МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи освітнього ступеня «фаховий молодший бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
на тему:

«Інформаційна система «Магазин автозапчастин»»

Виконав здобувач освіти групи П-41
Кемський Богдан Олександрович

Керівник роботи:
Устименко Людмила Миколаївна

Рецензент:
Устименко Ярослав Іванович

Житомир – 2025

Зміст

РЕФЕРАТ

Дипломна робота на тему «Інформаційна система «Магазин автозапчастин»» присвячена розробці програмного забезпечення для автоматизації бізнес-процесів невеликих магазинів автозапчастин. Обсяг роботи становить 65 сторінок, включає 35 рисунків, 1 додаток і 18 літературних джерел.

Метою роботи є створення настільного застосунку на базі Windows Forms, .NET 8 і SQLite для управління товарами, замовленнями, клієнтами та категоріями. Проведено аналіз аналогічних систем («1С:Торгівля і склад», «AutoIntellect», «Торгсофт»), що дозволило визначити вимоги до економічного та простого у використанні рішення. Система спроєктована з використанням модульної архітектури, включає п’ять форм для роботи з даними та базу даних SQLite, нормалізовану до третьої форми. Реалізовано алгоритми CRUD-операцій, пошуку та транзакційної обробки, що забезпечують швидкодію для невеликих обсягів даних. Інтерфейс користувача є інтуїтивно зрозумілим, з навігаційним меню та обробкою винятків.

Робота містить інструкцію для користувачів і підтверджує практичну цінність системи для малого бізнесу. Результати дозволяють автоматизувати облік і підвищити ефективність роботи магазинів автозапчастин.

Ключові слова: інформаційна система, магазин автозапчастин, Windows Forms, .NET 8, SQLite, автоматизація, база даних.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

1. Скорочення

Скорочення	Повна назва	Пояснення
БД	База даних	Організована структура для зберігання та управління даними
СУБД	Система управління базами даних	Програмне забезпечення для взаємодії з БД (у роботі - SQLite)
API	Application Programming Interface	Інтерфейс програмного забезпечення для взаємодії компонентів системи
UI	User Interface	Інтерфейс користувача
DTO	Data Transfer Object	Об'єкт передачі даних між рівнями архітектури системи
ORM	Object-Relational Mapping	Технологія зв'язку об'єктів програми з даними БД

2. Технічні терміни

Термін	Пояснення
Транзакція	Атомарна операція з даними, що виконується повністю або не виконується взагалі
Нормалізація даних	Процес оптимізації структури БД для зменшення надмірності
Репозиторій	Шаблон проектування для інкапсуляції логіки доступу до даних
Ін'єкція залежностей	Патерн проектування для реалізації слабкозв'язаних архітектур

3. Позначення технологій

Позначення	Пояснення
.NET 8	Платформа для розробки програмного забезпечення
Windows Forms	Графічна бібліотека для створення десктопних додатків
SQLite	Вбудована реляційна система управління базами даних
Entity Framework	Об'єктно-орієнтована технологія доступу до даних

4. Позначення сутностей системи

Позначення	Пояснення
Товар	Автозапчастина, що реалізується магазином
Категорія	Класифікаційна група товарів
Клієнт	Особа, що здійснює замовлення
Замовлення	Транзакція купівлі товарів

5. Додаткові позначення

Позначення	Пояснення
UML	Unified Modeling Language - мова графічного моделювання
ISBN	International Standard Book Number - ідентифікатор джерела
ДСТУ	Державний стандарт України

ВСТУП

Сучасний розвиток інформаційних технологій відкриває широкі можливості для автоматизації бізнес-процесів у різних галузях, зокрема в торгівлі. Однією з таких галузей є продаж автозапчастин, де ефективне управління асортиментом, замовленнями, клієнтами та складськими запасами є ключовим фактором успіху. Магазини автозапчастин стикаються з необхідністю швидкого оброблення інформації, точного обліку товарів і забезпечення якісного обслуговування клієнтів. Ручне ведення обліку та управління процесами часто призводить до помилок, затримок і зниження продуктивності. У зв’язку з цим розробка інформаційних систем, які автоматизують основні бізнес-процеси, є актуальною задачею.

Метою дипломної роботи є створення інформаційної системи «Магазин автозапчастин» на базі технології Windows Forms із використанням платформи .NET 8 та бази даних SQLite. Система призначена для автоматизації процесів управління асортиментом товарів, оброблення замовлень, ведення бази клієнтів і категорій товарів, а також забезпечення зручного інтерфейсу для працівників магазину. Використання Windows Forms дозволяє створити інтуїтивно зрозумілий графічний інтерфейс, адаптований для настільних застосунків, а SQLite забезпечує легковагу та ефективну базу даних для зберігання інформації. .NET 8, як сучасна платформа, надає надійні інструменти для розробки, забезпечуючи високу продуктивність і підтримку новітніх стандартів програмування.

Актуальність роботи зумовлена зростанням попиту на автоматизовані рішення в роздрібній торгівлі, зокрема в сегменті автозапчастин, де точність і швидкість оброблення даних безпосередньо впливають на задоволеність клієнтів і прибутковість бізнесу. Розроблена система дозволить оптимізувати облік товарів, спростити створення та управління замовленнями, а також забезпечити гнучкий доступ до інформації про клієнтів і категорії товарів.

У процесі виконання роботи було проаналізовано предметну область, визначено основні вимоги до системи, спроєктовано архітектуру програми,

реалізовано функціонал із використанням об’єктно-орієнтованого підходу та протестовано систему на відповідність поставленим завданням. Робота має практичну цінність, оскільки створена інформаційна система може бути використана в реальних магазинах автозапчастин для підвищення ефективності їхньої діяльності.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка інформаційної системи «Магазин автозапчастин» спрямована на створення програмного забезпечення для автоматизації основних бізнес-процесів у сфері роздрібної торгівлі автозапчастинами. Основна задача полягає в реалізації настільного застосунку, який забезпечить зручне управління асортиментом товарів, оброблення замовлень, ведення бази клієнтів і категорій товарів, а також облік складських запасів. Система має надавати інтуїтивно зрозумілий графічний інтерфейс, бути ефективною в обробці даних і адаптованою до потреб невеликих і середніх магазинів автозапчастин.

Для досягнення поставленої мети необхідно вирішити такі завдання: спроєктувати архітектуру системи з урахуванням принципів модульності та розширюваності; реалізувати функціонал для створення, редагування та видалення записів про товари, клієнтів, категорії та замовлення; забезпечити збереження даних у локальній базі даних із можливістю швидкого доступу та пошуку; розробити зручний інтерфейс користувача, який мінімізує час на освоєння системи. Вибір технологій для розробки має враховувати вимоги до продуктивності, простоти розгортання та підтримки, а також відповідність сучасним стандартам програмування.

Інформаційна система повинна відповідати функціональним вимогам, серед яких автоматизація обліку товарів і замовлень, підтримка пошуку та фільтрації даних, а також генерація звітів про стан складських запасів. Нефункціональні вимоги включають стабільність роботи, швидкодію при обробці великих обсягів даних і можливість легкого оновлення системи в майбутньому. У процесі розробки необхідно провести аналіз предметної області, щоб визначити ключові сутності (товари, клієнти, замовлення, категорії) та їх взаємозв’язки, а також врахувати особливості роботи магазинів автозапчастин, такі як необхідність швидкого пошуку запчастин за назвою чи категорією та контроль наявності товарів на складі.

Вибір технологічного стеку для реалізації системи базується на потребі створення надійного настільного застосунку з локальним зберіганням даних. Технологія Windows Forms на платформі .NET 8 обрана як основа для розробки графічного інтерфейсу завдяки її простоті, широкій підтримці та наявності готових компонентів для створення форм. SQLite використовується як база даних через її легковагість, відсутність потреби в сервері та достатню продуктивність для обробки даних невеликих магазинів. Платформа .NET 8 забезпечує сучасні інструменти розробки, підтримку об’єктно-орієнтованого програмування та високу продуктивність, що робить її оптимальним вибором для реалізації поставлених завдань.

1.2. Огляд та порівняння аналогічних систем.

Для розробки інформаційної системи «Магазин автозапчастин» було проведено аналіз аналогічних систем, які використовуються для автоматизації бізнес-процесів у сфері роздрібної торгівлі автозапчастинами. Розглянуто кілька популярних рішень, що застосовуються в Україні та за кордоном, з метою визначення їхніх сильних і слабких сторін, а також для формулювання вимог до власної системи. Серед проаналізованих систем виділено три основні: «1С:Торгівля і склад», «AutoIntellect» та «Торгсофт».

Система «1С:Торгівля і склад» є одним із найпоширеніших рішень для автоматизації торгівлі в Україні. Вона підтримує облік товарів, управління замовленнями, клієнтською базою та складськими запасами. Програма має гнучкі налаштування, можливість інтеграції з бухгалтерськими системами та підтримку звітності. Однак її складний інтерфейс і необхідність тривалого навчання ускладнюють використання для невеликих магазинів. Крім того, висока вартість ліцензії та залежність від платформи 1С можуть бути обмеженнями для малого бізнесу.

