МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ КОМПЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«_____» _____ 2025р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**

Тема: "Автоматизована база даних онлайн магазину з використанням методів машинного навчання"

Виконавець: Пушко Владислав Максимович

Керівник: доктор філософії Положенцев Артем Анатолійович

Нормоконтролер: ст. викл. Олександр ШЕВЧЕНКО

Київ – 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ “КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ”

Факультет *Комп’ютерних наук та технологій*

Кафедра *Комп’ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Аліна САВЧЕНКО

«_____»_____2025 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи (проєкту)

Пушка Владислава Максимовича

1. Тема роботи «Автоматизована база даних онлайн магазину з використанням методів машинного навчання» затверджена наказом ректора від 23.04.2025 р. за № 600/ст.

2. Термін виконання роботи з 12.05.2025 р. по 12.06.2025 р.

3. Вихідні дані до роботи: середовище розробки *Visual Studio,* мова розробки *C#,* ML.NET, WinForms.

4. Зміст пояснювальної записки: огляд та аналіз онлайн-магазинів, їх принципи роботи, дослідження технологій та інструментів для розробки програми, реалізація програми автоматизованої бази даних онлайн магазину з використанням методів машинного навчання.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу: вікно інструментів WinForms ; робоче вікно VisualStudio; робоче вікно SQL Server, онлайн системи управління онлайн комерцією.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Проаналізувати джерела за темою кваліфікаційної роботи та зробити аналіз аналогічних рішень та вимог користувача	12.05.2025 – 13.05.2025	Виконано
2	Написати перший розділ	14.05.2025 – 15.05.2025	Виконано
3	Провести аналіз сучасних рішень, обрати інструменти розробки	16.05.2025 – 17.05.2025	Виконано
4	Написати другий розділ	18.05.2025 – 20.05.2025	Виконано
5	Розробити проєкт архітектури системи, бази даних та користувацького інтерфейсу	21.05.2025 – 28.05.2025	Виконано
6	Реалізація функціональних модулів системи	29.05.2025 – 05.06.2025	Виконано
7	Оформлення третього розділу	06.06.2025 – 08.06.2025	Виконано
8	Оформити пояснювальну записку	09.06.2025 – 10.06.2025	Виконано
9	Підготувати графічний матеріал	10.06.2025 – 11.06.2025	Виконано
10	Отримання допуску до захисту та подача роботи в ДЕК	11.06.2025 – 12.06.2025	Виконано

7. Дата видачі завдання: «12» травня 2025 р.

Керівник кваліфікаційної роботи _____ Артем ПОЛОЖЕНЦЕВ
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Владислав ПУШКО
(підпис здобувача вищої освіти) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Автоматизована база даних онлайн магазину з використанням методів машинного навчання»: 67 с., 34 рис., 21 бібліографічних джерел.

Об’єкт дослідження – процес керування та аналізу даних про продажі й товари в онлайн-магазині.

Предмет дослідження – методи та засоби автоматизованої обробки бази даних онлайн-магазину з використанням мови програмування C# та елементів машинного навчання.

Мета кваліфікаційної роботи – розробка програми для ефективної автоматизованої обробки бази даних онлайн-магазину з використанням методів машинного навчання.

Методи дослідження – аналіз літературних і нормативних джерел, інтеграція та налаштування ML-алгоритмів, проектування програмного забезпечення, огляд і порівняння сучасних технологій для автоматизації обробки.

Отримані результати та їх новизна. Отримані результати та їх новизна: створено програму на C# для автоматизованої обробки даних онлайн-магазину з інтегрованими в реальному часі алгоритмами машинного навчання, що підвищує точність і своєчасність прогнозів закупівель.

Практичне значення роботи полягає у створенні програми для автоматизованої обробки бази даних онлайн-магазину, що може бути використано для підвищення ефективності обробки замовлень, зменшення кількості помилок при введені інформації та оптимізації процесу прогнозу закупівель.

ПРОГРАМА, БАЗА ДАНИХ, ОНЛАЙН МАГАЗИН, АВТОМАТИЗАЦІЯ, МАШИННЕ НАЧВАННЯ, ПРОГНОЗ ЗАКУПІВЕЛЬ, ОБРОБКА ДАНИХ WINFORMS, C#, .NET, SQL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

БД	База даних.
СУБД	Система управління базами даних.
ПЗ	Програмне забезпечення.
SQL	Structured Query Language - мова структурованих запитів.
.NET	Програмна платформа, розроблена компанією Microsoft.
WinForms	Windows Forms - технологія для створення графічного інтерфейсу користувача на платформі
.NET.IDE	Integrated Development Environment - інтегроване середовище розробки.
Visual Studio	Середовище розробки додатків від Microsoft.
CRUD	Create, Read, Update, Delete Базові операції з базами даних.
CLR	Common Language Runtime - середовище виконання .NET-програм.
ML.NET	Бібліотека машинного навчання для платформи.
.NET.UI	User Interface інтерфейс користувача.
AutoComplete	Функція автоматичного доповнення введеного тексту.
Forecasting	Прогнозування (в контексті машинного навчання - передбачення майбутніх значень на основі історичних даних).

ВСТУП

Станом на сьогодні онлайн-торгівля є ключовим елементом глобальної економіки. Вона є однією з найпопулярніших галузей у світі. Щомісяця інтернет-магазини обробляють сотні замовлень, але при цьому часто стикаються з проблемами, пов’язаними з ручним введенням даних про клієнтів, товари та залишки на складі. Невірно зазначений артикул товару чи несвоєчасне оновлення інформації про залишки може призвести до того, що клієнт дізнається про помилку лише після оплати. Згідно статистики 34% покупців відмовляється від повторних замовлень після негативного досвіду, а 15% залишають негативні відгуки що знижує рейтинг магазину в пошукових системах. Крім того, багато існуючих систем обробки замовлень не включають інструментів для прогнозування попиту. Це змушує менеджерів орієнтуватися на мінімальні історичні дані або інтуїтивні оцінки, що часто призводить до надлишкових запасів або дефіциту товарів. Помилки які пов'язані при введенні інформації, затримками в оновленні залишків товарів та відсутність інструментів для прогнозування попиту призводить до фінансових втрат і зниження лояльності клієнтів.

Актуальність теми полягає у написанні програми мовою програмування C#, яка дає можливість вирішити велику кількість проблем пов’язаних з онлайн-торгівлею. Це програмне забезпечення автоматизує більшість процесів, а саме процеси пов’язані з додаванням даних про товари, покупців, категорії товарів, також вона дозволяє зберігати історію покупок клієнтів та здійснювати прогноз закупівель на основі цієї історії. Саме таке рішення покращить якість виконання операцій з даними, що в свою чергу дозволяє зменшити велику кількість проблем з введенням бізнесу.

Об’єктом дослідження є процес автоматизації роботи онлайн-магазину, а саме обробка інформації про товари, категорій, замовлень, клієнтів та продажі.

Предметом дослідження є методи та засоби автоматизованої обробки БД онлайн-магазину з використанням мови програмування C# та елементів машинного навчання.

Метою роботи є розробка програми для ефективної автоматизованої обробки БД онлайн-магазину з використанням методів машинного навчання. Програма має можливість працювати з БД, що містить таблиці товарів, клієнтів, замовлень, категорій товарів. Програма матиме можливість створювати чеки, а дані, що були в чеках, зберігатимуться в БД. А вже на основі даних в чеках можна здійснювати прогноз закупівель на певний місяця, шляхом аналізу попередніх продажів за рік.

Основні завдання дослідження :

— Аналіз стану онлайн торгівлі та виявлення основних проблем.

— Створення структури БД для зберігання інформації.

— Розробка програми для автоматизації.

— Інтеграція елементів машинного навчання для передбачення майбутніх закупівель.

— Тестування програми та проаналізувати отримані результати.

Методи дослідження: аналіз літературних і нормативних джерел, інтеграція та налаштування ML-алгоритмів, проектування ПЗ, огляд і порівняння сучасних технологій для автоматизації обробки.

Науковою новизною є пряма інтеграція модуля машинного навчання ML.NET у десктопний інтерфейс WinForms на C#. Завдяки цьому кожне нове замовлення автоматично оновлює набір вхідних ознак у локальній базі SQL Server, і прогноз обсягів закупівель формується безпосередньо під час роботи програми. У традиційних рішеннях аналітика виконується як відкладене пакетне завдання або на віддаленому сервері, а тут весь цикл «від введення даних до отримання прогнозу» відбувається в одному процесі. Така архітектура усуває потребу в сторонніх веб-сервісах чи хмарних обчисленнях, знижуючи витрати на інфраструктуру та скорочуючи час обробки інформації. В результаті користувачі отримують актуальні рекомендації щодо закупівель у режимі, близькому до реального часу, що підвищує оперативність управлінських рішень.

Практичне значення отриманих результатів полягає в створенні програмного продукту для автоматизованої обробки бази даних онлайн-магазину та оперативного прогнозування закупівель. Система централізовано зберігає дані про товари, клієнтів і замовлення, забезпечуючи валідацію на рівні інтерфейсу WinForms і контроль цілісності в SQL Server. Вбудований ML-модуль дає змогу менеджерам отримувати рекомендації щодо необхідного обсягу товарів наступного місяця, що знижує ризик дефіциту або надлишкових запасів. Це сприяє оптимізації обігу капіталу, зменшенню операційних витрат і підвищенню якості обслуговування клієнтів. Результати роботи можуть стати основою для подальшої розробки подібних систем у інших галузях і використовуватися як навчальний приклад при вивченні автоматизації торгових процесів із застосуванням C# та ML-методів.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальний огляд об’єкта дослідження

Онлайн-магазин це досить складна ,але дуже ефективна електронна платформа ,яка дозволяє клієнтам здійснювати покупки в Інтернеті. Станом на сьогодні онлайн-торгівля стала невід’ємною частиною мільйонів людей. Вона зручна, доступна та швидко виконує операції , що робить її надзвичайно популярною для клієнтів та бізнесу. Для традиційних продажів потрібно купити, побудувати або орендувати магазин, що вже робить його неефективним з точки зору витрат, а для онлайн-магазину потрібен лише спеціальний сайт, який і буде виступати торговим майданчиком. Навіть з мінімальними навичками в інтернеті робить такі сайти зручними для користувачів.

Онлайн магазин являє собою велику систему, в якій відбуваються велика кількість процесів, у якій продавці та споживачі взаємодіють за чіткими алгоритмами обміну даних. Онлайн магазини оперують з великою кількість даних, починаючи від заповнення даними товарів в онлайн магазині, до формування чеків для покупців.

В цій великій системі діють чіткі механізми, такі як облік товарів, контроль доступу, зберігання даних, авторизація дій та валідація операцій. Сама архітектура онлайн-магазину використовує рішення з використанням реляційних БД, серверної логіки, механізми захисту й резервне копіювання. Усе це підтримує роботу онлайн-магазину особливо під час великих навантажень на систему, наприклад під час сезонних розпродажів.

Автоматизовані процеси в онлайн-магазинах є дуже важливими, саме вони дозволяють оброблювати велику кількість замовлень з малою командою співробітників, зменшуючи витрати на персонал.

Одним із прикладів такого рішення є система планування ресурсів підприємства, яка на основі єдиної інформаційної бази: автоматизує основні бізнес-процеси, забезпечує швидкий та надійний обмін даними між підрозділами компанії, дозволяє масштабуватися без втрати продуктивності.

Завдяки таким системам онлайн-торгівля стає максимально ефективною, що стимулює зростання обороту та підвищує довіру клієнтів до сервісу.

1.2. Опис і аналіз предметної області

Сучасна електронна комерція успішно поєднує класичні бізнес-процеси з найновішими технічними рішеннями, охоплюючи всі етапи роботи з даними: від створення каталогу товарів та реєстрації профілів клієнтів до розміщення замовлень, обробки платежів та проведення аналітики. Назви товарів, описи, ціни та наявність зберігаються в реляційних БД , тоді як журнали взаємодії, відгуки та дані про поведінку клієнтів можуть бути розміщені в нереляційних базах даних для гнучкої роботи з напівструктурованою інформацією.

Для обробки замовлень та платежів використовуються серверні сервіси на базі .NET/C#, які відповідають за валідацію даних, маршрутизацію транзакцій та інтеграцію із зовнішніми платіжними шлюзами та логістичними API. Асинхронний зв'язок між підсистемами забезпечується чергами повідомлень, а масштабованість та стійкість до навантаження гарантуються контейнеризацією та оркестрацією.

Аналітика та прогнозування попиту відіграють вирішальну роль у зниженні операційних витрат та покращенні задоволення клієнтів. Алгоритми машинного навчання допомагають аналізувати історію продажів, сезонні коливання та маркетингові кампанії, автоматично адаптуючи запаси та рекомендації до індивідуальних уподобань клієнтів.

Моделі регресії та кластеризації дозволяють оптимізувати процес закупівель та уникати дефіциту або надлишкових залишків.

Інтеграція онлайн-магазину з ERP-системою на єдиній інформаційній базі забезпечує узгодженість даних між відділами: автоматичне оновлення балансів, синхронізація фінансових потоків та координація складів. Таке рішення звільняє співробітників від рутинних завдань та дозволяє їм зосередитися на стратегічних та маркетингових завданнях.

Основними викликами в предметній області є забезпечення високої якості даних, підтримка узгодженості в розподіленій архітектурі та захист персональних та фінансових даних клієнтів. Тільки поєднання сучасних технологій розробки, надійної СУБД, інтелектуальних модулів машинного навчання та оркестрації контейнерів дозволяє створити гнучку та безпечну платформу, яка забезпечує оперативну обробку замовлень, мінімізує витрати та підвищує лояльність клієнтів.

1.3. Аналіз програмних засобів для предметної області

В рамках аналізу програмних засобів для реалізації автоматизованої бази даних для інтернет-магазину, спочатку слід оцінити систему БД MS SQL Server . Інтерфейс платформи на рис. 1.1.

Вона рекомендується як надійна платформа для зберігання транзакційних даних завдяки підтримці стовпцевих індексів, масштабованих ETL-процесів через та можливості проведення розширеного аналізу безпосередньо всередині СУБД. Якщо потрібно запускати скрипти машинного навчання безпосередньо на сервері, можливо використовувати PostgreSQL разом з розширенням PL/Python, яке забезпечує доступ до функціональності Python безпосередньо в базі даних.

Для реалізації бізнес-логіки та взаємодії з базою даних використовують найчастіше C# у поєднанні з .NET 6 або 7 [1]. Це забезпечує потужну платформу для створення сервісів, додатків та логіки обробки даних. Entity Framework Core дозволяє зручно працювати з базою даних за допомогою абстракції у вигляді класів, що значно спрощує розробку.

