НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2025 р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою “Цифрові технології в енергетиці”

Спеціальності 122 “Комп’ютерні науки”

на тему: “Застосунок з інтегрованою рекомендаційною системою для обліку товарів”

Виконав: студент 4 курсу, групи ТР-12
Калінін Ярослав Олегович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асистент каф. ЦТЕ, Костянтин ЗДОР _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент професор каф. ТАЕ, д.т.н., професор Михайло АБДУЛІН _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль доцент каф. ЦТЕ, Артем ГУРІН _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
(підпис)
“___” _____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

Калініну Ярославу Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Застосунок з інтегрованою рекомендаційною системою для обліку товарів”
Науковий керівник Здор Костянтин Андрійович
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “ 02 ” червня 2025 року № 1875-с
2. Термін подання студентом роботи 09.06.2025
3. Вихідні дані до роботи мова програмування C#, фреймворк .NET Framework, фреймворк машинного навчання ML.NET, ORM Entity Framework Code First, графічний інтерфейс Windows Forms, СУБД Microsoft SQL Server, середовище розробки JetBrains Rider
4. Перелік питань, які потрібно розробити _____
1) Проведення аналізу існуючих підходів та наявних аналогів до автоматизації обліку товарів, вибір найкращих технологічних рішень.
2) Формування та деталізація функціональних і технічних вимог до програмного продукту.

3) Обґрунтування вибору технологічного стеку, алгоритму проєктування та архітектури застосунку.
4) Проєктування та розробка функціональних модулів системи, включаючи реалізацію бази даних.
5) Впровадження інтегрованої рекомендаційної системи та механізму шифрування даних.
6) Тестування застосунку, аналіз отриманих результатів та оцінка ефективності впровадженого рішення.

5. Орієнтований перелік ілюстративного матеріалу ER-діаграма бази даних, діаграма прецедентів (ролей), діаграма класів системи, схема взаємодії компонентів застосунку, скріншоти інтерфейсу взаємодії користувачів з застосунком.

6. Дата видачі завдання 13.09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	11.03.2025-12.04.2025	
2.	Аналіз методів та засобів розв'язання задачі	17.04.2025-18.04.2025	
3.	Розробка архітектури та загальної структури системи	19.04.2025-22.04.2025	
4.	Розробка окремих підсистем	22.04.2025-28.04.2025	
5.	Програмна реалізація системи	29.04.2025-07.05.2025	
6.	Оформлення пояснювальної записки	08.05.2025-12.05.2025	
7.	Захист програмного забезпечення	12.05.2025-16.05.2025	
8.	Передзахист	26.05.2025-28.05.2025	
9.	Захист	16.06.2025-20.06.2025	

Студент _____ Ярослав КАЛІНІН
(підпис) (ім'я ПРІЗВИЩЕ)

Керівник _____ Костянтин ЗДОР
(підпис) (ім'я ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 55 сторінках, містить 18 ілюстрацій, 1 таблицю, 1 додаток, 18 джерел в переліку посилань.

Мета роботи – розробка програмного забезпечення для автоматизованого обліку товарів із вбудованою рекомендаційною системою товарів для підвищення ефективності управління запасами та оптимізації процесу продажів у локальному бізнесі.

Методи та засоби – алгоритм k-ближчих сусідів із використанням фреймворку системи машинного навчання ML.NET, проектування структури бази даних за допомогою Entity Framework Code First, реалізація графічного інтерфейсу засобами Windows Forms, організація взаємодії з базою даних Microsoft SQL Server, криптографічний захист даних із використанням алгоритму SHA256.

Результат – програмний засіб автоматизованого обліку товарів з інтегрованою рекомендаційною системою товарів для підвищення ефективності продажів і аналітики.

Ключові слова: ОБЛІК ТОВАРІВ, АВТОМАТИЗАЦІЯ, ПРОДАЖІ, АНАЛІТИКА.

ANNOTATION

The diploma thesis consists of 55 pages, includes 18 illustrations, 1 table, 1 appendix, and 18 sources in the reference list.

The purpose of the work is to develop software for automated inventory management with an integrated goods recommendation system to increase stock management efficiency and optimize the sales process in a local business.

Methods and tools: k-nearest neighbors algorithm using the ML.NET machine learning framework, database structure design with Entity Framework Code First, implementation of a graphical interface using Windows Forms, integration with a Microsoft SQL Server database, and cryptographic data protection using the SHA256 algorithm.

The result is a software tool for automated inventory management with an integrated goods recommendation system aimed at improving sales efficiency and analytics.

Keywords: INVENTORY MANAGEMENT, AUTOMATION, SALES, ANALYTICS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ООП (об’єктно-орієнтоване програмування) – парадигма, в якій дані й функції поєднані в об’єкти, що підвищує модульність, повторне використання коду й спрощує розробку складних систем.

DI (Dependency Injection) – архітектурний підхід, при якому залежності між компонентами програми задаються зовні, що підвищує гнучкість, тестованість і безпеку системи.

CRM (Customer Relationship Management) – система управління взаємовідносинами з клієнтами, що дозволяє централізовано зберігати дані про клієнтів, відстежувати історію взаємодії та автоматизувати процес продажів.

POS (Point of Sale) – система автоматизації торгових операцій, обліку товарів і розрахунків із покупцями.

ORM (Object-Relational Mapping) – технологія відображення об'єктів програмного коду у структури бази даних; у роботі використано Entity Framework.

EF (Entity Framework) – ORM-фреймворк для платформи .NET, що забезпечує роботу з базою даних у вигляді об'єктів.

LINQ (Language Integrated Query) – технологія для здійснення запитів до колекцій даних у C# та інших мовах .NET.

WinForms (Windows Forms) – бібліотека для створення графічного інтерфейсу користувача у Windows-додатках.

SQL Server – система управління реляційними базами даних, розроблена компанією Microsoft.

SHA256 – криптографічний хеш-алгоритм для захисту паролів та інших конфіденційних даних.

ML.NET – фреймворк для машинного навчання на платформі .NET, що дозволяє реалізовувати рекомендальні системи та інші алгоритми штучного інтелекту.

ВСТУП

У сучасних умовах розвитку цифрової економіки особливої значущості набуває завдання автоматизації бізнес-процесів у малому та середньому бізнесі. Більшість наявних досліджень і прикладних рішень, описаних в літературі з автоматизації [1], підкреслюють необхідність переходу до інноваційних методів обліку товарів і оптимізації продажів, зокрема через впровадження рекомендаційних систем. Традиційні методи ведення товарного обліку, які передбачають ручну обробку інформації, вже не відповідають зростаючим вимогам ефективності, точності та оперативності. Вони часто призводять до численних помилок, втрат даних, невчасного оновлення інформації щодо запасів і залишків товару, що, у свою чергу, негативно впливає на прибутковість та конкурентоспроможність бізнесу.

Саме тому розробка застосунку для обліку товарів є актуальною задачею сьогодення, особливо враховуючи гнучкість програми і інтеграцію рекомендаційної системи. Автоматизація процесів обліку товарних позицій, ведення клієнтської бази, управління персоналом і контроль за продажами дозволяють суттєво оптимізувати внутрішні процеси компаній, зменшити ризики помилок, підвищити ефективність використання ресурсів і забезпечити прозорість усіх операцій. Інтеграція рекомендаційної системи дозволяє додатково стимулювати продажі, пропонуючи клієнтам супутні або популярні товари, що сприяє збільшенню середнього чеку та загальної лояльності покупців.

Метою дипломної роботи є розробка програмного забезпечення для автоматизованого обліку товарів із інтегрованою рекомендаційною системою, що забезпечує оптимізацію управління запасами й підвищення ефективності продажів у локальному бізнесі. Підвищення ефективності досягнуто шляхом впровадження сучасних інформаційних технологій та залученню інтеграційної системи. Для досягнення цієї мети були поставлені наступні завдання:

– аналіз підходів, методів та алгоритмів розв’язання задачі автоматизації обліку товарів і впровадження рекомендаційних систем;

– аналіз програмних засобів для реалізації інформаційної системи обліку товарів;

– розробка програмного забезпечення для автоматизованого обліку товарів із рекомендаційною підсистемою;

– забезпечення надійного підключення до бази даних для зберігання та оперативного доступу до інформації;

– впровадження модулю звітності та відстеження історії змін для ефективного контролю бізнес-процесів;

– реалізація рекомендаційної системи вибору товарів на основі алгоритму k-ближчих сусідів із використанням фреймворку системи машинного навчання ML.NET;

– тестування розробленого програмного забезпечення та оцінка ефективності впровадженого рішення.

На сьогодні існує ряд програмних продуктів, які частково задовольняють потреби автоматизації малого бізнесу, однак їх більшість орієнтована на більш великі підприємства або не має інтегрованих систем рекомендацій. Тому розробка комплексного рішення, яке враховує специфіку локального бізнесу і містить інтегровану рекомендаційну систему, залишається відкритим завданням.

Об’єктом дослідження для написання роботи стали сучасні програмні рішення з автоматизації товарного обліку, які використовуються малими та середніми підприємствами. Аналіз цих рішень дозволив визначити оптимальні підходи до створення застосунку, а також визначити прогалини, якими варто доповнити систему, задля відповідності вимогам локального бізнесу.

1 АНАЛІЗ ВИМОГ ТА ІСНУЮЧИХ РІШЕНЬ З ОБЛІКУ ТОВАРІВ

Грамотний аналіз вимог є передумовою успіху будь-якого програмного продукту, адже саме він дозволяє зрозуміти реальні бізнес-процеси й виявити слабкі місця в існуючих системах обліку товарів. Замість простої автоматизації рутинних операцій крамниць і складів необхідно створити рішення, яке гнучко адаптується до потреб різних користувачів – від власника магазину, що потребує оперативної інформації про залишки, до комірника, для якого критично швидко знайти потрібну позицію. При цьому не меншу роль відіграють і менеджери, які зацікавлені в аналітиці продажів, формуванні персоналізованих пропозицій і контролі за промоакціями. Щоб забезпечити таку універсальність, варто проаналізувати, які дані формуватимуть основу системи: перелік товарів, картки клієнтів, історія транзакцій, умови знижок. Окрім того, слід оцінити, які підходи були випробувані на практиці – від традиційних реляційних баз даних до сучасних рекомендаторів, здатних підказувати клієнтам супутні товари. Завдяки детальному аналізу вимоги та існуючих рішень можна обрати оптимальні архітектурні рішення, алгоритми обробки даних і технологічний стек, що дозволить зменшити час розробки та забезпечить надійність готового застосунку.

1.1 Призначення та сфера застосування програмного забезпечення

Розроблене програмне забезпечення призначене для автоматизації та оптимізації внутрішніх процесів обліку товарних запасів, ведення детальної інформації про клієнтів, управління персоналом, контролю за продажами та адміністрування пропозицій зі знижкою у локальному бізнесі. Його основними користувачами є малі та середні підприємства різних галузей – магазини роздрібної торгівлі, оптові склади, торговельні точки й інтернет-магазини.

Застосунок дозволяє керівництву та співробітникам компаній оперативно відстежувати зміни в асортименті, залишки товарів, аналізувати тренди покупок і управляти товарними запасами у режимі реального часу. Значною перевагою програмного рішення є наявність інтегрованої рекомендаційної системи, яка, на основі аналізу історії покупок клієнтів та загальних статистичних даних, пропонує додаткові супутні товари або вигідні пропозиції. Це суттєво сприяє збільшенню середнього чеку покупок та підвищує рівень лояльності клієнтів.

Крім того, програмне забезпечення дозволяє автоматично формувати аналітичні звіти з історією транзакцій, проводити систематизацію інформації про клієнтів, персоналізувати підхід до кожного покупця та оперативно реагувати на зміни попиту і пропозиції на ринку. Важливим аспектом реалізації є забезпечення високого рівня безпеки даних, що гарантується сучасними методами шифрування інформації, зокрема SHA256, а також гнучкої системи розмежування доступу до функцій і даних у залежності від ролі користувача.

Завдяки продуманому та інтуїтивно зрозумілому інтерфейсу система є зручною у використанні навіть для співробітників, які не володіють спеціальними навичками роботи з інформаційними технологіями. Це дозволяє впроваджувати програму швидко і без зайвих витрат часу на навчання персоналу. Таким чином, застосунок охоплює широкий спектр завдань, необхідних для ефективного функціонування та розвитку малого і середнього бізнесу.

1.2 Характеристика потенційних користувачів системи

Потенційними замовниками та користувачами розробленого програмного забезпечення є малі та середні підприємства, які здійснюють свою діяльність у сфері роздрібної та оптової торгівлі. До таких бізнесів належать локальні магазини, торгові точки, склади, а також інтернет-магазини, які потребують систематизації та оптимізації внутрішніх процесів обліку товарів.

Для невеликих магазинів і торгових точок впровадження такого програмного рішення дозволить оперативно контролювати залишки товарів, запобігати їх

нестачі чи надлишку, а також зменшити час на формування звітів. Відтак, власники приділятимуть більше уваги розвитку, маркетингу та взаємодії з клієнтами.

Для складів та оптових компаній інтеграція рекомендаційної системи допоможе визначати найпопулярніші товари і своєчасно оновлювати асортимент, що сприятиме підвищенню ефективності роботи та уникненню збитків через застарілі або неліквідні товарні залишки.

Інтернет-магазинам ця система є корисною завдяки автоматизації формування пропозицій супутніх товарів, що дозволить збільшити середній чек, покращити клієнтський досвід та підвищити лояльність покупців.

Завдяки цьому можна заявити, що потенційні замовники системи – це компанії, які прагнуть оптимізувати процеси управління товарними потоками, покращити взаємодію з клієнтами і посилити свої конкурентні позиції на ринку за рахунок сучасних технологічних рішень.

1.3 Завдання, що вирішуються програмним продуктом

Розроблений програмний застосунок орієнтований на вирішення широкого спектру завдань, пов’язаних з автоматизацією та оптимізацією процесів обліку й управління товарними ресурсами на підприємствах малого і середнього бізнесу.

До основних завдань, які вирішуються системою, належать:

– автоматизація процесів обліку товарів: контроль залишків, руху товарів, поповнення асортименту, мінімізація помилок, що виникають при ручному веденні записів;

– управління клієнтською базою: зберігання актуальної інформації про клієнтів, аналіз історії покупок, персоналізація обслуговування для підвищення лояльності клієнтів;

– інтегрована рекомендаційна система: автоматичне формування рекомендацій щодо супутніх товарів на основі аналізу даних про попередні покупки, що дозволяє підвищити продажі;

– управління персоналом та рівнями доступу: забезпечення захищеного доступу до інформації та функціоналу системи, чіткий розподіл прав і обов’язків серед співробітників відповідно до їх ролей;

– система звітності: автоматичне створення аналітичних і статистичних звітів щодо продажів, залишків та змін у товарних запасах, що допомагає ефективно керувати бізнес-процесами;

– впровадження гнучкої системи знижок та акцій: можливість встановлення різноманітних акційних пропозицій, що допомагає залучати нових клієнтів та утримувати постійних;

– безпека та збереження даних: використання сучасних методів шифрування для захисту інформації, що гарантує безпечне зберігання та обробку даних.

Комплексне вирішення цих завдань дозволяє суттєво підвищити ефективність та конкурентоспроможність підприємства, а також забезпечує стабільність і прозорість бізнес-процесів.

1.4 Вхідні та вихідні дані системи

Для ефективної роботи програмного застосунку важливе чітке визначення вхідних і вихідних даних, що забезпечує коректність обробки інформації й прийняття управлінських рішень.

До вхідних даних системи належать:

– відомості про товари: назва, категорія, ціна, характеристики, кількість на складі, постачальник;

– дані клієнтів: ПІБ, контактний номер телефону, електронна пошта, адреса доставки;

– інформація про співробітників: ПІБ, контактні дані, посада, рівень доступу, пароль і статус співробітника;

– відомості про транзакції: дата продажу, перелік товарів, кількість проданого товару, інформація про клієнта та суму покупки;

– дані для формування знижок: розмір знижки, товари, на які вона поширюється, а також термін дії акції.

До вихідних даних програмного продукту входять:

– аналітичні та статистичні звіти про рух запасів, продажі й поповнення товарів;

– звіти з історії покупок клієнтів, що дозволяють аналізувати поведінку споживачів та покращувати якість обслуговування;

– автоматично згенеровані рекомендації щодо супутніх або додаткових товарів на основі аналізу історичних даних;

– дані про роботу співробітників, зокрема інформація про виконані дії в системі.