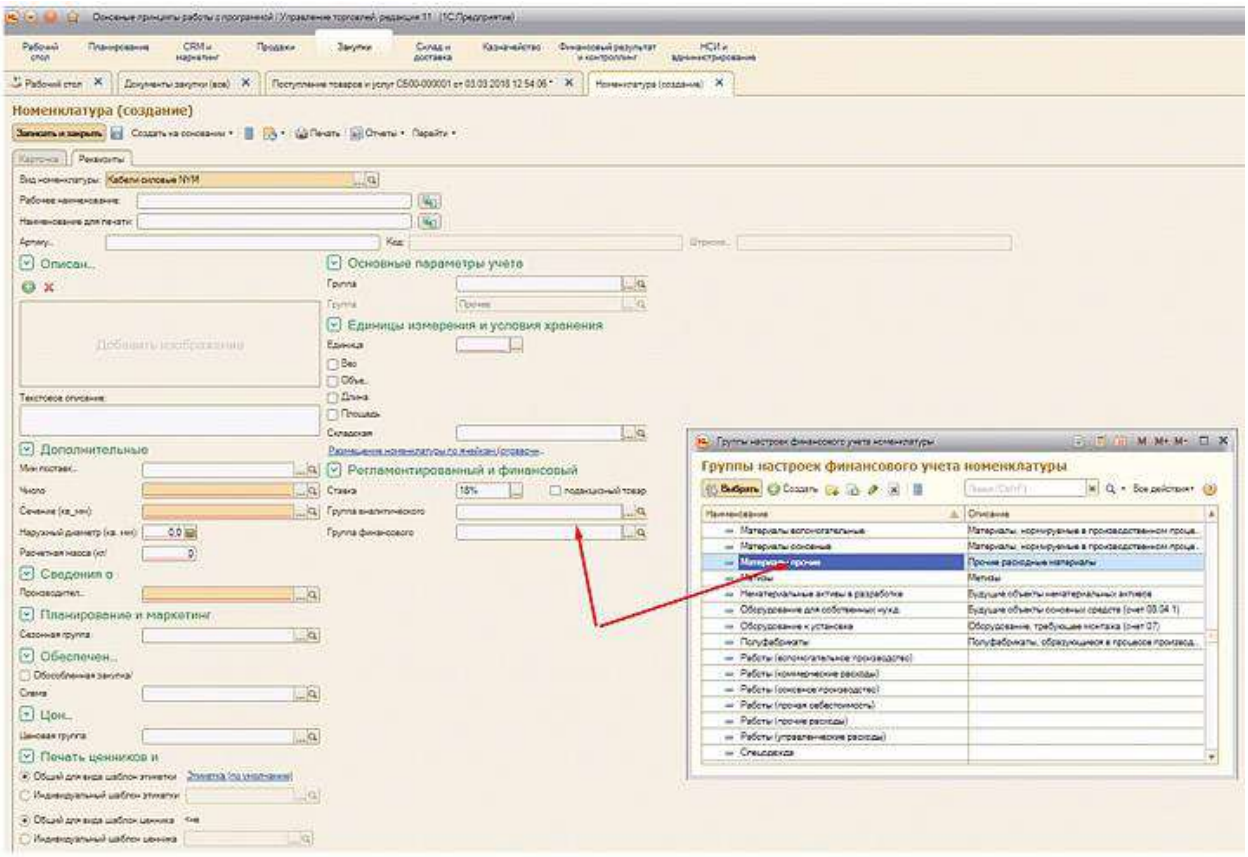

Рисунок 1.2.1 – Система «1С:Торгівля і склад»

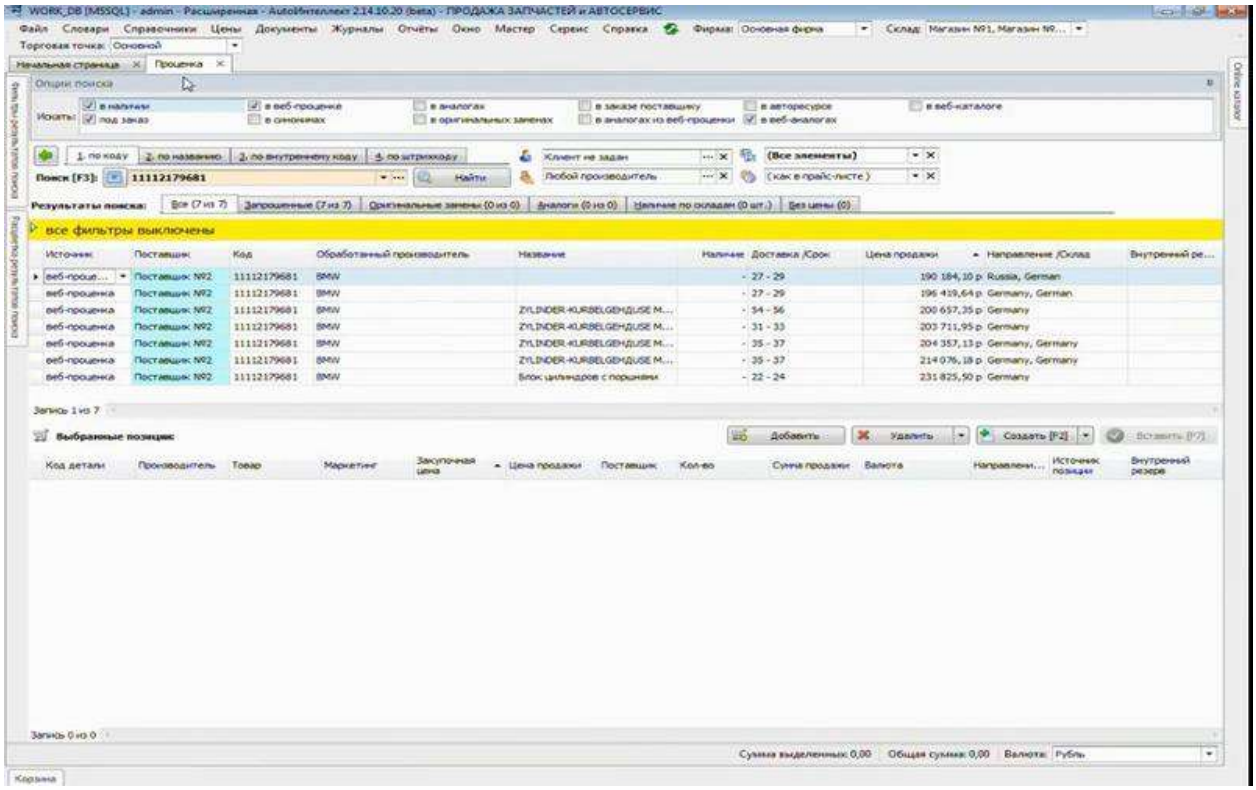

Рисунок 1.2.2 – AutoIntellect

«AutoIntellect» — спеціалізована система для магазинів автозапчастин, яка пропонує функції пошуку запчастин за VIN-кодом, каталогами та категоріями, а також управління замовленнями й інтеграцію з постачальниками. Система має зручний інтерфейс і адаптована до специфіки торгівлі автозапчастинами. Проте її орієнтація на великі мережі магазинів робить її менш доступною для малих підприємств через високу ціну та складність розгортання. Також система потребує постійного підключення до інтернету для роботи з онлайн-каталогами.

«Торгсофт» — українське рішення для автоматизації торгівлі, яке підходить для невеликих і середніх магазинів. Воно забезпечує облік товарів, продажів, клієнтів і складських операцій, а також має простий інтерфейс і відносно низьку вартість. Проте функціонал для специфічних потреб магазинів автозапчастин, таких як пошук за технічними характеристиками чи інтеграція з каталогами запчастин, обмежений. Крім того, система може бути менш гнучкою для масштабування порівняно з іншими рішеннями.

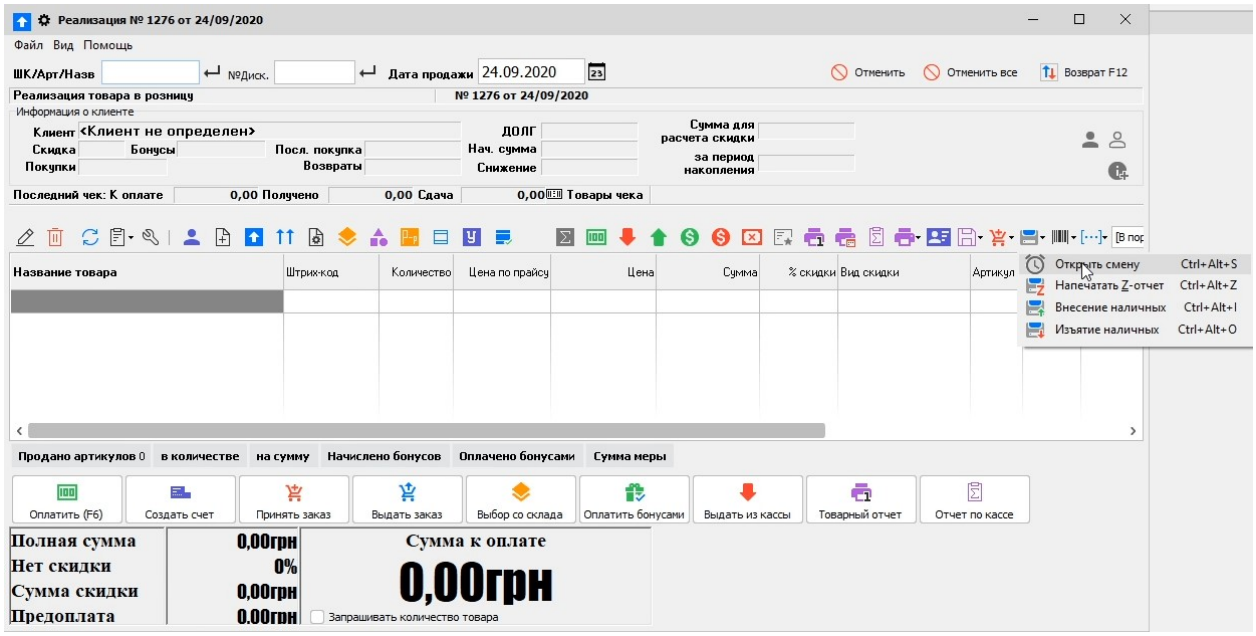

Рисунок 1.2.3 – Торгсофт

Порівняння цих систем показало, що більшість із них орієнтовані на великі підприємства або мають обмежений функціонал для специфічних потреб магазинів автозапчастин. «1С:Торгівля і склад» є надмірно складною і