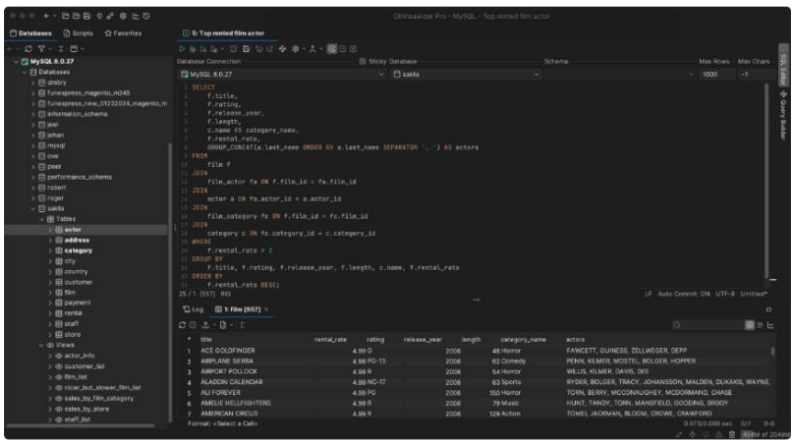

Рис. 1.1. Інтерфейс платформи MySQL

У цьому середовищі можна застосовувати ML.NET для розробки моделей класифікації або регресійних моделей - наприклад, для прогнозування попиту або формування рекомендацій щодо продуктів - без необхідності переходу на інші мови програмування [2].

Що стосується адміністративної панелі, то варіант залежить від потреб: WinForms підходить для швидкого прототипу з мінімальними залежностями, а WPF краще для гнучкого інтерфейсу з панелями інструментів та графіками.

Обидві технології добре інтегруються з сервісами REST, що важливо для взаємодії з іншими компонентами системи. Приклад створеного інтерфейсу зображено на рис. 1.2.

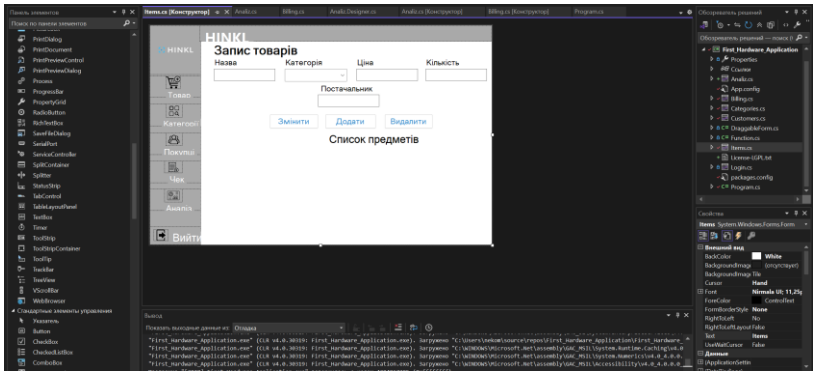

Рис. 1.2. Створенний інтерфейсу за допомогою бібліотеки WinForms

Якщо завдання аналітики або машинного навчання виходять за межі функціональності C#, доцільно виділити їх в окремий мікросервіс Python.

Для цього можна використовувати популярні бібліотеки, такі як pandas, scikit-learn або TensorFlow.

Такий сервіс може взаємодіяти з основною базою даних через REST API, отримувати дані для навчання або прогнозування у вигляді dataframes та повертати результати у форматі JSON.

Нарешті, для візуалізації аналітичних даних та моніторингу бізнес-метрик можна використовувати інтеграцію інструментів бізнес-аналітики, таких як Power BI або Grafana.

Вони можуть підключатися як до операційної бази даних, так і до окремого сховища даних, і дозволяють створювати інтерактивні панелі інструментів для відображення продажів, балансів, прогнозів попиту та ефективності моделей машин у режимі реального часу.

Цей набір програмних інструментів створює єдину, стабільну та гнучку платформу: реляційна СУБД та рівень C# відповідають за зберігання та цілісність даних, бібліотеки ML – за інтелектуальну обробку, WinForms/WPF – за зручну роботу, а Business Intelligence services– за прийняття обґрунтованих рішень на основі аналітики. Це дозволяє поступово розвивати інтернет-магазин, додаючи нові сервіси та масштабуючи систему в міру зростання бізнесу.

1.4. Огляд існуючих рішень

1.4.1. nopCommerce

nopCommerce - це система управління електронною комерцією з відкритим кодом, побудована на ASP.NET Core з підтримкою MS SQL Server [3]. Проєкт було офіційно запущено в жовтні 2008 року та розповсюджується безкоштовно за ліцензією nopCommerce Public License V3.

Станом на лютий 2025 року понад 54 000 сайтів використовують nopCommerce, а загальна кількість завантажень перевищила 3 мільйони.

Платформа довела свою надійність на прикладі брендових магазинів Volvo, Puma, Reebok, DHC skincare, Columbia, Medindia та Speedo.

Активна спільнота користувачів та розробників доповнює платформу: учасники допомагають один одному, розробляють плагіни та теми, а також беруть участь у плануванні дорожньої карти.

Наразі є 145 партнерів у 78 країнах, понад 1000 запитань з тегом «nopCommerce» на StackOverflow та понад тисячу розширень на маркетплейсі.

З 2015 року конференції #NopDevDays проводяться в різних містах, а з 2016 року вебінари та зустрічі регулярно організовуються по всьому світу.

nopCommerce спрощує створення та управління онлайн-магазином будь-якого масштабу завдяки гнучкій системі каталогу товарів, яка підтримує різні атрибути, опції та варіанти товарів. Платформа має вбудовані інструменти для маркетингу - купони, акції, рекомендації товарів та SEO-оптимізацію - які допомагають збільшити конверсію та залучити більше відвідувачів. Завдяки функції кількох магазинів ви можете керувати кількома онлайн-магазинами з однієї адміністративної панелі: налаштовувати окремі набори товарів, ціни, способи оплати та податки для кожного. Інтеграція з ERP, CRM та постачальниками платіжних послуг розширює можливості автоматизації бізнес-процесів та дозволяє зберігати дані про клієнтів та замовлення в єдиному середовищі. Платформа підтримує багатомовність та мультивалютність, що робить її ідеальною для виходу на міжнародні ринки та локалізації відповідно до різних регіональних вимог. Інтерфейс цієї системи зображений на рис. 1.3.

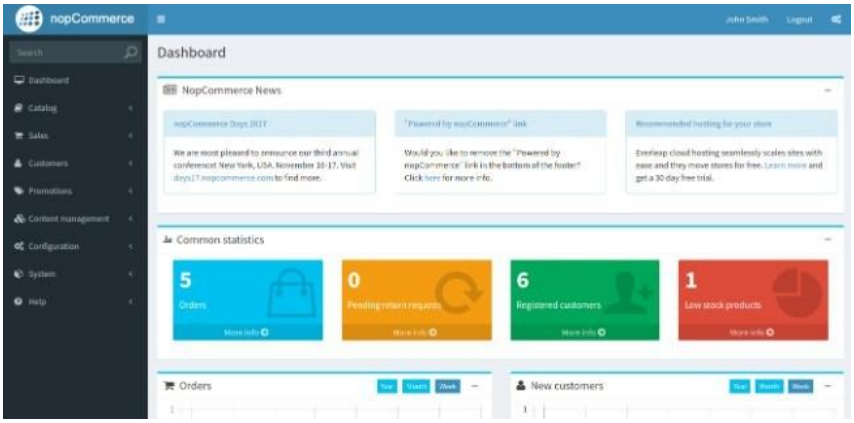

Рис. 1.3. Інтерфейс системи nopCommerce

1.4.2. Smartstore

Smartstore - це модульна, масштабована та надзвичайно швидка платформа електронної комерції, яка побудована на ASP.NET Core 7, Entity Framework Core та Vue.js [4].

Завдяки своїй кросплатформній архітектурі, Smartstore працює на Windows, Linux та macOS, а також легко розгортається в контейнерах Docker .

Платформа розповсюджується як Community Edition за ліцензією AGPL-3.0 та доступна безкоштовно; для корпоративних клієнтів надаються платні пакети підтримки та довгострокові релізи з розширеними послугами від розробників Smartstore.

Smartstore підтримує мультимагазини, мультивалютність та багатомовність, надаючи можливість керувати кількома онлайн-магазинами з різними доменами, мовами та валютами через єдину адміністративну панель . Вбудований редактор сторінок дозволяє створювати адаптивні та сучасні інтерфейси без написання коду, поєднуючи блоки зображень, тексту та товарів. Інтерфейс цієї системи на рис. 1.4.

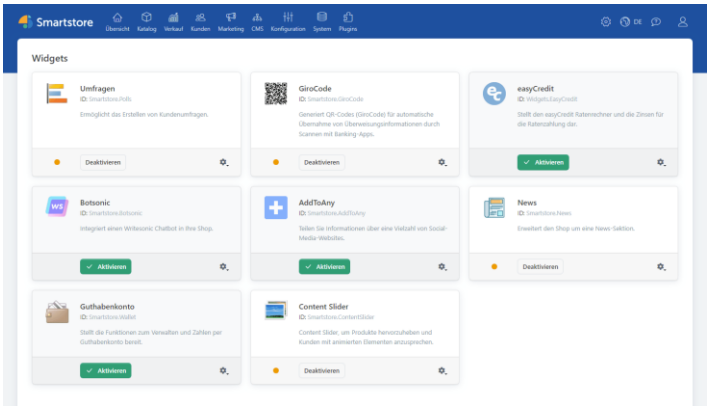

Рис. 1.4. Інтерфейс системи Smartstore

Екосистема Smartstore включає широкий спектр інструментів для управління каталогом товарів, зі зворотним зв'язком, знижками, купонами, SEO-оптимізацією, функціями CRM та CMS, що робить платформу універсальним рішенням для бізнесу будь-якого масштабу.

Smartstore потрібен для швидкого та гнучкого запуску онлайн-магазину з мінімальними зусиллями з налаштування, з вбудованими маркетинговими інструментами, високою продуктивністю та можливістю масштабування в майбутньому, що робить його привабливим як для стартапів малого бізнесу, так і для великих корпоративних проектів.

1.4.3. Virto Commerce

Virto Commerce - це відкрита платформа електронної комерції корпоративного рівня, побудована на ASP.NET Core та .NET, яка використовує архітектуру мікросервісного API для реалізації рішень на торгових майданчиках та багатокористувацьких рішень [5].

Платформу можна розгортати в різноманітних інфраструктурних сценаріях: на власних серверах компанії, у публічних чи приватних хмарах (зокрема в Microsoft Azure), а також у контейнеризованому вигляді з використанням Kubernetes. Така гнучкість розгортання дозволяє балансувати між контролем над інфраструктурою, експлуатаційними витратами та рівнем масштабованості, необхідним для обробки пікових навантажень. Використання Kubernetes автоматизує процеси розгортання й самовідновлення сервісів у разі збоїв, що гарантовано підтримує безперервність бізнес-процесів.

Для організацій із підвищеними вимогами до рівня обслуговування та стабільності платформи пропонуються платні версії з професійною підтримкою. Professional Edition включає гарантії SLA, довгострокові релізи з оновленнями та вичерпну документацію, а Virto Commerce Cloud у моделі SaaS забезпечує повністю кероване середовище з моніторингом, резервним копіюванням і автоматизованими оновленнями, що знімає навантаження з внутрішніх IT-команд клієнта.

Архітектурна модель Virto Commerce передбачає глибоку розширюваність: розробники можуть створювати власні модулі, теми оформлення й інтеграційні коннектори до ERP, CRM, платіжних та інших зовнішніх систем.

Це дозволяє адаптувати платформу під специфічні бізнес-процеси, інтегруватися з внутрішніми сервісами й реалізовувати унікальні сценарії роботи з клієнтськими даними.

Платформа має активну спільноту розробників і глобальну партнерську екосистему, що сприяє швидкому впровадженню нових розширень, обміну передовими практиками й регулярному оновленню функціоналу. Завдяки своїй надійності, гнучкості та широким можливостям кастомізації Virto Commerce є оптимальним рішенням для середніх і великих підприємств, які потребують масштабованої та високонастроюваної платформи електронної комерції з корпоративним рівнем підтримки. Інтерфейс цієї системи зображений на рис. 1.5.

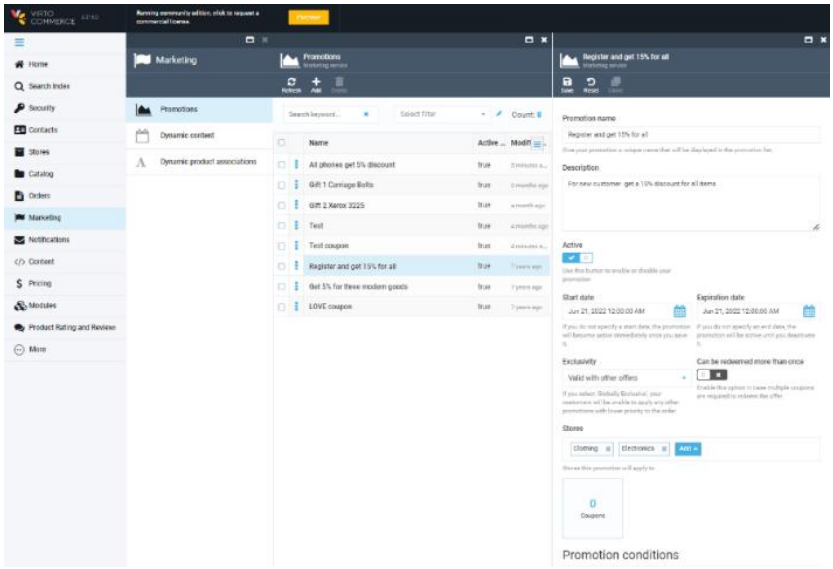

Рис. 1.5. Інтерфейс системи Virto Commerce

1.5. Висновок до розділу 1

У першому розділі було проведено комплексний огляд об’єкта та предмета дослідження - системи онлайн-торгівлі як багатокомпонентного середовища, що потребує автоматизації, масштабованості та аналітики. Визначено ключові особливості функціонування онлайн-магазинів, зокрема взаємодію користувача з системою, управління товарами, обробку замовлень та необхідність інтеграції з ERP-рішеннями.

Було проаналізовано сучасні підходи до реалізації таких систем, що поєднують реляційні й нереляційні бази даних, серверну логіку, технології контейнеризації (Docker, Kubernetes), інструменти прогнозування (ML.NET), а також засоби інтеграції з фінансовими, логістичними та аналітичними платформами.

Огляд існуючих платформ (nopCommerce, Smartstore, Virto Commerce) дозволив визначити поточний стан ринку і сформувати вимоги до функціональності майбутнього рішення. Також оцінено найдоцільніші інструменти розробки - MS SQL Server, .NET, C#, ML.NET, Python, WinForms/WPF - які забезпечують стабільну, модульну та масштабовану архітектуру.

Результати розділу підтверджують доцільність створення спеціалізованого програмного продукту для автоматизованої обробки даних онлайн-магазину з використанням інструментів машинного навчання, що сприятиме підвищенню ефективності бізнесу та задоволенню клієнтів.

РОЗДІЛ 2
ОПИС ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМИ

2.1. Загальний опис засобів реалізації програми

Мова програмування C#, платформа .NET Framework, середовище розробки Visual Studio, графічний інтерфейс Windows Forms (WinForms), система керування базами даних Microsoft SQL Server і бібліотека ML.NET.

Для розробки даної програми було обрано саме ці засоби через їхню ефективність, простоту у використанні та широку підтримку спільноти розробників.

