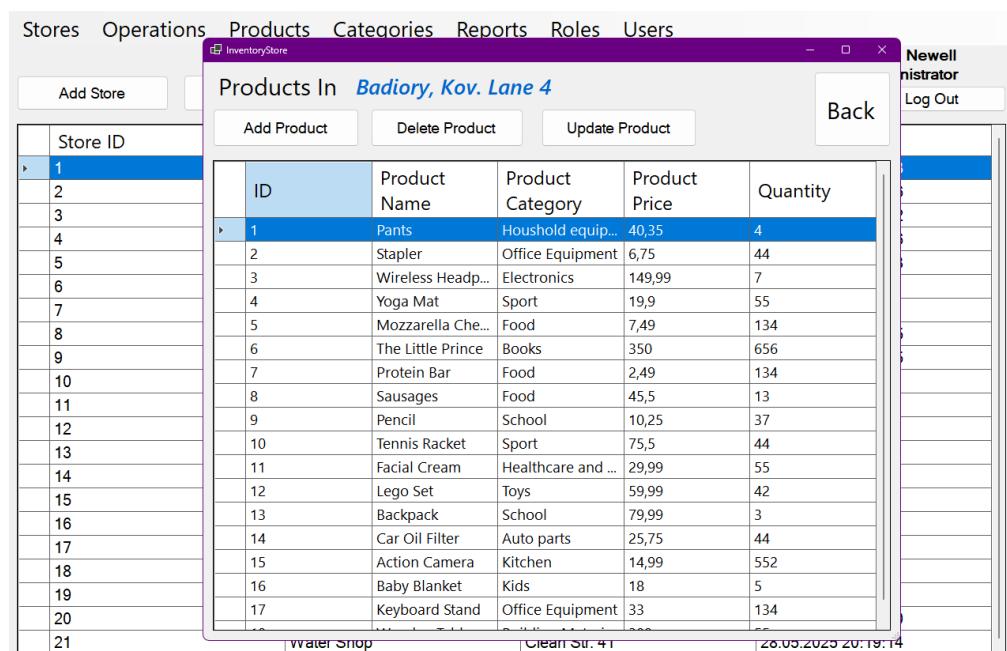


Рисунок 3.22 – Меню перегляду всіх продуктів обраної торгової точки.

В цьому ж пункті меню можна керувати інвентарем торгової точко. Можна додати, видалити та редагувати продукти на складі. На рисунку 3.23 зображено форму для роботи з інвентарем.

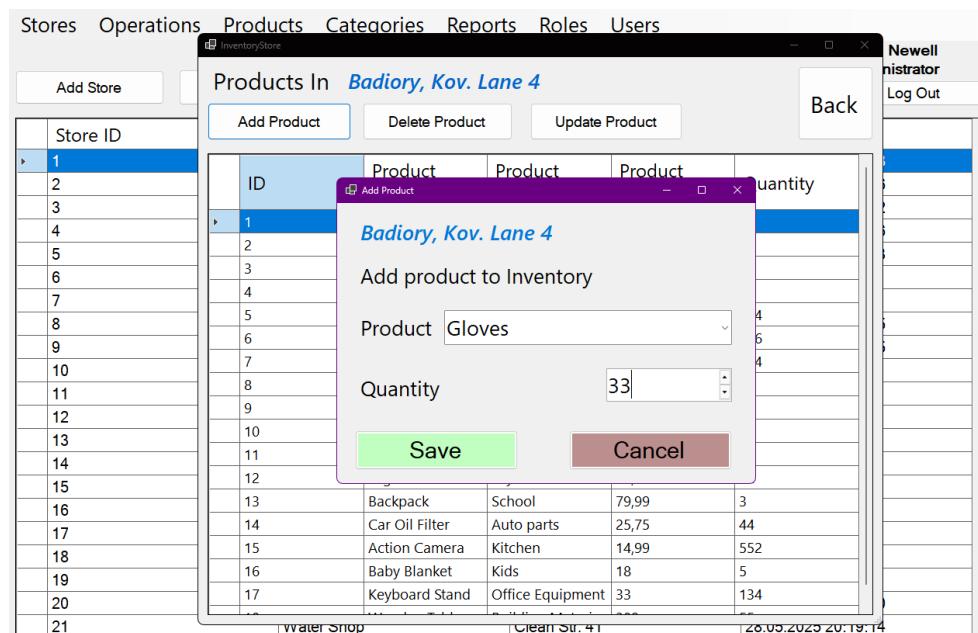


Рисунок 3.23 – Форма для роботи з інвентарем

Для кожного пункту меню забезпечено обробку подій кнопок, що відповідають за проведення операцій над даними. Основна мета створення цих

Зм	Лист	№ докум.	Підп.	Лата

**ІАЛЦ.045440.003 ПЗ**

Лис  
53

обробок подій — взаємодія користувача з графічним інтерфейсом для створення, редагування, видалення та перегляду інформації. Усі дії реалізовані з використанням асинхронних методів та патерну репозиторію, який здійснює обмін даними з API.

- Як приклад можна привести наступні методи для роботи з меню «Stores»:
- Метод `btnStoreAdd_Click` ініціює відкриття форми створення нової торгової точки (`StoresCreateEditForm`). У разі підтвердження змін користувачем, відбувається оновлення таблиці торгових точок шляхом виклику методу `ReadShowStores()`.
- Метод `btnStoreDelete_Click` реалізує логіку видалення торгової точки, попередньо отримуючи ідентифікатор обраного запису з `DataGridView`. Після підтвердження дії користувачем, запис видаляється через API, і дані оновлюються.
- Метод `btnStoreUpdate_Click` відповідає за редагування торгової точки. Він формує об'єкт `Store` на основі даних обраного рядка таблиці, передає його у форму редагування і, після внесення змін, оновлює відображення даних.
- Методи `showAllProducts_Click` та `showAllSales_Click` відповідають за перегляд пов'язаної з обраною торговою точкою інформації — відповідно списку товарів та продажів. Для цього відкриваються відповідні форми з передачею об'єкта `Store`.

Ключовою особливістю є те, що всі операції виконуються через репозиторій `StoreRepository`, який інкапсулює логіку звернення до API. Це забезпечує розділення відповідальностей між представленням (інтерфейсом користувача) та логікою обробки даних.

За цим самим принципом побудовані всі інші форми додатку, що відповідають за керування іншими сутностями, зокрема категоріями, товарами, користувачами, складами тощо. Такий підхід дозволяє забезпечити уніфікований,

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
54

зручний та масштабований механізм взаємодії з даними в межах клієнтської частини застосунку.

Варто розглянути пункт меню «Reports», що відповідає за генерацію звітів у форматі .xlsx або .html. Вікно для отримання звітів складається з виведення основних цифрових даних про систему та кнопок, натискаючи на які користувач запускає алгоритм конвертації даних в потрібний формат, а також збереження файлу з результатом. На рисунку 3.24 зображені інтерфейс для генерації звітів.

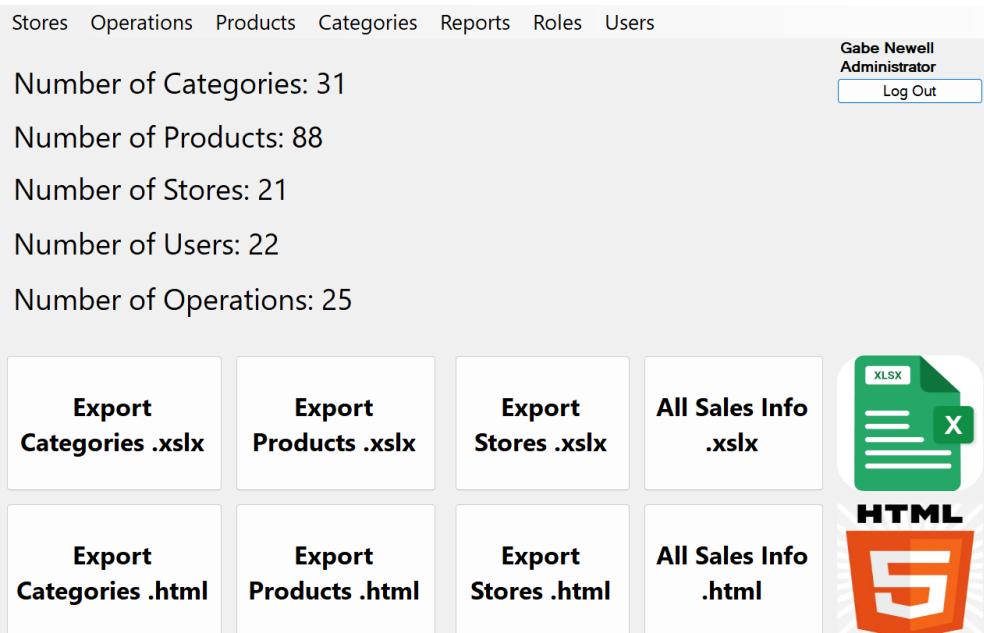


Рисунок 3.24 – Інтерфейс для генерації звітів.

Основні дії алгоритму генерації XLSX-звіту про продажі:

1. Отримання даних з API:

Ініціалізуються відповідні репозиторії: StoreRepository, ProductRepository, SalesRepository, UsersRepository, SaleProductRepository. Кожен з них асинхронно завантажує відповідні колекції об'єктів (торгові точки, товари, продажі, користувачі та зв'язки продажів з товарами).

2. Формування базової моделі звіту (SalesInfo):

На основі кожного запису про продаж (Sale) створюється об'єкт SalesInfo, до якого одразу додається основна інформація: ID продажу, ID користувача, ПІБ користувача, ID магазину, назва магазину, адреса та дата створення магазину. Ці об'єкти додаються до списку salesInfo.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
55

3. Збагачення об'єктів SalesInfo деталями товарів:

Для кожного об'єкта SalesInfo виконується пошук відповідних записів у колекції saleProducts. Якщо ідентифікатори продажу збігаються, до SalesInfo додається інформація про товар: назва, SKU, ціна, кількість і загальна сума (кількість × ціна).

4. Додавання категорії товару:

Для кожного об'єкта SalesInfo виконується пошук відповідного товару у списку products, і витягується назва категорії, яка записується у властивість CategoryName.

5. Додавання ролі користувача:

Для кожного об'єкта SalesInfo виконується пошук користувача за ID у списку users, після чого додається назва його ролі до RoleName.

6. Ініціалізація діалогу збереження файлу:

За допомогою SaveFileDialog користувач обирає місце та ім'я майбутнього Excel-файлу. Формат файла фіксований — .xlsx.

7. Створення Excel-документа:

Ініціалізується об'єкт ExcelPackage. У ньому створюється аркуш під назвою "SalesInfo". У першому рядку аркуша розміщаються заголовки колонок: ID, ім'я користувача, роль, товар, категорія, кількість, ціна, SKU, підсумкова сума, назва магазину, адреса, дата продажу.

8. Заповнення таблиці:

Дані з кожного об'єкта SalesInfo по черзі додаються до рядків таблиці, починаючи з другого рядка. Кожне поле вставляється у відповідну колонку.

9. Форматування таблиці:

За допомогою методу AutoFitColumns() встановлюється автоматичне підбирання ширини колонок згідно з їх вмістом.

10. Збереження файла:

Створений Excel-файл зберігається у вибране користувачем місце, а після успішного збереження виводиться повідомлення з підтвердженням операції.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

56

Користувач отримує сформований .xlsx-файл з деталізованим звітом про всі продажі, включаючи інформацію про продавця, товар, його категорію, магазин, загальну вартість, а також дату операції. Такий підхід забезпечує зручне представлення даних у вигляді таблиці та можливість подальшого аналізу, друку або архівування.