дорогою для малого бізнесу, «AutoIntellect» потребує значних ресурсів і підключення до інтернету, а «Торгсофт» не повною мірою відповідає вимогам спеціалізованого обліку автозапчастин. Розроблювана інформаційна система «Магазин автозапчастин» на базі Windows Forms, .NET 8 і SQLite спрямована на подолання цих недоліків. Вона пропонує простий і зрозумілий інтерфейс, локальне зберігання даних без залежності від інтернету, підтримку базового функціоналу для управління товарами, замовленнями, клієнтами та категоріями, а також низькі витрати на розгортання й обслуговування. Система розробляється з урахуванням потреб невеликих магазинів, забезпечуючи достатню функціональність і можливість подальшого розширення.

1.3. Вибір та обґрунтування технологій розробки

Вибір технологій для розробки інформаційної системи «Магазин автозапчастин» ґрунтується на вимогах до функціональності, продуктивності, простоти розгортання та підтримки, а також на специфіці цільової аудиторії — невеликих і середніх магазинів автозапчастин. Після аналізу предметної області та порівняння аналогічних систем було визначено, що система має бути настільним застосунком із локальним зберіганням даних, зручним інтерфейсом і мінімальними витратами на впровадження. Для реалізації обрано технологічний стек, що включає Windows Forms, платформу .NET 8 і базу даних SQLite.

Windows Forms обрано як основну технологію для створення графічного інтерфейсу користувача завдяки її простоті, зрілості та широкій підтримці в екосистемі .NET. Ця технологія дозволяє швидко розробляти настільні застосунки з готовими компонентами для форм, таблиць і елементів керування, що відповідає потребам системи в інтуїтивно зрозумілому інтерфейсі. Windows Forms забезпечує стабільну роботу на платформі Windows, яка є основною операційною системою для більшості магазинів автозапчастин. Порівняно з іншими технологіями, такими як WPF чи UWP,

Windows Forms має нижчий поріг входження для розробки та не потребує складного налаштування, що є перевагою для проєкту з обмеженими ресурсами.

Платформа .NET 8 обрана як основа для розробки завдяки її сучасним можливостям, високій продуктивності та підтримці об’єктно-орієнтованого програмування. .NET 8 пропонує розширені бібліотеки для роботи з даними, файлами та інтерфейсом, а також інструменти для створення модульної архітектури системи. Ця платформа забезпечує крос-версійну сумісність і можливість оновлення системи в майбутньому без значних змін у коді. Порівняно з іншими платформами, такими як Java чи Python, .NET 8 має переваги у швидкості виконання та інтеграції з Windows Forms, що робить її оптимальним вибором для настільного застосунку.

Для зберігання даних обрано SQLite — легковагу реляційну базу даних, яка не потребує окремого серверного програмного забезпечення. SQLite забезпечує достатню продуктивність для обробки даних невеликих магазинів, підтримує SQL-запити для пошуку та фільтрації, а також дозволяє зберігати всю базу даних у єдиному файлі, що спрощує резервне копіювання та перенесення даних. Порівняно з іншими базами даних, такими як Microsoft SQL Server чи MySQL, SQLite не вимагає додаткових витрат на серверне обладнання чи ліцензії, що відповідає вимогам економічності. Її простота інтеграції з .NET через бібліотеки, такі як Microsoft.Data.Sqlite, додатково полегшує розробку.

Обґрунтування вибору технологій базується на їхній відповідності ключовим вимогам проєкту: створення надійного, економічного та простого у використанні рішення для невеликих магазинів автозапчастин. Windows Forms забезпечує швидку розробку зручного інтерфейсу, .NET 8 гарантує продуктивність і гнучкість, а SQLite дозволяє ефективно працювати з даними без додаткових витрат. Такий технологічний стек дає змогу реалізувати всі необхідні функції системи, включаючи облік товарів, управління

замовленнями, клієнтською базою та категоріями, а також забезпечує можливість подальшого розширення функціоналу в разі потреби.

Висновки до розділу 1

У результаті аналізу предметної області та вимог до інформаційної системи «Магазин автозапчастин» було сформульовано основну задачу розробки — створення настільного застосунку для автоматизації управління асортиментом товарів, замовленнями, клієнтською базою та категоріями. Система має забезпечувати зручний інтерфейс, швидке оброблення даних і мінімальні витрати на впровадження, що відповідає потребам невеликих і середніх магазинів автозапчастин.

Огляд аналогічних систем, таких як «1С:Торгівля і склад», «AutoIntellect» і «Торгсофт», показав, що більшість наявних рішень або надмірно складні й дорогі для малого бізнесу, або не повною мірою відповідають специфіці торгівлі автозапчастинами. Розроблювана система спрямована на подолання цих недоліків шляхом створення економічного рішення з локальним зберіганням даних і простим інтерфейсом.

Для реалізації системи обрано технологічний стек, що включає Windows Forms, .NET 8 і SQLite. Windows Forms забезпечує швидку розробку зручного графічного інтерфейсу, .NET 8 гарантує високу продуктивність і гнучкість архітектури, а SQLite дозволяє ефективно працювати з даними без додаткових витрат на серверне обладнання. Вибір цих технологій обґрунтовано їхньою відповідністю функціональним і нефункціональним вимогам проєкту, а також можливістю створення стабільного й економічного рішення для цільової аудиторії. Проведений аналіз підтвердив доцільність використання обраного технологічного стеку для реалізації поставлених завдань.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Розробка інформаційної системи «Магазин автозапчастин» потребує вибору алгоритмів, які забезпечують ефективне виконання основних функцій: управління асортиментом товарів, оброблення замовлень, ведення бази клієнтів і категорій, а також пошуку даних. Враховуючи специфіку системи, яка працює з відносно невеликими обсягами даних у локальній базі SQLite, основна увага приділялася простоті реалізації, стабільності та швидкодії алгоритмів. У цьому підрозділі розглянуто ключові алгоритми, їхню ефективність і обґрунтування вибору.

Оброблення даних (CRUD-операції)

Для управління даними про товари, клієнтів, категорії та замовлення використано стандартні операції створення, читання, оновлення та видалення (CRUD). Ці операції реалізовано через прямий доступ до бази даних SQLite за допомогою SQL-запитів, які оптимізовано для швидкого виконання. Наприклад, для створення замовлення застосовується транзакційна обробка, що забезпечує атомарність операцій додавання замовлення та його позицій. Алгоритм створення замовлення включає послідовне виконання таких кроків: перевірка наявності клієнта, збереження заголовка замовлення, додавання позицій замовлення та оновлення складських запасів. Використання транзакцій дозволяє уникнути неконсистентності даних у разі збою, а часова складність алгоритму становить O(n), де n — кількість позицій у замовленні, оскільки кожна позиція обробляється окремо.

Для читання даних, наприклад, при завантаженні списку товарів чи замовлень, використовуються SQL-запити з індексацією за первинними ключами, що забезпечує швидкий доступ до записів. SQLite автоматично оптимізує такі запити, а часова складність операції читання наближається до O(1) для пошуку за ідентифікатором і O(m) для вибірки всіх записів, де m — кількість записів у таблиці. Оновлення та видалення даних також виконуються за ідентифікаторами, що забезпечує аналогічну ефективність.

Пошук і фільтрація

Функція пошуку реалізована для швидкого знаходження замовлень, товарів, клієнтів і категорій за ключовими полями, такими як назва, ідентифікатор чи статус. Для цього використано SQL-запити з умовою LIKE для текстового пошуку, наприклад, SELECT * FROM Orders WHERE Status LIKE '%searchTerm%'. Щоб підвищити ефективність, у базі даних створено індекси для полів, які часто використовуються в пошуку, таких як назва товару та статус замовлення. Це знижує часову складність пошуку з O(m) до O(log m) для великих наборів даних. Однак, враховуючи невеликий обсяг даних у цільових магазинах (до кількох тисяч записів), навіть неоптимізований пошук забезпечує прийнятну швидкодію.

Для фільтрації замовлень за статусом чи клієнтом застосовується алгоритм лінійного відбору на основі результатів SQL-запиту. Дані фільтруються на рівні клієнтського коду (у пам’яті), що має часову складність O(m), але завдяки малому обсягу даних затримки не виникають. У разі потреби масштабування системи можна оптимізувати фільтрацію шляхом перенесення умов на рівень SQL-запиту, що зменшить обсяг даних, переданих із бази.