Мова C# є однією з найкращих для створення застосунків під Windows. Вона поєднує простоту синтаксису з потужним функціоналом для роботи з даними, інтерфейсом та логікою програми. Платформа .NET надає багаті можливості для інтеграції з базами даних, роботи з графіками, обробки помилок і забезпечення безпеки.

Visual Studio - це зручне середовище розробки, яке містить усі необхідні інструменти для написання, налагодження, тестування та запуску програми.

Для створення графічного інтерфейсу було використано технологію Windows Forms, оскільки вона дозволяє швидко і просто створювати зручний та зрозумілий інтерфейс користувача. Це особливо важливо для програм, орієнтованих на бізнес-користувачів, які не мають технічної підготовки.

Для збереження та обробки даних про продажі використано Microsoft SQL Server. СКБД добре інтегруються з C#, забезпечують швидку обробку запитів та високу надійність при зберіганні великих обсягів інформації.

ML.NET дасть змогу прогнозувати майбутні продажі або аналізувати поведінку клієнтів на основі даних.

Таким чином, вибрані інструменти забезпечують:

- швидку розробку програми;
- зрозумілий інтерфейс;
- зручну роботу з базами даних;
- можливість аналітики та побудови графіків;
- простоту в оновленні та масштабуванні в майбутньому.

2.2. Платформа Microsoft SQL Server для управління базами даних

SQL Server - одна з найвідоміших систем керування базами даних, яка активно використовується в різних сферах - від розробки простих вебсайтів до побудови масштабних сервісів із високими вимогами до продуктивності. Система була створена Microsoft ще в 1987 році. З того часу вона пройшла чималий шлях: сучасна версія SQL Server 2022 вийшла у листопаді 2022 року і саме її можливості було використано в рамках цієї [6] .

Раніше SQL Server працював виключно під Windows, але з виходом 16-ї версії підтримка була розширена й на Linux. Це зробило платформу більш гнучкою й універсальною - тепер вона доступна розробникам, незалежно від того, яку операційну систему вони використовують.

Microsoft SQL Server надає повноцінну підтримку реляційної моделі даних. Це дозволяє створювати взаємозв’язані таблиці з чіткими обмеженнями, зовнішніми ключами, первинними ключами, що забезпечує логічну цілісність на всіх рівнях. Завдяки цьому помилки, пов’язані з дублікатами, некоректними значеннями або відсутніми зв’язками, виявляються ще на етапі запису в базу, що значно підвищує надійність системи.

Одна з важливих функцій SQL Server — підтримка транзакційного виконання операцій. Це означає, що будь-яка група дій над базою даних виконується як єдине ціле: або всі зміни вносяться успішно, або не вноситься нічого, якщо сталася помилка. Такий підхід особливо важливий для електронної комерції, де відсутність цілісності даних може призвести до втрати замовлень, неправильного розрахунку залишків товарів або помилок в обліку оплат.

Перевагою Microsoft SQL Server є також підтримка індексів, зокрема кластеризованих і не кластеризованих. Вони дозволяють значно пришвидшити виконання запитів, які містять сортування, пошук або агрегацію за певними колонками. Це особливо актуально для систем, які активно обробляють фільтрацію замовлень, пошук товарів за категоріями чи аналіз історичних даних.

Серверна частина SQL Server також підтримує збережені процедури та тригери, які реалізують частину бізнес-логіки безпосередньо на рівні бази даних. Це дозволяє винести типові або критичні операції (наприклад, автоматичну зміну статусу замовлення чи оновлення залишків на складі) в централізовану та захищену форму, яка не залежить від зовнішнього застосунку.

Ще однією перевагою є високий рівень безпеки. Microsoft SQL Server реалізує сучасні механізми аутентифікації, авторизації та контролю доступу до об’єктів. Користувачам можна надавати лише ті права, які потрібні для їхньої роботи, обмежуючи доступ до чутливої інформації. Крім того, передбачено механізми шифрування, захисту від SQL-ін’єкцій і журналювання змін, що важливо для захисту персональних та фінансових даних.

Microsoft SQL Server добре масштабовується як вертикально (за рахунок збільшення ресурсів сервера), так і горизонтально (через розподілення навантаження між кількома інстанціями). Це дозволяє системі витримувати зростання обсягів даних і кількості одночасних користувачів, не втрачаючи продуктивності.

У межах взаємодії з іншими частинами застосунку SQL Server підтримує численні механізми інтеграції, серед яких робота через ADO.NET, Entity Framework, REST API, зовнішні драйвери.

Завдяки цьому стає можливим гнучке включення бази даних у загальну архітектуру системи з підтримкою як десктопних клієнтів, так і вебслужб.

2.3. Використання SQL Server Management Studio для роботи з базою даних

Для роботи з реляційною системою управління базами даних Microsoft SQL Server застосовується спеціалізоване середовище — SQL Server Management Studio (SSMS). Це офіційний інструмент, що забезпечує повний спектр можливостей для адміністрування, налаштування, розробки й супроводу бази даних у візуальній формі. Інтерфейс програми SQL Server Management Studio зображений на рис. 2.1.

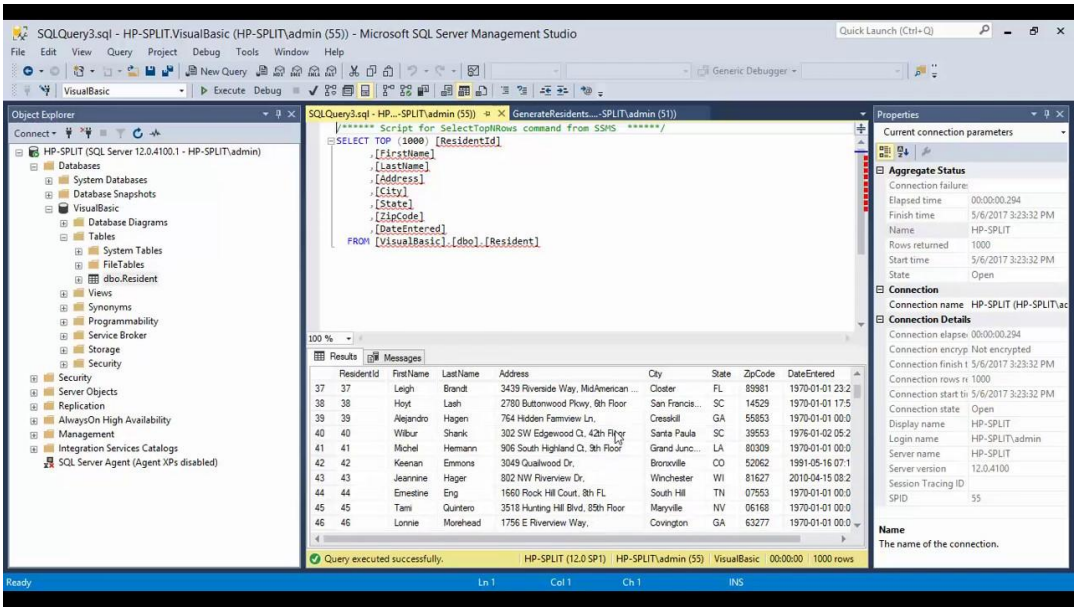

Рис. 2.1. Інтерфейс програми SQL Server Management Studio

SSMS надає користувачеві зручний графічний інтерфейс для взаємодії з базою: створення таблиць, перегляд схеми даних, виконання SQL-запитів, управління об'єктами бази, створення тригерів, функцій, представлень та інших компонентів. Завдяки цьому значно спрощується процес проєктування бази даних і знижується ймовірність помилок при ручному створенні складних конструкцій у коді.

Однією з важливих функцій SSMS є редактор запитів, який підтримує підсвічування синтаксису, автодоповнення команд і зручне форматування SQL-коду. Це дозволяє розробникам ефективно тестувати запити, проводити аналітику над даними, виконувати модифікації таблиць, а також створювати збережені процедури й функції. Наявність вкладок дозволяє одночасно працювати з кількома запитами або об'єктами бази, що особливо зручно під час налагодження складної логіки взаємозв’язків між таблицями.

Також середовище дозволяє переглядати структуру об'єктів, визначати зовнішні ключі, індекси, обмеження, перевіряти наявність зв’язків між таблицями та виявляти логічні помилки в структурі БД. Розробник може швидко вносити зміни у схему, переглядати попередні версії об’єктів, створювати скрипти для автоматизації оновлень чи міграцій бази між середовищами.

У межах адміністрування SQL Server Management Studio забезпечує доступ до засобів моніторингу навантаження, журналів подій, резервного копіювання та відновлення.

За допомогою вбудованих засобів можна створювати повні, диференціальні або транзакційні бекапи, що дозволяє захистити дані від втрати внаслідок збоїв або помилок користувача [6].

SSMS також має вбудовані механізми для управління безпекою: створення облікових записів, призначення прав доступу до об’єктів бази, встановлення ролей, налаштування аутентифікації. Це дозволяє ефективно контролювати, хто і які дії може виконувати з даними, що є критично важливим для інформаційної безпеки в електронній комерції.

Завдяки широкому набору функцій SQL Server Management Studio виступає як основний інструмент для розробника, адміністратора баз даних та аналітика. Його використання в рамках системи інтернет-магазину дозволяє швидко реагувати на зміни в структурі даних, відслідковувати продуктивність і забезпечувати контроль за цілісністю всієї інформаційної моделі. Простота у використанні поєднується з високим рівнем функціональності, що робить SSMS незамінним компонентом у реалізації реляційної бази даних на платформі Microsoft SQL Server.

2.4. Платформа .NET як середовище розробки прикладної частини

.NET Framework є комплексною платформою для створення і запуску додатків, орієнтованих на Windows-середовище. Основна мета цієї технології - забезпечити узгоджене об'єктно-орієнтоване середовище виконання, яке дозволяє розробникам створювати, зберігати, запускати та розгортати додатки локально, в мережі або через Інтернет [1].

До основних завдань .NET Framework належать:

- уніфікація програмного середовища для розробки локальних і веб-застосунків;
- мінімізація конфліктів при встановленні програмного забезпечення та під час оновлення версій;
- забезпечення безпечного запуску додатків, зокрема коду від третіх сторін;
- усунення обмежень продуктивності, характерних для сценарних або інтерпретованих мов;
- забезпечення підтримки галузевих стандартів і сумісності з іншим програмним кодом.

Архітектурно платформа складається з двох ключових компонентів - середовища виконання загальномовних програм CLR - Common Language Runtime та бібліотеки класів .NET Framework [1].

Середовище виконання CLR виконує роль керуючого агента, що відповідає за управління виконанням коду. Воно виконує низку важливих функцій:

- керування пам’яттю;
- управління потоками виконання;
- обробка викликів між процесами;
- забезпечення суворої типізації та перевірки безпеки.

Код, який обробляється CLR, називається керованим кодом, тоді як код, що працює поза межами CLR - некерованим.

Завдяки цій архітектурі можлива інтеграція як з нативними компонентами Windows, так і з компонентами, розробленими іншими мовами програмування, що підтримують стандарт CLI Common Language Infrastructure.

Бібліотека класів .NET Framework містить об'єктно-орієнтовану колекцію типів, що можуть бути повторно використані у створенні програм різного призначення - від консольних застосунків до складних багаторівневих веб-сервісів. Завдяки наявності потужної базової бібліотеки значно спрощується розробка користувацького інтерфейсу, взаємодії з файлами, базами даних, мережею та веб-ресурсами.

.NET Framework може розміщуватися всередині інших програм, які не є частиною самої платформи. У таких випадках некеровані компоненти ініціюють завантаження CLR у власному процесі та організовують виконання керованого коду. Це дозволяє поєднувати кероване та некероване середовище в межах одного застосунку, що розширює функціональні можливості системи.

Одним із прикладів такої інтеграції є ASP.NET - технологія для створення веб-застосунків, яка використовує CLR для запуску серверного керованого коду [7].

Вона забезпечує масштабоване, безпечне середовище для розробки веб-додатків та XML-вебсервісів [8].

На рис. 2.2 умовно показано, як взаємодіють компоненти .NET Framework, , CLR та бібліотеку класів. Схема демонструє загальну архітектуру середовища, що підтримує багатоаспектну розробку і запуск сучасних програмних рішень у Windows-екосистемі.

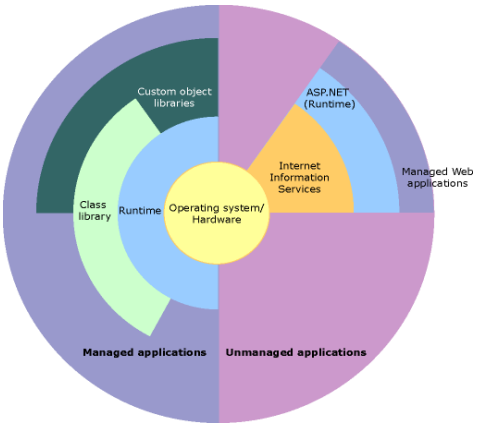

Рис. 2.2. Модель виконання .NET Framework

Середовище виконання загальномовних програм є фундаментальним компонентом платформи .NET Framework, який забезпечує виконання керованого коду з дотриманням вимог безпеки, типізації та ефективного управління ресурсами. CLR відповідає за такі критично важливі аспекти, як керування пам’яттю, виконання потоків, перевірка коду на безпечність, компіляція та реалізація системних служб [8].

Безпека виконання коду в CLR реалізується через механізм призначення ступеня довіри, який залежить від джерела походження компонента.

Наприклад, компоненти, що завантажуються з локального комп’ютера, можуть мати більше прав доступу порівняно з тими, що надходять з Інтернету. Завдяки цьому навіть в межах однієї програми окремі модулі можуть мати різні рівні доступу до ресурсів операційної системи, таких як файлові системи чи реєстр.

В основу архітектури CLR покладено інфраструктуру Common Type System (CTS), яка забезпечує сувору типізацію та гарантує сумісність між мовами програмування, що підтримують .NET Framework.

CTS дозволяє компонентам, написаним різними мовами, взаємодіяти між собою в межах одного середовища, що відкриває широкі можливості для повторного використання коду. Крім того, дотримання єдиної типізації дозволяє системі надійно перевіряти коректність коду ще до його виконання.

Управління пам’яттю є ще однією ключовою функцією CLR. Середовище автоматично розподіляє пам’ять під об’єкти та звільняє її, коли об’єкти більше не використовуються. Це дозволяє уникнути двох найпоширеніших помилок - витоків пам’яті та недійсних посилань. Завдяки цьому програмне забезпечення, що виконується під керуванням CLR, демонструє підвищену стабільність та захищеність.

Суттєвим є й те, що CLR дозволяє програмістам використовувати будь-яку з підтримуваних мов (наприклад, C#, Visual Basic .NET, F#), не втрачаючи при цьому доступу до переваг бібліотеки класів .NET та написаних іншими мовами компонентів. Компілятори мов, орієнтовані на CLR, транслюють вихідний код у проміжну мову (CIL), яка потім компілюється у машинний код безпосередньо під час виконання за допомогою механізму Just-in-Time (JIT) [9].

Це дозволяє досягти продуктивності, порівнянної з нативним виконанням, і водночас зберегти гнучкість та універсальність середовища [9].