Завдяки такій структурі даних система забезпечує оперативний обмін інформацією, сприяє автоматизації бізнес-процесів та підвищує ефективність управління підприємством.

1.5 Аналіз методів, підходів і програмних засобів для автоматизації обліку товарів

У розробці програмного забезпечення для автоматизованого обліку товарів було застосовано низку сучасних інженерних підходів, що забезпечують надійність, масштабованість і гнучкість системи при подальшому супроводі та розвитку. Ключовим принципом стала орієнтація на ООП, що дозволило виділити основні бізнес-сутності, такі як товари, клієнти, співробітники, транзакції і знижки у вигляді окремих класів, кожен із яких відповідає за власну частину функціоналу та даних.

Додатково було впроваджено принципи багаторівневої архітектури, окремо виділено рівень взаємодії з базою даних, рівень бізнес-логіки та рівень користувацького інтерфейсу. Такий підхід гарантує чітке розмежування відповідальності між модулями й дозволяє масштабувати систему без порушення вже існуючого функціоналу. Централізований контекст даних забезпечує

уніфікований доступ до об'єктів бази та дозволяє керувати всіма змінами у транзакціях через єдиний шлюз.

Особливу увагу під час проєктування системи приділено принципам модульності та гнучкості, відтак основна функціональність розподілена між тематичними модулями. Кожен модуль є відносно незалежним, що полегшує впровадження нових функцій або редагування наявних. Це підвищує стабільність системи в цілому та скорочує ризик виникнення помилок при оновленнях.

Для підвищення безпеки системи реалізовано систему контролю доступу на основі ролей (адміністратор, менеджер, комірник). Ролі визначають набір дозволених дій і гарантують, що співробітники мають доступ лише до тих функцій, які необхідні для виконання їхніх обов’язків. Це не лише підвищує захищеність інформації, а й спрощує аудит дій користувачів та адміністрування системи.

В системі обліку впроваджено автоматизовану валідацією даних та впровадженням dependency injection – всі критично важливі операції, такі як оформлення продажу, надання знижки, чи реєстрація нового клієнта супроводжуються перевіркою на коректність введених даних та відповідність діловим правилам. Завдяки цьому вдається мінімізувати людський фактор і забезпечити достовірність даних.

Ключовим інноваційним елементом застосунку є інтеграція рекомендаційної підсистеми. Для її реалізації було обрано алгоритм k-ближчих сусідів, що реалізовано з використанням фреймворку машинного навчання ML.NET та технології LINQ. Це дозволяє в реальному часі аналізувати історію покупок кожного клієнта, виділяти патерни купівельної поведінки й формувати персоналізовані рекомендації супутніх товарів. Таким чином, система не лише автоматизує облік і звітність, а й підвищує ефективність продажів, сприяючи зростанню середнього чеку та загальної лояльності клієнтів.

Загалом застосування сучасних методів і підходів до архітектури, модульності, контролю доступу та інтеграції алгоритмів машинного навчання дозволяє забезпечити не лише відповідність розробленої системи сучасним вимогам бізнесу, а й її гнучкість, масштабованість і готовність до подальшого розвитку.

1.6 Аналіз використаних програмних засобів та технологій

Вибір технологічного стеку для реалізації програмного забезпечення здійснювався з урахуванням вимог до продуктивності, надійності, масштабованості, безпеки й зручності підтримки системи в умовах реальної експлуатації. Кожен із використаних інструментів має стратегічне значення для досягнення поставлених у роботі цілей.

Базовою мовою програмування обрано C#, яка є лідером для розробки сучасних корпоративних і десктопних рішень завдяки синтаксичній зрозумілості і гнучкості при роботі із зовнішніми бібліотеками. Застосування платформи .NET Framework забезпечує високий рівень сумісності з різними версіями Windows, стабільність, а також ефективну роботу із системними ресурсами ПК.

Графічний інтерфейс програми створено за допомогою Windows Forms. Дана технологія дозволяє швидко будувати зрозумілі та адаптивні інтерфейси для настільних додатків, використовуючи drag-and-drop компоненти, дієву модель і автоматичну інтеграцію з обробкою даних. Такий вибір сприяє як швидкому впровадженню, так і легкому подальшому супроводженню застосунку.

Зберігання даних організовано на базі Microsoft SQL Server, що є сучасною реляційною СУБД, яка підтримує транзакційність, забезпечує захист від втрати інформації, має розвинуті інструменти для резервного копіювання, масштабування та аудиту. Для інтеграції програмної логіки з базою даних застосовується ORM Entity Framework із підходом Code First, що дозволяє управляти структурою БД безпосередньо з коду, здійснювати автоматичні міграції та синхронізацію схем.

Для підвищення захищеності та гнучкості архітектури у проєкті реалізовано технологію dependency injection, що дає змогу впроваджувати залежності між компонентами програми через абстракції, а не жорсткі прив’язки. Такий підхід дозволив мінімізувати ризик витоку або несанкціонованого доступу до критичних об’єктів та сервісів, підвищити тестованість модулів, а також спростити обслуговування й масштабування системи за рахунок зниження зв’язаності між окремими компонентами.

Інтеграція ML.NET надала можливість імплементувати рекомендаційну підсистему із використанням алгоритму k-ближчих сусідів, а також забезпечити подальшу адаптацію системи до нових вимог без значних змін у коді основного застосунку. Саме використання фреймворку машинного навчання відкриває перспективи для розвитку й інших інтелектуальних функцій, серед яких прогнозування попиту і автоматизація визначення трендів.

Додатково в роботі використовується технологія LINQ для здійснення складних запитів до об’єктів, реалізації фільтрації, сортування та агрегації даних у зручному синтаксисі. Це скорочує обсяг коду й підвищує читабельність бізнес-логіки.

Безпека реалізована через комплекс заходів зі зберіганням паролів користувачів у вигляді хешів із використанням стійкого алгоритму SHA256. Багаторівнева система розмежування доступу і перевірка даних на всіх етапах обробки додатково підсилює систему і дозволяє відповідати сучасним стандартам захисту інформації й гарантувати цілісність бази даних.

У комплексі перелічені технології формують сучасний, адаптивний та захищений програмний продукт, що не лише виконує основні бізнес-завдання, а й закладає надійний фундамент для подальшого розвитку та інтеграції з сервісами.

2 ТЕХНОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка якісного програмного забезпечення ґрунтується на ретельному виборі технологій вибору технологій та методологій, які забезпечать стабільність, безпеку й зручність у подальшій підтримці. У цьому розділі йдеться про ті методи, підходи та алгоритми, які стали основою реалізації даного застосунку з рекомендаційною системою. У випадку розробки застосунку з вбудованою рекомендаційною системою для обліку товарів вкрай важливим є акцент на платформи, фреймворки та підходи до побудови архітектури, що допоможуть у майбутньому швидко масштабувати продукт і адаптувати його до змін потреб бізнесу.

Для успішної реалізації проєкту було обрано алгоритми, архітектурниі рішення та програмні засоби, які якнайкраще впливають на працездатність системи і задовольняють виконання поставлених завдань. Важливим є особливості впровадження рекомендаційної підсистеми, забезпечення безпеки даних, а також використання інноваційних технологій, які забезпечують гнучкість і масштабованість розробленого продукту.

2.1 Огляд існуючих рішень і літературних джерел

Автоматизація обліку товарів у малому та середньому бізнесі є одним із ключових чинників підвищення конкурентоспроможності сучасних підприємств. Згідно з аналітичними оглядами й результатами досліджень у сфері інформаційних систем, більшість успішних проєктів побудовані на засадах централізованого зберігання даних, багаторівневої архітектури та застосування модульного підходу до формування функціоналу [2].

У літературі питання автоматизації бізнес-процесів розглядаються в контексті впровадження ERP-систем, оптимізації логістики, управління складом і

підвищення якості обслуговування клієнтів [3]. Значна увага приділяється розмежуванню між рівнями бізнес-логіки й інтерфейсної частини, що є фундаментом для масштабованості, стійкості до змін та подальшого розвитку проєктів. Як зазначає М. Фаулер у класичній праці "Patterns of Enterprise Application Architecture", розподіл відповідальності між компонентами та використання шаблонів проектування (design patterns) дозволяють підвищити гнучкість і підтримуваність системи [1].

З огляду на потреби малого бізнесу, відзначають важливість простоти інтерфейсу та можливості швидкого впровадження без залучення дорогих консалтингових ресурсів. Особливо затребуваними є рішення, які дозволяють автоматизувати основні операції, серед яких визначають облік товарів, оформлення продажів і ведення клієнтської бази, а також забезпечують гнучку звітність. Значна частина сучасних застосунків орієнтована на хмарні платформи, однак локальні системи зберігають популярність через простоту адміністрування, відсутність залежності від інтернету й більший контроль над безпекою даних.

З технічного боку більшість рішень реалізовані із застосуванням реляційних СУБД та мов програмування загального призначення, таких як C#, а також сучасних фреймворків для створення інтерфейсів користувача [4]. Практика показує, що вибір технології здебільшого визначається не лише вартістю впровадження, а й наявністю технічної підтримки, документації та можливістю інтеграції із зовнішніми модулями CRM, бухгалтерії та платіжних шлюзів.

Окремий напрям сучасних систем – впровадження рекомендаційних підсистем, які дозволяють використовувати накопичені дані для формування персоналізованих пропозицій клієнтам. У наукових роботах [5, 6] описуються різні підходи до рекомендацій: від простих механізмів, як от пропозиція супутніх товарів на основі попередніх покупок, до складних алгоритмів колаборативної фільтрації, машинного навчання та використання моделей штучного інтелекту. Саме рекомендаційні системи є тією конкурентною перевагою, яка дозволяє бізнесу не лише реєструвати продажі, а й активно впливати на динаміку попиту та середній чек за рахунок таргетованих пропозицій.

Значна увага також приділяється питанням інформаційної безпеки. На сучасному етапі захист даних стає пріоритетом для будь-якої бізнес-системи, особливо якщо вона працює з персональними даними клієнтів і фінансовими транзакціями. Базові вимоги – це зберігання паролів у вигляді хешів, захищене з'єднання, багаторівнева система доступу та аудит дій користувачів.

Підсумовуючи, аналіз літературних джерел і сучасних рішень свідчить про доцільність використання в облікових системах модульної архітектури, застосування реляційних СУБД, інтеграції рекомендаційних алгоритмів і акценту на безпеку та персоналізацію. Ці підходи взято за основу при створенні власної системи автоматизації обліку товарів, що дає змогу поєднати перевірені на практиці технології із сучасними вимогами бізнесу.

2.2 Порівняльний аналіз функціональних можливостей конкурентних систем

На сучасному ринку програмних продуктів для автоматизації обліку товарів представлено значну кількість як локальних, так і хмарних рішень. Для порівняння було обрано два найбільш розповсюджених продукти – Poster POS та Zoho Inventory, які користуються популярністю серед малого та середнього бізнесу.

Poster POS позиціонується як хмарна POS-система, орієнтована на автоматизацію торговельних операцій у роздрібній торгівлі, кафе, ресторанах і точках громадського харчування [7]. Система надає зручний веб-інтерфейс, дозволяє вести облік товарів, контроль складу, оформлення продажів, а також формувати детальні аналітичні звіти щодо руху запасів і фінансів. Серед ключових переваг Poster POS – простота налаштування та використання, мобільність, підтримка інтеграцій з CRM, платіжними сервісами та бухгалтерським обліком. Однак система має і певні недоліки: відсутність можливості глибокої адаптації під унікальні бізнес-процеси, платна підписка у вигляді щомісячної оплати, а також брак вбудованої рекомендаційної підсистеми для персоналізації пропозицій покупцям [8]. Приклад функціоналу системи зображено на рисунку 2.1.

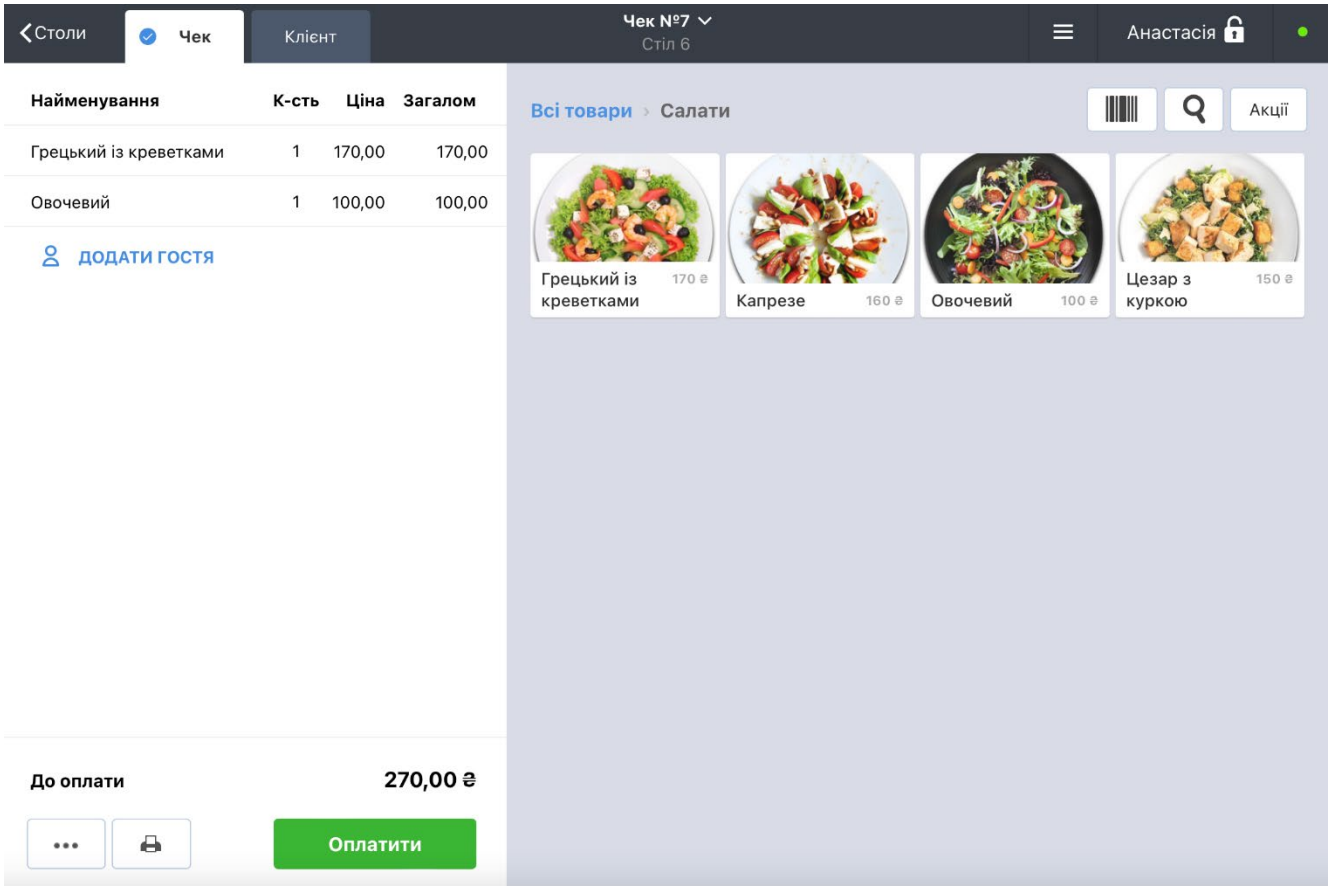


Рисунок 2.1 – Інтерфейс системи Poster POS

Zoho Inventory – це частина великої екосистеми Zoho, яка охоплює інструменти для автоматизації складу, обліку замовлень, доставки та інтеграції з різними онлайн-платформами, серед яких є маркетплейси, CRM та бухгалтерія [9]. Серед основних функцій системи – підтримка штрихкодів і серійних номерів, мультивалютність, налаштування прав доступу, автоматичне формування документів, гнучка аналітика та онлайн-доступ із будь-якої точки світу. До переваг сервісу можна віднести широку інтеграцію з іншими сервісами, зручний інтерфейс, зображений на рисунку 2.2 , масштабованість для міжнародного бізнесу.