Обчислення загальної вартості замовлення

Алгоритм обчислення загальної вартості замовлення використовується для відображення суми в інтерфейсі та збереження в базі даних. Він полягає в ітеративному підсумовуванні вартості кожної позиції замовлення, де вартість позиції обчислюється як добуток кількості товару на ціну за одиницю. Часова складність цього алгоритму становить O(n), де n — кількість позицій у замовленні. Оскільки кількість позицій у замовленні зазвичай невелика (до кількох десятків), алгоритм є достатньо ефективним. Для забезпечення точності обчислень використано тип даних decimal, який запобігає помилкам округлення, характерним для типів із плаваючою комою.

Оновлення складських запасів

При створенні замовлення реалізовано алгоритм оновлення складських запасів, який перевіряє наявність товару та зменшує його кількість на складі.

Алгоритм включає перевірку умови Product.Quantity >= OrderItem.Quantity для кожної позиції та оновлення запису товару в базі даних. Часова складність становить O(n), де n — кількість позицій у замовленні, оскільки для кожної позиції виконується окремий запит до бази. Для підвищення ефективності використано транзакції, що мінімізують кількість операцій із базою даних і забезпечують консистентність даних.

Оцінка ефективності

Обрані алгоритми є оптимальними для системи з невеликими обсягами даних, оскільки вони прості в реалізації та не потребують складних структур даних чи оптимізацій. Використання індексів у SQLite забезпечує швидкий доступ до даних, а транзакційна обробка гарантує надійність операцій. Для великих обсягів даних (понад 10 тисяч записів) може знадобитися додаткова оптимізація, наприклад, кешування даних у пам’яті чи пакетна обробка запитів, але для цільової аудиторії (невеликі магазини) поточні алгоритми забезпечують достатню швидкодію та стабільність.

Обґрунтування вибору алгоритмів базується на їхній відповідності вимогам системи: простота реалізації, мінімальні витрати ресурсів і достатня продуктивність для обробки даних у реальному часі. Використання стандартних SQL-запитів і транзакцій у поєднанні з лінійними алгоритмами для обробки невеликих наборів даних дозволяє досягти балансу між ефективністю та легкістю підтримки коду.

2.2. База даних

База даних інформаційної системи «Магазин автозапчастин» є ключовим компонентом, що забезпечує зберігання, оброблення та швидкий доступ до даних про товари, клієнтів, замовлення та категорії. Для реалізації обрано SQLite — легковагу реляційну базу даних, яка відповідає вимогам системи щодо простоти, економічності та ефективності. У цьому підрозділі

описано структуру бази даних, її проєктування, нормалізацію та особливості реалізації.

База даних має забезпечувати надійне зберігання інформації про основні сутності системи: товари, клієнтів, замовлення, позиції замовлень і категорії товарів. Основні вимоги включають підтримку CRUD-операцій (створення, читання, оновлення, видалення), швидкий пошук за ключовими полями, збереження цілісності даних і можливість роботи без серверного програмного забезпечення. SQLite обрано завдяки її здатності працювати з локальними файлами бази даних, що усуває потребу в додатковому сервері, спрощує розгортання та знижує витрати для невеликих магазинів автозапчастин.

На етапі проєктування було визначено основні сутності та їхні взаємозв'язки. База даних складається з п'яти таблиць: Products, Customers, Orders, OrderItems і Categories. Структура бази даних розроблена з урахуванням принципів нормалізації для уникнення надмірності даних і забезпечення цілісності.

1. **Таблиця Products** (Товари) зберігає інформацію про автозапчастини.

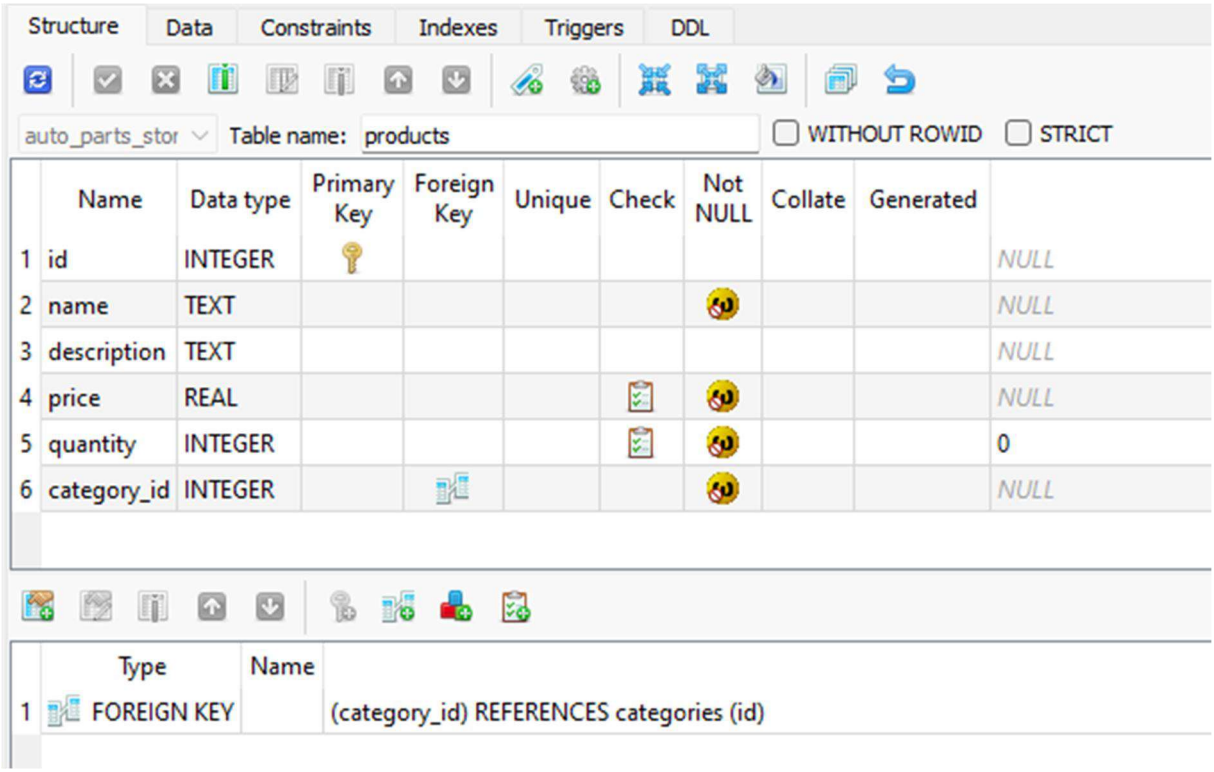

Рисунок 2.2.1 – Таблиця Products

Вона містить поля: id (унікальний ідентифікатор, первинний ключ), name (назва товару), description (опис), price (ціна), quantity (кількість на складі) і category_id (зовнішній ключ до таблиці Categories). Поле category_id пов'язує товар із категорією, забезпечуючи швидкий доступ до групування товарів.

2. **Таблиця Customers** (Клієнти) містить дані про клієнтів магазину.

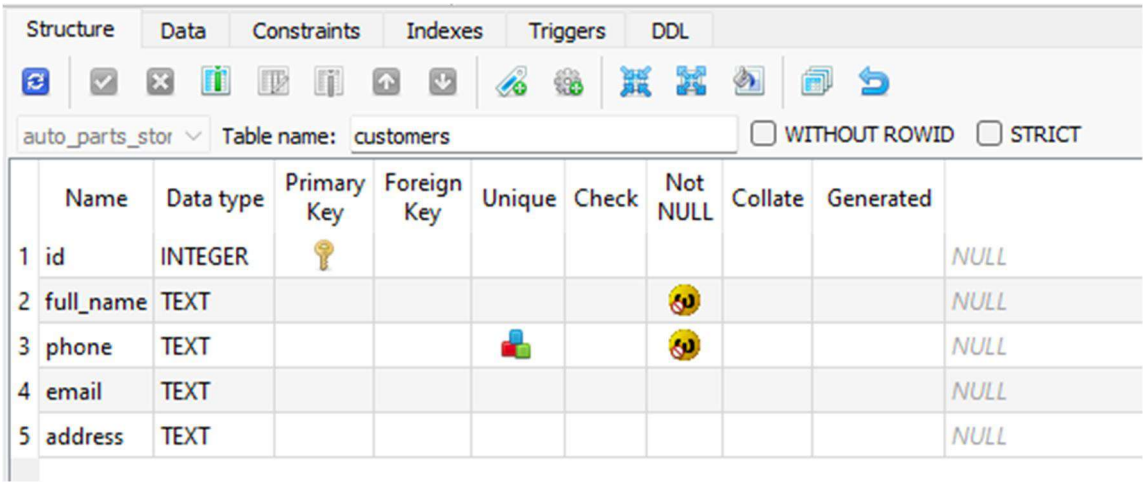

Рисунок 2.2.2 – Таблиця Customers

id (унікальний ідентифікатор, первинний ключ), full_name (ПІБ), phone (номер телефону), email (електронна пошта) і address (адреса). Ця таблиця використовується для ідентифікації клієнтів під час створення замовлень.

3. **Таблиця orders** (Замовлення) зберігає заголовки замовлень.