На рисунку 3.25 зображене результат звіту в форматі .xlsx, отриманого після генерації.

A	B	C	D	E	F	G	H	I	J	K	L	
1	ID	User Name	User Role	Product Name	Product Category	Product Count	Product Price	Product SKU	Total Charge	Store Name	Store Address	Date Of Sale
2	1	Pawl Jack	Cashier	Service Pruduct Test	Toys	55	350	A88798909205W3134-jj31	19250	Fora	Zhilyanska 8	24.05.2025 16:37:55
3	2	Man Mansky	Manager	Yoga Mat	Sport	4	19,9	SKU-782134	79,6	Fora	Zhilyanska 8	24.05.2025 16:37:55
4	3	Nancy Law	Cashier	The Little Prince	Books	7	350	A88798909205W3134-jj31	2450	Fora	Zhilyanska 8	24.05.2025 16:37:55
5	4	Slay Man	Administrator	Garden Shovel	Garden	2	24,5	SKU-349820	49	Fora	Zhilyanska 8	24.05.2025 16:37:55
6	5	Jhon Mars	Manager	Yoga Mat	Sport	100	19,9	SKU-782134	1990	Badiory	Kov. Lane 4	27.05.2025 18:38:03
7	6	Katty Kat	Cashier	Garden Shovel	Garden	5	24,5	SKU-349820	122,5	Pidvalchyk	Metalystiv 23	27.05.2025 18:47:36
8	7	Slay Man	Administrator	Baby Diapers Pack	Kids	44	32	SKU-984321	1408	Pidvalchyk	Metalystiv 23	27.05.2025 18:47:36
9	8	Nancy Law	Cashier	Cupcake Box	Bakery	3	10,5	SKU-984023	31,5	Badiory	Kov. Lane 4	27.05.2025 18:38:03
10	9	Slay Man	Administrator	Bluetooth Keyboard	Computer Parts	3	45,9	SKU-238901	137,7	Badiory	Kov. Lane 4	27.05.2025 18:38:03
11	10	Jhon Mars	Manager	Baby Diapers Pack	Kids	2	32	SKU-984321	64	Badiory	Kov. Lane 4	27.05.2025 18:38:03
12	11	Sarah Early	Cashier	Pencil	School	1	10,25	PNO9904882147828OP	10,25	Badiory	Kov. Lane 4	27.05.2025 18:38:03
13	12	Michael Hills	Cashier	Wall Paint Roller	Tools	3	15	SKU-934812	45	Badiory	Kov. Lane 4	27.05.2025 18:38:03
14	13	Gabe Newell	Administrator	Dog Leash	Animals	1	13,25	SKU-938401	13,25	Badiory	Kov. Lane 4	27.05.2025 18:38:03
15	14	Jim Hook	Cashier	Ball	Sport	1	15,99	BL0998BASA31	15,99	Badiory	Kov. Lane 4	27.05.2025 18:38:03
16	15	Danielle Murfy	Administrator	Mozzarella Cheese	Food	3	7,49	SKU-903284	22,47	Badiory	Kov. Lane 4	27.05.2025 18:38:03
17	16	Gabe Newell	Administrator	Cucumbers	Organic food	3	33,45	FOODORG12113312	100,35	Badiory	Kov. Lane 4	27.05.2025 18:38:03
18	17	David Sanchez	Manager	Baby Diapers Pack	Kids	5	32	SKU-984321	160	Badiory	Kov. Lane 4	27.05.2025 18:38:03
19	18	Dave Willson	Manager	Facial Cream	Healthcare and beauty	2	29,99	SKU-102394	59,98	Badiory	Kov. Lane 4	27.05.2025 18:38:03
20	19	Katty Kat	Cashier	Cucumbers	Organic food	4	33,45	FOODORG12113312	133,8	Badiory	Kov. Lane 4	27.05.2025 18:38:03

Рисунок 3.25 – Результат звіту в форматі .xlsx.

## 4. АНАЛІЗ ТА ТЕСТУВАННЯ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК

### 4.1. Тестування функціоналу клієнтської частини

Клієнтська частина програмного забезпечення відіграє ключову роль у взаємодії користувача з системою, забезпечуючи зручний інтерфейс для роботи з даними. Для підтвердження коректності її роботи було проведено тестування основного функціоналу, що охоплює створення, перегляд, редагування та видалення даних, а також взаємодію з API. Метою тестування є виявлення можливих помилок, перевірка стабільності роботи інтерфейсу та забезпечення відповідності програмного продукту поставленим вимогам.

На рисунку 4.1. зображено спробу додавання нового продукту.

Create Product	
Product Name	ABC TEST PRODUCT
Category Name	Food
Price	3,15
SKU	SSWSF11212321
<button>Save</button>	<button>Cancel</button>

Рисунок 4.1 – Спроба додати новий продукт.

На рисунку 4.2 продемонстровано, що новий продукт було успішно додано.

Stores Operations Products Categories Reports Roles Users						
Add Product		Delete Product		Update Product		
				Gabe Newell Administrator		
	Add Product	Delete Product	Update Product			Log Out
	Product ID	Product Name	Category Name	Price	SKU	Created At
▶	89	ABC TEST PR...	Food	3,15	SSWSF11212...	30.05.2025 21:...
	39	Action Camera	Kitchen	14,99	SKU-948203	28.05.2025 1:1...
	76	Almond Milk	Groceries	3,6	SKU-129384	28.05.2025 1:3...
	83	Antivirus Softw...	Software	139	SKU-384920	28.05.2025 1:3...
	87	Apple+	Software	15,99	APPLP111312	28.05.2025 4:1...
	22	Avocado Oil	Organic food	8,0	SKU-221000	28.05.2025 4:1...

4.2 – Відображення нового продукту.

На рисунку 4.3 зображено спробу оновити доданий продукт, шляхом введення нових тестових даних.

	Product ID	Product Name	Category Name	Price	SKU	Created At
89	ABC TEST PR...	Food	3,15	SSWSF11212...	30.05.2025 21...	
39	Action Camera	Kitchen	14.99	SKU-948203	28.05.2025 1:1...	
76						28.05.2025 1:3...
83						28.05.2025 1:3...
87						28.05.2025 4:1...
33						28.05.2025 1:1...
59						28.05.2025 1:2...
42						28.05.2025 1:1...
21						28.05.2025 1:0...
77						28.05.2025 1:3...
6						31 27.05.2025 19...
61						28.05.2025 1:2...
78						28.05.2025 1:3...
25						28.05.2025 1:0...
16						28.05.2025 1:0...
85	Car Oil Filter	Auto parts	25,75	SKU-847392	28.05.2025 1:3...	

Рисунок 4.3 – Спроба оновити продукт.

На рисунку 4.4 зображено результат оновлення продукту. Замість старих даних відображені нові, введені під час оновлення.

	Product ID	Product Name	Category Name	Price	SKU	Created At
40	Protein Bar	Food	2,49	SKU-583920	28.05.2025 1:1...	
88	Spaghetti	Food	17,99	SPGT2212312	29.05.2025 6:0...	
89	UPDATED	Food	3,15	UPD1121	30.05.2025 21:00	
18	Fishing Rod	Fishing	199,99	SKU-128374	28.05.2025 1:0...	
19	Wireless Head...	Electronics	149,99	SKU-203948	28.05.2025 1:0...	
31	Graphic Tablet	Electronics	229	SKU-293840	28.05.2025 1:1...	
53	HDMI Switch	Electronics	27,5	SKU-839210	28.05.2025 1:2...	

Рисунок 4.4 – Відображення оновленого продукту.

Також необхідно протестувати видалення. На рисунку 4.5 зображено всі наявні продукти, окрім того, що був доданий для тесту. Продукт із оновленою назвою «Updated» було видалено.

	Product ID	Product Name	Category Name	Price	SKU	Created At
▶	3	Service Prudoc...	Toys	350	A8879890920...	26.05.2025 16:...
	11	Car Toyota Co...	Toys	10,44	97857554354	21.05.2025 18:...
	52	Lego Set	Toys	59,99	SKU-584039	28.05.2025 1:2...
	69	Toy Dinosaur	Toys	16,99	SKU-928374	28.05.2025 1:3...
	81	Toy Robot	Toys	88,5	SKU-832947	28.05.2025 1:3...
	28	Thermos Flask	Tourism	21	SKU-384902	28.05.2025 1:1...
	54	Travel Backpack	Tourism	129,9	SKU-192837	28.05.2025 1:2...
	24	Wall Paint Roller	Tools	15	SKU-394812	28.05.2025 1:0...
	45	Wrench Set	Tools	44,99	SKU-473829	28.05.2025 1:2...
	58	Tool Box	Tools	54,9	SKU-384928	28.05.2025 1:2...
	37	Monthly Magaz...	Subscribe serv...	5	SKU-239401	28.05.2025 1:1...

Рисунок 4.5 – Оновлений список продуктів після видалення.

На рисунку 4.6 відображене спробу додати нову торгову точку до переліку вже існуючих.

	Store ID	Store Name	Address	Created At
	19	Allo	Pecherska 9	28.05.2025 1:47:59
	10			28.05.2025 1:40:57
	1			27.05.2025 18:38:03
	6			28.05.2025 1:39:03
	20			26.05.2025 16:48:40
	18			28.05.2025 1:47:37
	16			28.05.2025 1:46:45
	4			27.05.2025 19:13:56
	9			24.05.2025 16:37:55
	12			28.05.2025 1:44:34
	14			28.05.2025 1:45:37
	15			28.05.2025 1:46:12
▶	22			30.05.2025 21:55:20
	3			27.05.2025 19:13:42
	5			27.05.2025 22:12:53

Add Store

Store Name: Test Store

Address: Test Address

Save Exit

Рисунок 4.6 – Спроба додавання нової торгової точки.

Додавання пройшло успішно. На рисунку 4.7 продемонстровано оновлений перелік торгових точок, відсортований за назвою та за алфавітом в протилежному порядку, для того, щоб швидше знайти новий запис «Test Store».

	Store ID	Store Name	Address	Created At
	21	Water Shop	Clean Str. 41	28.05.2025 20:19:14
	7	Varus	Bazhana Ave 34	28.05.2025 1:39:37
▶	23	Test Store	Test Address	30.05.2025 21:56:00
	17	Sport Move	Chokolivsky Ave 44	28.05.2025 1:47:17
	8	Silpo	Shulyavskaya 78	24.05.2025 16:37:55

Рисунок 4.7 – Спроба додавання нової торгової точки «Test Store».

Зм	Лист	№ докум.	Підп.	Лата

ІАЛЦ.045440.003 ПЗ

Лис  
60

Після додавання нового запису була проведена спроба перейменувати торгову точку. Для цього було введено нові дані після натискання на кнопку «Update Store». Результат оновлення продемонстровано на рисунку 4.8. Назва точки «Test Store» було змінено на називу «Updated Test Store».

	Store ID	Store Name	Address	Created At
	21	Water Shop	Clean Str. 41	28.05.2025 20:19:14
	7	Varus	Bazhana Ave 34	28.05.2025 1:39:37
▶	23	Updated Test Store	Test Address	30.05.2025 21:56:00
	17	Sport Move	Chokolivsky Ave 44	28.05.2025 1:47:17
	8	Silpo	Shulyavksa 78	24.05.2025 16:37:55

Рисунок 4.8 – Спроба оновлення торгової точки «Test Store» на «Updated Test Store».

Проведено спробу видалення торгової точки. Для видалення було обрано той самий запис із називою «Updated Test Store». Результат видалення продемонстровано на рисунку 4.9.