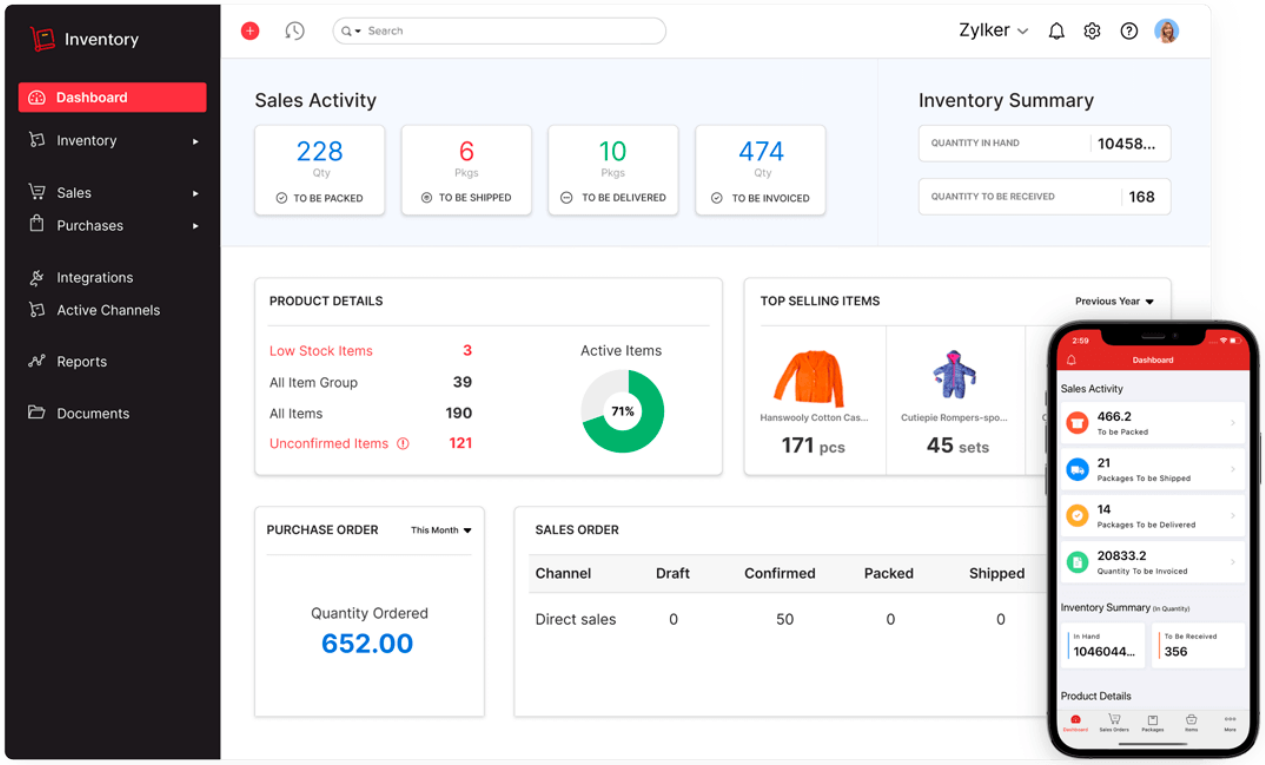


Рисунок 2.2 – Інтерфейс системи Zoho Inventory

Попри наявність переваг і зручного інтерфейсу, адаптація Zoho Inventory до локальних особливостей бізнесу може бути складною, безплатна версія сильно обмежена у своїх функціональних можливостях, а рекомендаційна підсистема якщо і наявна, то реалізована у вигляді простих інструментів і не орієнтована на складні шаблони купівельної поведінки [5]. Узагальнені результати функціонального аналізу конкурентних систем наведено у таблиці 2.1, яка ілюструє основні відмінності та переваги Poster POS, Zoho Inventory і розробленого рішення.

Таблиця 2.1 – Порівняльна таблиця функціональних можливостей

Критерій	**Poster POS**	**Zoho Inventory**	**Власна розробка**
Тип системи	Хмарна	Хмарна	Десктопна/локальна
Облік товарів	+	+	+
Облік клієнтів	+	+	+
Інтеграція з CRM	+	+	Можлива
Автоматичні звіти	+	+	+

Продовження таблиці 2.1

Критерій	Poster POS	Zoho Inventory	Власна розробка
Підтримка знижок/акцій	+	+	+
Мобільний доступ	+	+	-
Можливість кастомізаці	-	+	+
Вбудована рекомендаційна система	-	частково	+
Вартість	Передплата	Передплата/обмежена безкоштовна версія	Безкоштовна для підприємства після реалізації
Мова інтерфейсу	EN/UA	EN/UA (частково)	EN/UA
Локалізація під бізнес	Обмежена	Обмежена	Повна

Проведений аналіз показує, що хоча ринкові продукти Poster POS та Zoho Inventory пропонують широкий набір стандартних функцій для обліку товарів і клієнтів, вони не завжди дозволяють гнучко адаптувати систему під специфіку локального бізнесу. Основні обмеження – це відсутність індивідуалізованої системи рекомендацій, складність доопрацювання під нестандартні бізнес-процеси та залежність від сторонніх сервісів і постійної оплати. Розробка власного програмного продукту дозволяє врахувати ці нюанси, забезпечити інтеграцію з аналітикою та рекомендаційними алгоритмами, а також адаптувати систему під конкретні вимоги підприємства без зайвих витрат.

2.3 Обґрунтування вибору методів і технологій

Фреймворк, на базі якого створено програмний продукт, визначає ключові підходи до реалізації бізнес-логіки, взаємодії з базою даних та побудови інтерфейсу користувача. У рамках розробки системи обліку товарів для локального бізнесу

було обрано .NET Framework як основну платформу для програмної реалізації. Такий вибір обумовлений стабільністю, доведеною ефективністю в розробці десктопних додатків, широкою підтримкою інструментів розробника та простотою інтеграції з різноманітними бібліотеками й сервісами. .NET Framework дозволяє фокусуватися на реалізації бізнес-функціоналу без необхідності вирішення складних сумісних питань, а велика база навчальних матеріалів і документації спрощує впровадження навіть складних рішень, зокрема інтеграцію з базами даних та UI [10].

Щодо організації зберігання інформації було обрано Microsoft SQL Server як основну СУБД, оскільки ця система забезпечує високу продуктивність, стабільність, підтримує транзакційність, масштабування та контроль цілісності даних. Особливо актуальним для локального бізнесу є застосування версії SQL Server, яка легко розгортається локально, не потребує плати за хмарне зберігання та дає підприємству повний контроль над власною інформацією. Багатий функціонал, що включає індексацію, тригери, розмежування прав доступу та аудит дій користувачів, робить Microsoft SQL Server ідеальним вибором для систем автоматизації обліку [11].

Для ефективної взаємодії із СУБД застосовується ORM-фреймворк Entity Framework, що дозволяє розробляти програмний код на C# і безпосередньо відображати об’єкти програми у таблиці бази даних. Це значно зменшує обсяг ручного SQL-коду, знижує імовірність помилок і підвищує прозорість бізнес-логіки. Entity Framework підтримує звичний синтаксис розробки LINQ для здійснення запитів до даних, що спрощує фільтрацію, агрегацію й аналітику.

В межах реалізації системи застосовано підхід Code First, який дозволяє спочатку створювати класи-сутності у програмному коді, а вже потім на їх основі автоматично генерувати структуру БД. Такий підхід значно спрощує супровід системи, адже усі зміни в моделях зберігаються у коді, а при оновленні структури БД за допомогою міграцій, автоматично синхронізуються з базою. Такий підхід мінімізує людський фактор і дає змогу під час розробки максимально швидко вносити правки, не втрачаючи зв'язок із реальними даними. Крім того, Entity

Framework підтримує всі основні типи зв'язків, що забезпечує гнучкість структурування даних.

Механізм міграцій у Entity Framework є особливо зручним для проектів, які постійно вдосконалюються. Міграції дозволяють відслідковувати й впроваджувати зміни в структурі БД без втрати даних і ручного втручання в SQL-код. Таким чином, супровід програми стає гнучким та передбачуваним [12].

2.4 Рекомендаційна система та безпека даних

Сучасні системи управління бізнес-процесами поступово інтегрують інтелектуальні підсистеми, що дозволяють підвищувати ефективність і персоналізувати роботу з клієнтами. Одним із таких рішень є рекомендаційна система, яка реалізована на основі ML.NET та алгоритму k-ближчих сусідів. Рекомендаційна підсистема аналізує історію транзакцій клієнтів та формує динамічні пропозиції супутніх товарів. Це дозволяє не лише фіксувати факт покупки, а й активно впливати на поведінку клієнта, підвищувати середній чек і загальну лояльність.

Причина вибору ML.NET полягає в його повній інтеграції з екосистемою .NET, підтримці популярних алгоритмів машинного навчання, включаючи кластеризацію, класифікацію та колаборативну фільтрацію. На рисунку 2.3 зображено принцип дії машинного навчання з використанням технології ML.NET. Це дає можливість легко розширювати функціонал системи з легкою інтеграцією архітектури програми.

У реалізації використано гібридний підхід, відтак для найшвидших рекомендацій застосовано оптимізовані LINQ-запити до історії покупок, тоді як для більш складних сценаріїв задіяні механізми машинного навчання, що дозволяє системі адаптуватися до змін у поведінці клієнтів [13].

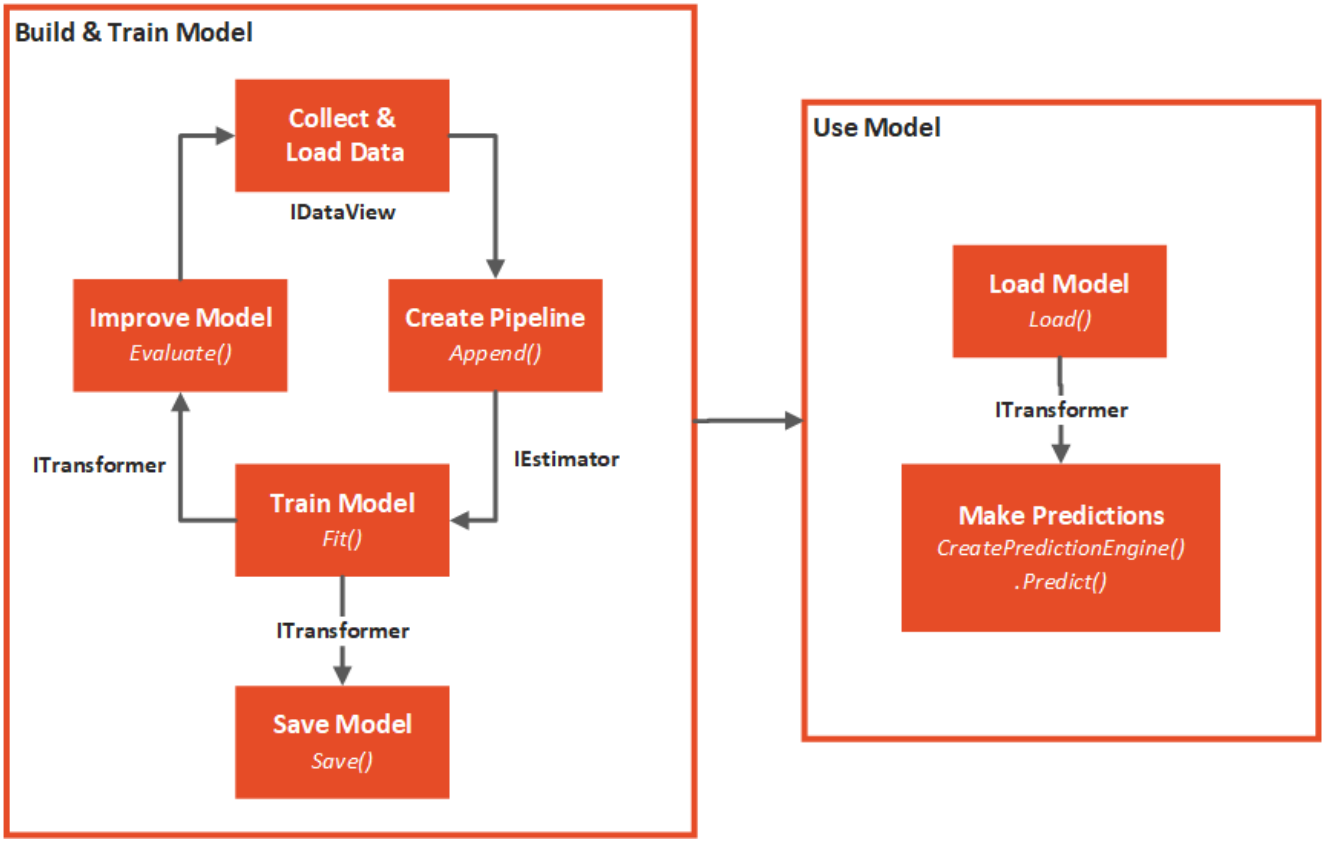


Рисунок 2.3 – Загальна схема побудови та застосування моделі в ML.NET

Оскільки програмний застосунок має бути не лише сучасним, а й безпечним, інформаційна безпека є ключовою вимогою до систем автоматизації в бізнесі, особливо тих, що працюють із персональними та фінансовими даними. На етапі проєктування системи значна увага була приділена питанням інформаційної безпеки. Зокрема, для захисту облікових записів користувачів у програмі реалізовано збереження паролів у вигляді криптографічних хешів, сформованих за допомогою алгоритму SHA256 [14]. Такий підхід гарантує, що справжній пароль не можна отримати навіть у разі отримання доступу до бази даних, оскільки хеш-функція є незворотною [15].

Додатково в системі впроваджено багаторівневу систему контролю доступу: кожна роль користувача отримує лише ті права, які необхідні для виконання посадових обов’язків, що значно знижує ймовірність несанкціонованих дій та втрати важливої інформації.

Ще однією важливою особливістю є локальне розміщення бази даних на комп’ютері підприємства. Це рішення дозволяє забезпечити повний контроль над

процесами зберігання та обробки даних, мінімізуючи ризики втрати чи несанкціонованого доступу через зовнішні мережі або сервіси..

Застосування таких технологій і підходів забезпечує не лише стійкість системи до атак чи помилок, а й відповідність сучасним стандартам захисту персональних та бізнес-даних [16].

2.5 Реалізація алгоритмів рекомендаційної системи

Інтеграція рекомендаційної підсистеми є однією з ключових інновацій розробленої системи обліку товарів. Вона дозволяє перейти від пасивного ведення статистики до активного формування персоналізованих пропозицій клієнтам, що підвищує ефективність продажів і покращує взаємодію з користувачем.

Для реалізації рекомендаційного механізму було обрано алгоритм k-ближчих сусідів, який має гарні рекомендації для вирішення завдань класифікації та прогнозування на основі історичних даних. Головною перевагою є здатність знаходити схожі патерни поведінки клієнтів, використовуючи інформацію про попередні покупки [17].

Процес впровадження алгоритму починається з підготовки й попередньої обробки даних: з бази даних витягується інформація про всі транзакції, де для кожного клієнта фіксується перелік придбаних товарів, час та сума покупки. Дані структуровано таким чином, щоб кожен клієнт або окрема покупка була представлена у вигляді числового вектора, де кожен елемент означає наявність чи відсутність придбання конкретної позиції. Такі характеристики як назви товарів та їх категорія переводяться у числовий вигляд методом векторизації ознак. Такий підхід дозволяє формалізувати дані для роботи з моделлю машинного навчання.

Навчання моделі здійснюється за допомогою ML.NET, у межах якого формується простір клієнтів, де кожен з них представлений як вектор ознак. У момент здійснення нової покупки система проводить пошук k найближчих сусідів у цьому просторі за відповідною метрикою схожості. Знайшовши найбільш схожі транзакції, підсистема аналізує, які саме додаткові товари найчастіше купувалися

разом із вже вибраними, та ранжує ці товари за частотою появи у подібних комбінаціях. Сформований перелік рекомендацій передається у користувацький інтерфейс, де відображається у вигляді персоналізованих пропозицій.