Переваги керованого середовища не обмежуються лише новими застосунками. CLR також підтримує взаємодію з існуючими компонентами, такими як COM-об’єкти та динамічно підключувані бібліотеки (DLL). Це дає змогу поступово адаптувати застаріле програмне забезпечення до сучасних технологій без потреби повної його переробки.

Нарешті, можливість розміщення CLR всередині високонавантажених серверних платформ, таких як Microsoft SQL Server чи Internet Information Services (IIS), дає змогу реалізовувати власну бізнес-логіку з використанням керованого коду в умовах підвищених вимог до продуктивності.

Інтеграція JIT-компіляції та інтелектуального керування пам’яттю мінімізує фрагментацію та оптимізує використання ресурсів, забезпечуючи високу ефективність обробки даних.

2.5. Мова програмування C# як засіб реалізації додатку

C# є сучасною високорівневою мовою програмування, що розроблена компанією Microsoft спеціально для платформи .NET Framework. Вона поєднує у собі принципи об’єктно-орієнтованого, імперативного, функціонального та компонентно-орієнтованого програмуваннях [10]. Мову було стандартизовано міжнародними організаціями ECMA (ECMA-334) та ISO (ISO/IEC 23270:2006), що забезпечує її міжплатформену стабільність та сумісність.

Розробка мови здійснювалась під керівництвом Андерса Хейлсберга - одного з провідних архітекторів мов програмування в Microsoft. В основі C# лежить синтаксис, знайомий розробникам мов C, C++, Java, однак із суттєвими покращеннями щодо безпеки, структури та читабельності коду. Наприклад, мова підтримує сувору статичну типізацію, що мінімізує помилки типів на етапі компіляції, а також реалізує автоматичне збирання сміття, яке звільняє пам’ять без необхідності втручання програміста.

Однією з ключових цілей при створенні C# була розробка мови, яка б сприяла створенню безпечних, надійних програмних компонентів для розподілених систем. Для цього в мову інтегровано перевірку меж масивів, виявлення неініціалізованих змінних, обмежене використання вказівників, а також підтримку властивостей у класах. Останні забезпечують інкапсуляцію даних і полегшують роботу з об’єктами, зменшуючи ризик помилок при доступі до внутрішніх полів [11].

Мова є переносимою завдяки відповідності специфікації CLI (Common Language Infrastructure), що дозволяє компілювати код C# у проміжну мову, яку можна виконувати на будь-якій платформі, що підтримує .NET. Це забезпечує високу гнучкість при розробці кросплатформеного програмного забезпечення.

Синтаксичні особливості мови включають:

- використання крапки з комою для завершення інструкцій;
- фігурних дужок для формування програмних блоків;
- оператора «=» для присвоєння значення та «==» для порівняння;
- квадратних дужок для роботи з масивами;
- підтримку булевого типу як окремого логічного типу.

Приклад однієї з синтаксичних особливостей в мові C# а саме, використання крапки з комою для завершення інструкцій , показано на рис. 2.3.

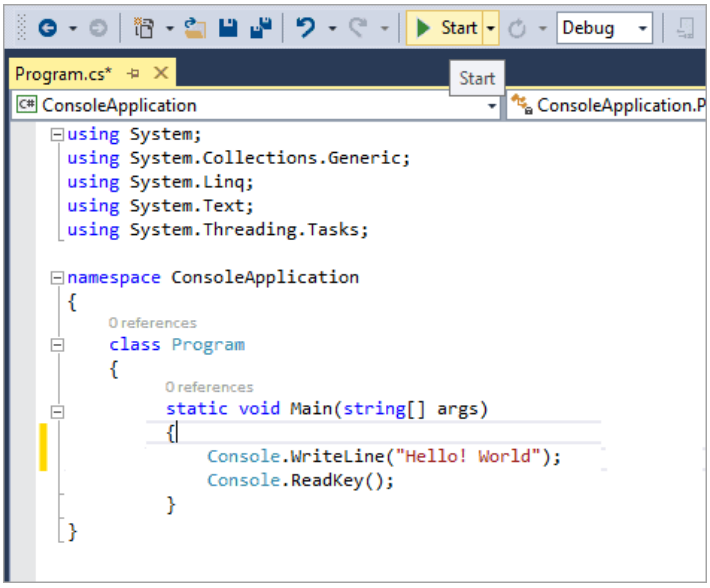

Рис. 2.3. Використання крапки з комою для завершення інструкцій

З моменту виходу першої версії C# 1.0 у 2002 році мова зазнала численних удосконалень. Наприклад, C# 8.0, випущена у 2019 році, додала нові функціональні можливості, такі як nullable reference types, asynchronous streams, ranges та індекси, що значно розширили можливості розробки додатків [12].

C# підтримується низкою сучасних середовищ розробки, серед яких найпоширенішим є Microsoft Visual Studio. Це інтегроване середовище забезпечує не лише комфортну роботу з кодом, а й доступ до розширених інструментів налагодження, тестування та керування проєктами. Альтернативні середовища включають JetBrains Rider, MonoDevelop, SharpDevelop тощо.

Сукупність особливостей мови, багатий набір стандартних бібліотек та підтримка від Microsoft роблять C# ефективним інструментом для створення програм будь-якої складності - від настільних додатків до хмарних рішень та мобільних застосунків.

2.6. Середовище розробки Visual Studio

Visual Studio - це потужне інтегроване середовище розробки (IDE), створене корпорацією Microsoft для створення програмного забезпечення різного призначення. Середовище підтримує розробку на численних мовах програмування, зокрема C#, Visual Basic .NET, C++, Python, JavaScript та інших [13]. Завдяки широкому набору інструментів і засобів автоматизації, Visual Studio значно підвищує ефективність процесу створення, налагодження та тестування програмного коду. Зовнішній вигляд Visual Studio зображено на рис. 2.4.

Однією з ключових особливостей Visual Studio є інтелектуальний редактор коду з підсвічуванням синтаксису, автодоповненням, інтерактивними підказками та підтримкою рефакторингу. Ці функції сприяють підвищенню продуктивності розробника, зниженню кількості синтаксичних помилок і забезпечують відповідність коду сучасним вимогам щодо структури та читабельності [14].

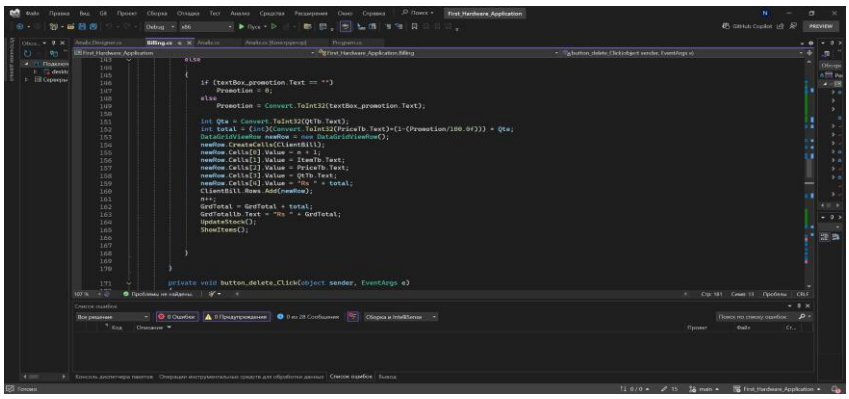

Рис. 2.4. Зовнішній вигляд Visual Studio

Однією з ключових особливостей Visual Studio є інтелектуальний редактор коду з підсвічуванням синтаксису, автодоповненням, інтерактивними підказками та підтримкою рефакторингу. Ці функції сприяють підвищенню продуктивності розробника, зниженню кількості синтаксичних помилок і забезпечують відповідність коду сучасним вимогам щодо структури та читабельності [14].

Невід'ємною складовою середовища є потужний засіб налагодження, що дозволяє покроково виконувати код, переглядати значення змінних у реальному часі та виявляти логічні помилки ще до початку етапу тестування. Такий підхід є надзвичайно корисним при розробці складних розподілених систем або проєктів з великою кількістю залежностей.

Visual Studio підтримує інтеграцію з системами контролю версій, зокрема Git та Team Foundation Version Control . Ця можливість дозволяє ефективно організовувати колективну роботу над програмними проєктами, відстежувати зміни та управляти конфігураціями коду. Підтримка розширень через Visual Studio Marketplace дає змогу розширити функціональність IDE, додавши підтримку нових мов програмування, інструментів для аналізу, генерації документації або автоматизованого тестування.

Середовище має вбудовані засоби для реалізації модульного та інтеграційного тестування, що дає змогу виявляти помилки ще на ранніх етапах життєвого циклу програмного забезпечення.

Крім того, IDE забезпечує високий рівень інтеграції з іншими продуктами Microsoft, такими як Azure, Microsoft Office 365 та GitHub, що дозволяє легко реалізовувати хмарні рішення, бізнес-додатки та інші масштабовані сервіси.

Visual Studio підтримує створення широкого спектру програмних рішень - від настільних додатків і вебсайтів до мобільних застосунків, ігор та хмарних сервісів. Таке розмаїття робить IDE універсальним інструментом для розробників, які працюють у різних галузях IT [15].

Завдяки великій спільноті розробників, багатій технічній документації та доступності навчальних ресурсів, середовище Visual Studio забезпечує не лише продуктивну розробку, але й підтримку з боку фахівців та ентузіастів.

Форми підтримки включають офіційну документацію, форуми, блоги та відеоуроки, що дозволяє швидко адаптуватися до роботи з IDE навіть розробникам-початківцям.

2.7. Фреймворк інтерейсу користувача

Windows Forms (WinForms) - це бібліотека .NET, яка надає засоби для створення графічного інтерфейсу користувача для настільних програм, що працюють у середовищі Windows [16]. Вигляд вибору шоблону інтерфейсу WinForm на рис. 2.5. Цей фреймворк широко використовується для розробки програм з багатим інтерфейсом, які мають доступ до ресурсів операційної системи, зокрема файлової системи, пристроїв введення/виведення та мережевих підключень.

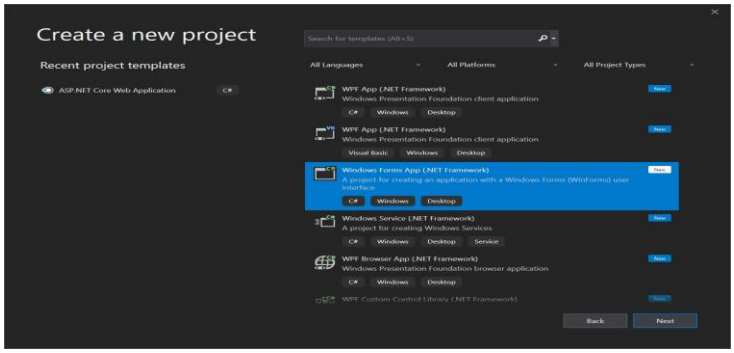

Рис. 2.5. Вибір шаблону інтерфейсу WinForm

У середовищі Visual Studio реалізовано візуальний конструктор форм, що дозволяє швидко компонувати елементи керування методом перетягування (drag-and-drop). Це значно скорочує час розробки інтерфейсу користувача та дозволяє зосередитись на реалізації бізнес-логіки додатка.

Основною складовою Windows Forms є форма - об’єкт, що виступає контейнером для елементів керування, таких як кнопки, текстові поля, списки, вкладки тощо. Кожен елемент керування взаємодіє з користувачем через події, наприклад, клацання миші або натискання клавіш, які обробляються програмним кодом [17].

Серед типових елементів керування, що надаються Windows Forms, виділяють TextBox, Button, ComboBox, ListBox, RadioButton, CheckBox, а також елементи для побудови інтерфейсу в стилі офісних додатків: MenuStrip, ToolStrip, StatusStrip. Інструменти, як-от FlowLayoutPanel, TableLayoutPanel та SplitContainer, забезпечують гнучку побудову макета форми без потреби в ручному вирівнюванні компонентів [18]. Приклад вигляду додадного елемента керування до форми зображений на рис.2.6.

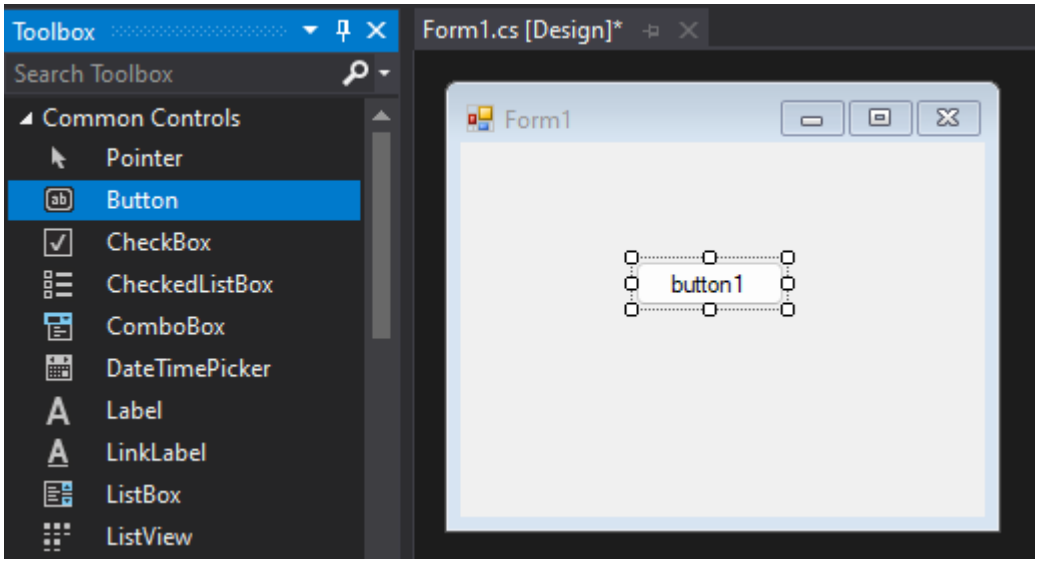

Рис. 2.6. Додавання кнопки до форми

Для створення унікальних візуальних елементів можливе використання класу UserControl, що дозволяє створювати власні компоненти інтерфейсу.

Крім того, простір імен System.Drawing надає доступ до засобів програмного малювання, що дає змогу створювати векторні елементи графіки безпосередньо на формі.

У програмах, де необхідно відображати структуровані дані з джерел, таких як бази даних, XML/JSON-файли або веб-сервіси, ефективним інструментом є елемент керування DataGridView. Він підтримує вивід табличних даних, форматування клітинок, фіксацію рядків або стовпців, а також вбудовування інших елементів керування безпосередньо в таблицю [18].

Прив’язка даних реалізується через компонент BindingSource, що виступає посередником між джерелом даних та елементами інтерфейсу.

Це дає змогу реалізовувати як односторонню, так і двосторонню прив’язку, з підтримкою навігації, редагування, збереження змін. Додатково компонент BindingNavigator надає інтерфейс для зручного переміщення між записами джерела даних.

Важливою особливістю є інтеграція Windows Forms із вікном «Джерела даних» у Visual Studio, що дозволяє здійснювати прив’язку даних до форм шляхом перетягування з представлення об'єктів.

Це спрощує створення форм, які автоматично відображають або змінюють інформацію з баз даних або інших структурованих джерел.