	Store ID	Store Name	Address	Created At
▶	21	Water Shop	Clean Str. 41	28.05.2025 20:19:14
	7	Varus	Bazhana Ave 34	28.05.2025 1:39:37
	17	Sport Move	Chokolivsky Ave 44	28.05.2025 1:47:17

Рисунок 4.9 – Перелік торгових точок після видалення торгової точки «Updated Test Store».

На рисунку 4.10 продемонстровано всі продукти, які наявні в торговій точці «Badiory» за адресою «Kov. Lane 4».

The screenshot shows a window titled "Products In Badiory, Kov. Lane 4". At the top, there are three buttons: "Add Product", "Delete Product", and "Update Product". A "Back" button is located in the top right corner. The main area is a table with columns: ID, Product Name, Product Category, Product Price, and Quantity. The table contains 10 rows of data.

ID	Product Name	Product Category	Product Price	Quantity
1	Pants	Houshold equip...	40,35	4
2	Stapler	Office Equipment	6,75	44
3	Wireless Headp...	Electronics	149,99	7
4	Yoga Mat	Sport	19,9	55
5	Mozzarella Che...	Food	7,49	134
6	The Little Prince	Books	350	656
7	Protein Bar	Food	2,49	134
8	Sausages	Food	145,5	13
9				
10				

Рисунок 4.10 – Продукти, наявні в торговій точці «Badiory» за адресою «Kov. Lane 4».

На рисунку 4.11 відображені спробу додавання нового продукту до інвентарю торгової точки «Badiory» за адресою «Kov. Lane 4».

The screenshot shows the same window as Figure 4.10. A modal dialog box is open in the center, titled "Add Product". It contains fields for "Product" (set to "Action Camera") and "Quantity" (set to "11"). At the bottom of the dialog are two buttons: "Save" (green) and "Cancel" (brown).

Рисунок 4.11 – Спроба додавання нового продукту до інвентаря торгової точки «Badiory» за адресою «Kov. Lane 4».

На рисунку 4.12 продемонстровано результат успішного додавання продукту до інвентарю.

Products In <i>Badiory, Kov. Lane 4</i>					
		Add Product	Delete Product	Update Product	Back
	ID	Product Name	Product Category	Product Price	Quantity
>	18	Action Camera	Kitchen	14,99	11
	15	Baby Blanket	Kids	18	5
	13	Backpack	School	79,99	3

Рисунок 4.12 – Результат успішного додавання продукту до інвентарю.

На рисунку 4.13 продемонстровано спробу оновити кількість продукту, що було додано в попередній дії.

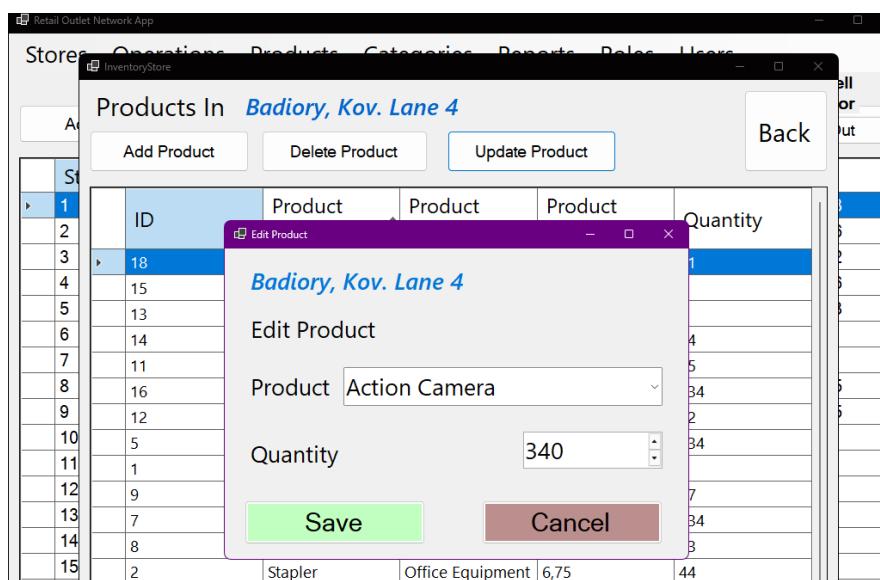


Рисунок 4.13 – Спроба оновлення кількості продуктів «Action Camera».

На рисунку 4.14 продемонстровано результат оновлення. Операція пройшла успішно.

Products In <i>Badiory, Kov. Lane 4</i>					
		Add Product	Delete Product	Update Product	Back
	ID	Product Name	Product Category	Product Price	Quantity
>	18	Action Camera	Kitchen	14,99	340
	15	Baby Blanket	Kids	18	5
	13	Backpack	School	79,99	3
	14	Car Oil Filter	Auto parts	25,75	44

Рисунок 4.14 – Результат оновлення кількості продуктів «Action Camera».

Проведено спробу видалення продукту. Результат видалення продемонстровано на рисунку 4.15. Продукт було успішно видалено з інвентарю торгової точки.

Зм	Лист	№ докум.	Підп.	Дата	Лис
					63

**ІАЛЦ.045440.003 ПЗ**

Products In <i>Badiory, Kov. Lane 4</i>					
		<a href="#">Add Product</a>	<a href="#">Delete Product</a>	<a href="#">Update Product</a>	
	ID	Product Name	Product Category	Product Price	Quantity
▶	15	Baby Blanket	Kids	18	5
	13	Backpack	School	79,99	3
	14	Car Oil Filter	Auto parts	25,75	44
	11	Facial Cream	Healthcare and ...	29,99	55
	16	Keyboard Stand	Office Equipment	33	134
	12	Lego Set	Toys	59,99	42
	5	Mozzarella Che...	Food	7,49	134
	1	Pants	Houshold equip...	40,35	4
	9	Pencil	School	10,25	37
	7	Protein Bar	Food	2,49	134
	8	Sausages	Food	45,5	13
	2	Stapler	Office Equipment	6,75	44

Рисунок 4.15 – Результат видалення продукту «Action Camera».

## 4.2. Тестування API та запитів до БД

Оскільки система побудована за принципом клієнт-серверної архітектури, важливим етапом є тестування серверної частини – зокрема, API, який забезпечує обмін даними між інтерфейсом користувача та базою даних. Правильна робота API гарантує, що дані будуть оброблятися надійно, а функціонал – працювати безпомилково. У межах цього розділу здійснюється перевірка основних HTTP-запитів (GET, POST, PUT, DELETE), які використовуються для взаємодії з даними в базі.

Тестування API буде проводитися за допомогою Swagger – інтерактивного інструменту для візуалізації, перевірки та документування REST API. Swagger автоматично генерується на основі атрибутів контролера в ASP.NET Core, дозволяючи в зручній формі тестувати всі доступні методи контролера без необхідності створення зовнішнього клієнта. Це значно пришвидшує процес перевірки роботи серверної логіки та надає наочне уявлення про доступні маршрути, параметри запитів і відповіді.

На рисунку 4.16 відображені всі записи таблиці «Stores». Ці записи мають бути отримані шляхом надсилання `HttpGet` запиту у форматі JSON.

Зм	Лист	№ докум.	Підп.	Лата	Лис
					64

**ІАЛЦ.045440.003 ПЗ**

The screenshot shows the pgAdmin interface with a query editor window. The query is:

```
1 ✓ SELECT * FROM public.stores
2 ORDER BY "StoreID" ASC
```

The results table has columns: StoreID [PK] uuid, StoreName text, Address text, and CreatedAt timestamp with time zone. The data shows 22 rows of store information.

	StoreID [PK] uuid	StoreName text	Address text	CreatedAt timestamp with time zone
1	01971309-ff94-77b1-8436-838170428bd7	Badiory	Kov. Lane 4	2025-05-27 21:38:03.725646+03
2	01971312-4efb-7a45-824b-542a7ec84bc6	Pidvalchyk	Metalystiv 23	2025-05-27 21:47:36.362036+03
3	0197132a-34b4-7df0-fbc28-1acaffd38ea	Novus	Boryspilska 43	2025-05-27 22:13:42.523349+03
4	0197132a-6c73-7e6b-be61-287337b7f43d	Food Market	Chavdar 5	2025-05-27 22:13:56.846564+03
5	019713ce-3fe1-70fa-9740-3f7440659e75	Novus	Khrechatyk 5	2025-05-28 01:12:53.34839+03
6	0197148a-ff8e-7a92-9567-480a011c7fde	Bdzhilka	Khmelnytskogo 53	2025-05-28 04:39:03.171694+03
7	0197148b-84de-7c6a-b133-89cfbb928c66	Varus	Bazhana Ave 34	2025-05-28 04:39:37.29814+03
8	0197148c-be14-7599-82bf-35de0654424	ATB	Kozatska 17	2025-05-28 04:40:57.488202+03
9	0197148d-3a4c-7023-bb10-f180c2189e63	Products	Heroiv Dnipro 44	2025-05-28 04:41:29.287832+03
10	01971490-0cc3-72ab-a44b-eb23285b7...	iBook	Pravdy Ave 1	2025-05-28 04:44:34.203699+03
11	01971490-7758-7adf-a7b2-0e9443e557d3	Ortas	Saksahanskogo 4	2025-05-28 04:45:01.523789+03
12	01971491-02b8-741f-b330-c0f6fb5b7fec	ISLANDSHOP	Zoologichna 5A	2025-05-28 04:45:37.203657+03
13	01971491-8df9-738e-adc3-1b5fdbaa7313	Master Zoo	Pryvatna 41	2025-05-28 04:46:12.852947+03
14	01971492-0e0c-7e05-81ff-1f89d95a4f1f	DNIPRO	Telihy 7	2025-05-28 04:46:45.639478+03
15	01971492-89e6-74fa-8bab-4d2e063e18ad	Sport Move	Chokolivsky Ave 44	2025-05-28 04:47:17.345864+03
16	01971492-da3a-7227-b892-cc91a4476c...	Dance Foods	Bohatyriska 48	2025-05-28 04:47:37.909667+03
17	01971493-2d9c-785d-873b-5d2e8f573d7e	Allo	Pecherska 9	2025-05-28 04:47:59.255754+03
18	0197188c-918d-7513-b434-a7dc2dad63...	Water Shop	Clean Str. 41	2025-05-28 23:19:14.948721+03
19	01972331-42f4-7dc6-8670-0c859e6c5046	New Store	Address 44	2025-05-31 00:55:20.424492+03
20	337ed825-f391-465a-a87e-16575f9c6563	Slipo	Shulyavskaya 78	2025-05-24 19:37:55.363+03
21	920db6be-c6dd-4172-96c2-02bd351911...	Fora	Zhilyanska 8	2025-05-24 19:37:55.363+03
22	bb6636bc-3ec7-4ee6-9e35-e863685360ae	BRUSNYCH...	Kozachok 766	2025-05-26 19:48:40.683+03

Рисунок 4.16 – Записи таблиці «Stores».

На рисунку 4.17 показано результат HttpGet-запиту за посиланням <https://localhost:7015/api/Stores>. Отримано код 200, що свідчить про успішність запиту. Також на рисунку 4.17 показано отримані дані у форматі JSON.