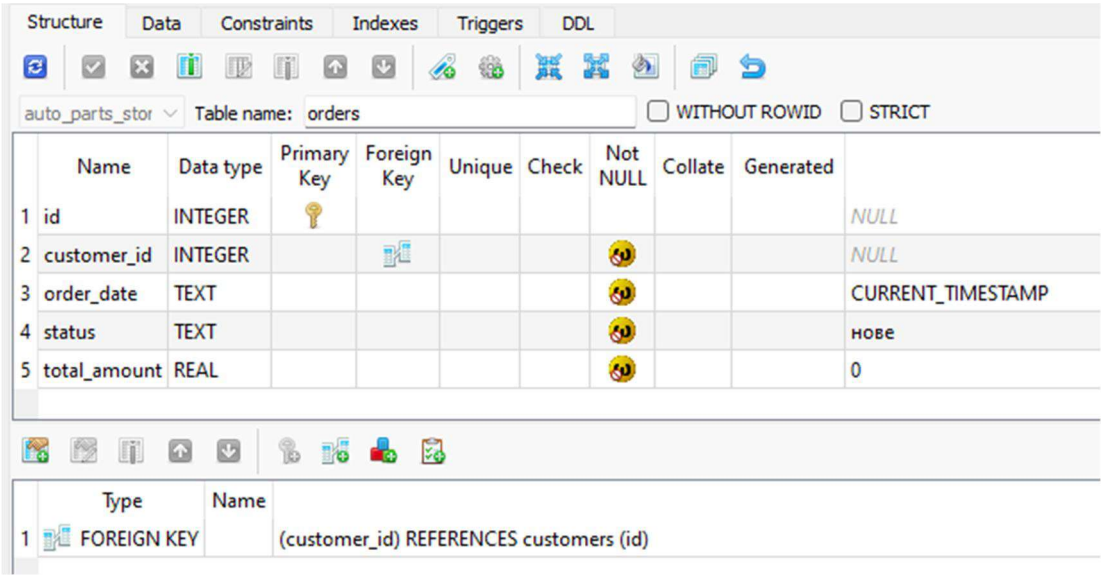

Рисунок 2.2.3 – Таблиця orders

id (унікальний ідентифікатор, первинний ключ), customer_id (зовнішній ключ до таблиці customers), order_date (дата замовлення), status (статус замовлення, наприклад, «нове», «в обробці»), total_amount (загальна сума). Поле customer_id забезпечує зв'язок із клієнтом, а total_amount обчислюється на основі позицій замовлення.

4. **Таблиця order_items** (Позиції замовлень) описує товари в замовленні/

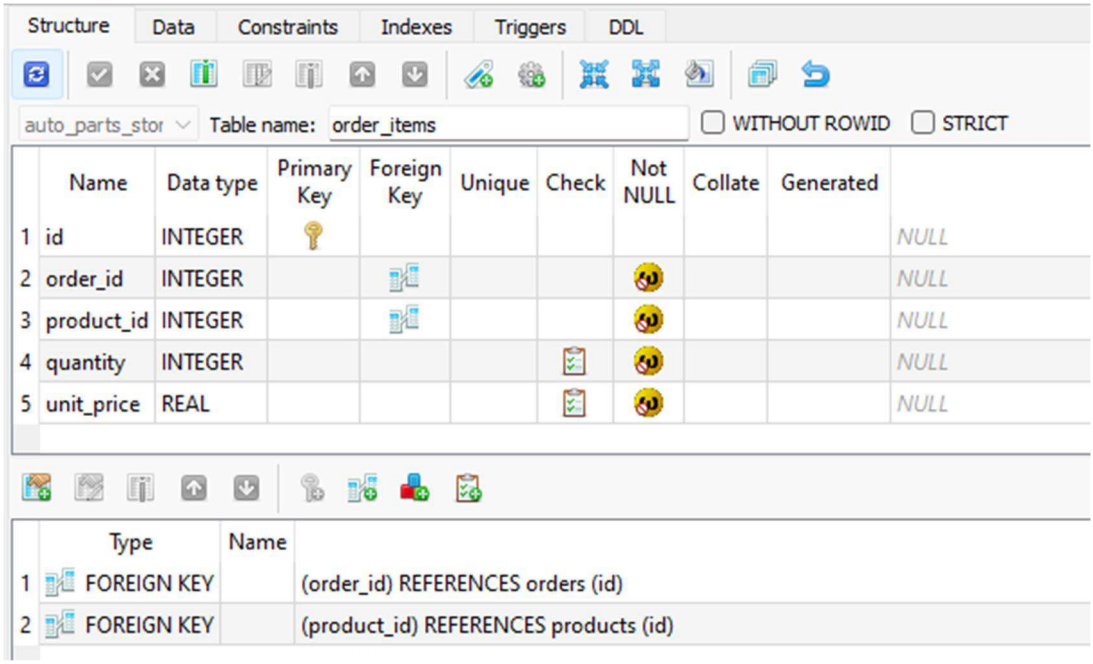

Рисунок 2.2.4 – Таблиця order_items

id (унікальний ідентифікатор, первинний ключ), order_id (зовнішній ключ до таблиці orders), product_id (зовнішній ключ до таблиці products), quantity (кількість товару), unit_price (ціна за одиницю). Ця таблиця забезпечує зв'язок «багато до одного» між замовленнями та товарами.

5. **Таблиця categories** (Категорії) містить перелік категорій товарів.

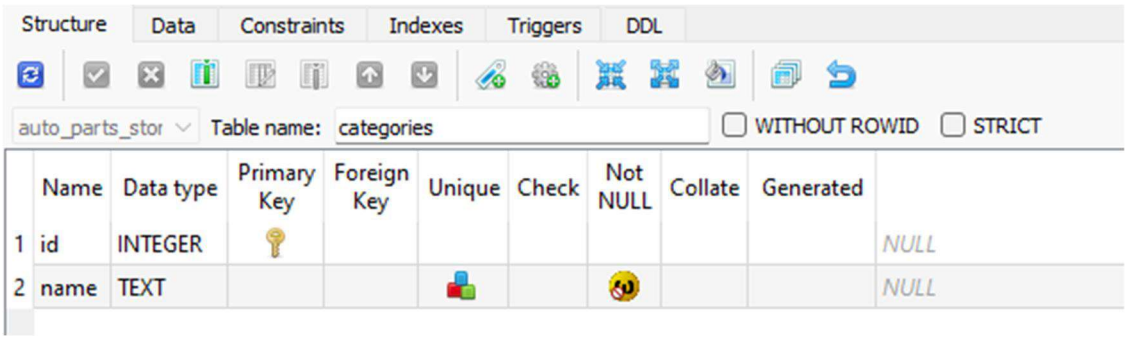

Рисунок 2.2.5 – Таблиця categories

id (унікальний ідентифікатор, первинний ключ) і name (назва категорії). Вона використовується для класифікації товарів і спрощення пошуку.

База даних спроєктована відповідно до третьої нормальної форми (3NF), що усуває надмірність і залежності між неключовими атрибутами. Наприклад, інформація про клієнтів зберігається лише в таблиці customers, а не дублюється в orders, що зменшує обсяг даних і спрощує оновлення. Аналогічно, ціна товару в orderitems зберігається як unit_price, щоб зафіксувати значення на момент замовлення, незалежно від подальших змін у таблиці Products. Зовнішні ключі (categoryid, customerid, orderid, productid) забезпечують референтну цілісність, а SQLite автоматично перевіряє їх під час операцій.

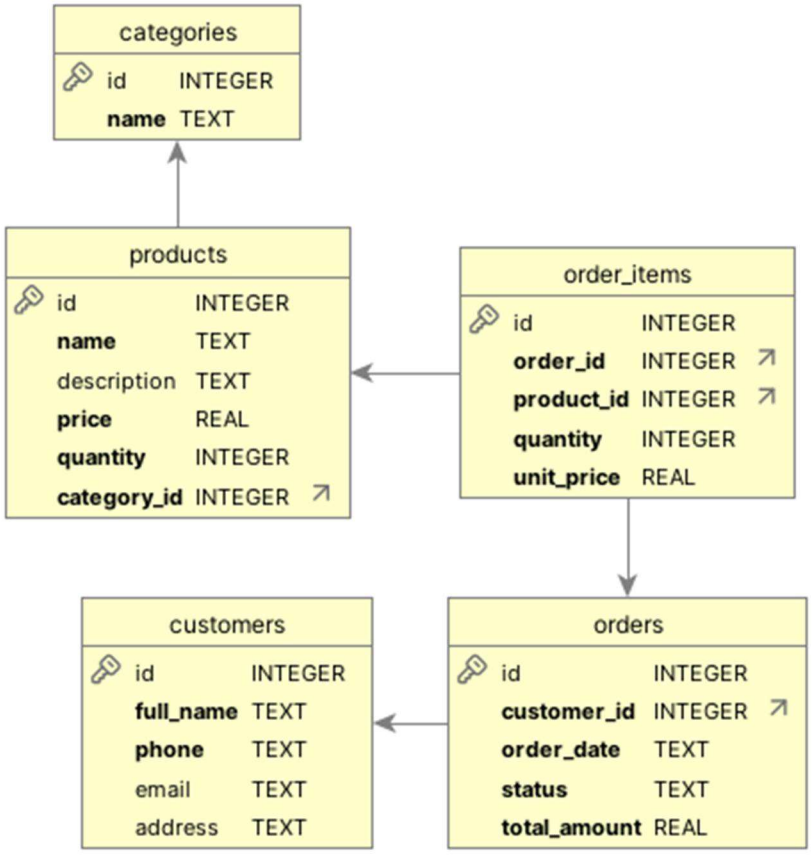

Рисунок 2.2.6 – Схема бази даних

База даних реалізовано за допомогою SQLite, а доступ до неї здійснюється через бібліотеку Microsoft.Data.Sqlite у складі .NET 8. Для роботи з даними використано патерн Repository, який ізолює логіку доступу

до бази від бізнес-логіки. Наприклад, клас ProductRepository містить методи для створення, читання, оновлення та видалення записів у таблиці Products. SQL-запити оптимізовані для швидкого виконання: створено індекси для полів Name у таблицях Products і Categories, а також для Status у таблиці Orders, що прискорює пошук і фільтрацію.