Для зберігання конфігураційних параметрів додатків передбачено механізм «Налаштування програми», який дозволяє зберігати як глобальні, так і користувацькі параметри. Визначені налаштування зберігаються у форматі XML і автоматично підвантажуються при наступному запуску програми.

Таким чином, Windows Forms забезпечує ефективне середовище для створення десктопних застосунків із багатим інтерфейсом, широкою підтримкою елементів керування, розширеною роботою з даними та інтеграцією з візуальними засобами розробки.

2.8. Використання ML.NET у системі на C#

ML.NET - це бібліотека з відкритим кодом, призначена для реалізації алгоритмів машинного навчання у додатках, що розробляються на платформі .NET [19]. Основною перевагою цієї технології є глибока інтеграція з мовою програмування C# та зручна підтримка розробки в середовищі Visual Studio. ML.NET дозволяє виконувати навчання моделей, здійснювати передбачення та реалізовувати повний цикл машинного навчання без необхідності використовувати зовнішні інструменти чи сервіси.

Бібліотека ML.NET підтримує широкий спектр завдань машинного навчання, серед яких:

- Класифікація (наприклад, визначення категорії товару на основі опису);
- Регресія (прогнозування числових значень, наприклад, ціни товару);
- Кластеризація (групування подібних об’єктів, наприклад, клієнтів за поведінкою);
- Виявлення аномалій (ідентифікація відхилень у транзакціях чи поведінці користувачів);
- Обробка тексту та зображень (з використанням попередньо навчених моделей або власного навчання).

Завдяки модульній структурі, ML.NET забезпечує підтримку як власноруч створених моделей, так і попередньо навчених, включно з можливістю їх імпорту з інших платформ (наприклад, TensorFlow, ONNX). Це дозволяє легко адаптувати вже наявні рішення або створювати нові моделі, які відповідають конкретним задачам прикладного ПЗ [20].

Особливістю ML.NET є підтримка різних типів вхідних даних: табличних, текстових, зображень, аудіо та відео. Таким чином, розробник може інтегрувати машинне навчання до програм різного призначення - від аналізу поведінки користувача до систем автоматичної класифікації зображень

Для роботи з даними ML.NET використовує трубопроводи обробки даних (Data Processing Pipelines), які дозволяють послідовно виконувати трансформації, вибірки ознак і виклик моделей. Крім того, бібліотека надає високорівневий API (ModelBuilder, AutoML) для швидкого прототипування без глибоких знань у сфері машинного навчання, а також низькорівневий API для побудови кастомізованих рішень.

У середовищі Visual Studio реалізовано інструмент ML.NET Model Builder, що дозволяє створювати моделі через графічний інтерфейс - шляхом імпорту набору даних, вибору типу завдання і навчання моделі в кілька кліків. Створена модель зберігається у вигляді ZIP-файлу та може бути завантажена й використана в будь-який момент виконання програми.

Інтеграція з екосистемою .NET робить ML.NET природним вибором для розробників, які створюють десктопні (Windows Forms, WPF), веб- (ASP.NET Core) або серверні додатки. Модель машинного навчання може бути викликана безпосередньо у коді C# для реалізації автоматизованих рішень на основі даних [21].

2.9. Висновок до розділу 2

У другому розділі було детально проаналізовано та обґрунтовано вибір інструментів для реалізації автоматизованої системи обробки даних онлайн-магазину. В якості основних технологій було обрано мову програмування C#, платформу .NET, середовище розробки Visual Studio, систему керування базами даних Microsoft SQL Server, бібліотеку ML.NET для машинного навчання та фреймворк WinForms для створення графічного інтерфейсу.

Розгляд архітектури та функціоналу SQL Server підтвердив його доцільність для використання як стабільної реляційної СУБД з високим рівнем безпеки та продуктивності. Окрему увагу було приділено створенню структури бази даних, таблиць і формуванню SQL-запитів для роботи з даними.

.NET Framework та CLR забезпечують ефективне виконання коду, захист пам’яті та інтеграцію між модулями, що дозволяє створювати надійні й масштабовані рішення. Завдяки ML.NET система отримує можливість реалізації інтелектуальних функцій, зокрема прогнозування закупівель на основі історії продажів.

Фреймворк WinForms дозволив створити зручний та інтуїтивно зрозумілий інтерфейс, що важливо для кінцевих користувачів. Візуальне середовище Visual Studio значно спростило розробку, налагодження та підтримку проєкту.

Усі обрані інструменти разом формують єдину технологічну екосистему, яка забезпечує високу продуктивність, надійність, масштабованість і можливість подальшого розвитку системи.

РОЗДІЛ 3
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Налаштування структури БД через середовище управління та підключення

Початковий етап створення бази даних відбувся через графічний інтерфейс середовища управління SQL Server. Після встановлення серверного компонента обрали пункт «Бази даних» та в діалоговому вікні ввели назву HardwareBD, залишивши стандартні розміри файлів даних і журналу транзакцій. Після підтвердження з’явилася порожня база даних у списку.

Перехід до створення таблиць виконано через пункт «Таблиці» та команду «Створити таблицю». У конструкторі послідовно додано поля для таблиці категорій: ідентифікатор та назва. Для поля ідентифікатора встановлено ознаку «Первинний ключ». Після збереження таблиця отримала ім’я CategoryTbl.

Створення таблиці товарів відбувалося аналогічно. У конструкторі додано поля: ідентифікатор, назва, ціна, код категорії, кількість, постачальник . Далі у властивостях зв’язків налаштовано зовнішній ключ на таблицю категорій із каскадним видаленням залежних записів. Таблиця збережена як ItemsTbl.

Таблицю клієнтів оформлено через конструктор таким чином: ідентифікатор, ім’я та прізвище, електронна пошта, дата реєстрації. Назву таблиці встановлено Customers.

Для таблиці замовлень додано поля: ідентифікатор, ім’я клієнта, кількість покупок, гендер, номер телефону.

Створення таблиці чеків виконано через конструктор: ідентифікатор замовлення, дата замовлення, категорія, кількість, метод оплати та номер.

У таблиці аналітики Sales додано поля: ідентифікатор, дата, назва товару, категорія, кількість, ціна, сезон та акція. Структура всіх таблиць зображена на рис.3.1.

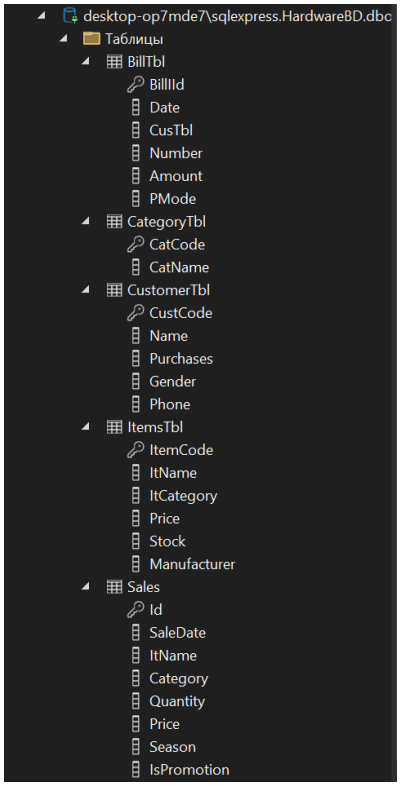

Рис. 3.1. Структура таблиць бази даних

Підключення бази до Windows Forms-додатку здійснено через копіювання рядка підключення з властивостей бази. У конфігураційному файлі проєкту додано секцію з вказанням сервера, назви бази та облікових даних користувача з правами лише на читання та запис, також є можливість змінювати структуру.

У файлі виділено окремий клас для взаємодії з базою даних, що поєднує об’єкти керування підключенням і виконанням команд. На початку простору імен прописані необхідні директиви – звернення до базових колекцій, до даних ADO.NET та до провайдера SQL Server. Це дозволяє працювати з типами SqlConnection, SqlCommand, SqlDataAdapter та класом DataTable без додаткових префіксів. Клас для підключення та роботи з базою даних зображений на рис. 3.2.

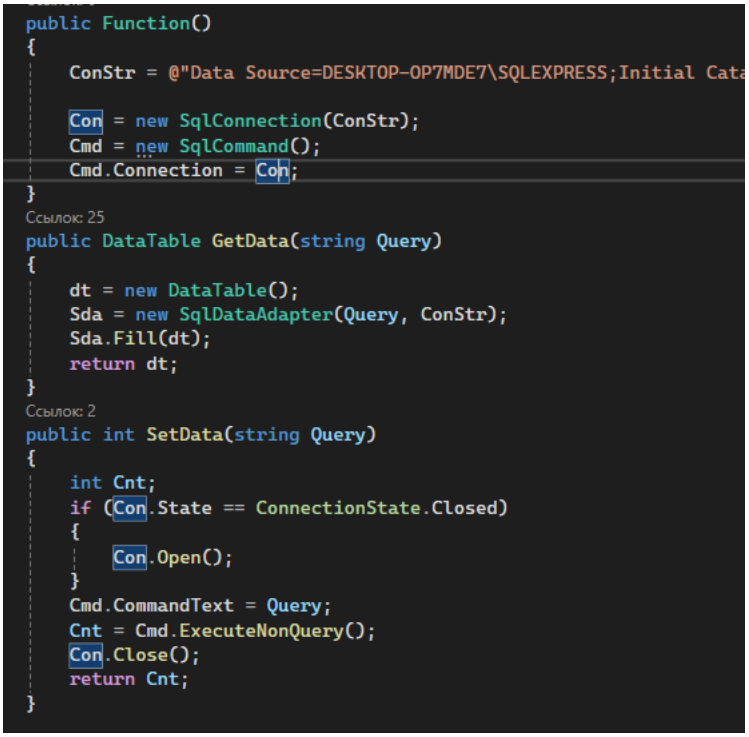

Рис. 3.2. Клас для підключення та роботи з базою даних

У конструкції класу вказано приватне поле для рядка підключення. Воно містить назву сервера та інстансу SQL Server Express, назву цільової бази даних, параметри безпеки (використання автентифікації Windows, шифрування каналу та довіру до сертифіката). Далі створюється екземпляр підключення за цим рядком і командний об’єкт, який одразу ж отримує посилання на створене підключення.

Метод, що повертає дані, спроектовано так, щоб виклик будь-якого SQL-запиту повертав структуровану таблицю результатів. Усередині створюється порожня таблиця, потім ініціалізується адаптер із переданим рядком запиту та тим самим рядком підключення. Виклик методу заповнення адаптера автоматично встановлює з’єднання, виконує запит, переносить усі результати в таблицю та закриває підключення. Повернення готової таблиці дозволяє використати її у формі даних Windows Forms, підставити в елемент керування або обробити далі в логіці програми.

Операції зміни даних спрощено до іншого методу: перед виконанням перевіряється стан підключення, і якщо воно закрите, відбувається відкрите під’єднання до сервера.

У цьому методі текст запиту передається властивості командного об’єкта, після чого запускається виконання без очікування набору рядків. Результатом виконання стає кількість змінених рядків, що одразу повертається викликачу. Після завершення команда закриває з’єднання. Подібний підхід гарантує, що відкритий канал ніколи не залишиться активним після виконання, а кількість торкнутих записів підкаже успішність чи необхідність подальшої перевірки.

3.2. Головне меню програми

Головне меню програми у межах даної програми відіграє ключову роль у навігації між формами. Воно побудоване на основі панелей-кнопок, які в свою чергу активують відповідні форми, які завантажуються в основну область вікна. Ця реалізація забезпечує зручну взаємодію користувача з системою, також спрощує навігацію між розділами і дозволяє уникнути проблем з використанням окремих вікон для кожної форми.

Основна форма MainForm є контейнером для елементів панелі керування формами і самою панелю та має область для вмісту де відображаються відповідні форми. Також при вході до програми зустрічає текст, де коротко описуються основні функції програми. Вигляд основною форми зображений на рис. 3.3.

Рис. 3.3. Вигляд основної форми MainForm

Для забезпечення інтеграції форм було використано метод ShowFormInHost(Form form).

В цьому методі очищується вміст контейнера, завантажує нову форму як вкладений елемент, застосовує прив’язку для адаптації розміру до контейнера та відображає форму. Код цього методу зображений на рис. 3.4.

```
138     private void ShowFormInHost(Form form)
139     {
140         panelHost.Controls.Clear();
141         form.TopLevel = false;
142         form.FormBorderStyle = FormBorderStyle.None;
143         form.Dock = DockStyle.Fill;
144         panelHost.Controls.Add(form);
145         form.Show();
146     }
```

Рис. 3.4. Код методу ShowFormInHost(Form form)

Також було додано механізми для візуального виділення активної панелі. Замінюється фон та колір тексту, активного пункту меню. Також було забезпечено динамічний ефект наведення, у відповідній панелі у разі наведення курсору замінюється фон, якщо вона не активна. Робота цих механізмів зображено на рис. 3.5 та рис. 3.6.

Рис. 3.5. Наведення мишки на пункт меню

Рис. 3.6. Активний пункт меню

Також додано підтримку переміщення форми без рамки. Для цього форма реагує на подію MouseDown певних панелей та елементів.

Також кожен пункт меню пов'язаний із відповідними формами. Натискання на певну кнопку викликає методи для завантаження потрібної форми та для оновлення візуального статусу меню. Код реалізації зображений на рис. 3.7. Саме цей підхід дозволяє забезпечити єдину логіку переходів у межах програми без потреби відкриття нових вікон.

```
private void btnItems_Click(object sender, EventArgs e)
{
    ShowFormInHost(new Items());
    SetActivePanel(panelItems);
}
```

Рис. 3.7. Код для відображення форми Товарів та візуального статусу

3.3. Форма авторизації програми

Форма авторизації відіграє також ключову роль, а саме у забезпечені контролю доступу до функцій програми. Вона реалізує просту, але надійну схему входу для користувачів. Оформлена у вигляді окремої форми з елементами взаємодії та підвищеним зручності для користувача. Вигляд форми зображено на рис.3.8.

Рис. 3.8. Форма авторизації

Форма містить два основних поля введення: textbox_username для логіну та textbox_password для паролю. Після натиснення кнопки Увійти відбувається процес перевірки введених даних. В цьому випадку при введені значення Admin у два текстових поля, авторизація вважається пройденою. Невдалі спроби супроводжуються повідомленням про помилку. Також ця форма адаптується під дії користувача.

Під час наведення на кнопку входу змінюється її колір та колір тексту, кнопка button_pass_vis дозволяє перемикати видимість пароля, забезпечуючи як зручність, так і конфіденційність входу та видимість цієї кнопки динамічно контролюється залежно від вмісту поля пароля та остання кнопка Скинути видаляє введені значення, очищаючи форму.

3.4. Форма “Запис товарів”

Форма Items призначення для управління товарами в інформаційній системі онлайн-магазину. Її призначення це забезпечення відображення, пошук, додавання, редагування та видалення записів таблиці ItemsTbl у БД. Після завантаження компонента Items у конструкторі виконується InitializeComponent(), ініціалізується об’єкт Function для взаємодії з базою даних, а також викликаються ShowItems() і GetCategories() для початкового підвантаження даних і заповнення комбобоксу категорій. Вигляд форми Items зображений на рис. 3.9.