The screenshot shows a Postman request for the '/api/Stores' endpoint. The response code is 200, and the response body is a JSON array of store objects. Each object contains 'storeID', 'storeName', 'address', and 'createdAt' fields.

```

[{"storeID": "01971490-7758-7adf-a7b2-0e9443e557d3", "storeName": "Dance Foods", "address": "Bohatyriska 48", "createdAt": "2025-05-28T01:47:37.909667Z"}, {"storeID": "01971491-02b8-741f-b330-c0f6fb5b7fec", "storeName": "ISLANDSHOP", "address": "Zoologichna 5A", "createdAt": "2025-05-28T04:45:37.203657Z"}, {"storeID": "01971491-8df9-738e-adc3-1b5fdbaa7313", "storeName": "Master Zoo", "address": "Pryvatna 41", "createdAt": "2025-05-28T04:46:12.852947Z"}, {"storeID": "01971492-0e0c-7e05-81ff-1f89d95a4f1f", "storeName": "DNIPRO", "address": "Telihy 7", "createdAt": "2025-05-28T04:46:45.639478Z"}, {"storeID": "01971492-89e6-74fa-8bab-4d2e063e18ad", "storeName": "Sport Move", "address": "Chokolivsky Ave 44", "createdAt": "2025-05-28T04:47:17.345864Z"}, {"storeID": "01971492-da3a-7227-b892-cc91a4476c...", "storeName": "Dance Foods", "address": "Bohatyriska 48", "createdAt": "2025-05-28T04:47:37.909667Z"}, {"storeID": "01971493-2d9c-785d-873b-5d2e8f573d7e", "storeName": "Allo", "address": "Pecherska 9", "createdAt": "2025-05-28T04:47:59.255754Z"}, {"storeID": "0197188c-918d-7513-b434-a7dc2dad63...", "storeName": "Water Shop", "address": "Clean Str. 41", "createdAt": "2025-05-28T23:19:14.948721Z"}, {"storeID": "01972331-42f4-7dc6-8670-0c859e6c5046", "storeName": "New Store", "address": "Address 44", "createdAt": "2025-05-31T00:55:20.424492Z"}, {"storeID": "337ed825-f391-465a-a87e-16575f9c6563", "storeName": "Slipo", "address": "Shulyavskaya 78", "createdAt": "2025-05-24T19:37:55.363Z"}, {"storeID": "920db6be-c6dd-4172-96c2-02bd351911...", "storeName": "Fora", "address": "Zhilyanska 8", "createdAt": "2025-05-24T19:37:55.363Z"}, {"storeID": "bb6636bc-3ec7-4ee6-9e35-e863685360ae", "storeName": "BRUSNYCH...", "address": "Kozachok 766", "createdAt": "2025-05-26T19:48:40.683Z"}]

```

Рисунок 4.17 – Результат запиту HttpGet за адресою <https://localhost:7015/api/Stores>.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
65

На рисунку 4.18 показано спробу зробитиHttpPost-запит задля того, щоб створити новий рядок у таблиці Stores. Посилання для запиту <https://localhost:7015/api/Stores>.

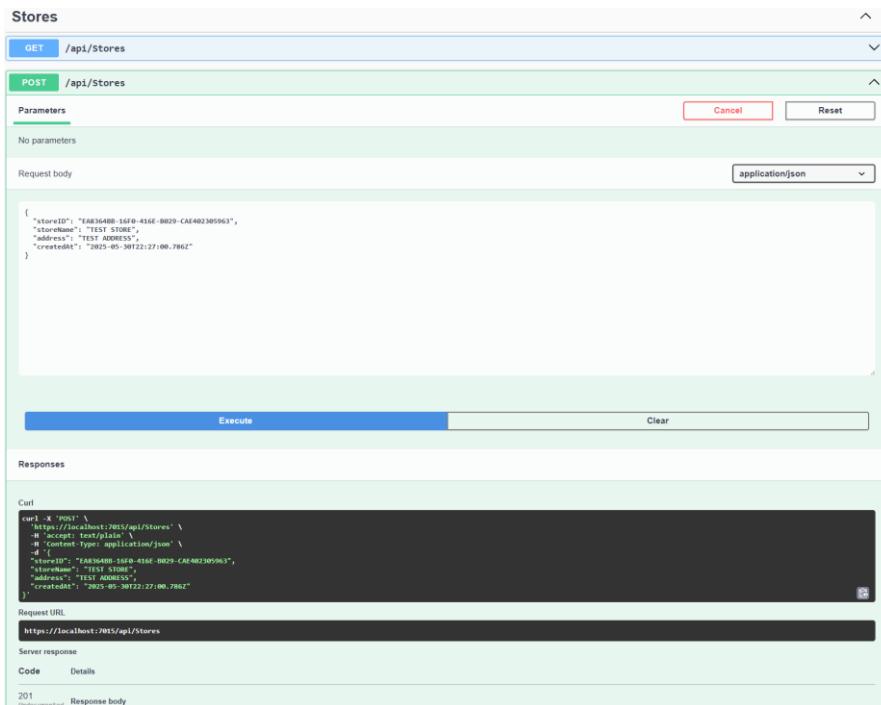


Рисунок 4.18 – Спроба надсиланняHttpPost-запиту.

На рисунку 4.19 зображено Request Body запиту. В ньому передається новий ідентифікатор запису, а також назва торгової точки, адреси та дата створення запиту.



Рисунок 4.19 – Request Body запиту.

Статус-код запиту 200, що означає про успішність створення нового рядка в таблиці. Також можна перевірити наявність нового запису безпосередньо в таблиці. На рисунку 4.20 зображено всі записи таблиці Stores. В кінці таблиці видно рядок з тими даними, які були у Request Body запиту попереднього запиту.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
66

query query history				
Data Output Messages Notifications				
	StoreID [PK] uid	StoreName text	Address text	CreatedAt timestamp with time zone
1	01971309-ff94-77b1-8436-838170428bd7	Badriy	Kov. Lane 4	2025-05-27 21:38:03.725646+03
2	01971312-4efb-7a45-824b-542a7ec84bc6	Pidvalchik	Metalystiv 23	2025-05-27 21:47:36.362036+03
3	0197132a-3484-7d0f-bc28-1aacaffd38ea	Novus	Boryspilska 43	2025-05-27 22:13:42.523349+03
4	0197132a-6c73-7e6b-be61-287337b7f4...	Food Market	Chavdar 5	2025-05-27 22:13:56.846564+03
5	019713ce-3fef-70f4-9740-3f7440659c75	Novus	Khrechatyk 5	2025-05-28 01:12:53.34839+03
6	0197148a-f8fe-7a92-9567-480a011c7fde	Bdzhilika	Khmelnytskogo 53	2025-05-28 04:39:03.171694+03
7	0197148b-84de-7c6b-133-89cfbb928c...	Varus	Bazhana Ave 34	2025-05-28 04:39:37.29814+03
8	0197148c-be14-7599-82bf-35de806544...	ATB	Kozatska 17	2025-05-28 04:40:57.488202+03
9	0197148d-3a4c-7023-bb10-f180c2189c...	Products	Heroiv Dnipro 44	2025-05-28 04:41:29.287832+03
10	01971490-0cc3-72ab-a44b-eb2328d5b7...	IBook	Pravdy Ave 1	2025-05-28 04:44:34.203699+03
11	01971490-7758-7adf-a7b2-0e9443e557...	Ortas	Saksahanskogo 4	2025-05-28 04:45:01.523789+03
12	01971491-02b8-7411-b530-c06fb95b7fe	ISLANDSHOP	Zoologichna 5A	2025-05-28 04:45:37.203657+03
13	01971491-8df9-738e-adc3-1b5fdbaa7313	Master Zoo	Pryvatna 41	2025-05-28 04:46:12.852947+03
14	01971492-0e0c-7e05-81ff-1fb9d95a4f1f	DNIPRO	Telihy 87	2025-05-28 04:46:45.639478+03
15	01971492-89e6-74fa-8bab-4d2e063e18...	Sport Move	Chokolivsky Ave 44	2025-05-28 04:47:17.345864+03
16	01971492-d3a3-7227-b892-c91a4476c...	Dance Foods	Bohatyriska 48	2025-05-28 04:47:37.909697+03
17	01971493-2d9c-785d-873b-5d2e0f573d...	Allo	Pecherska 9	2025-05-28 04:47:59.255754+03
18	0197188c-918d-7513-b434-a7dc2dad63...	Water Shop	Clean Str. 41	2025-05-28 23:19:14.948721+03
19	01972331-42f4-7dc6-8670-0c859e5c5046	New Store	Address 44	2025-05-31 00:59:20.424492+03
20	337ed825-f391-465a-a87e-16575f9c6563	Silpo	Shulyavksa 78	2025-05-24 19:37:55.5363+03
21	920db4e-3c6d-4172-96c2-02bd351911...	Fora	Zhilyanska 8	2025-05-24 19:37:55.5363+03
22	bb6636bc-3ec7-4ee6-9e35-e63685360...	BRUSNYCH...	Kozachok 766	2025-05-26 19:48:40.683+03
23	ea8364bb-16f0-416e-b029-cae402305963	TEST STORE	TEST ADDRESS	2025-05-31 01:27:00.786+03

Рисунок 4.20 – Всі записи таблиці Stores.

На рисунку 4.21 показано результат HttpGet-запиту за посиланням <https://localhost:7015/api/Stores/ea8364bb-16f0-416e-b029-cae402305963>. Цей запит містить ідентифікатор рядка, який треба отримати від програми. Отримано код 200, а разом з ним і дані рядка. Це свідчить про успішність запиту.

The screenshot shows a REST API testing interface. A GET request is made to `/api/stores/{id}` with the parameter `id` set to `ea8364bb-16f0-416e-b029-cae402305963`. The response code is 200 OK, and the response body is a JSON object representing a store:

```

{
  "storeId": "ea8364bb-16f0-416e-b029-cae402305963",
  "name": "TEST ADDRESS",
  "address": "TEST ADDRESS",
  "createdAt": "2025-05-30T22:29:01Z"
}

```

Рисунок 4.21 – Всі записи таблиці Stores.

На рисунку 4.22 показано результат HttpPut-запиту за посиланням <https://localhost:7015/api/Stores/ea8364bb-16f0-416e-b029-cae402305963>. HttpPut-

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
67

запит відповідає за оновлення існуючого рядка. Він містить ідентифікатор рядка, а також нові дані, які мають замінити записи старого рядка. Отримано код 200. Це свідчить про успішність запиту.

The screenshot shows a REST API client interface. In the 'Parameters' section, there is a field for 'Id' with the value 'ea8364bb-16f0-416e-b029-cae402305963'. Below it, the 'Request body' contains the following JSON:

```
{
  "storeId": "ea8364bb-16f0-416e-b029-cae402305963",
  "storeName": "Updated",
  "address": "Updated address",
  "createdAt": "2025-05-30T22:29:21.094Z"
}
```

At the bottom, there are 'Execute' and 'Clear' buttons. The 'Responses' section shows a 'Curl' command and a 'Request URL' (https://localhost:7805/api/Stores/ea8364bb-16f0-416e-b029-cae402305963). The 'Server response' section shows a status code of 204 and a timestamp of 'date: Fri, 30 May 2025 22:30:19 GMT'.

Рисунок 4.22 – Всі записи таблиці Stores.

На рисунку 4.23 відображені всі записи таблиці «Stores». Ці записи мають бути отримані шляхом надсилання HttpGet запиту у форматі JSON.