Для ініціалізації бази даних використано скрипт створення таблиць, який виконується під час першого запуску програми. Транзакції застосовуються для складних операцій, таких як створення замовлення з кількома позиціями, щоб забезпечити атомарність і консистентність даних. Наприклад, при створенні замовлення оновлення складських запасів і додавання позицій виконуються в одній транзакції, що запобігає невідповідностям у разі збою.

Структура бази даних є компактною та ефективною для невеликих магазинів, де обсяг даних не перевищує кількох тисяч записів. SQLite забезпечує швидкий доступ до даних завдяки локальному зберіганню, а розмір бази даних залишається мінімальним через відсутність серверних накладних витрат. Використання індексів і нормалізації дозволяє досягти часової складності O(log m) для пошукових запитів і O(1) для доступу за ідентифікатором, де m — кількість записів у таблиці. Для масштабування системи в майбутньому передбачено можливість додавання нових таблиць або міграції на серверну базу даних, таку як Microsoft SQL Server, без значних змін у коді.

Проєктування бази даних забезпечує баланс між простотою, продуктивністю та гнучкістю, відповідаючи потребам системи в надійному зберіганні даних і швидкому доступі до них.

2.3. Спроєктовані класи програми

Розробка інформаційної системи «Магазин автозапчастин» базується на об’єктно-орієнтованому підході з використанням принципів модульності, інкапсуляції та поділу відповідальностей. Для реалізації функціоналу системи спроєктовано набір класів, які відповідають за моделі даних, доступ до бази даних, бізнес-логіку та інтерфейс користувача. У цьому підрозділі описано основні класи програми, їх призначення, структуру та взаємозв’язки, що забезпечують виконання вимог до системи.

Моделі даних

Моделі даних відображають основні сутності системи та відповідають таблицям бази даних. Вони розміщені в просторі імен AutoPartsStore.Data.Models.

1. **Клас Product** представляє товар (автозапчастину).

```
18 references
public class Product
{
    7 references
    public int Id { get; set; }
    11 references
    public string Name { get; set; }
    6 references
    public string Description { get; set; }
    8 references
    public decimal Price { get; set; }
    12 references
    public int Quantity { get; set; }
    6 references
    public int CategoryId { get; set; }
    2 references
    public string? CategoryName { get; set; }
}
```

Рисунок 2.3.1 – Клас Product

Містить властивості Id (унікальний ідентифікатор), Name (назва), Description (опис), Price (ціна), Quantity (кількість на складі) і CategoryId (ідентифікатор категорії). Цей клас використовується для зберігання інформації про товари та їх відображення в інтерфейсі.

2. **Клас Customer** описує клієнта магазину.

```
public class Customer
{

    public int Id { get; set; }

    public string FullName { get; set; }

    public string Phone { get; set; }

    public string Email { get; set; }
    7 references
    public string Address { get; set; }
}
```

Рисунок 2.3.2 – Клас Customer

Включає властивості Id (унікальний ідентифікатор), FullName (ПІБ), Phone (номер телефону), Email (електронна пошта) і Address (адреса). Використовується для управління клієнтською базою та зв'язку з замовленнями.

3. **Клас Order** відображає замовлення.

```
15 references
public class Order
{
    6 references
    public int Id { get; set; }
    8 references
    public int CustomerId { get; set; }
    6 references
    public string OrderDate { get; set; }
    7 references
    public string Status { get; set; }
    6 references
    public decimal TotalAmount { get; set; }
    5 references
    public List<OrderItem> OrderItems { get; set; }
}
```

Рисунок 2.3.3 – Клас Order

Містить властивості Id (унікальний ідентифікатор), CustomerId (ідентифікатор клієнта), OrderDate (дата замовлення), Status (статус, наприклад, «нове» чи «доставлено»), TotalAmount (загальна сума) і OrderItems

(список позицій замовлення). Служить для управління замовленнями та обчислення їхньої вартості.

4. **Клас OrderItem** представляє позицію замовлення.

```
public class OrderItem
{
    6 references
    public int Id { get; set; } // Додано як первинний ключ
    7 references
    public int OrderId { get; set; }
    10 references
    public int ProductId { get; set; }
    8 references
    public int Quantity { get; set; }
    7 references
    public decimal UnitPrice { get; set; }
}
```

Рисунок 2.3.4 – Клас OrderItem

Включає властивості Id (унікальний ідентифікатор), OrderId (ідентифікатор замовлення), ProductId (ідентифікатор товару), Quantity (кількість) і UnitPrice (ціна за одиницю). Використовується для деталізації товарів у замовленні.

5. **Клас Category** описує категорію товарів.

```
public class Category
{

    public int Id { get; set; }
    7 references
    public string Name { get; set; }
}
```

Рисунок 2.3.5 – Клас Category

Містить властивості Id (унікальний ідентифікатор) і Name (назва). Застосовується для класифікації товарів і спрощення пошуку.

Додатково спроєктовано клас OrderItemViewModel у просторі імен AutoPartsStore.Models для відображення позицій замовлення в інтерфейсі. Він включає властивості ProductId, ProductName, Quantity, UnitPrice і TotalPrice

(обчислювальна властивість, що повертає Quantity * UnitPrice). Цей клас забезпечує зручне представлення даних у таблицях інтерфейсу.

Репозиторії

Для ізоляції логіки доступу до бази даних використано патерн Repository. Класи репозиторіїв розміщено в просторі імен AutoPartsStore.Data.Repository.

1. **Клас ProductRepository** реалізує операції з таблицею Products: створення, читання, оновлення та видалення товарів.

```
5 references
public class ProductRepository : IRepository<Product>
{
    private readonly DataContext _context;
    2 references
    public ProductRepository(DataContext context)...
    4 references
    public void Add(Product product)...
    8 references
    public Product GetById(int id)...
    6 references
    public IEnumerable<Product> GetAll()...
    5 references
    public void Update(Product product)...
    4 references
    public void Delete(int id)...
}
```

Рисунок 2.3.6 – Клас ProductRepository

Містить методи, такі як GetAll, GetById, Create, Update, Delete, які виконують SQL-запити через бібліотеку Microsoft.Data.Sqlite.

2. **Клас CustomerRepository** відповідає за роботу з таблицею Customers. Надає аналогічні методи для управління даними клієнтів.

```
5 references
public class CustomerRepository : IRepository<Customer>
{
    private readonly DataContext _context;
    2 references
    public CustomerRepository(DataContext context)...
    4 references
    public void Add(Customer customer)...
    8 references
    public Customer GetById(int id)...
    6 references
    public IEnumerable<Customer> GetAll()...
    5 references
    public void Update(Customer customer)...
    4 references
    public void Delete(int id)...
}
```

Рисунок 2.3.7 – Клас CustomerRepository

3. **Клас OrderRepository** обробляє операції з таблицею Orders, включаючи створення замовлення та отримання списку замовлень із пов'язаними даними клієнтів.

```
2 references
public class OrderRepository : IRepository<Order>
{
    private readonly DataContext _context;
    1 reference
    public OrderRepository(DataContext context)...
    3 references
    public void Add(Order order)...
    5 references
    public Order GetById(int id)...
    5 references
    public IEnumerable<Order> GetAll()...
    4 references
    public void Update(Order order)...
    3 references
    public void Delete(int id)...
}
```

Рисунок 2.3.8 – Клас OrderRepository

4. **Клас OrderItemRepository** управляє таблицею OrderItems, забезпечуючи додавання, оновлення та видалення позицій замовлення.

```
2 references
public class OrderItemRepository : IRepository<OrderItem>
{
    private readonly DataContext _context; // Єдине оголошення _context
    1 reference
    public OrderItemRepository(DataContext context)...
    3 references
    public void Add(OrderItem item)...
    5 references
    public OrderItem GetById(int id)...
    5 references
    public IEnumerable<OrderItem> GetAll()...
    4 references
    public void Update(OrderItem item)...
    3 references
    public void Delete(int id)...
}
```

Рисунок 2.3.9 – Клас OrderItemRepository

5. **Клас CategoryRepository** працює з таблицею Categories, надаючи методи для управління категоріями товарів.

```
7 references
public class CategoryRepository : IRepository<Category>
{
    private readonly DataContext _context;
    2 references
    public CategoryRepository(DataContext context)...
    4 references
    public void Add(Category category)...
    10 references
    public Category GetById(int id)...
    6 references
    public IEnumerable<Category> GetAll()...
    5 references
    public void Update(Category category)...
    4 references
    public void Delete(int id)...
}
```

Рисунок 2.3.10 – Клас CategoryRepository

Усі репозиторії отримують екземпляр DataContext (клас для ініціалізації підключення до SQLite) через конструктор, що забезпечує єдиний доступ до бази даних.