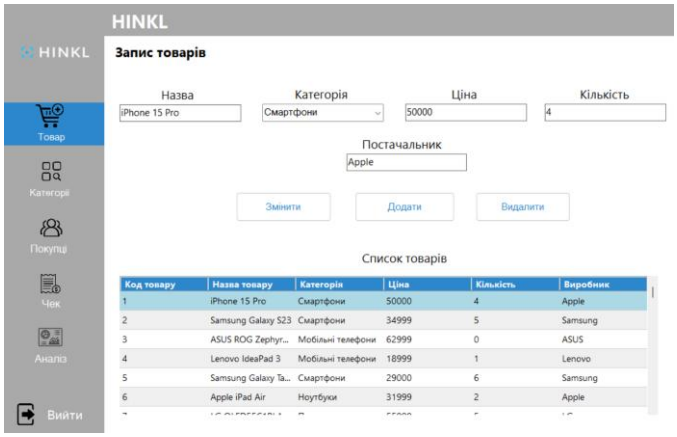

Рис. 3.9. Форма “Запис товарів”

Після стартових налаштувань викликається ShowItems(), який витягує дані з таблиць ItemsTbl і CategoryTbl через запит цей метод зображений на рис. 3.10.

```
private void ShowItems()
{

    string Query = @" SELECT I.ItemCode, I.ItName, C.CatName AS CategoryName, I.Price, I.Stock, I.Manufacturer
    FROM ItemsTbl I JOIN CategoryTbl C ON I.ItCategory = C.CatCode";

    var data = Con.GetData(Query);
    ItemList.DataSource = data;

    ItemList.Columns[0].HeaderText = "Код товару";
    ItemList.Columns[1].HeaderText = "Назва товару";
    ItemList.Columns[2].HeaderText = "Категорія";
    ItemList.Columns[3].HeaderText = "Ціна";
    ItemList.Columns[4].HeaderText = "Кількість";
    ItemList.Columns[5].HeaderText = "Виробник";
```

Рис. 3.10. Метод ShowItems з запитом

Результат потрапляє у DataGridView ItemList. Заголовки стовпців замінюються на: «Код товару», «Назва», «Категорія», «Ціна», «Кількість», «Виробник». Внаслідок цього інтерфейс стає максимально зручним для оператора.

Для того щоб ввімкнути швидкий пошук, назви товарів збираються в колекцію автодоповнення, а створена колекція зіставляє текст із рядком даних та дозволяє при введенні назви миттєво заповнювати поля CatCb, PriceTb, StockTb і ManufacturerTb; внаслідок цього відповідний рядок у таблиці підсвічується й прокручується в видиму область.

Додавання, редагування й видалення товарів здійснюються через відповідні SQL-запити INSERT, UPDATE … WHERE ItemCode = Key, DELETE … WHERE ItemCode = Key , після чого виконується повторне завантаження даних і очищення полів. Приклад зміни інформації в базі даних показано на рис. 3.11.

```
string Query = "update ItemsTbl set ItName = '{0}',ItCategory = '{1}',Price = '{2}',Stock = '{3}',Manufacturer = '{4}' where ItemCode = {5}";
Query = string.Format(Query, Name , Cat, Price , Stock, Man, Key);
Con.GetData(Query);
```

Рис. 3.11. Зміна інформації в базі даних Items

Кнопки отримали обробники MouseMove та MouseLeave, що змінюють їхнє оформлення при наведенні - колір фону та тексту змінюються на чіткий контрастний.

Натискання на рядок у таблиці (ItemList_CellClick) також заповнює поля форми, тож редагування можна розпочати без зайвих кліків. Завдяки такому підходу можна швидко й безпомилково вводити нові товари, виправляти помилки або видаляти застарілі позиції, не відриваючись від основного інтерфейсу.

3.5. Форма “Запис категорій”

Форма Categories призначенна для управління категоріями в інформаційній системі онлайн-магазину. Вона відповідає за додавання, редагування та видалення записів у таблиці категорій. Основною метою побудови даної форми стало забезпечення швидкої взаємодії з даними, підвищення наочності та інтуїтивної зрозумілості інтерфейсу. Зовнішній вигляд форми зображений на рис. 3.12.

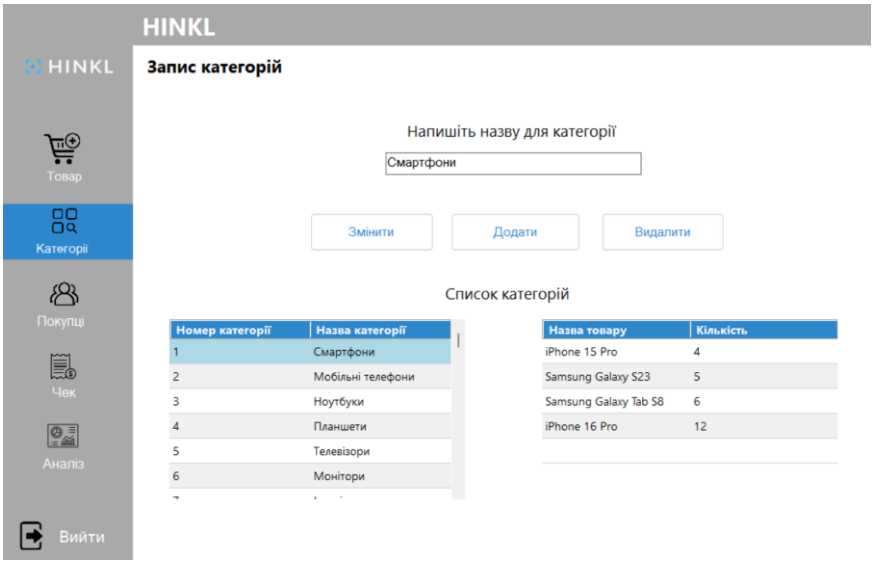

Рис. 3.12. Форма “Запис категорій”

Форма містить два візуальних компонента – таблиці для відображення списку категорія CategoryList та товарів ItemsList, а також текстове поле для введення або редагування назви категорії NameTb.

Вся взаємодія з базою даних здійснюється через об'єкт класу Function, який забезпечує з'єднання, виконання SQL-запитів і повернення результатів.

З моменту ініціалізації форми викликається метод ShowCategories(), який завантажує дані з таблиці CategoryTbl у DataGridView CategoryList. Для зручності додано автозаповнення: усі назви категорій передаються до AutoCompleteStringCollection, що дозволяє швидко знаходити потрібний запис. Також до події TextChanged текстового поля NameTb прив'язано логіку пошуку відповідної категорії та автоматичного заповнення таблиці товарів на основі обраного значення.

Операції у цій формі охоплюють три дії: перше це додавання нової категорії через кнопку button_additem. Код роботи видалення зображений на рис. 3.13. У разі порожнього поля NameTb виводиться повідомлення про помилку і запис не додається. Якщо назву введено, вона передається до SQL-запиту типу INSERT. Після цього викликається метод ShowCategories(), щоб оновити таблицю. Друга операція це редагування. Вона виконується лише для попередньо обраної категорії (ідентифікатор зберігається в змінній Key). Дані оновлюються через запит UPDATE. При успішному виконанні оновлення таблиця перезавантажується, і змінений запис залишається обраним. Третя дія - видалення. Якщо запис виділено, виконується SQL-запит DELETE, після чого таблиця оновлюється.

```
private void button_delete_Click(object sender, EventArgs e)
{
    if (NameTb.Text == " ")
        MessageBox.Show("Заповніть текстове поле!");
    else
        try
        {
            string Name = NameTb.Text;
            string Query = "Delete from CategoryTbl where CatCode = {0}";
            Query = string.Format(Query,  Key);
            Con.GetData(Query);
            ShowCategories();
            MessageBox.Show("Категорія видалена!");
            NameTb.Text = "";
        }
        catch (Exception Ex)
        {
            MessageBox.Show(Ex.Message);
        }
}
```

Рис. 3.13. Робота кнопки для видалення даних

При виборі рядка в таблиці категорій активується подія CategoryList_CellClick зображено на рис. 3.14.

На її основі в текстове поле NameTb завантажується назва категорії, а у таблиці ItemsList відображається список товарів, що належать до цієї категорії. Для цього формується запит до таблиці ItemsTbl з фільтрацією за категорією. Виводяться лише назва товару ItName і кількість на складі Stock. Це дає змогу швидко бачити вміст конкретної категорії без переходу на інші форми.

```
private void CategoryList_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        var selectedRow = CategoryList.Rows[e.RowIndex];

        NameTb.Text = selectedRow.Cells[1].Value?.ToString() ?? "Не вказано";

        if (!string.IsNullOrEmpty(selectedRow.Cells[0].Value?.ToString()))
        {
            Key = Convert.ToInt32(selectedRow.Cells[0].Value.ToString());

        }
        else
        {
            Key = 0;

        }

        ShowItemsByCategory(Key);
    }
}
```

Рис. 3.14. Метод CategoryList_CellClick

3.6. Форма “Покупці”

Форма Customers призначена для управління клієнтами в інформаційній системі онлайн-магазину, а саме адміністрування записів про клієнтів у базі даних.

Інтерфейс розроблений так, щоб забезпечити зручність перегляду, редагування та додавання нових покупців та швидкий пошук за іменем із використанням автозаповнення. Зовнішній вигляд форми зображений на рис. 3.15.

Після завантаження форми викликається метод ShowCustomers(), який підключається до БД і виконує запит до таблиці CustomerTbl. Результат відображається в таблиці CustomerList, яка стилізована відповідно до загального дизайну застосунку.

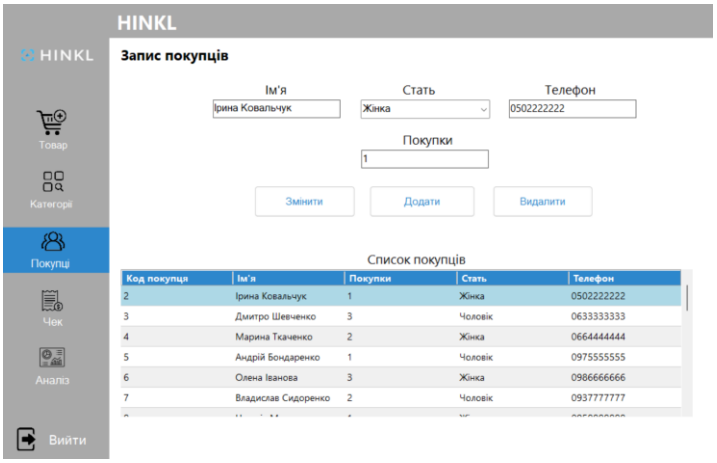

Рис. 3.15. Форма “Покупці”

Реалізовано події наведення курсора на кнопки: при наведенні змінюється фон і колір тексту, а при відведенні відновлюється попередній стиль із білим фоном і синіми межами. Це робить інтерфейс динамічнішим і приємнішим у використанні.

До цього поля підключено автозаповнення: список імен формується на основі унікальних значень з бази. При введенні імені відбувається перевірка, чи є відповідність. Якщо так автоматично підставляються стать і телефон у відповідні поля, а також виділяється відповідний рядок у таблиці.

Крім перегляду, реалізовано функції додавання, редагування та видалення покупців. Для створення нового запису достатньо заповнити поля і натиснути кнопку додавання. Усі дані передаються у SQL-запит INSERT. Якщо хоча б одне з полів не заповнене, з’являється попередження. Після успішного додавання таблиця оновлюється і поля очищуються. Редагування виконується лише після вибору запису в таблиці значення ключа зберігається в змінній Key. Змінені дані передаються у запит UPDATE. У разі помилки користувач бачить повідомлення, а при успішному збереженні форма оновлюється, і дані очищуються. Видалення працює за аналогічним принципом - при натисканні кнопки Delete формується запит до бази з умовою по ключу. Якщо запис успішно видалено, таблиця оновлюється, а поля скидаються до початкового стану.

3.7. Форма “Розрахунок чеку”

Форма Billing створенна для оформлення чеків та обліку продажів у програмі. Під час ініціалізації створюється екземпляр класу Function для роботи з базою, запускається побудова таблиць: ClientBill для формування чека та ItemList для перегляду наявного асортименту. Стовпці ClientBill налаштовано вручну - «Номер», «Предмет», «Ціна», «Кількість», «Остаточна ціна». Зовнішній вигляд форми зображений на рис. 3.16.

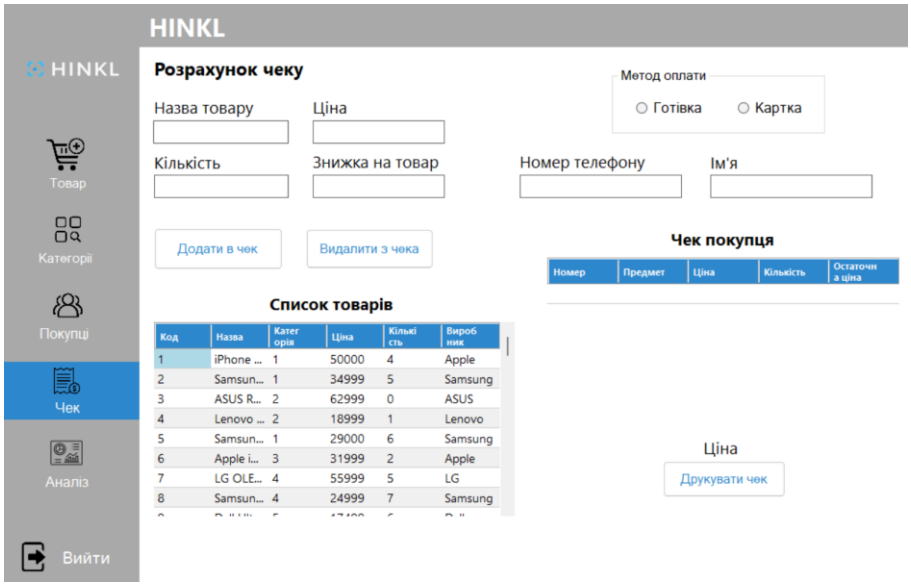

Рис. 3.16. Форма “Розрахунок чеку”

Автозаповнення клієнтів і товарів реалізовано через створення словників _customerByPhone та _itemDataCache. Після завантаження даних з CustomerTbl і ItemsTbl у відповідні колекції AutoCompleteStringCollection поле PhoneTb починає пропонувати номери телефонів, а при введенні існуючого контакту в NameTb автоматично відображається ім’я покупця. Аналогічно в полі ItemTb підказуються назви товарів, а поруч у PriceTb одразу з’являється його вартість. Вигляд методу для автозаповнення покупця зображений на рис. 3.17.

При виборі рядка в ItemList вказується код назва, ціна й кількість на складі , що дозволяє контролювати наявність.