Stores				
	StoreID [PK] uuid	StoreName text	Address text	CreatedAt timestamp with time zone
1	01971309-ff94-77b1-8436-838170428bd7	Badiory	Kov. Lane 4	2025-05-27 21:38:03.725646+03
2	01971312-4fe6-7845-824b-542a7ec84bc6	Pidvalchyk	Metalystiv 23	2025-05-27 21:47:36.362036+03
3	0197132a-3484-7d0f-bc28-1aacaffd38ea	Novus	Boryspilska 43	2025-05-27 22:13:42.523349+03
4	0197132a-6c73-76e6-be61-287337b7f4...	Food Market	Chavdar 5	2025-05-27 22:13:56.846564+03
5	019713ce-3fef-70f4-9740-3f7440659c75	Novus	Khrechatyk 5	2025-05-28 01:12:53.34839+03
6	0197148a-ff0e-7a92-9567-480a011c7fde	Bdzhilka	Khmelnytskogo 53	2025-05-28 04:39:03.171694+03
7	0197148b-84d6-7c6a-b133-89cfb928c...	Varus	Bazhana Ave 34	2025-05-28 04:39:37.29814+03
8	0197148c-be14-7599-82bf-35de806544..	ATB	Kozatska 17	2025-05-28 04:40:57.488202+03
9	0197148d-3a4c-7023-bb10-f180c2189c..	Products	Herivo Dnipro 44	2025-05-28 04:41:29.287832+03
10	01971490-0cc0-72ab-a44b-e6232d5b7...	iBook	Pravdy Ave 1	2025-05-28 04:44:34.203699+03
11	01971490-7758-7adff-a7b2-0e9443e557..	Ortas	Saksahanskogo 4	2025-05-28 04:45:01.523789+03
12	01971491-02b8-741f-b330-006fb95b57fec	ISLANDSHOP	Zoologichna 5A	2025-05-28 04:45:37.203657+03
13	01971491-98ff-738d-adc3-1b5fdbaa7313	Master Zoo	Pryvtna 41	2025-05-28 04:46:12.852947+03
14	01971492-0e0c-7e05-81ff-1f89d95a4f1f	DNIPRO	Telihy 87	2025-05-28 04:46:45.639478+03
15	01971492-89ee-674fa-8bab-4d2e063e18..	Sport Move	Chokolivsky Ave 44	2025-05-28 04:47:17.345864+03
16	01971492-da3a-7227-b892-cc91a4476c..	Dance Foods	Bohatyrskaya 48	2025-05-28 04:47:37.909697+03
17	01971493-2d9c-785d-873b-5d2e8f573d..	Allo	Pecherska 9	2025-05-28 04:47:59.255754+03
18	0197188c-918d-7513-b434-a7dc2dad63..	Water Shop	Clean Str. 41	2025-05-28 23:19:14.948721+03
19	01972331-42f4-7dc6-8670-0c859ec5c046	New Store	Address 44	2025-05-31 00:55:20.424492+03
20	337ed825-f391-465a-a87e-16575f9c6563	Silpo	Shulyavyska 78	2025-05-24 19:37:55.363+03
21	920d6b4e-c6dd-4172-96c2-02bd351911..	Fora	Zhilyanska 8	2025-05-26 19:48:40.683+03
22	bb6636bc-3ec7-4ee6-9e35-e863685360..	BRUSNYCH...	Kozachok 766	2025-05-26 19:48:40.683+03
23	ea8364bb-16f0-416e-b029-cae402305963	Updated	Updated	2025-05-31 01:27:00.786+03

Рисунок 4.23 – Всі записи таблиці Stores після оновлення.

Зм	Лист	№ докум.	Підп.	Дата	Лис
					68

**ІАЛЦ.045440.003 ПЗ**

В кінці таблиці присутній рядок із оновленими даними. Це підтверджує успішну операцію оновлення даних.

На рисунку 4.24 показано результат HttpDelete-запиту за посиланням <https://localhost:7015/api/Stores/ea8364bb-16f0-416e-b029-cae402305963>.

HttpDelete-запит відповідає за видалення існуючого рядка. Він містить тільки ідентифікатор рядка. Отримано код 200. Це свідчить про успішність запиту.

The screenshot shows a REST API testing interface. At the top, there's a red header bar with the URL `/api/Stores/{id}`. Below it, a 'Parameters' section has a single entry: `id * required string($uuid) {path}` with the value `ea8364bb-16f0-416e-b029-cae402305963`. There are 'Execute' and 'Clear' buttons below this. The 'Responses' section is expanded, showing a 'Curl' command, a 'Request URL' (`https://localhost:7015/api/stores/ea8364bb-16f0-416e-b029-cae402305963`), and a 'Server response' block containing the HTTP headers `Date: Fri, 10 May 2025 22:13:22 GRT` and `Server: Kestrel`. The 'Responses' table at the bottom shows a single row with code 200 and description 'OK'.

Рисунок 4.24 – Всі записи таблиці Stores після оновлення.

На рисунку 4.25 показано результат видалення. Рядок, з яким проводилось тестування в попередніх діях, не є присутнім в базі даних.

StoreID [PK] uuid	StoreName text	Address text	CreatedAt timestamp with time zone
1 01971309-ff94-77b1-8436-838170428bd7	Badiory	Kov. Lane 4	2025-05-27 21:38:03.725646+03
2 01971312-4efb-7a45-824b-542a7ec84bc6	Pidvalchyk	Metalystiv 23	2025-05-27 21:47:36.362036+03
3 0197132a-3484-7d0f-bc28-1aacaffd38ea	Novus	Boryspilska 43	2025-05-27 22:13:42.523349+03
4 0197132a-6c73-7e6b-be61-287337b7f43d	Food Market	Chavdar 5	2025-05-27 22:13:56.846564+03
5 019713ce-3fef-70f4-9740-3f7440659c75	Novus	Khrechatyk 5	2025-05-28 01:12:53.34839+03
6 0197148a-ff8e-7a92-9567-480a011c7fde	Bdzhilka	Khmelnytskogo 53	2025-05-28 04:39:03.171694+03
7 0197148b-84d6-7c6a-b133-89cfbb928c66	Varus	Bazhana Ave 34	2025-05-28 04:39:37.298144+03
8 0197148c-be14-7599-82bf-35de80654424	ATB	Kozatska 17	2025-05-28 04:40:57.488202+03
9 0197148d-3a4c-7023-bb10-f180c2189c63	Products	Herov Dnipro 44	2025-05-28 04:41:29.287832+03
10 01971490-0cc3-72ab-a44b-eb2328d5b7...	IBook	Pravyd Ave 1	2025-05-28 04:44:34.203699+03
11 01971490-7758-7adfc-a7b2-0e9443e557d3	Ortas	Sakhsanskogo 4	2025-05-28 04:45.01.523789+03
12 01971491-02b8-741f-b330-c06f9b5b7fec	ISLANDSHOP	Zoologichna 5A	2025-05-28 04:45:37.203657+03
13 01971491-8df9-738e-adc3-1b5fdbaa7313	Master Zoo	Priyatna 41	2025-05-28 04:46:12.852947+03
14 01971492-0e0c-7e05-81ff-1f89d95a4f1f	DNIPRO	Telihy 87	2025-05-28 04:46:45.639478+03
15 01971492-89e6-74fa-8bab-4d2e063e18ad	Sport Move	Chokolivsky Ave 44	2025-05-28 04:47:17.345864+03
16 01971492-da3a-7227-b892-cc91a4476c...	Dance Foods	Bohatyrskaya 48	2025-05-28 04:47:37.909697+03
17 01971493-2d9c-785d-873b-5d2e8f573d7e	Allo	Pecherska 9	2025-05-28 04:47:59.255754+03
18 0197188c-918d-7513-b434-a7dc2dad63...	Water Shop	Clean Str. 41	2025-05-28 23:19:14.948721+03
19 01972331-42f4-7dc6-8670-0c859ec5c046	New Store	Address 44	2025-05-31 00:55:20.424492+03
20 337ed825-f391-465a-a87e-16575f9c6563	Silpo	Shulyavskaya 78	2025-05-24 19:37:55.363+03
21 9206b64e-c6dd-4172-96c2-02bd351911...	Fora	Zhilanska 8	2025-05-24 19:37:55.363+03
22 bb6636bc-3ec7-4ee6-9e35-e863685360ae	BRUSNYCH...	Kozachok 766	2025-05-26 19:48:40.683+03

Рисунок 4.25 – Всі записи таблиці Stores після оновлення.

Зм	Лист	№ докум.	Підп.	Дата	Лис
					69

**ІАЛЦ.045440.003 ПЗ**

## 5. ШЛЯХИ ПОЛІПШЕННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

### 5.1. Можливості розширення функціоналу

У подальшому розвиток системи може бути спрямований на додавання нових модулів та сервісів, які значно поглиблять та розширяють її можливості. Кожне з наведених покращень ґрунтуються на конкретній меті – підвищити ефективність бізнес-процесів, гнучкість налаштування під користувача або надійність і безпеку роботи. Конкретні варіанти покращення описано далі:

1. Розширена підтримка онлайн-режimu з автоматичною синхронізацією даних.

Таке покращення забезпечить безперебійну роботу в регіонах зі слабким або нестабільним інтернет-з'єднанням. Кешування даних та черга операцій на клієнті дозволять проводити продажі та оновлювати залишки навіть без доступу до сервера, а після відновлення з'єднання всі дані будуть автоматично синхронізовані. Таке впровадження підвищить стійкість системи та задоволеність користувачів у віддалених магазинах.

2. Розгортання контейнеризованих мікросервісів із CI/CD. Перехід до мікросервісної архітектури в Docker/Kubernetes (AKS, EKS) із налаштованим конвеєром CI/CD дозволить автоматично тестувати, збирати та розгортати оновлення в кілька кліків, мінімізуючи час простою. Метою є прискорення циклів розробки, підвищення стабільності та легкість масштабування окремих компонентів системи.
3. Інтеграція зі службами єдиного входу (SSO) та багатофакторною автентифікацією. Забезпечення централізованого керування доступом через корпоративні LDAP/Active Directory або протоколи OAuth2/OpenID Connect дозволить підвищити рівень безпеки та спростити адміністрування великої кількості користувачів у великих мережах. Багатофакторна автентифікація зменшить ризик несанкціонованого доступу до конфіденційних даних про продажі та залишки й поліпшить існуючу ієрархію користувачів.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
70

4. Впровадження аналітики користувацької поведінки (User Behavior Analytics). Збирання та аналіз метрик щодо того, які функції системи використовуються найчастіше, допоможе пріоритизувати розвиток продукту та оптимізувати інтерфейс під реальні сценарії застосування. Це підвищить загальну продуктивність персоналу та скоротить час навчання нових співробітників. Усі запропоновані покращення системи вплинуть на підтримку стійкості роботи та масштабування функціоналу. Поточні варіанти удосконалення є реалістичними й заплановано відповідно до характеристик розробленої системи.

## 5.2. Інтеграція з хмарними сервісами

Окремим елементом покращення функціонування системи можна виділити використання хмарних сервісів. Вони дозволяють швидко розгорнути інфраструктуру, забезпечують автоматичне масштабування ресурсів у відповідь на зміну навантаження та гарантують високу надійність за рахунок розподіленого зберігання та резервування даних. Інтеграція системи з хмарною інфраструктурою забезпечує її високу доступність, гнучкість і здатність автоматично адаптуватися до змін навантаження.

Одним з варіантів використання хмарного середовища є перенесення серверної частини та бази даних на такі платформи як IaaS чи PaaS дозволяє звільнитися від ручного керування фізичними серверами, зосередившись на розвитку бізнес-логіки і функціоналу. А використання контейнеризації (Docker, Kubernetes), як було запропоновано в пункті 5.1, у поєднанні з хмарними менеджерами допоможе автоматично масштабувати мікросервіси відповідно до обсягу запитів, що критично підвищує продуктивність у пікові години продажів. Застосування об'єктного сховища (наприклад, Amazon S3 або Azure Blob Storage) для архівування звітних даних, медіафайлів і резервних копій дає змогу гарантувати цілісність інформації та швидку доставку контенту через CDN-мережі без навантаження основної бази даних. Крім того, хмарні функції (Azure Functions чи AWS Lambda) стають потужним інструментом для фонового

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
71

оброблення подій – надсилання сповіщень, формування щоденних зведень чи оптимізація кешу – без необхідності завантажувати основний API.