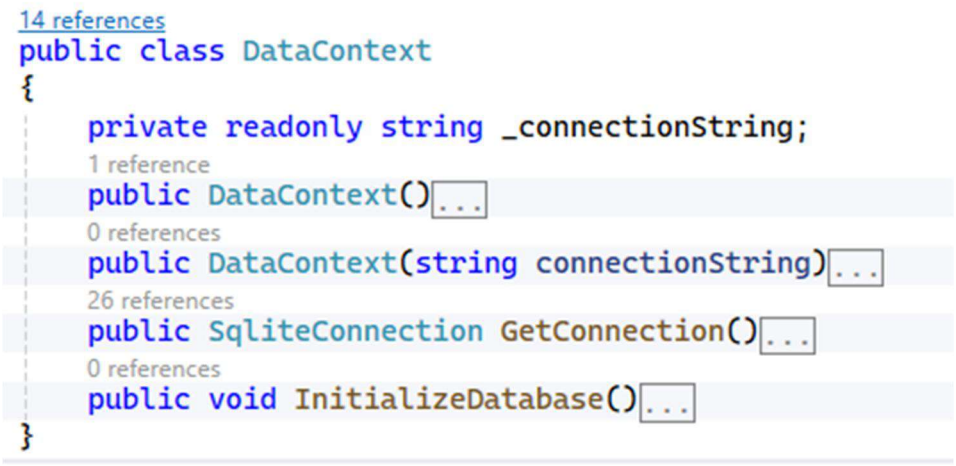

Рисунок 2.3.11 – Клас DataContext

Бізнес-логіка системи реалізована через класи сервісів у просторі імен AutoPartsStore.Services. Вони використовують репозиторії для доступу до даних і виконують складні операції.

1. **Клас ProductService** відповідає за управління товарами.

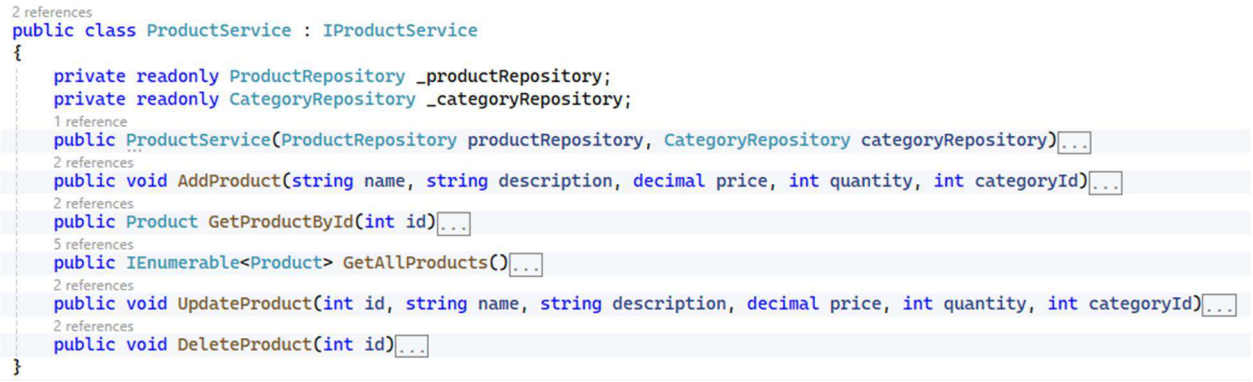

Рисунок 2.3.12 – Клас ProductService

Містить методи для отримання списку товарів, створення, оновлення та видалення записів. Використовує ProductRepository і CategoryRepository для забезпечення зв’язку товарів із категоріями.

2. **Клас CustomerService** управляє клієнтською базою, надаючи методи для роботи з клієнтами через CustomerRepository.

```
2 references
public class CustomerService : ICustomerService
{
    private readonly CustomerRepository _customerRepository;
    1 reference
    public CustomerService(CustomerRepository customerRepository)...
    2 references
    public void AddCustomer(string fullName, string phone, string email, string address)...
    1 reference
    public Customer GetCustomerById(int id)...
    4 references
    public IEnumerable<Customer> GetAllCustomers()...
    2 references
    public void UpdateCustomer(int id, string fullName, string phone, string email, string address)...
    2 references
    public void DeleteCustomer(int id)...
}
```

Рисунок 2.3.13 – Клас CustomerService

3. **Клас OrderService** реалізує логіку оброблення замовлень.

```
2 references
public class OrderService : IOrderService
{
    private readonly IRepository<Order> _orderRepository;
    private readonly IRepository<OrderItem> _orderItemRepository;
    private readonly IRepository<Product> _productRepository;
    private readonly IRepository<Customer> _customerRepository;
    private static readonly string[] ValidStatuses =
        { "нове", "в обробці", "відправлено", "доставлено", "скасовано" };
    1 reference
    public OrderService(
        IRepository<Order> orderRepository,
        IRepository<OrderItem> orderItemRepository,
        IRepository<Product> productRepository,
        IRepository<Customer> customerRepository)...
    2 references
    public void CreateOrder(int customerId, List<(int productId, int quantity, decimal unitPrice)> orderItems)...
    1 reference
    private void ValidateOrderCreation(int customerId, List<(int productId, int quantity, decimal unitPrice)> orderItems)...
    1 reference
    private decimal CalculateTotalAmount(List<(int productId, int quantity, decimal unitPrice)> orderItems,
        Dictionary<int, Product> products)...
    1 reference
    private void CreateOrderItems(int orderId,
        List<(int productId, int quantity, decimal unitPrice)> orderItems,
        Dictionary<int, Product> products)...
    2 references
    public void UpdateOrderStatus(int orderId, string status)...
    2 references
    public void DeleteOrder(int orderId)...
    2 references
    public Order GetOrderById(int orderId)...
    2 references
    public IEnumerable<Order> GetAllOrders()...
}
```

Рисунок 2.3.14 – Клас OrderService

Містить методи CreateOrder, UpdateOrderStatus, DeleteOrder і GetOrderById, які координують роботу OrderRepository, OrderItemRepository і ProductRepository. Наприклад, метод CreateOrder перевіряє наявність товарів, створює замовлення та оновлює складські запаси в одній транзакції.

4. **Клас CategoryService** забезпечує управління категоріями через CategoryRepository.

```
2 references
public class CategoryService : ICategoryService
{
    private readonly CategoryRepository _categoryRepository;
    1 reference
    public CategoryService(CategoryRepository categoryRepository)...
    2 references
    public void AddCategory(string name)...
    1 reference
    public Category GetCategoryById(int id)...
    3 references
    public IEnumerable<Category> GetAllCategories()...
    2 references
    public void UpdateCategory(int id, string name)...
    2 references
    public void DeleteCategory(int id)...
}
```

Рисунок 2.3.15 – Клас CategoryService

Інтерфейс користувача

Класи інтерфейсу користувача реалізовано як форми Windows Forms у просторі імен AutoPartsStore.Forms. Кожна форма відповідає окремому модулю системи.

1. **Клас OrderForm** відображає головний інтерфейс для роботи з замовленнями. Містить таблиці для відображення замовлень (dgvOrders) і позицій (dgvOrderItems), списки для вибору клієнтів, товарів і статусів (cmbCustomer, cmbProduct, cmbStatus), а також кнопки для додавання, оновлення, видалення та очищення даних. Використовує OrderService, CustomerService і ProductService для виконання операцій.
2. **Клас ProductForm** забезпечує управління товарами, дозволяючи створювати, редагувати та видаляти записи через ProductService і CategoryService.
3. **Клас CustomerForm** реалізує інтерфейс для роботи з клієнтською базою, використовуючи CustomerService.
4. **Клас CategoryForm** дозволяє управляти категоріями через CategoryService.
5. **Клас AboutForm** відображає інформацію про програму, включаючи назву, автора та рік створення.

Архітектура системи побудована за принципом поділу відповідальностей. Форми взаємодіють із сервісами, які, у свою чергу, використовують репозиторії для доступу до бази даних. Моделі даних передаються між рівнями для забезпечення консистентності. Наприклад, при створенні замовлення OrderForm викликає метод CreateOrder у OrderService, який координує роботу OrderRepository, OrderItemRepository і ProductRepository для збереження даних і оновлення запасів. Такий підхід забезпечує модульність і полегшує тестування та розширення системи.

Спроєктовані класи відповідають вимогам системи щодо простоти, продуктивності та розширюваності. Використання патерну Repository ізолює логіку доступу до даних, дозволяючи легко замінити SQLite на іншу базу даних. Сервіси забезпечують чітке розділення бізнес-логіки, а форми надають зручний інтерфейс для користувача. Структура класів дозволяє додавати нові функції, такі як звіти чи інтеграція з зовнішніми системами, без значних змін у коді. Часова складність операцій у сервісах і репозиторіях залежить від SQL-запитів, які оптимізовані за допомогою індексів, забезпечуючи швидкодію для невеликих обсягів даних.

Проєктування класів забезпечує надійну основу для реалізації функціоналу системи, відповідаючи потребам невеликих магазинів автозапчастин у зручному та ефективному програмному забезпеченні.

2.4. Програмна реалізація

Програмна реалізація інформаційної системи «Магазин автозапчастин» виконана з використанням технологічного стеку Windows Forms, .NET 8 і SQLite. Система розроблена як настільний застосунок, що забезпечує автоматизацію управління товарами, замовленнями, клієнтами та категоріями. У цьому підрозділі описано склад проєкту, структуру програми та детальну реалізацію форм інтерфейсу користувача, які є ключовими компонентами системи.

Склад проєкту

Проєкт організовано в модульній структурі з чітким поділом на логічні шари: моделі даних, доступ до бази даних, бізнес-логіку та інтерфейс користувача.