Після введення кількості та промокоду кнопка «Додати» формує новий рядок у ClientBill: розрахунок вартості враховує знижку за промо, оновлює змінні GrdTotal та _baseTotal, а віджет GrdTotallb показує підсумок з префіксом «Rs ». Внаслідок цього за кілька кліків додаються одиниці товару із коректним підрахунком вартості.

```
private void LoadCustomersForAutocomplete()
{
    var dt = Con.GetData("SELECT Name, Phone FROM CustomerTbl");
    _customerByPhone = new Dictionary<string, DataRow>();
    var phones = new AutoCompleteStringCollection();

    foreach (DataRow row in dt.Rows)
    {
        var phone = row["Phone"]?.ToString();
        if (!string.IsNullOrEmpty(phone) && !_customerByPhone.ContainsKey(phone))
        {
            _customerByPhone[phone] = row;
            phones.Add(phone);
        }
    }

    PhoneTb.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
    PhoneTb.AutoCompleteSource = AutoCompleteSource.CustomSource;
    PhoneTb.AutoCompleteCustomSource = phones;


    PhoneTb.TextChanged += PhoneTb_TextChanged;
}
```

Рис. 3.17. Метод LoadCustomersForAutocomplete

При виборі рядка в ItemList вказується код назва, ціна й кількість на складі , що дозволяє контролювати наявність. Після введення кількості та промокоду кнопка «Додати» формує новий рядок у ClientBill: розрахунок вартості враховує знижку за промо, оновлює змінні GrdTotal та _baseTotal, а віджет GrdTotallb показує підсумок з префіксом «Rs ». Внаслідок цього за кілька кліків додаються одиниці товару із коректним підрахунком вартості.

При натискані кнопки «Друк чека» запускає генерацію нового номера через GenerateUniqueBillId(), що бере максимум BillIId з BillTbl і додає одиницю. Під час формування запису в BillTbl обираються дата, ім’я клієнта, номер телефону, сума з урахуванням знижки 2% для постійних покупців та спосіб оплати Cash або Card. Кожен рядок чека паралельно зберігається в таблиці Sales з додатковими полями: сезон продажу визначається методом GetSeason, чи застосована промо-знижка, категорія товару отримаємо з ItemsTbl.

Після збереження рахунку оновлюється лічильник покупок у CustomerTbl через CASE WHEN, відбувається виклик UpdateStock(), що зменшує кількість у складі, і

відбувається перезавантаження ItemList. Врешті-решт форма повертається в початковий стан: поля очищені, таблиця чека порожня, загальний підсумок обнулений, радіокнопки скинуті. Код описаний у абзаці на рис. 3.18.

```
string upd = $@"UPDATE CustomerTbl SET Purchases = CASE  WHEN ISNUMERIC(Purchases) = 1 THEN CAST(Purchases AS INT) + 1 ELSE 1 END WHERE Phone = '{phone}'";

Con.SetData(upd);
UpdateStock();
ShowItems();
```

Рис. 3.18. Код для оновлення лічильника покупок у покупця

3.8. Форма “Аналіз”

Форма Analiz у проекті виступає центральним елементом для взаємодії з користувачем і виконує одразу декілька ключових завдань: завантаження даних, відображення графіків, налаштування моделі машинного навчання та обробку подій вибору місяця. Зовнішній вигляд форми зображений на рис. 3.19.

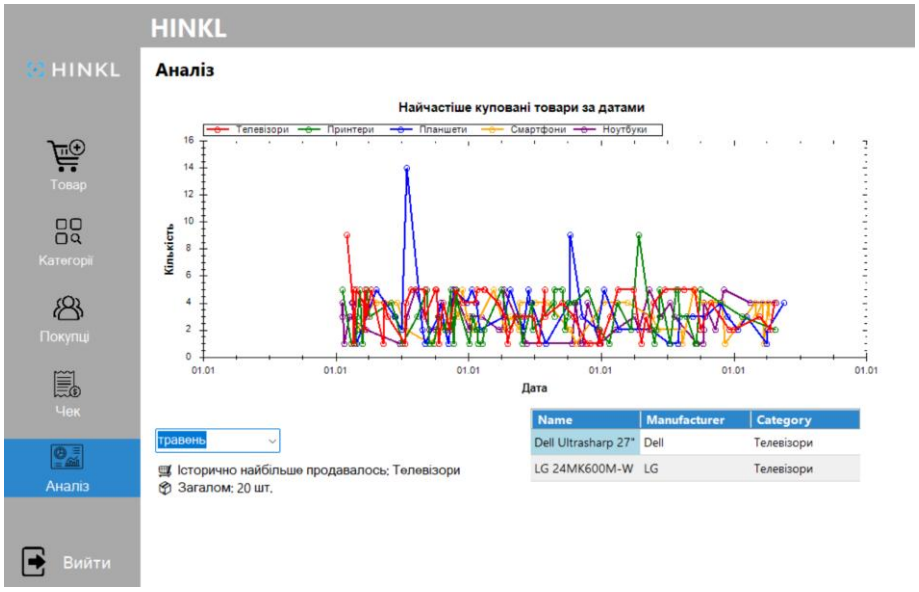

Рис. 3.19. Форма “Аналіз”

Після ініціалізації компонентів інтерфейсу виконується звернення до функції GetData, яка за вказаним SQL-запитом повертає повний набір записів історії продажів із полями SaleDate, Quantity, Category, IsPromotion.

Паралельно отримується таблиця itemsData зі списком товарів разом із їхніми категоріями та виробниками. Веб-запити всередині конструктора виконуються послідовно, що гарантує наявність усіх необхідних даних до подальших етапів.

У наступному кроці заповнюється випадаючий список comboBox1 назвами місяців із CultureInfo поточної культури. Take(12) забезпечує виключення порожнього рядка, властивого .NET-формату, і дає можливість відобразити лише реальні місяці від січня до грудня.

Запускається метод DrawTopProductsOverTimeChart, що відповідає за побудову динамічного графіку обсягів продажів за часом. Спочатку salesData перетворюється на послідовність анонімних об’єктів із властивостями Date (лише дата без часу) та Category, після чого відбувається агрегація сумарної кількості продажів для кожної комбінації дати і категорії. Отримана сукупність групується окремо за категоріями для визначення топ-5 найпопулярніших, які потім виводяться як окремі криві на графіку. Код зображений на рис. 3.20. Використання масиву кольорів гарантує відмінність ліній, а налаштування форматування осі X у вигляді “dd.MM” робить шкалу інтуїтивно зрозумілою.

```
var topCategories = grouped
    .GroupBy(g => g.Category)
    .Select(g => new { Category = g.Key, Total = g.Sum(x => x.Quantity) })
    .OrderByDescending(x => x.Total)
    .Take(5)
    .Select(x => x.Category)
    .ToList();
```

Рис.3.20. Код для групування категорій топ-5

Далі відбувається налаштування ML-моделі в методі SetupMlModel. Всі записи salesData знову обробляються, але цього разу в анонімних об’єктах фіксуються навіть годинникові мітки, які перетворюються на рік і місяць. Потім об’єкти групуються за трьома полями - Year, Month, Category - для обчислення TotalQty, тобто сукупної кількості проданого товару. Результат пакується в список SalesDataPoint, і цей список передається до MLContext для створення IDataView.

У pipeline спочатку виконується кодування категорії методом One-Hot Encoding, потім векторизуються фічі через Concatenate, а фінальним кроком стає FastTreeRegressor із параметрами numberOfLeaves = 20, numberOfTrees = 100 і мінімальною кількістю прикладів на лист 10. Це зображенно на рис. 3.20.

```
var pipeline = mlContext.Transforms.Categorical
        .OneHotEncoding(outputColumnName: "CategoryEncoded", inputColumnName: "Category")
    .Append(mlContext.Transforms.Concatenate(
        "Features",
        "Month",
        "Year",
        "CategoryEncoded"))
    .Append(mlContext.Regression.Trainers.FastTree(
        labelColumnName: "TotalQuantity",
        featureColumnName: "Features",
        numberOfLeaves: 20,
        numberOfTrees: 100,
        minimumExampleCountPerLeaf: 10));
```

Рис. 3.21. Конвеєр обробки даних і навчання моделі регресії у ML.NET

Перевірка File.Exists(_modelPath) визначає, чи слід завантажувати раніше збережену модель чи тренувати нову. Якщо файл SalesForecastModel.zip уже присутній на диску, модель просто завантажується за допомогою mlContext.Model.Load, а PredictionEngine створюється на основі завантаженого трансформера. Це зображенно на рис. 3.22.

У протилежному випадку модель тренується на зібраному IDataView і успішно зберігається через mlContext.Model.Save для подальшого повторного використання.

```
if (File.Exists(_modelPath))
{
    ITransformer loadedModel = mlContext.Model.Load(_modelPath, out var _);
    _predictionEngine = mlContext.Model.CreatePredictionEngine<SalesDataPoint, SalesPrediction>(loadedModel);
}
else
{
    var model = pipeline.Fit(dataView);
    mlContext.Model.Save(model, dataView.Schema, _modelPath);
    _predictionEngine = mlContext.Model.CreatePredictionEngine<SalesDataPoint, SalesPrediction>(model);
}
```

Рис. 3.22. Код для перевірки збереженої моделі

Обробка вибору місяця відбувається в обробнику comboBox1_SelectedIndexChanged. Першим кроком визначається selectedMonth як індекс вибраного елемента плюс одиниця. Якщо модель ще не готова (_predictionEngine == null), формується інформаційне повідомлення з проханням зачекати.

Інакше збираються всі унікальні категорії, які мали продажі в обраному місяці. Якщо їх немає, лейбл lblRecommendation інформує про відсутність даних, а таблиця dataGridViewTopModels очищується. У випадку наявності категорій визначається predictYear - це або максимальний рік із продажами в цьому місяці, або поточний рік. Далі для кожної категорії створюється новий екземпляр SalesDataPoint зі значеннями Month, Year і TotalQuantity = 0. Цей екземпляр подається на вхід PredictionEngine для отримання прогнозованої кількості продажів.

Поруч рахується histSum - історична сума продажів за всі роки для вибраного місяця й категорії. Кожна тріада (Category, Predicted, HistoricalSum) вкладається в список predictions. Алгоритм вибору остаточної категорії передбачає порівняння двох найкращих прогнозів. Якщо різниця менша за 0,05, перевага віддається категорії з більшою історичною сумою; інакше - з більшим прогнозованим значенням. Цей код зображений на рис. 3.23

```
if (sortedByPred.Count > 1 && (bestPred.Predicted - secondPred.Predicted) < 0.05f)
{

    var bestByHist = predictions
        .OrderByDescending(p => p.HistoricalSum)
        .First();
    chosenCategory = bestByHist.Category;
    finalQty = (int)Math.Round(bestByHist.HistoricalSum, 0);
    lblRecommendation.Text = $@"🛒 Історично найбільше продавалось: {chosenCategory}
📦 Загалом: {finalQty} шт.";
}
else
{

    chosenCategory = bestPred.Category;
    finalQty = (int)Math.Round(bestPred.Predicted, 0);
    lblRecommendation.Text = $@"🛒 Рекомендовано замовити: {chosenCategory}
📦 Очікуваний попит: {finalQty} шт.";
}
```

Рис. 3.23. Код для виведення результатів з порівняння двох прогнозів

Після визначення chosenCategory та підрахунку finalQty лейбл lblRecommendation отримує текст із рекомендацією. Наприкінці dataGridViewTopModels заповнюється списком моделей із itemsData, відфільтрованим за chosenCategory, із полями Name, Manufacturer і Category. Цей код зображений на рис. 3.23.

3.9. Тестування

Для забезпечення стабільного функціонування моделі машинного навчання в межах навчального проєкту виникла необхідність перевірити коректність роботи ключових компонентів без залучення зовнішніх фреймворків. Було ухвалено рішення розробити спрощений механізм тестування безпосередньо в WinForms-додатку. У результаті визначено дві основні одиниці верифікації: логіку агрегації даних, яка перетворює початкову структуру DataTable у формат, придатний для навчання моделі, та процес навчання і отримання прогнозів на основі синтетично згенерованих даних із чітко заданими математичними залежностями.

У якості вихідного формату для синтетичних даних створено клас SalesDataPoint із полями Year, Month, Category і TotalQuantity. Така структура прагматично дозволяла реалізувати лінійну залежність цільової змінної від ознак. Для категорії «A» було визначено формулу TotalQuantity = 10 × Year + 5 × Month, а для категорії «B» — TotalQuantity = 8 × Year + 3 × Month. Оскільки залежності жорстко фіксовані, прогнозування за допомогою FastTree-регресора забезпечує майже ідеальні результати, а будь-які відхилення легко виявляються.

Метод GenerateSyntheticData послідовно обходить фіксовані значення років 2022 і 2023, місяців 1, 2 і 3, а також категорії «A» та «B», створюючи таким чином 12 точок даних із заданими формулами. Після формування списку точок здійснюється перевірка: якщо кількість елементів не дорівнює 12, породжується виключення з повідомленням «Очікувалось 12 точок, але отримано інша кількість». Ця проста валідація гарантує, що будь-які зміни в логіці генератора синтетики одразу будуть виявлені. Код цього методу зображений на рис. 3.24.

```
private static void Test_SyntheticData_Count()
{
    var data = GenerateSyntheticData();
    if (data.Count != 12)
        throw new InvalidOperationException($"Очікувалось 12 точок, але отримано інша кількість {data.Count}");
}
```

Рис. 3.24. Код для перевірки кількості точок

Управління даними з DataTable реалізовано методом AggregateFromDataTable, який спочатку перетворює кожний рядок у анонімний об’єкт із полями Year, Month, Category і Quantity. Процес починається із розбору дати з поля SaleDate: якщо формат дати некоректний, такий рядок ігнорується, і цей факт фіксується в журналі. Після трансформації дані групуються за ключем (Year, Month, Category) із підрахунком суми Quantity, яка й стає TotalQuantity.

У результаті формується список об’єктів SalesDataPoint, де кожен елемент містить агреговану суму для відповідної комбінації.

Для верифікації алгоритму агрегації розроблено тест Test_AggregateFromDataTable, який створює DataTable із чотирьох рядків: дві записи з травня 2023 для категорії «X» (Quantity = 10 і 5), одну запис із травня 2023 для категорії «Y» (Quantity = 7) і одну запис із червня 2023 для категорії «X» (Quantity = 3). Очікується три групи: (2023, 5, X) = 15, (2023, 5, Y) = 7 і (2023, 6, X) = 3. Якщо кількість груп відрізняється від трьох або хоча б одна сума є неправильною, тест породжує виключення з детальним повідомленням про невідповідність. Аналогічно, передбачено граничні випадки: якщо у DataTable міститься рядок із некоректним форматом дати, агрегація пропускає його, і тест перевіряє, що неправильний запис не бере участі у формуванні результату; якщо для певного місяця взагалі відсутні дані, метод AggregateFromDataTable повертає порожній список, і це перевіряється тестом Test_NoDataForMonth, який породжує помилку, якщо агрегація повертає непорожній результат.

Після створення генератора синтетичних даних і логіки агрегації сформовано набір статичних методів-тестів, кожен із яких поміщений у блок try/catch для обробки виключень. Головний метод RunAllTests послідовно запускає чотири перевірки: Test_AggregateFromDataTable, Test_SyntheticData_Count (перевірка кількості згенерованих точок), Test_SyntheticPrediction_CategoryA і Test_SyntheticPrediction_CategoryB. Після виконання всіх тестів RunAllTests повертає список рядків із повідомленнями про виявлені помилки. Якщо список виходить порожнім, це свідчить про успішне проходження всіх перевірок.