За допомогою хмарних сервісів можна також покращити роботу з базою даних. Аналітичні сервіси Big Data, такі як Azure Synapse або Amazon Redshift, дозволяють виконувати складні обчислювальні запити на великих обсягах даних. Це відкриває можливість гнучко формувати дашборди та звіти для адміністраторів, використовуючи вбудовані інструменти візуалізації (Power BI, QuickSight). Звільнення від рутинної обробки даних та перехід до спеціалізованих скринь сприяє значному скороченню часу прийняття управлінських рішень. Описані ідеї інтеграції хмарних сервісів до розробленої системи допоможуть при масштабуванні системи, додаванні нових функціональних характеристик та при роботі з великими даними.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

72

## ВИСНОВКИ

У ході виконання дипломного проєкту було реалізовано повноцінну комп'ютерну систему для централізованого управління мережею торгових точок, яка охоплює ключові функціональні компоненти сучасних облікових та аналітичних систем. Основною метою розробки стало створення масштабованого, надійного та інтуїтивно зрозумілого інструменту, що дозволяє автоматизувати щоденні бізнес-процеси — від обліку товарів до формування звітності.

У першому розділі було проведено огляд існуючих рішень у галузі управління торговими точками, таких як Poster POS, 1C:Підприємство, RetailCRM, iiko тощо. Детально проаналізовано їх функціональні можливості та виявлено низку недоліків, серед яких: складність кастомізації, висока вартість, прив'язаність до хмарних сервісів та складність інтеграції. Саме ці аспекти стали підставою для розробки власної системи, орієнтованої на потреби малого та середнього бізнесу в Україні.

У другому розділі було спроектовано архітектуру програмного забезпечення, що базується на трикомпонентній клієнт-серверній моделі. Виділено клієнтську частину (WinForms-додаток), серверну частину (ASP.NET Core Web API) та базу даних (PostgreSQL). Обґрунтовано вибір усіх застосованих технологій: C# — як мови програмування; Entity Framework Core — як ORM для зручної роботи з базою; HTTP-протокол і RESTful API — для зв'язку між клієнтом і сервером. Також сформовано вимоги до інтерфейсу користувача, його зручності, функціональності та відповідності ролям користувачів системи.

Особливу увагу приділено проектуванню бази даних, яка відповідає першій, другій та третій нормальним формам, що забезпечує мінімізацію надмірностей та запобігає логічним помилкам. Таблиці логічно організовані, підтримують зв'язки через зовнішні ключі, дозволяючи легко масштабувати систему та виконувати складну аналітику. Розглянуто кожну сутність: користувачів, ролі, магазини, категорії товарів, залишки, продажі та їх деталізацію.

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

73

У третьому розділі було реалізовано всі компоненти системи. Серверна частина, написана на ASP.NET Core, забезпечує обробку HTTP-запитів, взаємодію з базою даних через EF Core та реалізацію бізнес-логіки. Детально описано контролери, сервісні класи, контекст бази даних та використані атрибути C#. Обґрунтовано використання асинхронності, яка дозволяє ефективно обробляти множинні запити та знижує навантаження на сервер.

Клієнтський застосунок реалізовано засобами WinForms. Наведено опис головного меню, роботи з API через репозиторії, побудови інтерфейсу з використанням DataGridView, форм взаємодії з користувачем. Також описано механізм генерації HTML та XLSX звітів з використанням EPPlus, що дозволяє створювати документи з детальною інформацією про продажі, товари, користувачів, магазини тощо. Усі запити до API обробляються централізовано, забезпечуючи цілісність і узгодженість даних між клієнтом і сервером.

У четвертому розділі розглянуто тестування системи: тестування клієнтської частини — шляхом перевірки роботи з формами, кнопками та відображенням даних; тестування API — через Swagger, який дає змогу інтерактивно надсиляти запити до серверної частини, перевіряти коректність маршрутів, відповідей та взаємодії з базою.

У процесі реалізації було виконано всі завдання, зазначені в технічному завданні: розроблено архітектуру системи, спроектовано та реалізовано базу даних, побудовано серверну частину з використанням RESTful API, реалізовано клієнтський застосунок з графічним інтерфейсом, забезпечено фільтрацію, авторизацію, підтримку ролей, реалізовано механізм звітності у форматах HTML/XLSX, роведено аналіз та тестування працездатності.

Розроблена система є гнучкою, масштабованою та зручною в користуванні. Вона може бути легко адаптована під реальні потреби малого або середнього бізнесу, а її архітектура дозволяє подальший розвиток — від підключення мобільного додатку до інтеграції з хмарними сервісами або сторонніми системами аналітики. Таким чином, поставлена мета досягнута повністю, а створена

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
74

комп'ютерна система відповідає сучасним вимогам до програмного забезпечення для обліку та управління торговою мережею.

Зм	Лист	№ докум.	Підр.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
75

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Poster POS [Електронний ресурс]. – Режим доступу до ресурсу: <https://joinposter.com/ua>
2. 1C Підприємство [Електронний ресурс]. – Режим доступу до ресурсу: <https://itez.com.ua/what-is-1c.html>
3. RetailCRM [Електронний ресурс]. – Режим доступу до ресурсу: <https://proficrm.com.ua/retailcrm/>
4. Iiko [Електронний ресурс]. – Режим доступу до ресурсу: <https://magnat-trade.com.ua/iiko-avtomatizaciya-restorana/>
5. Square POS [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.deliverect.com/en/blog/pos-systems/square-pos-system-101-the-complete-guide-2024>
6. Lightspeed [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.lightspeedhq.com/>
7. Shopify POS [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.shopify.com/pos>
8. RESTful API [Електронний ресурс]. – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>
9. Архітектура системи [Електронний ресурс]. – Режим доступу до ресурсу: [https://pupenasan.github.io/ProgIngContrSystems/%D0%9B%D0%B5%D0%BA%D1%86/6\\_httapi.html](https://pupenasan.github.io/ProgIngContrSystems/%D0%9B%D0%B5%D0%BA%D1%86/6_httapi.html)
10. Тип даних uuid [Електронний ресурс]. – Режим доступу до ресурсу: <https://uk.php.brj.cz/uuid-i-produktivnist-velikomasstabnih-dodatkiv>
11. Нормальні форми [Електронний ресурс]. – Режим доступу до ресурсу: <https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture02>
12. C# 4.0 The Complete Reference. — М. : «Вільямс», 2011. — С. 34 – 44.
13. PostgreSQL [Електронний ресурс]. – Режим доступу до ресурсу: <https://postgrespro.com/docs>
14. Windows Forms [Електронний ресурс]. – Режим доступу до ресурсу: <https://learn.microsoft.com/uk-ua/dotnet/desktop/winforms/?view=netframeworkdesktop-4.8>
15. ASP .NET Core [Електронний ресурс]. – Режим доступу до ресурсу:

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис

76

<https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0>

Зм	Лист	№ докум.	Підп.	Дата

**ІАЛЦ.045440.003 ПЗ**

Лис  
77

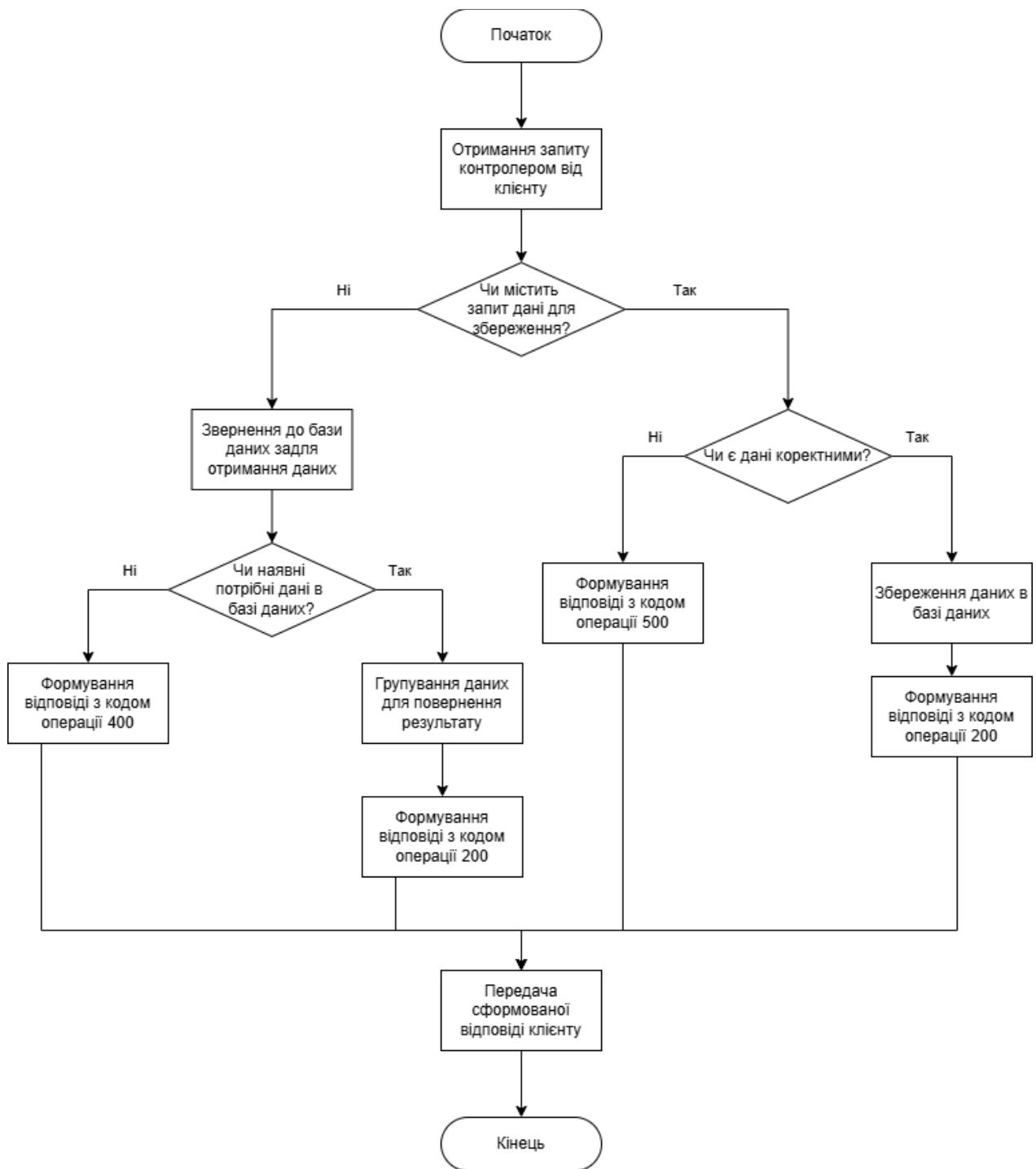
## ДОДАТОК 1

Комп'ютерна система управління мережею торгових точок

Схема алгоритму роботи серверного застосунку  
ІАЛЦ. 045440.004Д1

Аркушів 1

Київ 2025



Зм.	Лист	№ докум.	Підп.	Дата
Розроб.		Чебан М.Д.		
Перев.		Павловський В.І.		
Н. контр.		Клятченко Я.М.		
Затв.		Романкевич В.О.		