Для кращого аналізу даних і налаштування пропозицій рекомендаційної системи було реалізовано змішану модель формування рекомендацій. Відтак, дані про покупки фіксуються у БД, після чого, при здійсненні нової покупки система за допомогою ML.NET аналізує схожість транзакції з історичними покупками інших користувачів, а також покупок самого клієнта, за певними ознаками – категоріями товару, частотою купівлі товарів [18]. Після здійснення аналізу, на основі алгоритму k-ближчих сусідів система формує перелік додаткових товарів, які найчастіше купували разом із основними позиціями у схожих ситуаціях.

Для підвищення швидкості реагування та масштабованості рішення було впроваджено використання LINQ-запитів, що дозволяє швидко фільтрувати й маніпулювати даними без необхідності ручного SQL-програмування. Завдяки цьому підходу, було отримано можливість генерувати динамічні рекомендації у реальному часі, адаптувати пропозиції під зміни у поведінці покупців, працювати із відносно невеликими наборами даних, характерними для локального бізнесу, а також забезпечити модульність та розширюваність системи. Розширюваність системи дозволяє залишити функціонал для впровадження майбутніх, складніших моделей підвищеної точності, таких як колаборативна чи контентна фільтрації.

Усі етапи аналізу даних, від обробки даних і векторизації до навчання моделі й інтеграції у графічний інтерфейс реалізовано на основі екосистеми .NET, що забезпечує стабільність і швидкодію навіть на відносно невеликих вибірках даних, характерних для локального бізнесу. Детальне викладення програмного коду рекомендаційної системи подано у додатку (Додаток А).

Впровадження рекомендаційної системи дозволяє не лише підвищити рівень сервісу для постійних клієнтів, а й стимулює продажі за рахунок інтелектуального підбору супутніх товарів. Обрана технологія програмування й алгоритм аналізу даних оптимально відповідають потребам сучасного локального бізнесу й дозволяють зробити систему конкурентною порівняно з типовими обліковими рішеннями.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

Опис програмної реалізації є ключовим для розуміння того, як теоретичні напрацювання, обрані методи і технології втілюються у працюючий програмний продукт. У цьому розділі розглянуто принципи побудови архітектури системи, опис структури її основних модулів, особливості організації взаємодії між об’єктами й представленням даних у базі. Для кращого розуміння створеного застосунку також зображено логіку роботи підсистем, зокрема механізми обробки транзакцій, формування рекомендацій, реалізовані засоби контролю доступу та звітності, що поєднуються у цілісну, гнучку й надійну систему.

Окремий акцент зроблено на практичних рішеннях, які забезпечують адаптацію застосунку до реальних вимог локального бізнесу. Підтримка багаторівневої архітектури, гнучкість при внесенні змін, і автоматичне оновлення інтерфейсу надають програмному забезпеченню фінального, довершеного вигляду. Важливу роль у програмній реалізації відіграють принципи розмежування відповідальності, а також механізми захисту інформації та збереження цілісності даних у процесі експлуатації системи.

3.1 Архітектура програмної системи

Архітектура програмної системи є визначальним чинником її стабільності, масштабованості та простоти у підтримці. У процесі розробки застосунку для автоматизованого обліку товарів було обрано багаторівневу архітектурну модель, яка поєднує принципи об’єктно-орієнтованого проєктування, модульності та розмежування відповідальності між основними складовими.

Загальна структура побудована за схемою “інтерфейс – бізнес-логіка – дані”, де кожен рівень чітко відокремлений і має свою зону відповідальності. Користувач взаємодіє із застосунком через графічний інтерфейс, розроблений із використанням Windows Forms. Усі дії, які виконує користувач, такі як додавання чи редагування товарів, формування звітів або управління клієнтами, передаються на рівень бізнес-

логіки, де здійснюється перевірка правильності запитів, валідація даних, реалізується застосування знижок та облік історії транзакцій.

Важливою особливістю архітектури є впровадження єдиного центру управління даними – контексту бази (AppDbContext), який забезпечує взаємодію між бізнес-логікою та фізичним сховищем інформації. Всі зміни, які вносяться у системі, відображаються у базі даних через ORM Entity Framework за підходом Code First. Це дозволяє забезпечити цілісність і актуальність даних, уникати ручного написання SQL-коду та підтримувати автоматичні міграції структури сховища.

Модульність архітектури полягає у розподілі функціоналу між окремими логічними блоками: облік товарів, клієнтів, співробітників, продажів, знижок і звітності. Кожен із цих модулів реалізований як самостійний компонент із власним набором класів і методів. Це суттєво спрощує оновлення системи, адже зміни в одному модулі не впливають на роботу інших, що підвищує надійність та зменшує ризик виникнення помилок.

Особливе місце у архітектурі посідає рекомендаційна підсистема, інтегрована безпосередньо у логіку обробки транзакцій. Вона взаємодіє з історією покупок і оперативно формує персоналізовані пропозиції для клієнтів. Завдяки використанню алгоритмів машинного навчання, рекомендації оновлюються динамічно, враховуючи зміни у поведінці користувачів.

Архітектурна модель системи забезпечує зручність масштабування та супроводу, дозволяє гнучко адаптувати програмний продукт під різні бізнес-процеси та гарантує стабільну роботу у повсякденних умовах експлуатації.

3.2 Діаграма прецедентів та ролі користувачів

Діаграма прецедентів є одним із ключових інструментів для структуризації логіки взаємодії користувачів із програмною системою, особливо у системах із чітким розмежуванням прав доступу, як-от у випадку застосунків для обліку товарів із інтегрованою рекомендаційною підсистемою.

У розробленій системі виділено три основні ролі: адміністратор, менеджер і комірник. Адміністратор має найширший набір можливостей, серед яких виділено доступ до редагування інформації про співробітників, керування базою клієнтів, робота з модулем знижок, внесення інформації про товари і перегляд аналітичних звітів. Крім того, адміністратор здійснює активацію або деактивацію облікових записів співробітників, що гарантує безпеку й гнучкість в управлінні персоналом.

Менеджер – це користувач, який взаємодіє із клієнтською базою, може редагувати інформацію про товари, створювати і супроводжувати продажі, а також працювати зі знижками й переглядом звітів.

Роль комірника обмежена лише необхідними для роботи функціями – переглядом залишків товару, доступом до інформації про складські операції, а також базовим аналізом звітів без можливості вносити зміни до даних.

Сценарії використання визначаються рівнем доступу кожного користувача. Менеджер має змогу здійснювати продажі та додавати нові транзакції, що дозволяє оперативно обслуговувати клієнтів і реагувати на попит у магазині. Адміністратор може аналізувати й коригувати дані про клієнтів, контролювати роботу співробітників, запускати розширені звіти та впроваджувати акційні пропозиції. Комірник своєю чергою концентрується на підтримці порядку в обліку запасів, що знижує ризик помилок і забезпечує достовірність залишків у системі.

Діаграма прецедентів, представлена на рисунку 3.1, ілюструє всі ключові взаємозв’язки між користувачами й функціональними модулями програми: авторизація, робота з клієнтами та співробітниками, управління товарами, продажі, застосування знижок, формування й перегляд звітів.

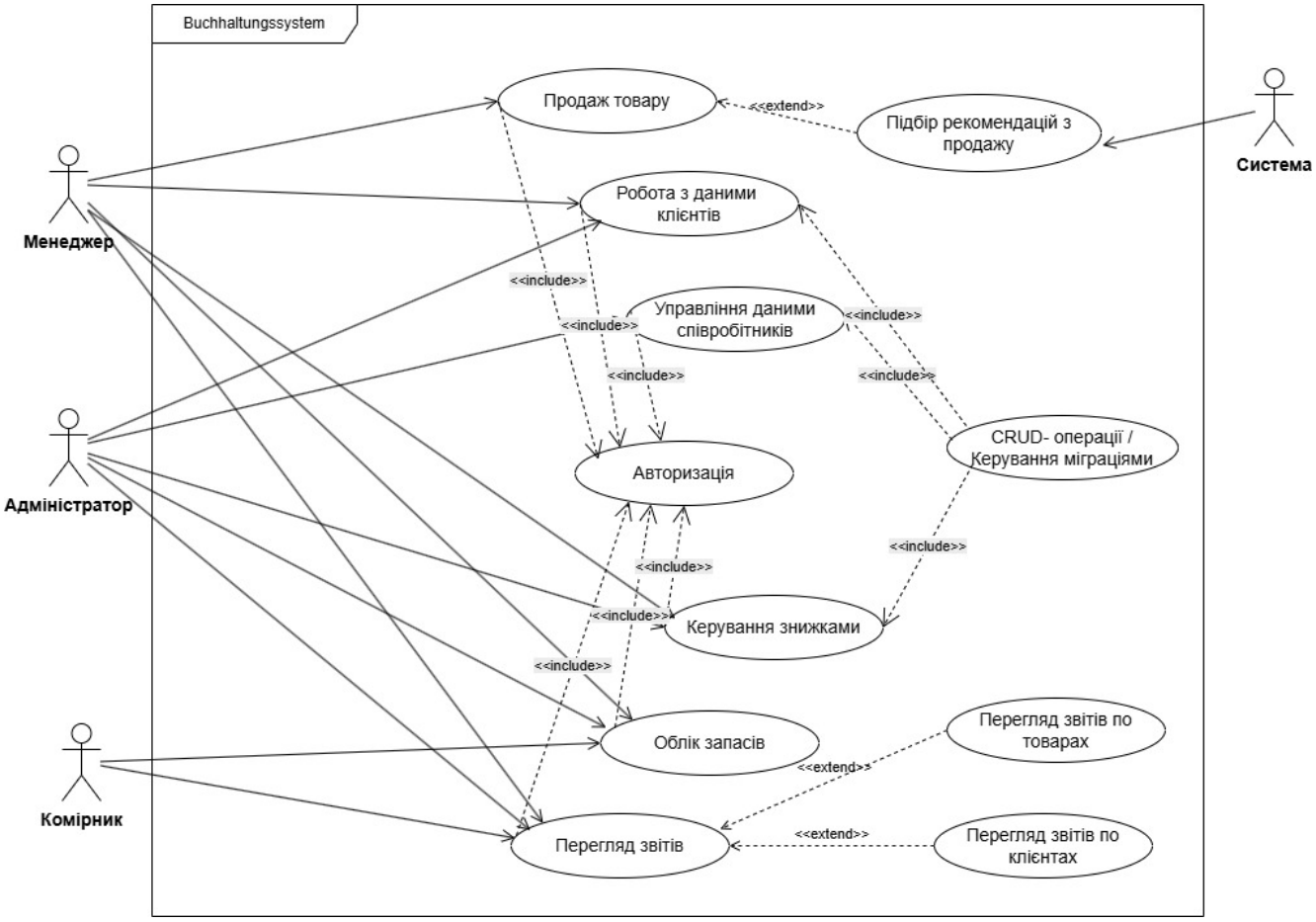


Рисунок 3.1 – Діаграма прецедентів створеної системи

Окремо варто відзначити сценарій “Підбір рекомендацій з продажу”, який відображає логіку роботи інтелектуальної підсистеми. Саме система, аналізуючи історію покупок та поточний склад кошика, автоматично пропонує релевантні додаткові товари, стимулюючи збільшення середнього чеку та підвищення лояльності клієнтів.

Для кожної ролі чітко окреслено, які саме дії дозволені, що забезпечує захист інформації та відповідає вимогам сучасних IT-систем щодо розмежування повноважень. У результаті, діаграма прецедентів є не лише візуальною частиною документації, а й основою для впровадження багаторівневої авторизації у програмному коді.

3.3. Модель даних та діаграма класів

Розробка моделі даних є фундаментальним етапом створення будь-якої інформаційної системи, оскільки саме від її структури залежить цілісність збережених відомостей, швидкодія запитів і зручність масштабування. В основі системи для обліку товарів покладено класичний реляційний підхід, реалізований через ORM-технологію Entity Framework із застосуванням методології Code First. Це дало змогу будувати модель безпосередньо у вигляді класів C#, які віддзеркалюють основні сутності та бізнес-процеси локального підприємства.

Структурна схема класів програмної системи, зображена на рисунку 3.2, ілюструє усю сукупність об’єктів, з якими працює система, а також зв’язки між ними. Основні сутності представлені класами Product, Client, Transaction, Employee, Promotion, які містять ключові властивості: ідентифікатори, поля для зберігання імен, контактної інформації, цін, категорій, статусів, а також додаткові допоміжні параметри.

Для наочності використано згенеровану схематику класів із середовища Visual Studio, що дозволяє швидко оцінити склад, зв’язки та ролі всіх основних елементів системи. Хоча така візуалізація не є класичною UML-діаграмою класів, вона повністю відображає структуру коду, реалізовану в програмі.

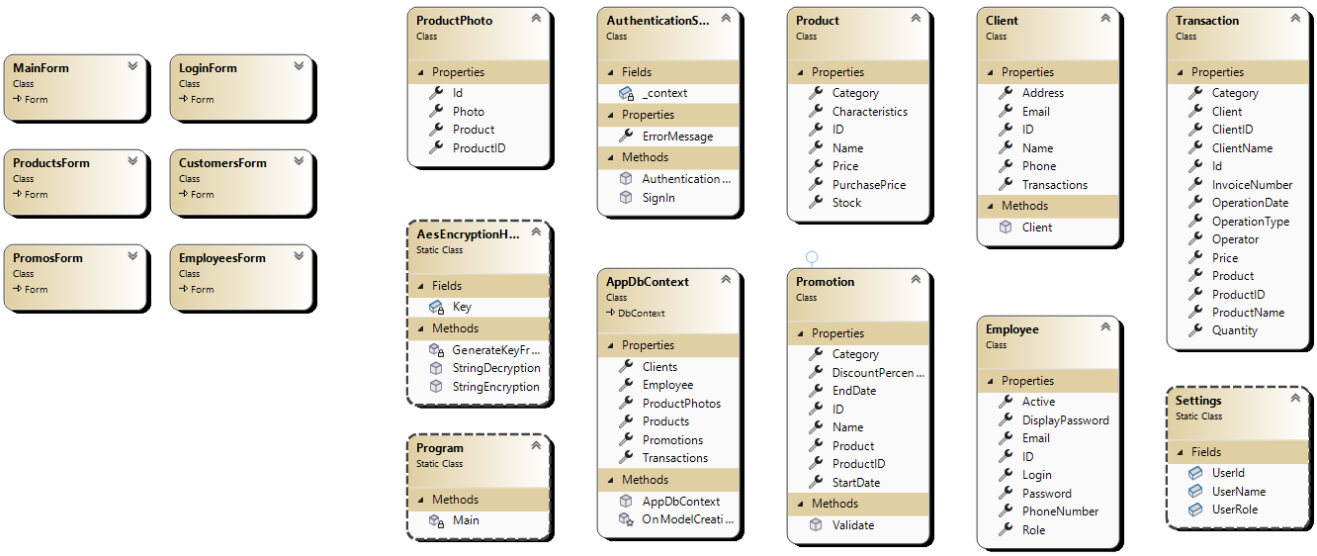


Рисунок 3.2 – Структурна схема класів програмної системи

Для організації зберігання фото товарів використовується окремий клас ProductPhoto, а централізовану роботу з базою забезпечує клас AppDbContext, що агрегує всі основні таблиці й виступає точкою входу для операцій над даними.

У структурі реалізовано всі необхідні типи зв’язків. Зокрема, між Client і Transaction діє відношення “один до багатьох”, що дозволяє для кожного клієнта вести історію покупок. Клас Product пов’язаний із транзакціями через деталізацію продажів, а також із сутністю Promotion для відображення динаміки знижок. Employee асоціюється зі своїми ролями та має відповідні дані для авторизації, які захищено зберігаються у вигляді хешів.

Кожна модель описана із дотриманням принципів інкапсуляції: поля даних захищені від прямого доступу, зміни здійснюються через відповідні методи, що гарантує консистентність і захищеність інформації.

Важливо, що така організація даних спрощує розширення та підтримку системи. Наприклад, у разі появи нових бізнес-вимог додавання додаткових полів чи зв’язків не потребує складної міграції вручну – досить оновити відповідний клас у коді, після чого система міграцій автоматично синхронізує ці зміни з реальною структурою бази.

Діаграма класів наочно показує, як реалізовано основні бізнес-логіки: облік руху товарів, формування історії транзакцій, керування знижками та взаємодія різних ролей із системою. Завдяки такому підходу забезпечується цілісність усієї архітектури, а система залишається гнучкою для подальшого розвитку.