Інтерфейс WinForms реалізовано в класі Analiz.cs, де додано кнопку «Run Tests». При натисканні ця кнопка викликає RunAllTests. Якщо список помилок порожній, відображається модальне вікно з повідомленням «Усі тести пройдено успішно!». Якщо ж є невідповідності, вміст помилок формується в текстовому звіті всередині багаторядкового елементу TextBox. Кожен рядок звіту має вигляд «Test_<назва> failed: <опис помилки>», що дозволяє користувачеві одночасно бачити всі збої без необхідності закривати кілька вікон. Код для кнопки зображений на рис. 3.25.

```
private void BtnRunTests_Click(object sender, EventArgs e)
{

    var errors = SelfTest.RunAllTests();

    if (!errors.Any())
    {
        MessageBox.Show("Всі тести успішно пройдено ! ☑",
                        "SelfTest Result",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Information);
    }
    else
    {
        // Соберём все ошибки в единый текст
        string msg = "Знайдені помилки:\n\n" + string.Join("\n\n", errors);
        MessageBox.Show(msg,
                        "SelfTest Failed",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
}
```

Рис. 3.24. Код для кнопки тестування

Алгоритм навчання та прогнозування на синтетичних даних реалізовано в методі TrainOnSynthetic. Спочатку створюється MLContext зі статичним параметром seed = 0 для відтворюваності результатів. Синтетичний набір даних завантажується в IDataView за допомогою LoadFromEnumerable. Далі виконується OneHotEncoding стовпця Category, після чого поля Month, Year і закодована категорія об’єднуються в єдиний вектор Features. Для навчання застосовується FastTree-регресор із помірною кількістю дерев і листків, що забезпечує як достатню точність, так і швидкість тренування.

Отриманий ITransformer використовується для створення PredictionEngine<SalesDataPoint, SalesPrediction>, який дозволяє передавати окремий об’єкт SalesDataPoint для отримання одного прогнозу.

У тестах Test_SyntheticPrediction_CategoryA і Test_SyntheticPrediction_CategoryB для категорії «A» береться точка з Year = 2024, Month = 2, Category = «A», а для категорії «B» — Year = 2024, Month = 3, Category = «B». Відповідні очікувані значення обчислюються за формулами 10 × 2024 + 5 × 2 = 20250 для «A» та 8 × 2024 + 3 × 3 = 16201 для «B». Після виклику engine.Predict повертається PredictedQuantity, яке порівнюється з очікуваним значенням із допуском ± 5 %. Якщо прогноз виходить за межі діапазону [20250 − 1012, 20250 + 1012] для категорії «A» або [16201 − 810, 16201 + 810] для категорії «B», тест порушується через виключення з повідомленням «Категорія A: очікувалось близько 20250, але отримано <значення>» або «Категорія B: очікувалось близько 16201, але отримано <значення>». Такий підхід гарантує, що будь-які несанкціоновані зміни в процесі побудови конвеєра або генерації синтетики виявляються негайно.

Для впевненості в механізмі перевірок було проведено експериментальне порушення очікувань: у Test_SyntheticPrediction_CategoryA замість 20250 як очікуване значення було вказано 100. Таким чином реальний прогноз 20250 вийшов за допустимий інтервал, і тест згенерував повідомлення «Test_SyntheticPrediction_CategoryA failed: Категорія A: очікувалось близько 100, але отримано 20250». Цей сценарій підтвердив коректність механізму виявлення відхилень.

На початковому етапі налагодження в Test_SyntheticPrediction_CategoryA після отримання передбаченого значення викликався MessageBox.Show із текстом «Predicted = 20255.32, Expected = 20250». Це дозволяло миттєво оцінити, наскільки прогноз наближений до очікуваного, перед остаточною перевіркою. Після завершення налаштування інформаційні діалоги було видалено, залишивши лише фінальну перевірку з виключенням у разі невідповідності.

У фінальній структурі проєкту передбачено такі файли: SalesDataPoint.cs, який містить опис моделі вхідних даних з полями Year, Month, Category та TotalQuantity; SalesPrediction.cs, що визначає структуру результату регресора з полем PredictedQuantity; SelfTest.cs, у якому зібрано набір статичних методів для перевірки агрегування, генерації синтетичних даних і прогнозування; Analiz.cs, де реалізовано кнопку «Run Tests».

Усі тести виконуються в оперативній пам’яті без залучення зовнішніх бібліотек, що забезпечує високий рівень автономності та простоту використання в навчальному середовищі. Такий підхід дозволяє чітко контролювати алгоритми обробки даних і навчання моделі машинного навчання: будь-які зміни в коді, які порушують очікувану поведінку, одразу виявляються завдяки жорстким умовам перевірок, а відсутність зовнішніх залежностей робить систему оптимальною для освітніх проєктів.

3.10. Висновок до розділу 3

У межах третього розділу було реалізовано прикладну частину програмного продукту для автоматизованої обробки даних онлайн-магазину, що поєднує інструменти баз даних, графічного інтерфейсу та алгоритмів машинного навчання. Процес розробки охоплював побудову ключових форм, які забезпечують повноцінне функціонування системи: від автентифікації користувача до управління товарами, категоріями, клієнтами, чеками та прогнозами закупівель.

На початковому етапі було визначено логіку інтерфейсної взаємодії, що враховує зручність користування, мінімізацію помилок під час введення даних і контроль за коректністю транзакцій.

Застосування C# у середовищі WinForms дало змогу створити інтерактивні форми з автоматичним доповненням, стилізованими таблицями та функціоналом, орієнтованим на потреби операторів і менеджерів онлайн-магазину.

У підсистемі Billing реалізовано логіку генерації чеків, збереження транзакцій у базі даних SQL Server та забезпечення цілісності інформації шляхом валідації на

рівні інтерфейсу. Особливої уваги заслуговує форма Analiz, у якій інтегровано модуль машинного навчання ML.NET для здійснення прогнозу закупівель. Побудована модель виконує передбачення обсягів замовлень на основі історичних даних, що дозволяє оптимізувати товарні залишки, підвищити точність закупівель і знизити ризики дефіциту чи надлишків.

Архітектура програми побудована за модульним принципом, що передбачає незалежну обробку різних сутностей: товарів, категорій, клієнтів і транзакцій. Такий підхід забезпечує гнучкість масштабування, спрощує тестування окремих компонентів і дозволяє легко адаптувати систему до нових бізнес-вимог.

Завдяки комплексному підходу до реалізації програмного забезпечення було досягнуто стабільної роботи всіх модулів.

Інтеграція бази даних з алгоритмами машинного навчання забезпечила системі аналітичну глибину, а реалізація багатоформного інтерфейсу надала користувачам інструменти для зручного управління всіма аспектами торгівлі.

Тестування в програмі інтегроване дуже просто. Будь-яка зміна в коді, що порушує очікування, негайно виявляється завдяки умовам перевірки. Простота реалізації і мінімальні залежності роблять такий механізм оптимальним для навчального середовища, де важлива впевненість у працездатності ключових компонентів

Загалом створений додаток демонструє приклад ефективного впровадження сучасних IT-рішень у процеси електронної комерції.

ВИСНОВКИ

У дипломній роботі на тему «Автоматизована база даних онлайн-магазину з використанням методів машинного навчання» було проведено комплексне дослідження, спрямоване на створення програмного забезпечення для підвищення ефективності обробки замовлень та управління даними в електронній комерції. В роботі розв’язано актуальну інженерну проблему: автоматизацію процесів прийняття рішень щодо закупівель в умовах динамічної зміни попиту за рахунок впровадження інтелектуальних моделей на базі машинного навчання.

Аналіз предметної області дозволив виявити основні виклики, що стоять перед сучасними онлайн-магазинами: необхідність оперативного реагування на зміни попиту, обробка великих обсягів даних, потреба у прогнозуванні продажів, оптимізація логістичних процесів. Було обґрунтовано актуальність автоматизованих інформаційних систем у сфері електронної комерції як засобу забезпечення стабільного зростання бізнесу при зниженні експлуатаційних витрат.

У теоретичній частині дослідження проведено огляд функціонування онлайн-магазинів як складних інформаційних систем, що поєднують клієнтську, серверну, аналітичну та логістичну складові. Розглянуто сучасні архітектурні підходи до побудови таких систем, використання реляційних і нереляційних баз даних, механізмів авторизації, масштабування, а також засоби аналітики на базі ML.

Здійснено аналіз популярних рішень для електронної комерції (nopCommerce, Smartstore, Virto Commerce) і визначено, що їх ключовими перевагами є модульність, підтримка інтеграції з ERP/CRM-системами, а також вбудовані засоби маркетингу та аналітики. Однак більшість з них потребують складного налаштування або не адаптовані до конкретних бізнес-задач малого або середнього бізнесу, що обґрунтовує доцільність створення кастомізованої системи.

У практичній частині реалізовано програмне забезпечення, яке забезпечує: зберігання і обробку інформації про товари, замовлення, клієнтів; інтерфейс керування даними, реалізований на базі WinForms; інтеграцію з Microsoft SQL Server для надійного управління даними; навчання та використання моделі регресії FastTree через ML.NET для прогнозування майбутніх обсягів продажу.

Модель машинного навчання була навчена на історичних даних продажів, з використанням змінних «місяць», «рік» та «категорія товару». Результати тестування моделі показали достатній рівень точності: похибка прогнозу не перевищує 15–20% у контрольних наборах, що є прийнятним для прикладних завдань середнього бізнесу. Це дозволяє підприємству робити обґрунтовані закупівельні рішення, зменшити дефіцит або перевищення запасів і, відповідно, оптимізувати витрати.

Програма протестована в умовах обмеженої кількості користувачів та демонструє стабільну роботу з базою даних, швидкий час відповіді інтерфейсу, коректну авторизацію доступу та можливість масштабування в разі подальшої розробки (наприклад, через переведення логіки в мікросервісну архітектуру чи хмарну інфраструктуру).Практичне значення роботи полягає в:створенні прикладного ПЗ, яке може бути впроваджено в реальний онлайн-магазин без значних витрат; можливості адаптації системи під специфіку конкретного бізнесу; інтеграції прогнозних алгоритмів у звичайний процес управління продажами; зниженні витрат на персонал та збільшенні точності обліку.

Наукова новизна отриманих результатів полягає в:застосуванні методів машинного навчання для прогнозування закупівель на базі історичних даних у форматі реального часу; побудові цілісної системи, що поєднує класичне ПЗ та ML-моделі у єдиному середовищі .NET; використанні ML.NET як інструмента прямої інтеграції алгоритмів у програмний продукт без залучення зовнішніх ML-платформ.

Рекомендації щодо використання результатів:

- запропоноване ПЗ доцільно впроваджувати в невеликих та середніх онлайн-магазинах як автономне або додаткове рішення до існуючої ERP-системи;

- система може бути основою для подальшої розробки web-версії або мобільного клієнта;
- передбачено можливість розширення функціональності за рахунок додавання CRM-модуля, логістичного трекінгу або розсилок на основі аналітики.

Таким чином, усі завдання дипломної роботи виконано повністю, поставлену мету досягнуто, а результати мають як наукову новизну, так і практичну значущість. Розроблена система демонструє перспективність використання машинного навчання в обробці комерційних даних та може слугувати платформою для подальших досліджень і впроваджень у сфері електронної торгівлі.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. .NET Blog. Офіційний блог платформи .NET від Microsoft: веб-сайт. URL: https://devblogs.microsoft.com/dotnet/ (дата звернення: 24.05.2025).

2. .NET Framework. Офіційна документація Microsoft: веб-сайт. URL: https://docs.microsoft.com/dotnet/framework/ (дата звернення: 24.05.2025).

3. nopCommerce. Відкрита платформа електронної комерції: веб-сайт. URL: https://www.nopcommerce.com (дата звернення: 29.04.2025).

4. Smartstore. Масштабована система електронної комерції: веб-сайт. URL: https://www.smartstore.com (дата звернення: 29.04.2025).

5. Virto Commerce. Корпоративна платформа для електронної комерції : веб-сайт. URL: https://virtocommerce.com (дата звернення: 29.04.2025).

6. Microsoft SQL Server. Документація СУБД: веб-сайт. URL: https://docs.microsoft.com/en-us/sql/sql-server/ (дата звернення: 25.04.2025).

7. ASP.NET Community Standup. Офіційні дискусії та анонси спільноти ASP.NET: веб-сайт. URL: https://live.asp.net/ (дата звернення: 24.05.2025).

8. Common Language Runtime (CLR). Основи та архітектура CLR від Microsoft: веб-сайт. URL:https://docs.microsoft.com/dotnet/standard/clr (дата звернення: 24.05.2025).

9. Microsoft Docs. “Assembly Loading and Binding in the CLR”: веб-сайт.URL: https://docs.microsoft.com/dotnet/standard/assembly/ (дата звернення: 28.05.2025).

10. Microsoft Docs. “C# Guide”: офіційна документація з мовою програмування C#: веб-сайт.URL: https://docs.microsoft.com/dotnet/csharp/ (дата звернення: 24.05.2025).

11. Microsoft Docs. “C# Language Reference”: повний довідник по синтаксису та особливостям C#: веб-сайт.URL: https://docs.microsoft.com/dotnet/csharp/language-reference/ (дата звернення: 27.05.2025).

12. C# Programming Guide. ".NET Guide": практичні приклади та рекомендації по C#: веб-сайт.URL: https://docs.microsoft.com/dotnet/csharp/programming-guide/ (дата звернення: 25.05.2025).

13. Visual Studio. Офіційна документація Microsoft: веб-сайт.URL: https://docs.microsoft.com/visualstudio/ (дата звернення: 26.05.2025).

14. Visual Studio Blog. Офіційний блог команди Visual Studio: веб-сайт.URL: https://devblogs.microsoft.com/visualstudio/ (дата зверення: 26.05.2025).

15. Pluralsight. "Visual Studio 2022 Fundamentals": онлайн-курс.URL: https://www.pluralsight.com/courses/visual-studio-2022-fundamentals (дата зверення: 28.05.2025).

16. Microsoft Docs. "Windows Forms Overview": офіційна документація Microsoft: веб-сайт. URL: https://docs.microsoft.com/dotnet/desktop/winforms/ (дата звернення: 28.05.2025).

17. Microsoft Docs. "Walkthrough: Create a Windows Forms application": офіційний гайд Microsoft: веб-сайт. URL: https://docs.microsoft.com/dotnet/desktop/winforms/quickstart (дата зверення: 23.05.2025).

18.CodeProject. "Windows Forms Tutorials and Articles": спільнота розробників, практичні приклади: веб-сайт. URL: https://www.codeproject.com/KB/windows/ (дата зверення: 29.05.2025).

19. Microsoft Docs. "ML.NET Overview": офіційна документація Microsoft: веб-сайт. URL: https://docs.microsoft.com/dotnet/machine-learning/ (дата звернення: 24.05.2025).

20. Microsoft Docs. "ML.NET Tutorial: Time Series Forecasting with SSA": веб-сайт. URL: https://docs.microsoft.com/dotnet/machine-learning/tutorials/time-series-ssa (дата звернення: 24.05.2025).

21. Microsoft Blog. "ML.NET Community Standup & Updates": офіційний блог Microsoft: веб-сайт. URL: https://devblogs.microsoft.com/dotnet/category/ml-net/ (дата зверення: 24.05.2025).