**ІАЛЦ. 045440.004 Д1**

Комп'ютерна система управління мережею  
торгових точок

*Схема алгоритму роботи  
серверного застосунку*

Літ.	Аркуш	Аркушів
	1	1
НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-11		

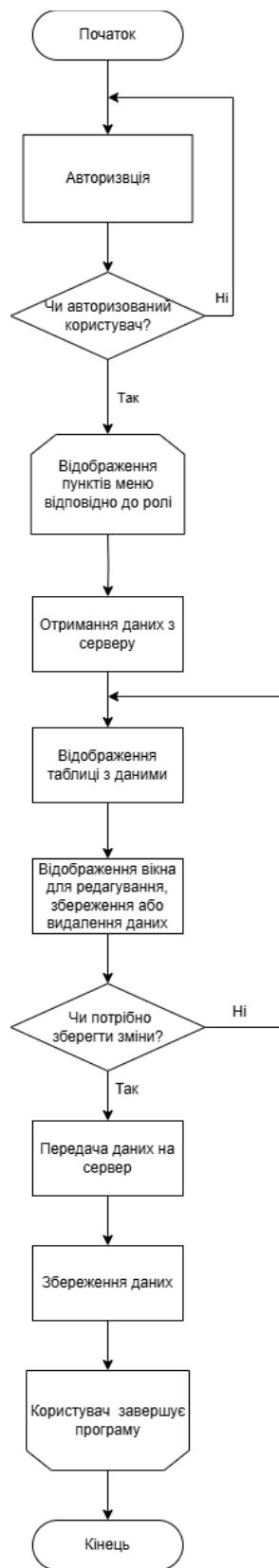
## ДОДАТОК 2

Комп'ютерна система управління мережею торгових точок

Схема алгоритму роботи клієнтського застосунку  
ІАЛЦ. 045440.005 Д2

Аркушів 1

Київ 2025



Зм	Лист	№ докум.	Підп.	Дата
Розроб.		Чебан М.Д.		
Перев.		Павловський В.І.		
Н. контр.		Клятченко Я.М.		
Затв.		Романкевич В.О.		

**ІАЛЦ. 045440.005 Д2**

Комп'ютерна система управління мережею  
торгових точок  
*Схема алгоритму роботи  
клієнтського застосунку*

Літ.	Аркуш	Аркушів
	1	1

НТУУ «КПІ ім. Ігоря  
Сікорського», ФПМ, КВ-11

## ДОДАТОК 3

Комп'ютерна система управління мережею торгових точок

Алгоритм роботи системи  
ІАЛЦ. 045440.006 ДЗ

Аркушів 1

Київ 2025



Зм.	Лист	№ докум.	Підп.	Дата	ІАЛЦ. 045440.006 ДЗ		
Розроб.	Чебан М.Д.				Літ.	Аркуш	Аркушів
Перев.	Павловський В.І.					1	1
Н. контр.	Клятченко Я.М.				Алгоритм роботи системи		
Затв.	Романкевич В.О.				НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-11		

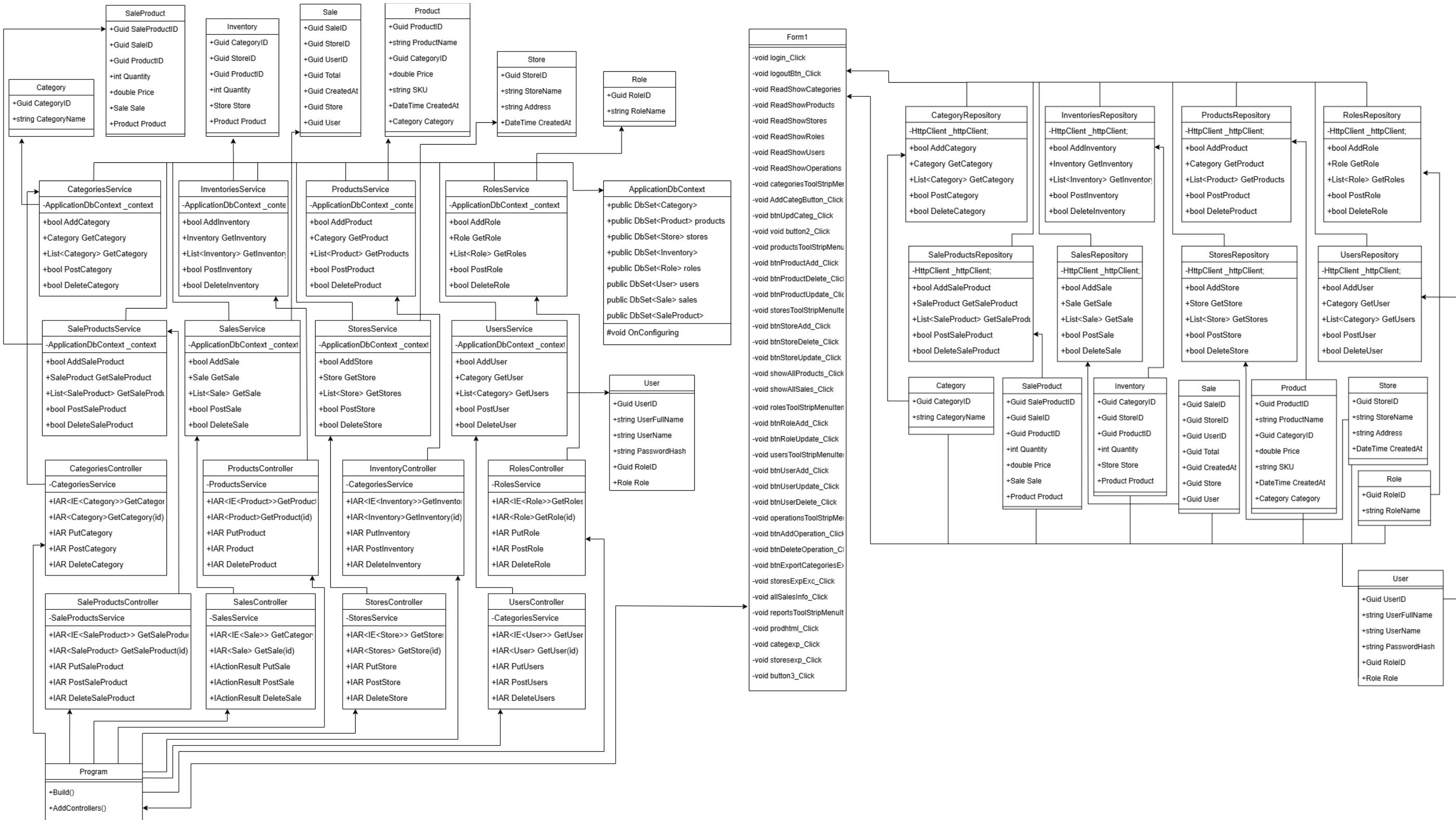
## ДОДАТОК 4

Комп'ютерна система управління мережею торгових точок

Діаграма класів клієнтського та серверного додатків  
ІАЛЦ. 045440.007 Д4

Аркушів 1

Київ 2025



Зм.	Арк.	№ документа	Підпис	Дата	Літ.	Аркуш	Аркушів
Розроб.		Чебан М.Д.					
Перев.		Павловський В.І.					
Н. контр.		Клятченко Я.М.					
Зам.		Романкевич В.О.					

**Діаграма класів клієнтського та серверного додатків**

Комп'ютерна система управління мережею торгових точок

НТУУ «КПІ ім. Ігоря Сікорського», ФІМ, КВ-11

## ДОДАТОК 5

Комп'ютерна система управління мережею торгових точок

Презентація

Аркушів 7

Київ 2025

# **Комп'ютерна система управління мережею торгових точок**

Студент:

**Чебан Максим Дмитрович**

Група: КВ-11

Дипломний керівник:

к.т.н., доцент каф. СПСКС

Павловський Володимир Ілліч

## **АКТУАЛЬНІСТЬ РОБОТИ**

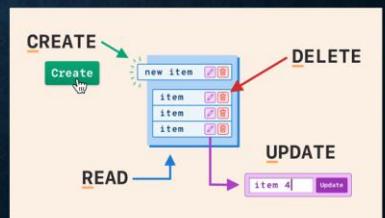
У сучасних умовах активного розвитку торгівлі та цифрових технологій ефективне управління мережею торгових точок стає важливою складовою успішної діяльності підприємств. Зростаючий обсяг даних, необхідність швидкого прийняття рішень та контроль за процесами продажу вимагають впровадження сучасних комп'ютерних систем обліку та управління.



**Метою** даного проекту є створення застосунку для централізованого управління мережею торгових точок, що дозволяє вести облік товарів, продажів, категорій, ролей користувачів, а також здійснювати базовий аналіз діяльності.

**Вимоги** до програмного продукту, що розробляється:

- Віконний інтерфейс користувача на базі .NET Windows Forms;
- Робота з базою даних PostgreSQL;
- Можливість створення, редагування та перегляду товарів, категорій товарів, торгових точок, користувачів, ролей та продажів.
- Підтримка обліку залишків товару;
- Аутентифікація користувачів;
- Фільтрація, сортування та пошук даних;
- Створення базових звітів.