3.4 Основні модулі системи та їх функціональність

Структура розробленого застосунку для автоматизованого обліку товарів побудована за модульним принципом. Такий підхід дозволяє чітко розмежувати зони відповідальності компонентів системи, полегшує її супровід та забезпечує простоту внесення змін. У межах дипломної роботи реалізовано шість основних

модулів, кожен з яких забезпечує певну функціональну частину загального бізнес-процесу.

Модуль “Редагування асортименту” є одним із ключових і дозволяє управляти асортиментом підприємства. В ньому реалізовані можливості додавання, редагування та видалення інформації про товари, їхню категорію, характеристики, ціну закупівлі й продажу, наявні залишки. Користувач може оперативно переглядати та фільтрувати інформацію за категорією або кількістю залишків, як показано на рисунку 3.3, а також здійснювати пошук за назвою або іншими параметрами. Після кожної операції інформація миттєво оновлюється в базі даних.

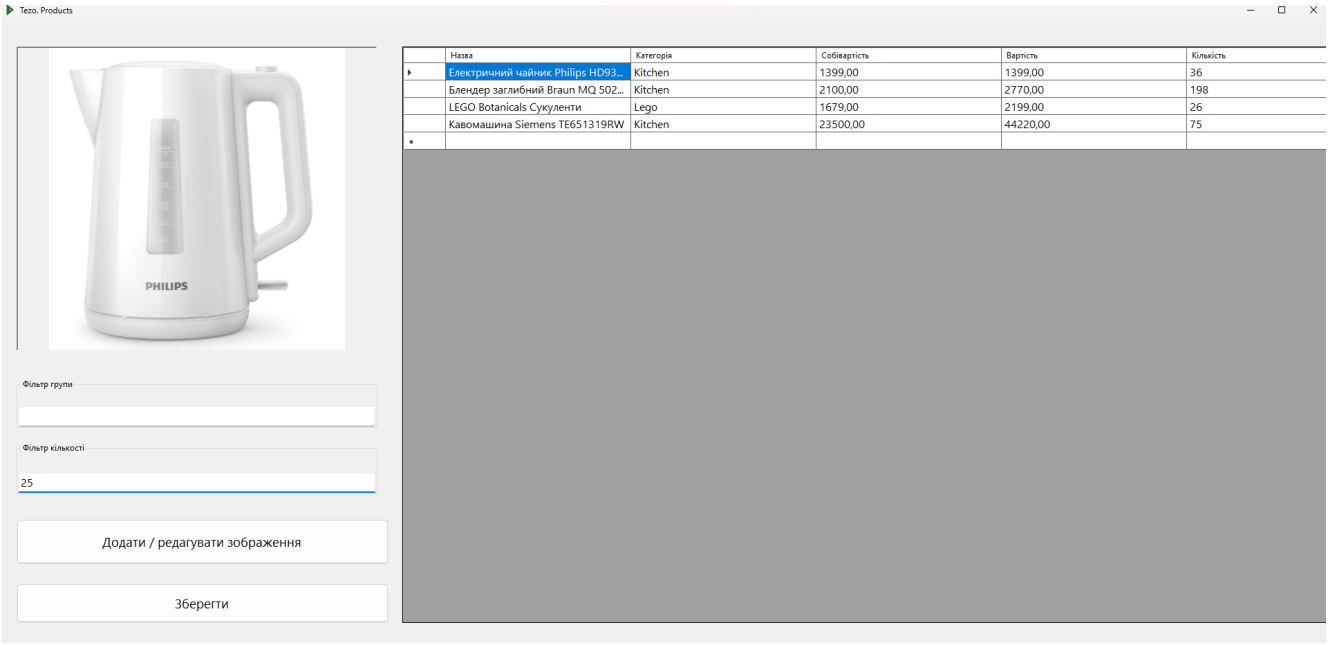


Рисунок 3.3 – Вигляд модулю “Редагування асортименту”

Модуль роботи з клієнтами містить функціонал для роботи з клієнтською базою підприємства. В ньому зберігаються персональні дані клієнтів, а також історія здійснених покупок. Система дозволяє легко знаходити клієнтів за їхніми даними, переглядати попередні замовлення й здійснювати швидке редагування даних. Приклад такого модулю зображено на рисунку 3.4.

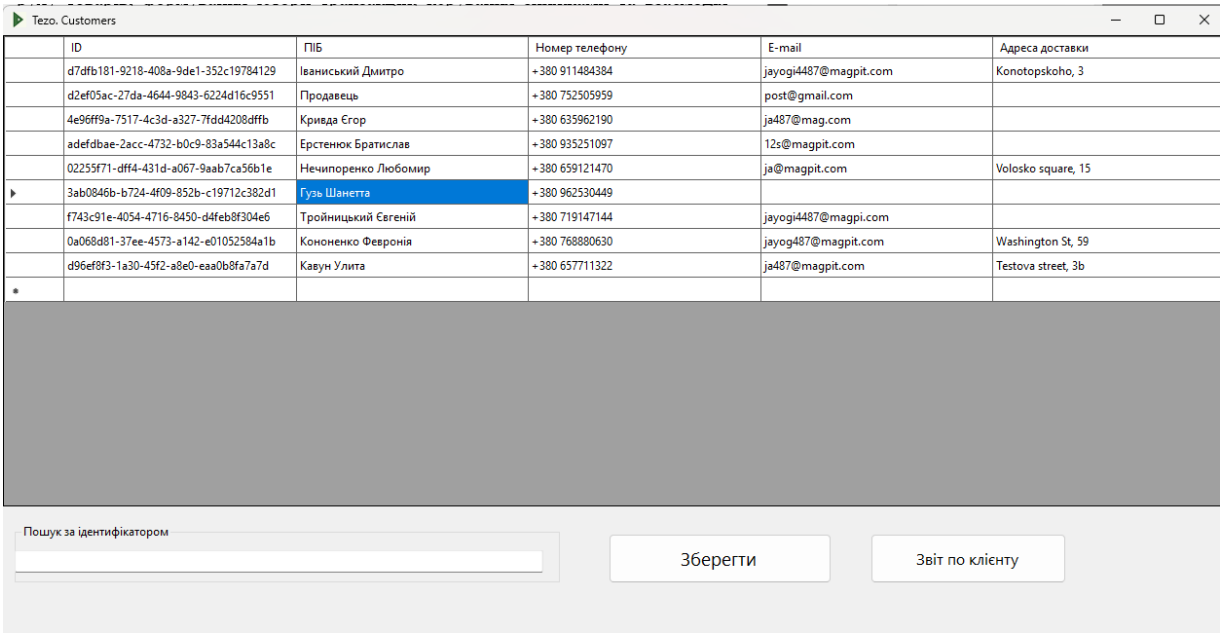


Рисунок 3.4 – Модуль взаємодії з даними клієнтів

До модулю роботи зі співробітниками має дозвіл лише адміністратор, і він дозволяє адміністратору переглядати, додавати, редагувати та деактивувати профілі співробітників. Функція активації та деактивації користувачів, зображена на рисунку 3.5, дозволяє керувати відстороненням звільнених користувачів, зберігаючи при тому їхні профілі в системі.

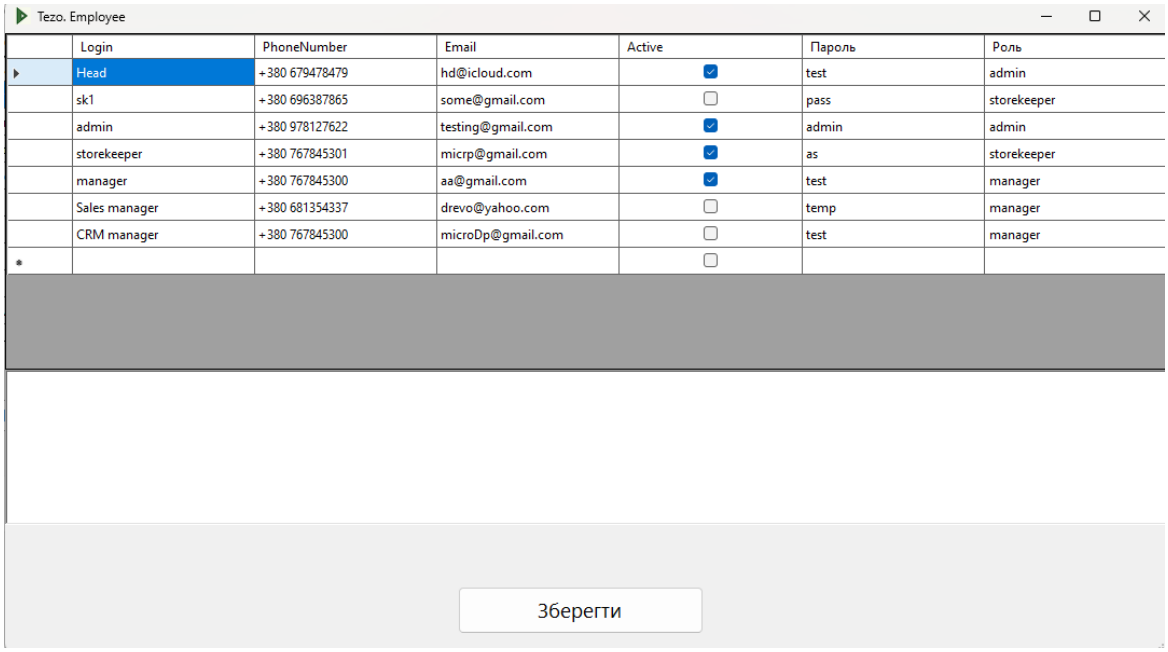


Рисунок 3.5 – Модуль редагування даних співробітників

Модуль “Знижки” дає змогу створювати акції з датами початку та завершення й відсотком знижки. Для існуючих товарів реалізований вибір через меню доступу, автозаповнення полів назви та категорії, а також валідація термінів дії знижки. Зображення модулю знижок подано на рисунку 3.6. Система автоматично перераховує ціну в накладній, якщо в момент продажу діє відповідна акція.

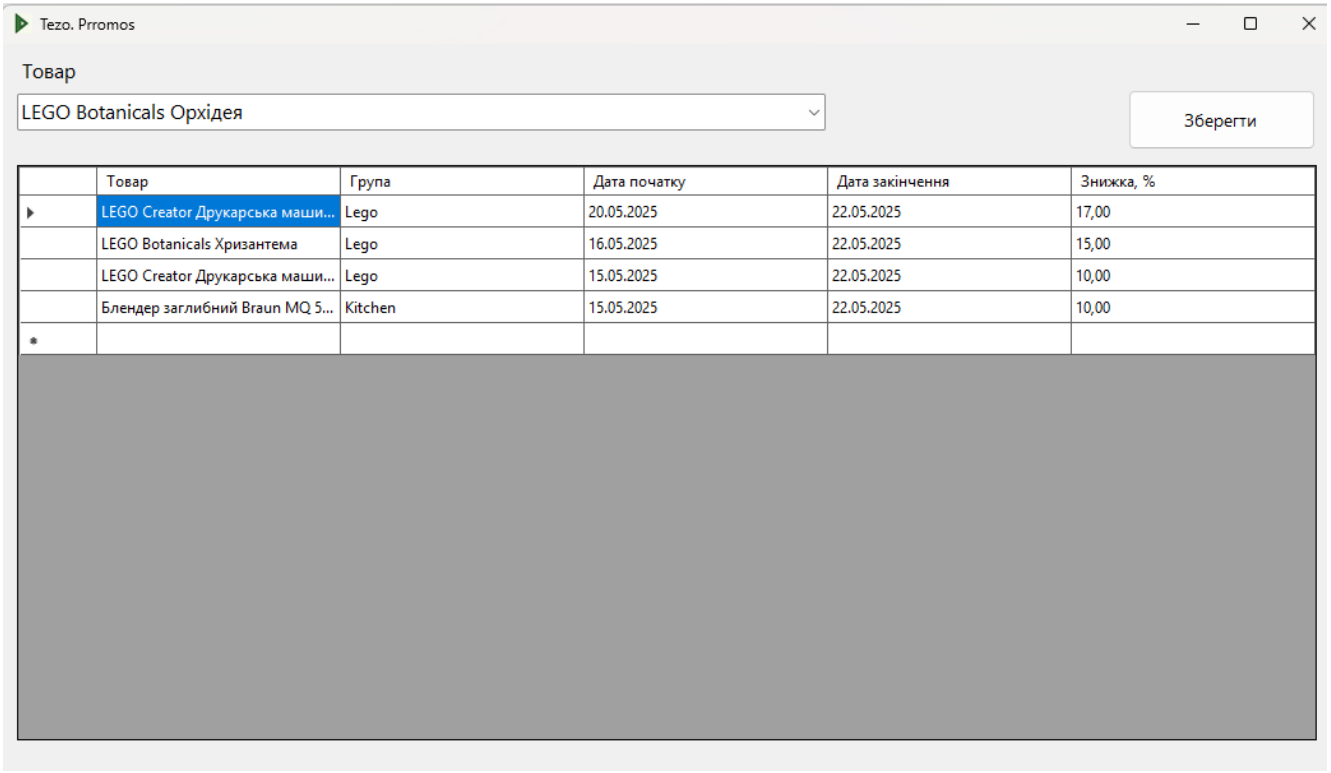


Рисунок 3.6 – Модуль керування знижками на товар

Ключовим модулем програмного застосунку є вкладка продажів, в якому користувач системи отримує змогу здійснювати продаж товару. Доступ до продажів товару має лише менеджер підприємства. При роботі з інтерфейсом, подвійний клік додає товар до поточної транзакції, де можна коригувати кількість товару та наповнення чеку. Крім того, саме в цьому вікні наявний графічний елемент, що відображає рекомендації додаткових товарів до покупки.

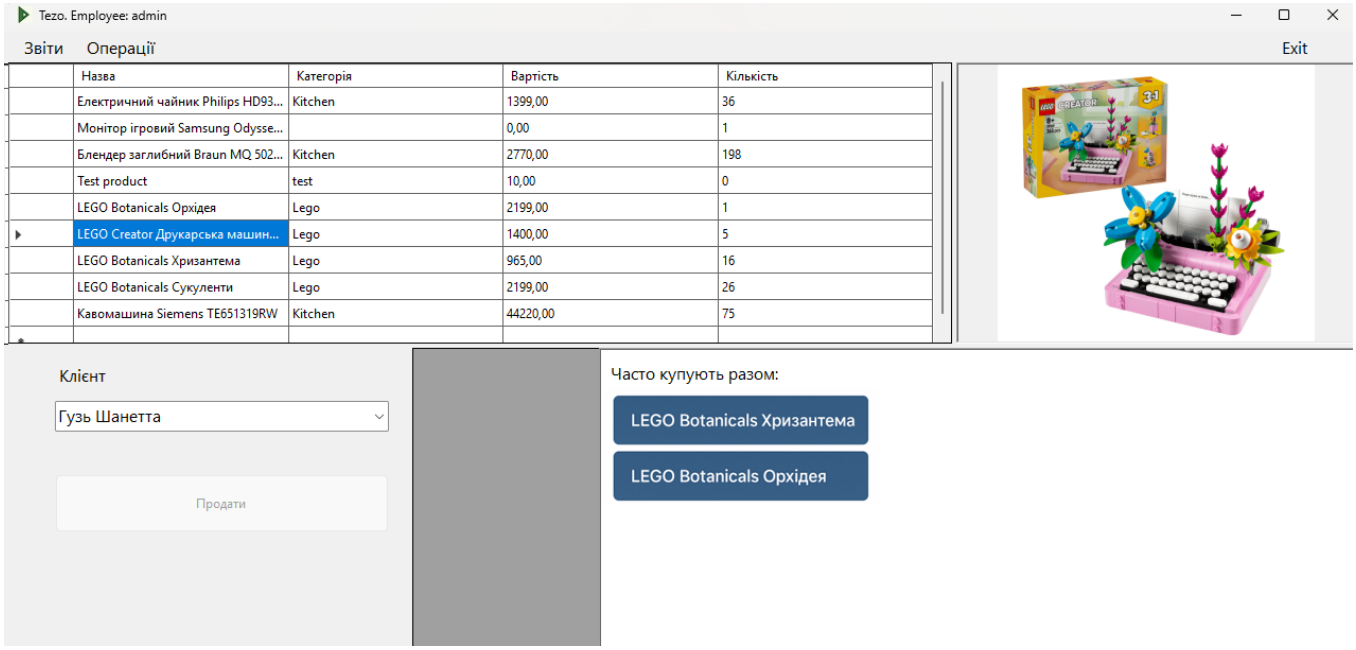


Рисунок 3.7 – Модуль продажів товару з рекомендаційною системою

Підтвердження продажу формує номер накладної, змінює залишок у БД та записує операцію з усіма деталями: дату, клієнта, ціну й користувача.

Модуль звітності реалізовано за рахунок накладних квитанцій, що генерує HTML-документи для різного роду трнзакцій.

Рисунок 3.8 – Сформований звіт продажів товару “Блендер заглибний Braun”

Сформовані документи містять таблиці з деталями операцій, підсумковими сумами, датами та підписом, що дозволяє швидко експортувати результати для друку або подальшого аналізу.

Усі модулі взаємодіють через єдиний контекст даних, підтримують автоматичне оновлення інтерфейсу після змін та відповідають принципам

мінімізації доступу – кожен користувач працює лише з тими функціями, які відповідають його ролі.

3.5 Рух даних і взаємодія компонентів

Організація внутрішнього руху даних та взаємодії між компонентами є критично важливою для стабільності, передбачуваності й ефективності функціонування програмної системи. У розробленому застосунку всі основні процеси побудовано на основі чіткої багаторівневої логіки, завдяки чому було розмежовано відповідальність між частинами програми, зменшено ризики порушення цілісності даних та полегшено обслуговування системи.

Користувач взаємодіє із застосунком через графічний інтерфейс, побудований на основі Windows Forms. При кожній дії – введенні нового товару, редагуванні клієнта або створенні продажу – інформація надсилається до модулів бізнес-логіки, які здійснюють валідацію, перевірку наявності пов’язаних записів, відповідність структурі та правам доступу. У разі успішного проходження логічного контролю дані передаються до контексту бази даних AppDbContext, через який, за допомогою механізмів ORM, відбувається запис або зчитування інформації з фізичного сховища БД Microsoft SQL Server.

Централізована взаємодія через AppDbContext дозволяє гарантувати єдину точку входу до даних, що запобігає дублюванню логіки доступу та забезпечує однакову поведінку в усіх модулях. Наприклад, після створення транзакції система одразу виконує зміну залишку товару, фіксує деталі покупки та зберігає зв’язок між клієнтом, оператором і купленими позиціями. Операція здійснюється у межах єдиної транзакції, що дозволяє забезпечити атомарність і консистентність.

Ефективне функціонування системи обліку товарів значною мірою залежить від логічної, стабільної та масштабованої структури бази даних. У межах розробленого застосунку використано реляційну модель, що дало змогу повністю керувати структурою БД з боку коду, уникати конфліктів на рівні схеми й забезпечити синхронізацію між логікою програми та таблицями сховища.

База даних в розробленому програмному продукті реалізує класичну логіку для бізнес-систем: довідники, журнали подій (транзакцій), користувацькі облікові записи, механізми безпеки, а також інструменти для обліку знижок і історії клієнтів. Вся інформація групується за основними категоріями:

- Довідники (Products, Clients, Employees);
- Операції (Transactions, TransactionDetails);
- Адміністративна частина (Promotions, Roles, Users).

У системі реалізовано низку таблиць, які відображають бізнес-сутності та логіку роботи програми. Кожна таблиця має первинний ключ та пов’язана з іншими через зовнішні ключі. Структура бази даних зображена на рисунку 3.9.

Особливе місце у моделі руху даних займає рекомендаційна підсистема. При оформленні покупки бізнес-логіка звертається до методу, який аналізує історію транзакцій за допомогою LINQ-запитів та моделі ML.NET. На основі результатів запиту формується перелік рекомендованих товарів, який повертається на рівень інтерфейсу й відображається користувачу в момент продажу.

Також важливу роль у взаємодії компонентів відіграє система авторизації. На стартовому етапі запуску застосунку користувач проходить автентифікацію, яка визначає його роль. Відповідно до ролі формується дозволений набір компонентів інтерфейсу. Таким чином, доступ до логіки продажів, знижок або звітів контролюється не лише на рівні UI, а й глибоко інтегровано в логіку обробки запитів.

До прикладу, коли менеджер керує продажем товару, інтерфейс формує список обраних товарів, передає їх до класу Transaction, який у свою чергу генерує відповідні записи до таблиць транзакцій, змінює дані у таблиці залишку товарів та формує зв’язок із конкретним клієнтом. Усе це відбувається в рамках однієї взаємодії між модулями, з подальшим миттєвим оновленням відображення даних у інтерфейсі.

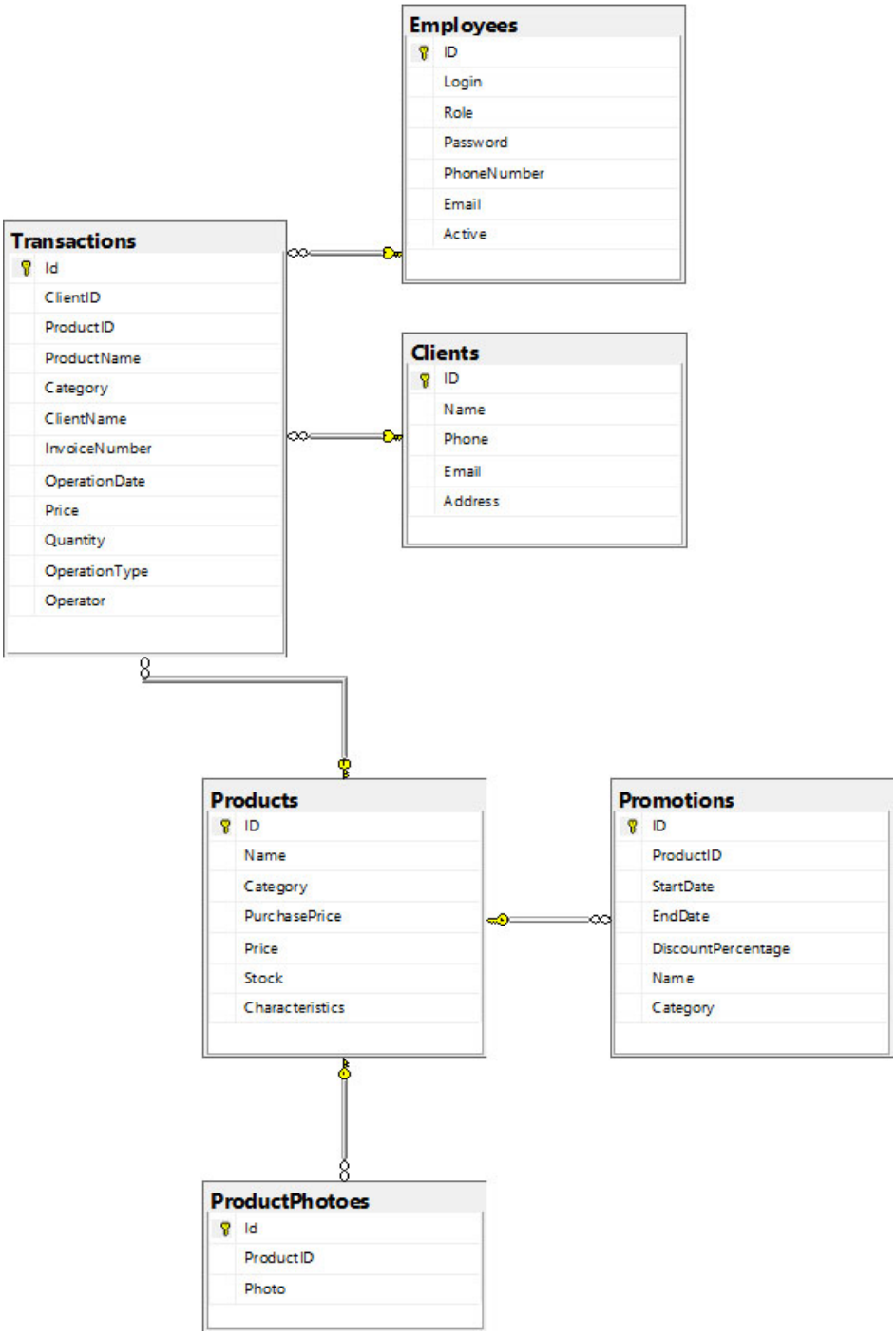


Рисунок 3.9 – ER-діаграма бази даних застосунку

Уся взаємодія між компонентами системи, а саме формами, класами-моделями, логікою авторизації, механізмом рекомендацій, а також зв’язком з базою даних реалізована так, аби гарантувати цілісність, швидкодію і передбачувану реакцію програми на дії користувача. Така структура дозволяє легко розширювати функціональність, зберігаючи чіткий порядок обробки даних на кожному етапі.

3.6 Інтерфейс користувача та структура навігації

Інтерфейс користувача є важливим елементом будь-якого програмного забезпечення, особливо якщо воно орієнтоване на широке коло співробітників, які можуть не мати глибокої технічної підготовки. Від простоти, логіки та зручності навігації безпосередньо залежить ефективність роботи з системою, швидкість навчання персоналу та мінімізація помилок під час виконання типових операцій.

У межах застосунку інтерфейс має чіткий та лаконічний вигляд, і забезпечує стабільну роботу на будь-якому комп’ютері з операційною системою Windows та не потребує встановлення додаткових компонентів. Кожна форма створена з урахуванням норм зручності користування – усі елементи розташовано з достатнім візуальним розмежуванням функціональних блоків і можливістю швидкого доступу до основних дій.

Після проходження авторизації, вікно якої зображено на рисунку 3.10, користувач потрапляє на головну панель навігації, яка відображає доступні модулі залежно від ролі.

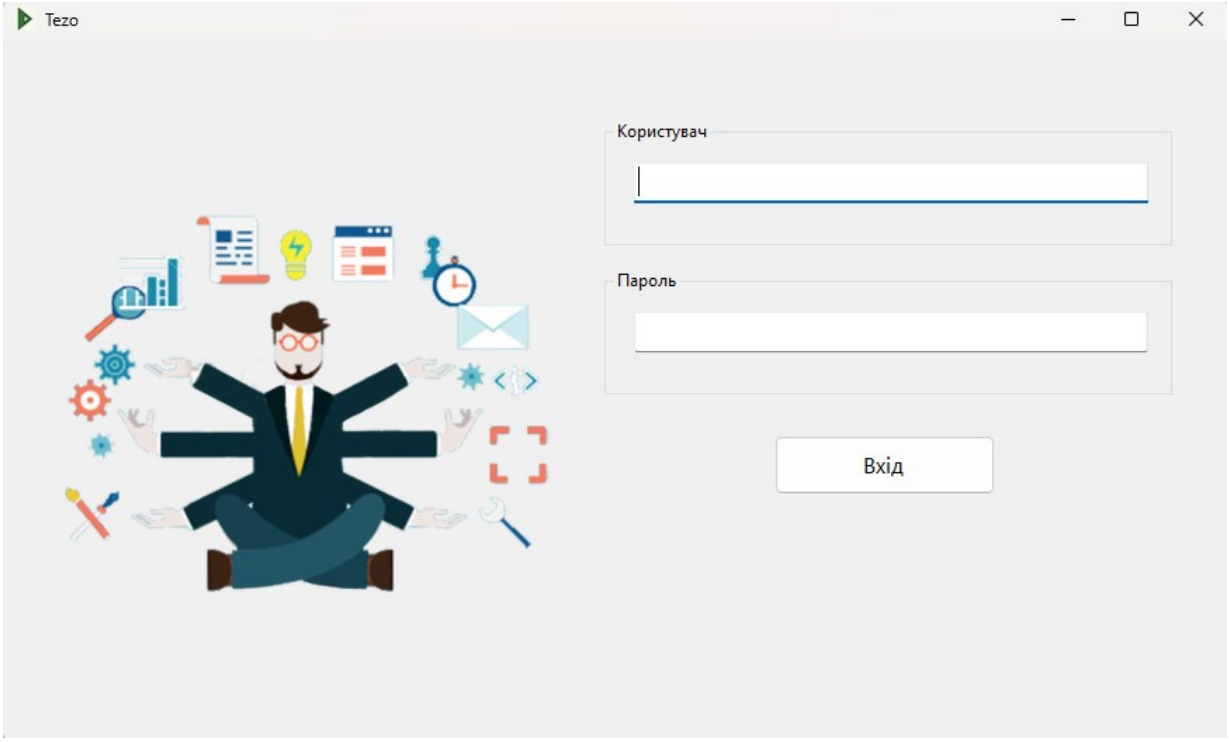


Рисунок 3.10 – Вікно авторизації в застосунок

Наприклад, менеджер бачить вкладки “Товари”, “Клієнти”, “Знижки” та “Звіти”, а адміністратор додатково отримує доступ до управління даними співробітників. Комірнику ж доступний лише перегляд товарів та форм звітності.

У структурі навігації застосовано модель з використанням вкладок, де кожен основний модуль відкривається у вигляді окремої форми. Така структура дозволяє легко перемикатися між модулями, зберігаючи контекст поточних дій, що є особливо корисним при паралельній роботі з різними блоками. До прикладу, формуванням транзакцій та переглядом історії клієнта.

Кожна форма містить різний набір елементів, та загалом, має наявні таблиці для перегляду даних, кнопки для базових CRUD-операцій, панелі фільтрів і поля для пошуку. Після будь-якої дії відбувається автоматичне оновлення даних, що миттєво відображається на екрані. Усі повідомлення про успішне виконання операцій, помилки або порушення доступу реалізовано у вигляді діалогових вікон, що дозволяє зберігати прозорість взаємодії.

Варто також відзначити реалізовану в інтерфейсі логіку рекомендацій. Після додавання товару до списку продажів система автоматично відображає праворуч блок рекомендованих позицій – ці дані підтягуються з відповідної підсистеми й змінюються динамічно залежно від вибору користувача.

Загалом структура інтерфейсу створена таким чином, щоб звести до мінімуму кількість кліків для досягнення цільової дії. При цьому програма підтримує навігацію клавіатурними комбінаціями та підказки для нових користувачів. Такий підхід забезпечує інтуїтивність, зменшує вірогідність помилок та покращує загальний користувацький досвід.

4 ВПРОВАДЖЕННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для забезпечення високої якості кінцевого продукту необхідно не тільки розробити програму, а й провести її коректне розгортання у робочому середовищі та переконатися в коректності роботи всіх функцій у реальних умовах. Впровадження системи починається з підготовки інсталяційного пакета, налаштування локальної бази даних та створення необхідних конфігурацій, що має супроводжуватись відповідною документацією.

Тестування реалізованого функціоналу застосунку засвідчує, що користувачі з різними рівнями доступу можуть виконувати лише свої дії, що всі бізнес-правила працюють згідно з технічним завданням, а критичні операції, серед яких формування транзакції, зберігання даних чи шифрування паролів, відбуваються без помилок.

Завдяки цьому відбувається демонстрація, як застосунок успішно переходить від розробки до реальної експлуатації, підтверджуючи свою готовність до безперебійного використання бізнес-користувачами.

4.1 Технічні вимоги до обчислювальної техніки

Для стабільної та безперебійної інсталяції застосунку достатньо дотримуватися помірних апаратних і програмних вимог. Програма поширюється у вигляді єдиного інсталяційного файлу (.exe), що містить усі необхідні бібліотеки й компоненти, та не потребує доступу до Інтернету, оскільки працює з локальною базою даних SQL Server Express. Під час встановлення інсталятор автоматично перевіряє й, за необхідності, завантажує відсутні .NET Framework та SQL Server LocalDB, що спрощує процес розгортання навіть у закритих корпоративних мережах або офлайн-магазинах. Завдяки використанню SQL Server Express

LocalDB, систему можна розгорнути без окремого серверного оточення або підписки на хмарні сервіси, і вона повністю функціонує автономно.

Оскільки застосунок побудовано на технологіях Windows Forms і .NET Framework, до складу інсталятора входить також Visual C++ Redistributable, тому додаткових дій з боку користувача виконувати не потрібно.