## АНАЛІЗ НАЯВНИХ РІШЕНЬ



Poster POS



1C:Підприємство



iiko



RetailCRM

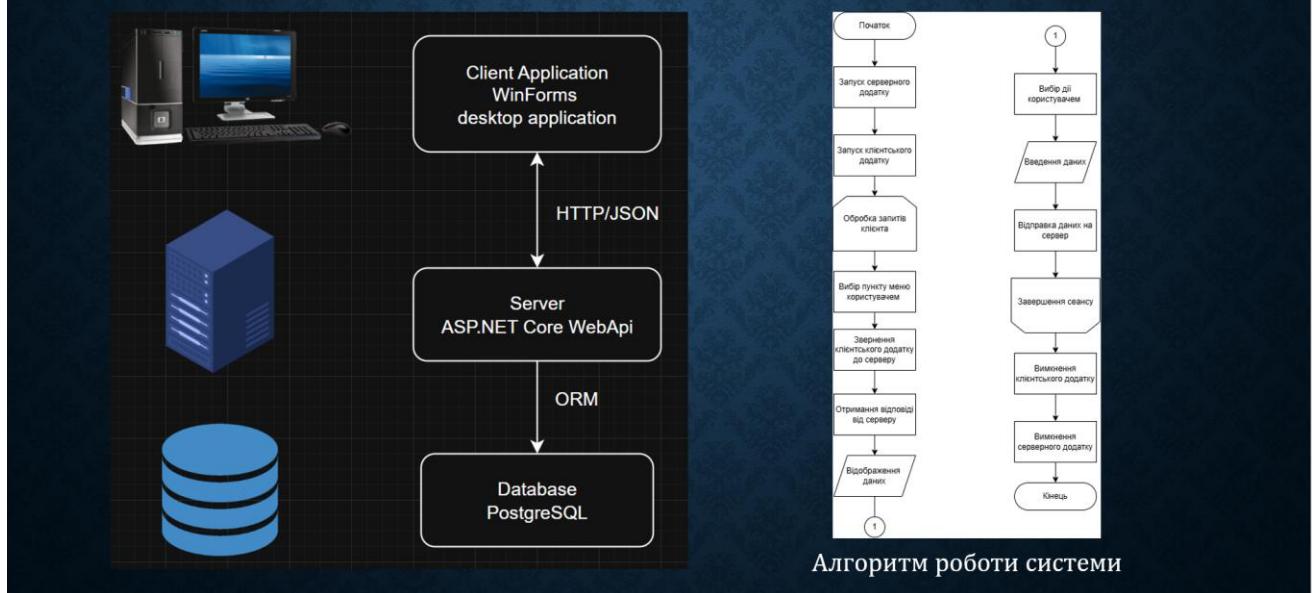


Square POS

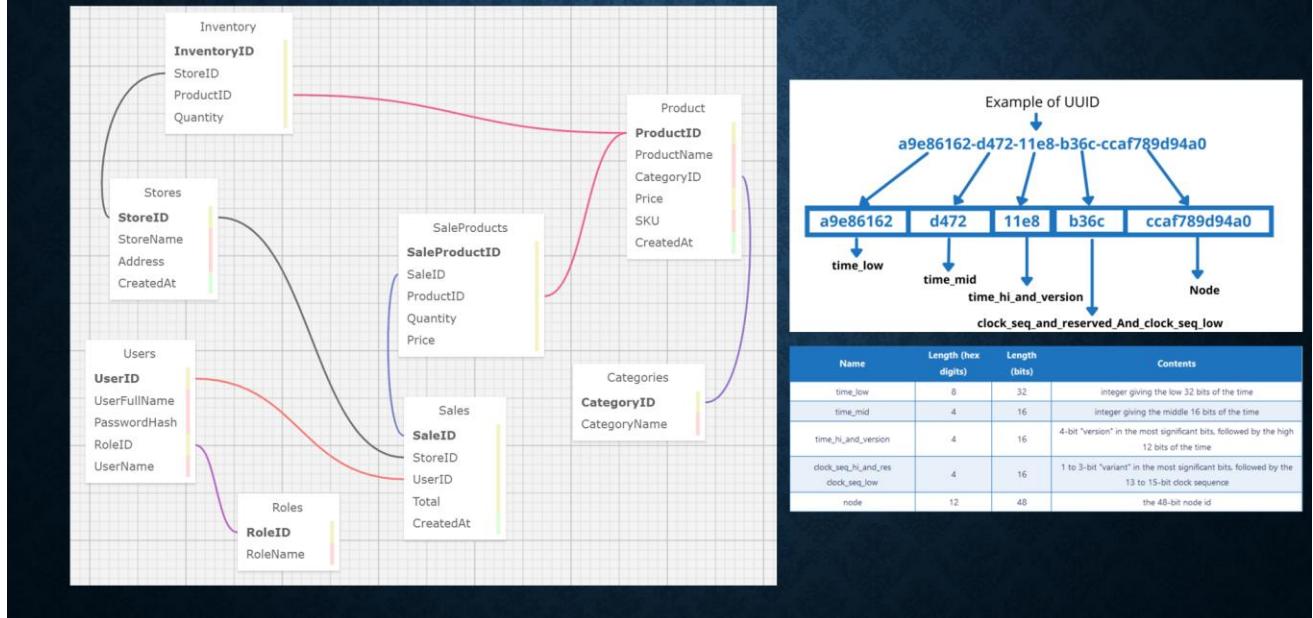


Lightspeed

# ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ТОРГОВИХ ТОЧОК



## ПРОЄКТУВАННЯ БАЗИ ДАНИХ



**Метою** даного проекту є створення комп'ютерної системи для централізованого управління мережею торгових точок, що дозволяє вести облік товарів, продажів, категорій, ролей користувачів, а також здійснювати базовий аналіз діяльності.

**Вимоги** до комп'ютерної системи, що розробляється:

- Віконний інтерфейс користувача на базі .NET Windows Forms;
- Робота з базою даних PostgreSQL;
- Можливість створення, редагування та перегляду товарів, категорій товарів, торгових точок, користувачів, ролей та продажів.
- Підтримка обліку залишків товару;
- Аутентифікація користувачів;
- Фільтрація, сортування та пошук даних;
- Створення базових звітів.



## РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ ЗАСОБАМИ ASP.NET CORE

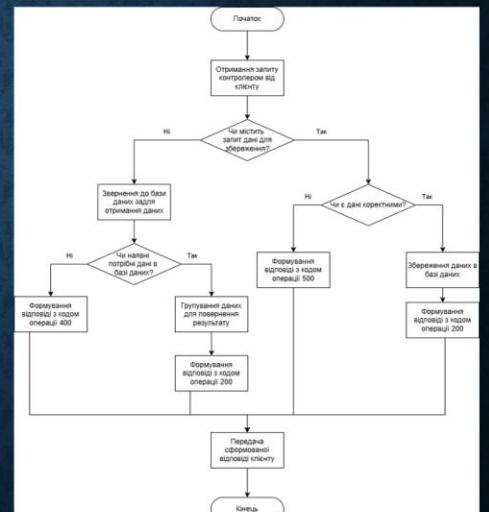
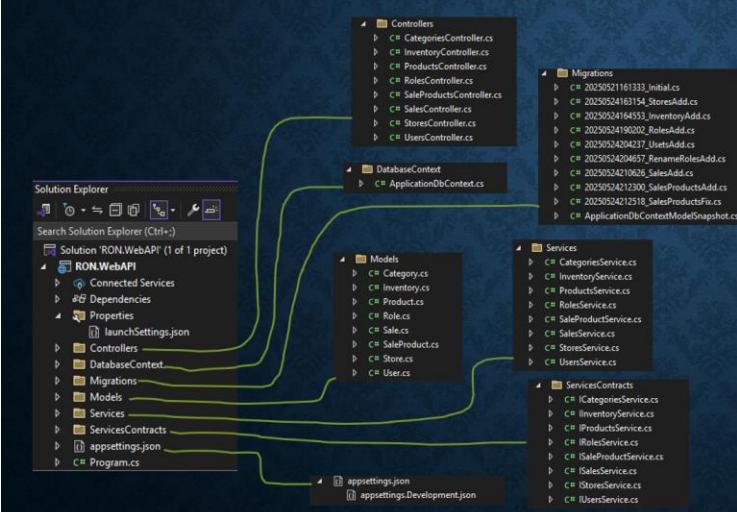


Схема алгоритму роботи серверного застосунку

# РЕАЛІЗАЦІЯ КЛІЄНТСЬКОГО ЗАСТОСУНКУ ЗАСОБАМИ .NET WINFORMS

```

11  public class CategoryRepository
12  {
13      private readonly HttpClient _httpClient; // Ін'єкція класу HttpClient
14      public CategoryRepository() // Конструктор
15      {
16          _httpClient = new HttpClient();
17      }
18      public Task<List<Category>> GetCategories() // Метод для отримання всіх
19      // категорій з бази даних
20      {
21          var categories = new List<Category>();
22          var response = await _httpClient
23              .GetAsync("https://localhost:7015/api/Categories"); // Надсилання запиту
24          response.EnsureSuccessStatusCode();
25
26          var json = await response.Content.ReadAsStringAsync(); // Отримання результату
27          var categories = JsonConvert.DeserializeObject<List<Category>>(json, new JsonSerializerOptions
28          {
29              PropertyNameCaseInsensitive = true
30          });
31
32          return categories; // Повернення інстансії класу Category
33      }
34
35      public async Task CreateCategory(Category category) // Метод для створення категорії
36      {
37          var json = JsonConvert.SerializeObject(category); // Сєріалізація даних
38          var content = new StringContent(json, Encoding.UTF8, "application/json");
39
40          var response = await _httpClient
41              .PostAsync("https://localhost:7015/api/Categories", content);
42          response.EnsureSuccessStatusCode();
43      }
44
45      public async Task UpdateCategory(Category category) // Метод для оновлення категорії
46      {
47          var json = JsonConvert.SerializeObject(category); // Сєріалізація даних
48          var content = new StringContent(json, Encoding.UTF8, "application/json");
49
50          var response = await _httpClient
51              .PutAsync($"https://localhost:7015/api/Categories/{category.CategoryID}", content);
52          response.EnsureSuccessStatusCode();
53      }
54
55      public async Task DeleteCategory(Guid guid) // Метод для видалення категорії
56      {
57          var response = await _httpClient
58              .DeleteAsync($"https://localhost:7015/api/Categories/{guid}"); // об'єкт для видалення
59      }
60  }

```

Приклад класу-репозиторію

Схема алгоритму роботи клієнтського застосунку



## ОСНОВНІ ЕКРАНИ ТА ФОРМИ

The screenshot displays several windows of a .NET WinForms application:

- Main Menu:** Shows links for Stores, Operations, Products, Categories, Reports, Roles, Users, and Log Out.
- Login Screen:** Contains fields for Username and Password, and a Log In button.
- Category Management:** A list view showing Category ID (1-29) and Category Name (Food, Sport, etc.). Buttons for Add Category, Delete Category, and Update Category are at the top.
- Create Category Dialog:** A modal window titled "Create category" with a "Category name" input field and Save/Cancel buttons.
- Product Management:** A list view showing Product Name (Ball, Keyboard, etc.), Category (Healthcare and beauty), and SKU (SKU-001, SKU-002, etc.). Buttons for Create, Edit, and Delete are visible.
- Product Details Dialog:** A modal window showing detailed information for a product like "Ball" with fields for Price, SKU, and Description.
- Category Details Dialog:** A modal window showing detailed information for a category like "Food" with fields for Category name and Description.
- Confirmation Dialog:** A modal window asking if the user wants to save changes with "Save changes" and "Cancel" buttons.

# ГЕНЕРАЦІЯ ЗВІТІВ

The screenshot shows a reporting interface with the following sections:

- Top Navigation:** Stores, Operations, Products, Categories, Reports, Roles, Users.
- User Information:** Gabe Newell, Administrator, Log Out.
- Summary Metrics:**
  - Number of Categories: 31
  - Number of Products: 88
  - Number of Stores: 21
  - Number of Users: 22
  - Number of Operations: 25
- Export Options:**
  - Export Categories .xlsx
  - Export Products .xlsx
  - Export Stores .xlsx
  - All Sales Info .xlsx (with XLS icon)
  - Export Categories .html
  - Export Products .html
  - Export Stores .html
  - All Sales Info .html (with HTML5 icon)
- Report Data:** A grid view showing columns for ID, User Name, User Role, Product Name, Product Category, Product Price, Product SKU, Total Charge, Store Name, Store Address, Date of Sale, and Status.

## ТЕСТУВАННЯ КЛІЄНТСЬКОГО ЗАСТОСУНКУ ТА API

The screenshot illustrates the testing of a client-side application and its API integration. It includes:

- Create Product:** A form to add a new product (ABC TEST PRODUCT) with category Food, price 3,15, and SKU SWSWF11212321. A green 'Save' button is highlighted.
- Operations Dashboard:** Shows a list of products with columns: Product ID, Product Name, Category Name, Price, SKU, and Created At. One entry is highlighted: ABC TEST PR, Food, 3,15, SWSWF112123, 28.05.2025 1:1.
- Edit Product:** A modal showing the updated product details (Product Name: UPDATED, Category Name: Food, Price: 3,15, SKU: UPD1121). A green 'Save' button is highlighted.
- Operations Dashboard (Updated):** Shows the updated product entry in the list.
- Stores API Requests:** A sequence of API calls to /api/stores and /api/stores. The first request shows parameters: storeID: "EAB364BB-16F0-416E-B029-CAE402305963", storeName: "TEST STORE", address: "TEST ADDRESS", createdAt: "2025-05-30T22:27:00.786Z". The second request shows a response body with a success message.
- Operations Dashboard (Final State):** Shows the final state of the product list with the updated entry.

## **Висновок:**

У ході дипломного проекту розроблено комп'ютерну систему для централізованого управління мережею торгових точок, яка автоматизує ключові бізнес-процеси — від обліку товарів до звітності. Проведено аналіз існуючих рішень і виявлено їхні недоліки, що спонукало створити власний продукт, орієнтований на малий та середній бізнес в Україні. Проект базується на клієнт-серверній архітектурі з використанням **C#, ASP.NET Core, WinForms і PostgreSQL**. Реалізовано функціонал авторизації, ролей, звітності у форматах **HTML** і **XLSX**, а також забезпечено масштабованість і гнучкість системи. Проведено тестування клієнтської частини та **API**.

У перспективі планується розширення функціоналу системи за рахунок переходу на мікросервісну архітектуру, інтеграції з сучасними засобами безпеки та впровадження аналітики користувачької поведінки. Також розглядається інтеграція з хмарними сервісами для підвищення надійності, масштабованості та ефективності роботи з великими обсягами даних.

Таким чином, система є сучасним, гнучким інструментом, готовим до подальшого розвитку і адаптації під реальні потреби бізнесу.



**ДЯКУЮ ЗА УВАГУ**