Крім того, за рахунок використання стійких та надійних для системи технологій, застосунок може бути встановлений навіть на комп'ютерах з базовими характеристиками, що робить його доступним для широкого кола користувачів. Відтак, серед умов до використання застосунку, можна виокремити:

– процесор: Intel Core i3 або AMD Ryzen 3 і вище;
– оперативна пам'ять: від 4 ГБ (рекомендовано 8 ГБ);
– вільне місце на диску: не менше 600 МБ для інсталяції програми;
– роздільна здатність екрана: мінімум 1366×768;
– операційна система: Windows 10 або новіша (64-bit).

Серед програмних вимог, для безпосереднього запуску застосунку, необхідно:

– фреймворк .NET Framework версії 4.7.2 або новішої;
– База даних Microsoft SQL Server Express LocalDB;
– інтегроване середовище розробки Microsoft Visual C++ Redistributable (автоматично додається при інсталяції);
– права адміністратора (під час першого встановлення).

4.2 Порядок інсталяції та налаштування програми

Правильність інсталяції й конфігурації застосунку має суттєвий вплив на подальшу стабільність і безпомилковість роботи системи. Для розгортання системи важливим етапом є встановлення необхідних компонентів середовища, а також правильність формування структури БД і її відповідність моделі даних програми. Акцент зроблено на механізмі ініціалізації облікового запису адміністратора, оскільки саме від нього починається керування всіма користувачами та їхніми

правами. Заходи з опис інсталяційних файлів спрямовані на мінімізацію ручної роботи на етапі розгортання та задля збереження програми від типових помилок, пов’язаних із недоінсталюванням залежностей або невірними правами доступу до бази даних.

Процес інсталяції та початкового налаштування програмного продукту починається з підготовки середовища, у якому він працюватиме без відмов і збоїв. Спершу необхідно запустити інсталяційний файл “Tezo.exe”, який містить повністю зібраний додаток разом із усіма залежностями. Після подвійного клацання на виконуваному файлі з’явиться діалогове вікно контролю облікових записів Windows, в якому слід підтвердити надання прав адміністратора і довіру до застосунку, як показано на рисунку 4.1. Інсталятор самостійно перевірить наявність на комп’ютері .NET Framework версії 4.7.2 або новішої. Якщо потрібний фреймворк відсутній, він буде завантажений і інстальований автоматично, без участі користувача. Аналогічним чином здійснюється перевірка присутності SQL Server Express LocalDB: у разі його відсутності інсталятор підхопить і встановить цю компоненту, що дозволить програмі створювати й зберігати базу даних локально.

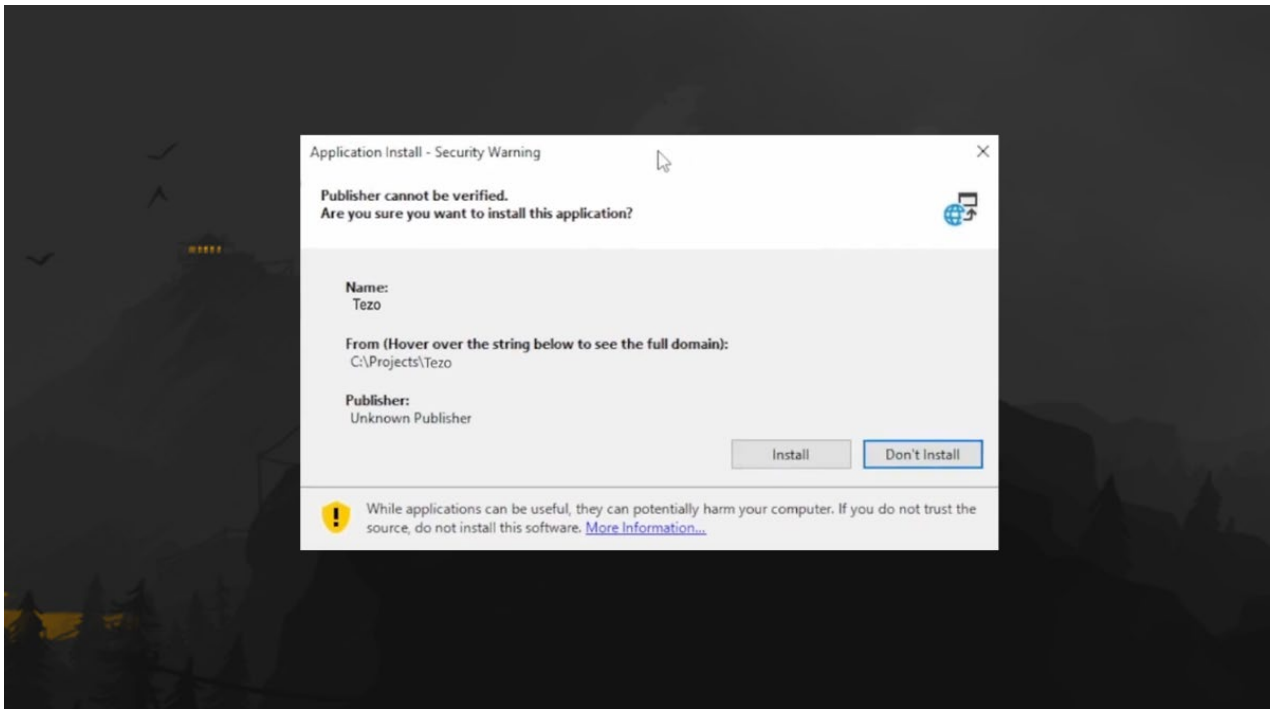


Рисунок 4.1 – Приклад встановлення застосунку “Tezo”

Після успішної перевірки середовища відкриється стандартне вікно вибору теки для встановлення основних файлів програми. Зазвичай використовується запропонований шлях, проте за потреби, користувач може вказати інший каталог. Як тільки місце розташування обрано, інсталятор переходить до копіювання файлів і налаштування необхідних служби. В процесі встановлення обов'язково формуються ярлики на Робочому столі та в меню "Пуск" для швидкого доступу до програми після інсталяції.

Щойно інсталяція основних файлів завершена, відбувається ініціалізація локальної бази даних. Перший запуск програми викликає перевірку наявності файлу бази (Tezo.mdf) у користувацькій теці AppData. Якщо цієї бази ще не існує, запускається механізм міграцій Entity Framework. Він створює структуру таблиць відповідно до моделі даних, прописаної в коді. Одразу після цього формується початковий запис для адміністратора, що є критично важливим, оскільки адміністративний обліковий запис керуватиме всіма подальшими налаштуваннями прав доступу та ролей.

В разі успішного створення й перезапуску програми користувач бачить екран авторизації. Після успішного входу відкривається головний екран програми, що дозволяє переходити до розділів з управління товарами, клієнтами, співробітниками, продажами, знижками та звітами. До цього моменту налаштування бази вважається завершеним, і система готова до подальшого конфігурування.

Щоб переконатися в коректності інсталяції та правильності налаштувань, необхідно перевірити роботу базових функціональних модулів. Для цього під обліковим записом адміністратора слід послідовно відкрити розділ "Товари" та додати тестову позицію: після введення назви, категорії, ціни та кількості товар з'явиться в таблиці асортименту. Відтак, схожий алгоритм необхідно здійснити і в інших модулях застосунку. Якщо всі ці дії виконуються без помилок і зміни відображаються миттєво, базові операції працюють коректно.

Щойно інсталяцію та відладку програми завершено, отримує працездатний продукт, що не потребує додаткових ручних втручань. Коректна ініціалізація бази даних, створення адміністративного облікового запису та налаштування

конфігураційних параметрів гарантують стійку роботу всіх бізнес-правил і виключають критичні збої під час експлуатації.

4.3 Демонстрація функціоналу системи

Наведені сценарії взаємодії з застосунком дають змогу переконатися в тому, що всі ключові модулі системи працюють відповідно до вимог технічного завдання і реалізовані згідно з поставленим завданням. Завдяки розгляду сценаріїв чітко видно, що застосунок відпрацьовує повний життєвий цикл облікової операції – від створення довідкових записів до генерації фінальних звітів. Кожна дія користувача на екрані супроводжується відповідним оновленням у базі даних, а результат миттєво відображається в інтерфейсі.

Перший сценарій відображає додавання нового товару до системи. Користувач із роллю “Адміністратор” переходить у вкладку “редагування асортименту”, і у формі, що відкривається, заповнює поля назви товару, його категорію і вартість. Також, за потреби, адміністратор може додати зображення товару, як показано на рисунку 4.2.

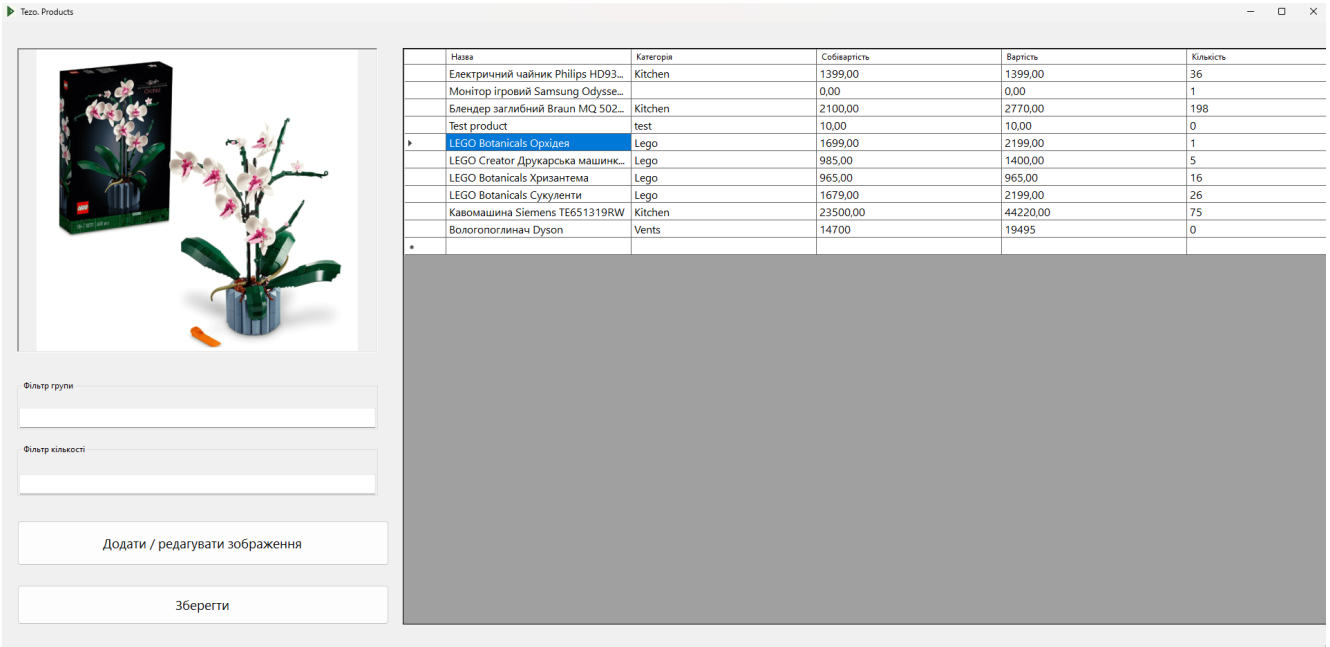


Рисунок 4.2 – Додавання товару “LEGO Botanicals Орхідея” до асортименту

Після натискання кнопки “Зберегти” відбувається оновлення бази даних: фреймворк Entity Framework виконує міграцію, створюючи або оновлюючи запис у таблиці Products. Новий рядок з’являється у табличному списку асортименту, зображеному на рисунку 4.3, де відображаються всі параметри товару. Адміністратор бачить повідомлення про успішне збереження, а програма автоматично повертає користувача до списку, де доданий товар підсвічується у загальному реєстрі.

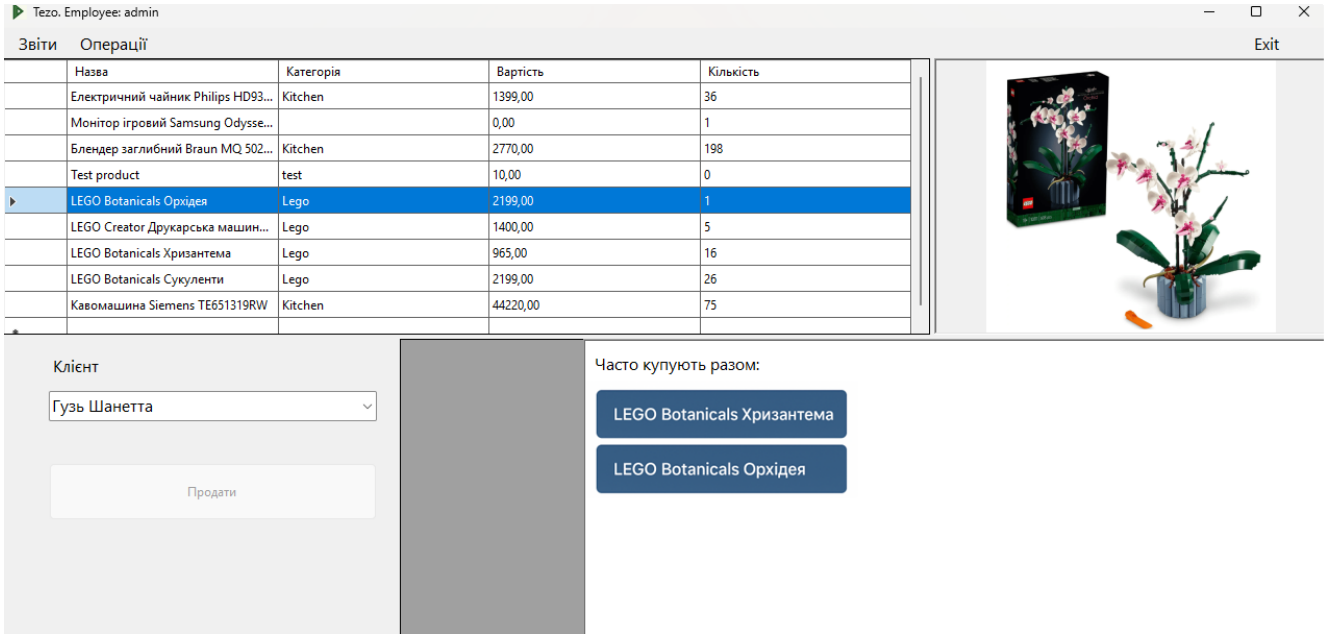


Рисунок 4.3 – Оновлення асортименту головного вікна продажів новим товаром

Сценарій реєстрації клієнта та його подальше залучення до продажу має наступний алгоритм: менеджер або адміністратор обирає опцію “Редгування даних клієнта”. У вікні введення даних вказує необхідні дані, такі як ПІБ, телефон і електронну пошту, а потім підтверджує дію кнопкою “Зберегти”. Після цього в базі даних створюється новий запис у таблиці Clients, а інформація одразу актуалізується в застосунку, про що свідчить рисунок 4.4. Відтак, коли менеджер переходить до вкладки продажів, в полі “Клієнти” з’являється новий запис з БД, що надає змогу взаємодіяти із клієнтом підприємства.

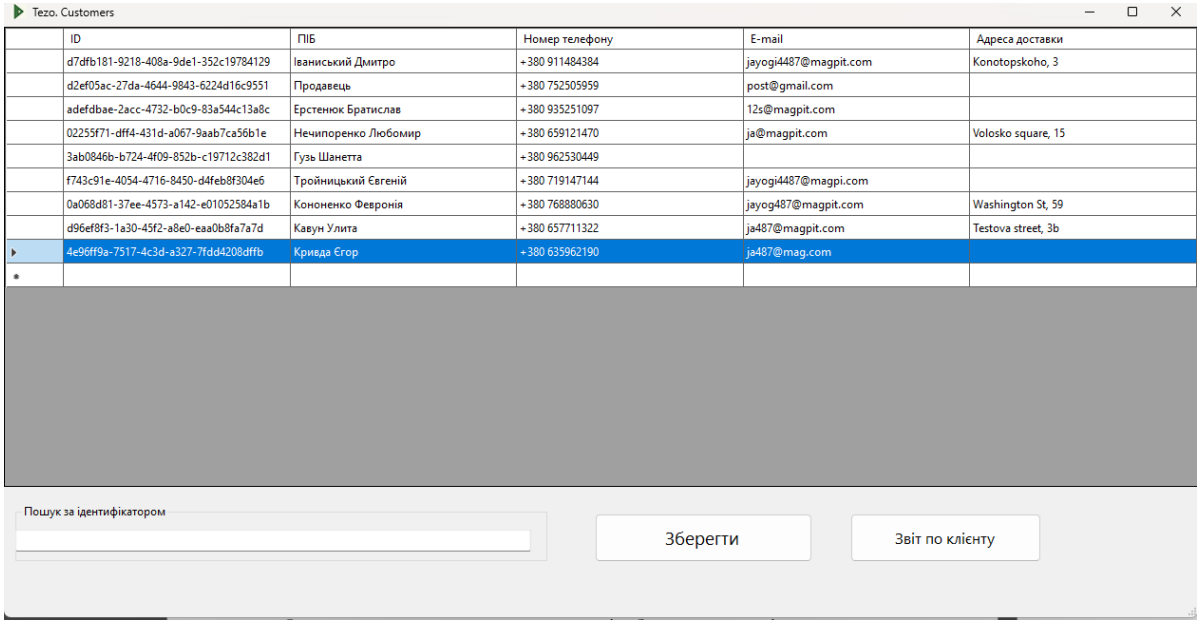


Рисунок 4.4 – Додавання інформації про нового клієнта в систему

Задля генерації звіту з історії покупок клієнта менеджер вибирає вкладку “Звіти”, де представлені різні типи аналітики.

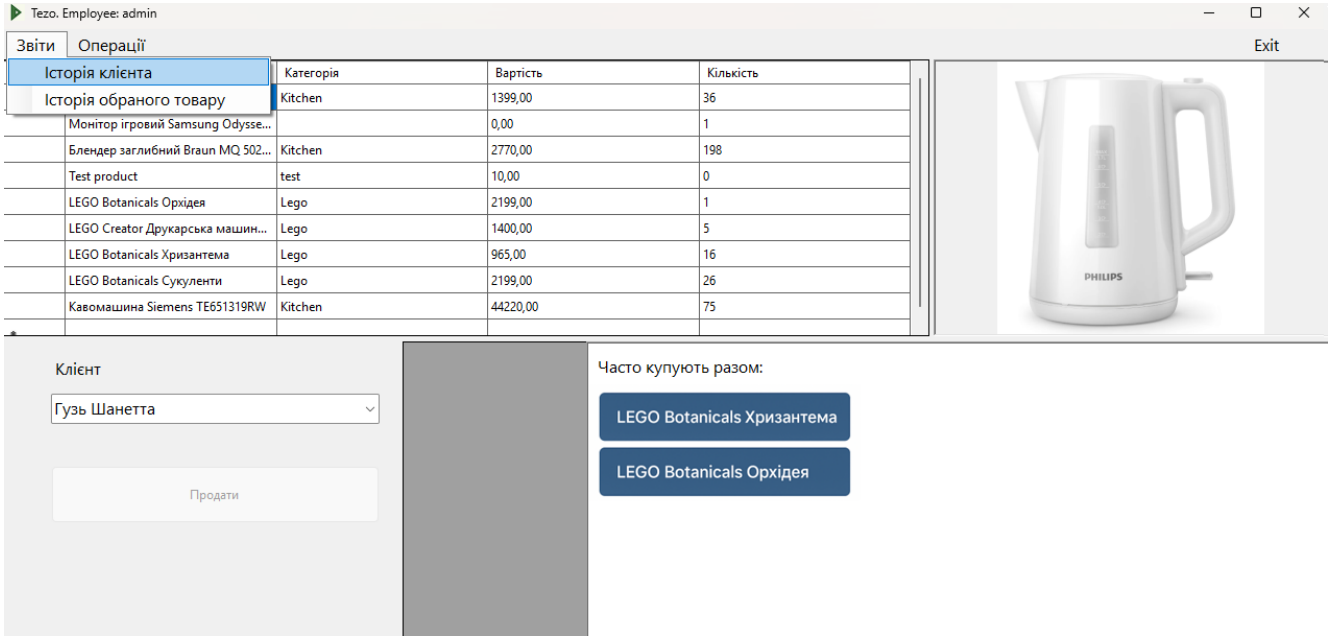
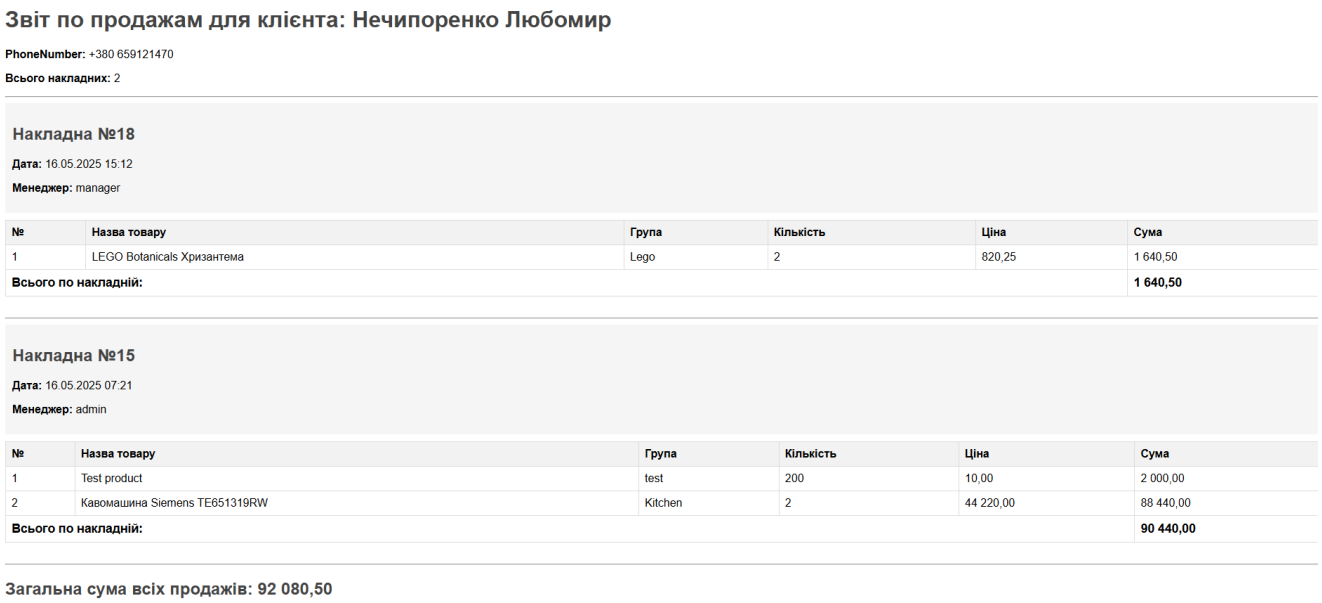


Рисунок 4.5 – Опція перегляду звітності транзакцій по клієнту

В разі якщо менеджер обирає опцію “Історія клієнта”, програма за допомогою LINQ-запиту до БД отримує необхідні дані, формує HTML-файл зі

структурованою таблицею та відкриває його у браузері за замовчанням. Вигляд такого звіту показано на рисунку 4.6.



Звіт по продажам для клієнта: Нечипоренко Любомир

PhoneNumber: +380 659121470

Всього накладних: 2

Накладна №18

Дата: 16.05.2025 15:12

Менеджер: manager

№	Назва товару	Група	Кількість	Ціна	Сума
1	LEGO Botanicals Хризантема	Lego	2	820,25	1 640,50
Всього по накладній:					**1 640,50**

Накладна №15

Дата: 16.05.2025 07:21

Менеджер: admin

№	Назва товару	Група	Кількість	Ціна	Сума
1	Test product	test	200	10,00	2 000,00
2	Кавомашина Siemens TE651319RW	Kitchen	2	44 220,00	88 440,00
Всього по накладній:					**90 440,00**

Загальна сума всіх продажів: 92 080,50

Рисунок 4.6 – Згенерований HTML-звіт транзакцій клієнта

У цьому звіті відображаються назви товарів, кількість проданих одиниць за обраний період, загальна сума, а також ім’я менеджера, що провів операцію. Якщо адміністратор обирає “Історія покупок клієнта”, він вводить ІД клієнта або обирає його зі списку, вказує діапазон дат і натискає “Згенерувати”. Після цього система формує окремий HTML-звіт із переліком усіх транзакцій цього клієнта, розділяючи їх за датою та надаючи деталі (скрін 4.3.6). Користувач може експортувати звіт у PDF або відразу роздрукувати.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено застосунок для автоматизованого обліку товарів із персоналізованими рекомендаціями, який відповідає сучасним потребам локального бізнесу і задовольняє поставлену мету. Всі етапи реалізації були підпорядковані завданню створити не лише стабільний інструмент для автоматизації товарного обліку, а й сучасний застосунок, що забезпечує підвищення ефективності продажів за рахунок персоналізованих рекомендацій.

У результаті аналізу існуючих програмних засобів і порівняння підходів до побудови систем обліку товарів у сфері торгівлі було обґрунтовано вибір стеку технологій, що включає мову програмування C#, фреймворки .NET Framework, Entity Framework та ML.NET. Саме поєднання цих технологій із багаторівневою архітектурою програмного продукту й алгоритмом k-ближчих сусідів для реалізації рекомендаційної підсистеми дало змогу створити застосунок, що відповідає сучасним вимогам надійності, масштабованості та простоти підтримки.

У межах роботи розроблено повнофункціональний програмний застосунок, який охоплює основні аспекти автоматизованого обліку: управління товарами, даними про клієнтів, персонал, продажі, знижки та формуванням звітів. Ключовою функціональною перевагою системи є інтегрована рекомендаційна підсистема, побудована на алгоритмі k-ближчих сусідів із використанням ML.NET. Це дозволяє формувати персоналізовані пропозиції супутніх товарів на основі історії покупок, що сприяє підвищенню середнього чеку й загальної ефективності бізнесу.

Ретельне тестування програмного забезпечення у середовищі Windows підтвердило його коректну роботу, стабільність функціонування всіх модулів і відповідність функціональним вимогам.

У перспективі застосунок може бути розширено шляхом впровадження веб-інтерфейсу для роботи через браузер, а також інтеграції зовнішніх сервісів електронної комерції та CRM-систем для збільшення важливості системи. З боку системи звітності, застосунок може бути доповнений модулем бізнес-аналітики з інтерактивними дашбордами і нативною звітністю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002, 560 с. (дата звернення: 06.05.2025).

2. Литвин О. М. Інформаційні системи і технології в обліку: підручник. Київ: Центр учбової літератури, 2016. с. (дата звернення: 06.05.2025).

3. Aggarwal C. C. Recommender Systems: The Textbook. Springer, 2016, 497 с. (дата звернення: 07.05.2025).

4. ASP. NET Core in Action, Third Edition. Manning Publications Co. LLC, 2023. 1003 с. (дата звернення: 17.05.2025).

5. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. 2nd ed. Springer, 2015. 1007 с. (дата звернення: 17.05.2025).

6. Content-Based Access to Multimedia Information. Springer, 2022, 338 с. (дата звернення: 18.05.2025).

7. What Does POS Mean? URL: https://www.touchbistro.com/blog/important-pos-terms/ (дата звернення: 19.05.2025).

8. Poster POS Features & Integrations. URL: https://joinposter.com/en/features (дата звернення: 19.05.2025).

9. Zoho Inventory – офіційний сайт. URL: https://www.zoho.com/inventory/ (дата звернення: 21.05.2025).

10.Albahari J., Albahari B. C# 10 in a Nutshell: The Definitive Reference. O'Reilly Media, 2022. 1040 с. (дата звернення: 22.05.2025).

11.Mistry R., Misner S. Introducing Microsoft SQL Server 2014. Microsoft Press, 2014. 144 с. (дата звернення: 24.05.2025).

12.Smith J. Entity Framework Core in Action. Manning Publications, 2018. 520 с. (дата звернення: 25.05.2025).

13.LINQ in depth: advanced features - Blexin. URL: https://blexin.com/en/blog-en/linq-in-depth-advanced-features/ (дата звернення: 25.05.2025).

14.Hashing and Validation of SHA-256 in C# Implementation. *MojoAuth - Secure Passwordless Authentication Platform*. URL: https://mojoauth.com/hashing/sha-256-in-c/ (дата звернення: 27.05.2025)

15. Baeldung – SHA256 hashing. URL: https://www.baeldung.com/sha-256-hashing-java (дата звернення: 27.05.2025).

16.National Institute of Standards and Technology (NIST) – FIPS 180-4. Secure Hash Standard (SHS), (дата звернення: 29.05.2025).

17.K-Nearest Neighbor(KNN) Algorithm - GeeksforGeeks. URL: https://www.geeksforgeeks.org/k-nearest-neighbours/ (дата звернення: 31.05.2025).

18.Перші кроки з ML.NET: як навчити машину розпізнавати об’єкти. *DOU*. URL: https://dou.ua/forums/topic/34961/ (дата звернення: 30.05.2025).

ДОДАТОК А

Реалізація підсистеми рекомендацій товарів із використанням ML.NET алгоритму розбору протоколів

Програмні засоби:

– мова програмування – C#;

– бібліотека машинного навчання – ML.NET;

– підсистема LINQ для аналізу транзакцій;

– ORM-технологія – Entity Framework;

– обробка табличних даних – DataView;

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.ML;
using Microsoft.ML.Data;
using MyApp.Data;

public class RecommendationSystem
{
   public uint ProductId { get; set; }
   public uint CoProductId { get; set; }
   public float Label { get; set; } // 1 — is bought together, 0 — is not
}

public class PurchasePrediction
{
   [ColumnName("Score")] public float Score { get; set; }
}

public class RecommendationEngine
```

```
{
    private readonly MLContext _mlContext;
    private ITransformer _model;
    private readonly List<Product> _products;
    private readonly List<Transaction> _transactions;

    public RecommendationEngine(List<Product> products, List<Transaction>
transactions)
    {
        _mlContext = new MLContext();
        _products = products;
        _transactions = transactions;
    }

    // dataset for studying
    private IEnumerable<PurchaseData> PrepareTrainingData()
    {
        var pairs = _transactions
            .SelectMany(t => t.Products.SelectMany(
                p1 => t.Products.Where(p2 => p2.ProductID != p1.ProductID),
                (p1, p2) => new { p1, p2 }))
            .GroupBy(x => new { x.p1.ProductID, x.p2.ProductID })
            .Select(g => new PurchaseData
            {
                ProductId = (uint)g.Key.ProductID,
                CoProductId = (uint)g.Key.ProductID,
                Label = g.Count()
            });
        return pairs;
    }
```

```
    // Тренування моделі на даних про покупки
    public void TrainModel()
    {
        var data = PrepareTrainingData();
        var trainingData = _mlContext.Data.LoadFromEnumerable(data);

        // Побудова пайплайну на основі алгоритму k-ближчих сусідів (ApproximateMatrixFactorization)
        var pipeline = _mlContext.Recommendation()
            .Trainers.MatrixFactorization(
                labelColumnName: "Label",
                matrixColumnIndexColumnName: "ProductId",
                matrixRowIndexColumnName: "CoProductId");

        _model = pipeline.Fit(trainingData);
    }

    public void SaveModel(string modelPath)
    {
        _mlContext.Model.Save(_model, null, modelPath);
    }



    public void LoadModel(string modelPath)
    {
        DataViewSchema schema;
        _model = _mlContext.Model.Load(modelPath, out schema);
    }

    public List<Product> GetRecommendedProducts(uint productId, int topN = 3)
    {
```

```
        var predictions = new List<(uint, float)>();

        foreach (var candidate in _products)
        {
            if (candidate.ProductID == productId) continue;
            var input = new PurchaseData { ProductId = productId, CoProductId = (uint)candidate.ProductID };
            var predictionEngine = _mlContext.Model.CreatePredictionEngine<PurchaseData, PurchasePrediction>(_model);
            var prediction = predictionEngine.Predict(input);
            predictions.Add(((uint)candidate.ProductID, prediction.Score));
        }

        // chosing topN products
        var recommendedIds = predictions.OrderByDescending(x => x.Item2)
            .Take(topN)
            .Select(x => x.Item1)
            .ToList();
        return _products.Where(p => recommendedIds.Contains((uint)p.ProductID)).ToList();
    }
}
```