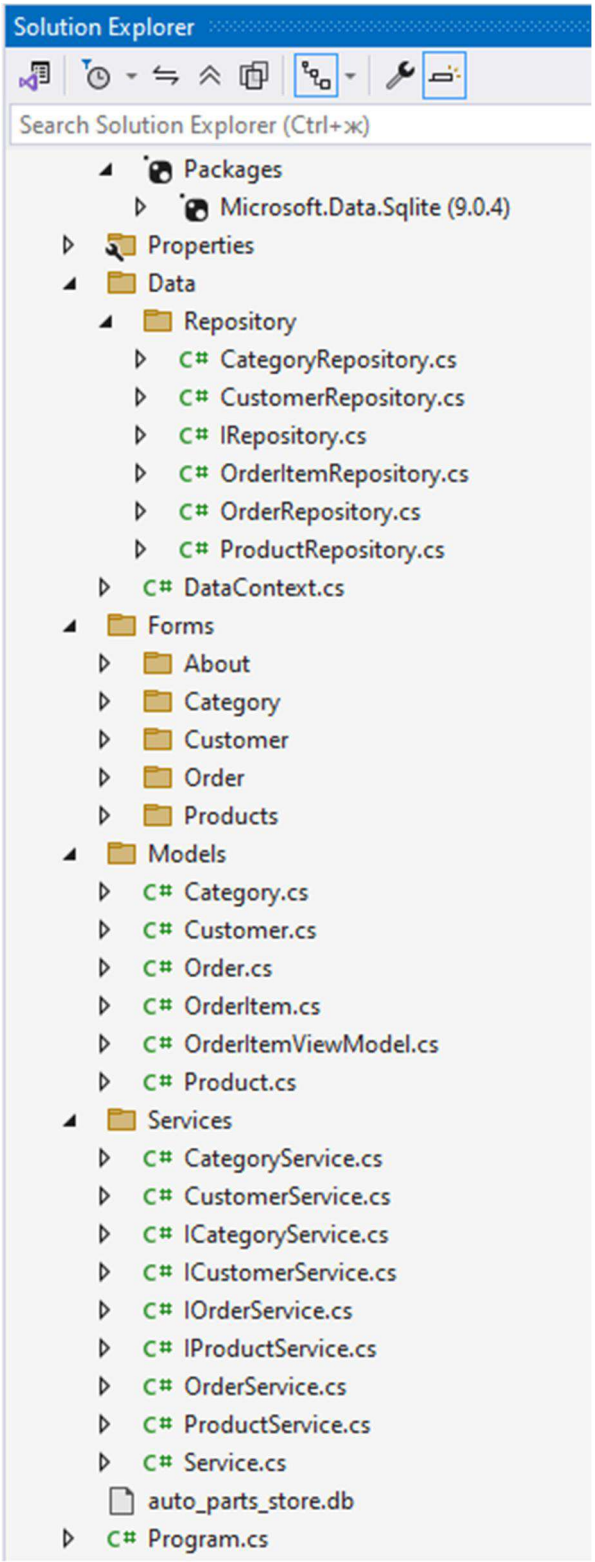

Рисунок 2.4.1 – Склад проєкту

Основні простори імен і їх призначення:

1. **AutoPartsStore.Data** містить класи для роботи з базою даних SQLite:

DataContext — ініціалізує підключення до бази даних і забезпечує доступ до SQLite.

Простір імен AutoPartsStore.Data.Models включає моделі даних: Product, Customer, Order, OrderItem, Category, що відображають таблиці бази даних.

Простір імен AutoPartsStore.Data.Repository містить класи репозиторіїв (ProductRepository, CustomerRepository, OrderRepository, OrderItemRepository, CategoryRepository) для виконання CRUD-операцій.

2. **AutoPartsStore.Services** реалізує бізнес-логіку через класи сервісів: ProductService, CustomerService, OrderService, CategoryService. Ці класи координують роботу репозиторіїв і виконують складні операції, такі як створення замовлення з оновленням складських запасів.
3. **AutoPartsStore.Models** містить допоміжні моделі для інтерфейсу, зокрема OrderItemViewModel, який використовується для відображення позицій замовлення з обчислювальною властивістю TotalPrice.
4. **AutoPartsStore.Forms** включає форми Windows Forms для реалізації інтерфейсу користувача. Кожна форма відповідає окремому модулю системи й описана нижче.
5. **AutoPartsStore** (кореневий простір імен) містить точку входу програми — клас Program із методом Main, який ініціалізує застосунок і запускає головну форму OrderForm.

Проєкт створено у Visual Studio 2022 як рішення .NET 8 Windows Forms App. Залежності включають бібліотеку Microsoft.Data.Sqlite для роботи з SQLite. Усі файли бази даних зберігаються локально в директорії застосунку, що забезпечує портативність системи.

Реалізація форм

Інтерфейс користувача реалізовано через п’ять форм Windows Forms, кожна з яких відповідає за окремий модуль системи. Форми використовують компоненти Windows Forms, такі як DataGridView, ComboBox, TextBox, Button

і MenuStrip, для забезпечення зручної взаємодії з користувачем. Нижче описано кожну форму, її функціонал і ключові елементи.

1. **OrderForm** (Головна форма)

Головна форма для управління замовленнями, включаючи створення, редагування, видалення та перегляд деталей.

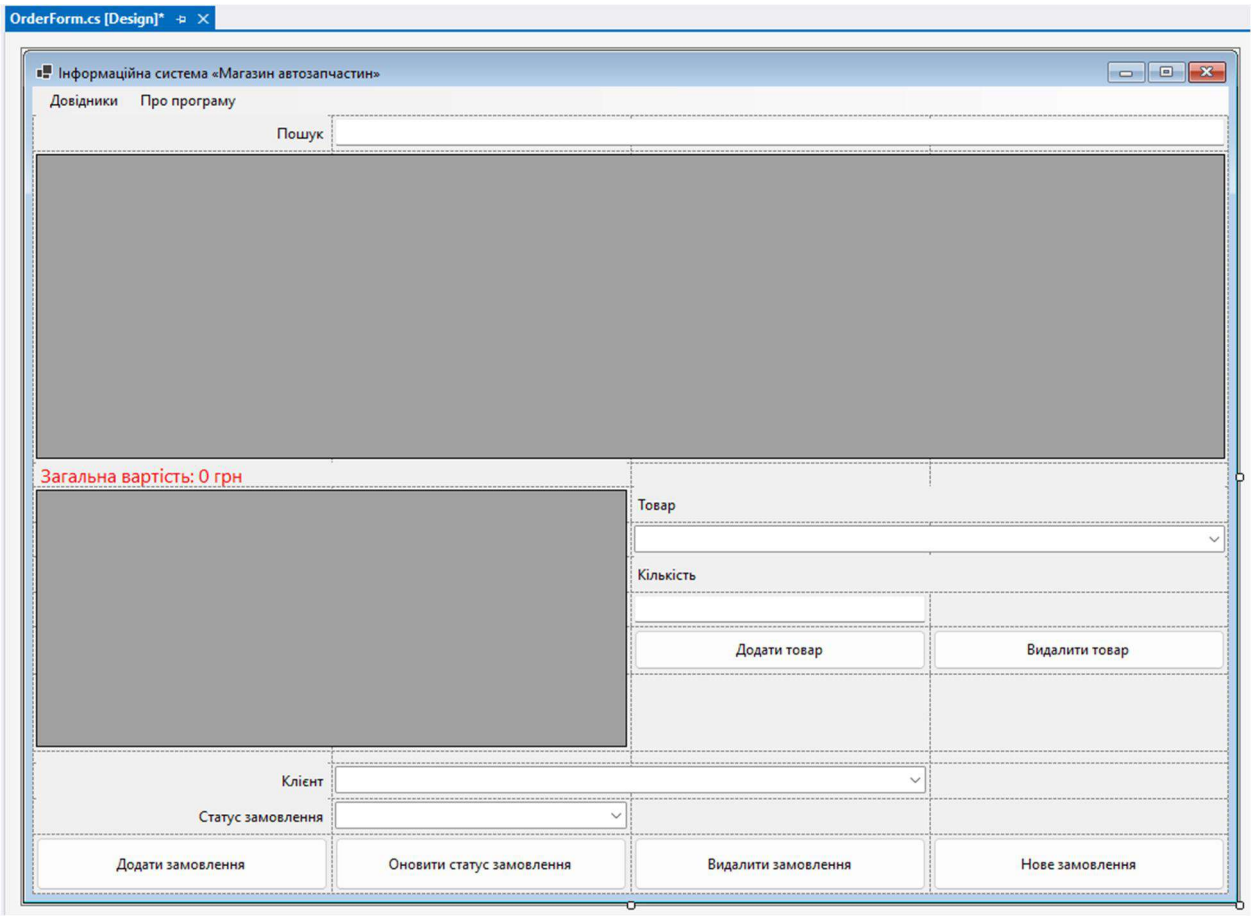

Рисунок 2.4.2 – Головна форма

Форма дозволяє створювати замовлення, додавати до них товари, видаляти позиції, змінювати статус і видаляти замовлення. Пошук замовлень реалізовано через подію TextChanged для txtSearch. Вибір замовлення в dgvOrders оновлює dgvOrderItems і поля форми. Форма взаємодіє з OrderService, CustomerService і ProductService для виконання операцій.

Використовує транзакції для створення замовлення та оновлення запасів, забезпечуючи консистентність даних. Колонка TotalPrice у dgvOrderItems обчислюється динамічно через OrderItemViewModel.

2. **ProductForm** (Форма товарів)

Управління асортиментом товарів, включаючи створення, редагування та видалення.

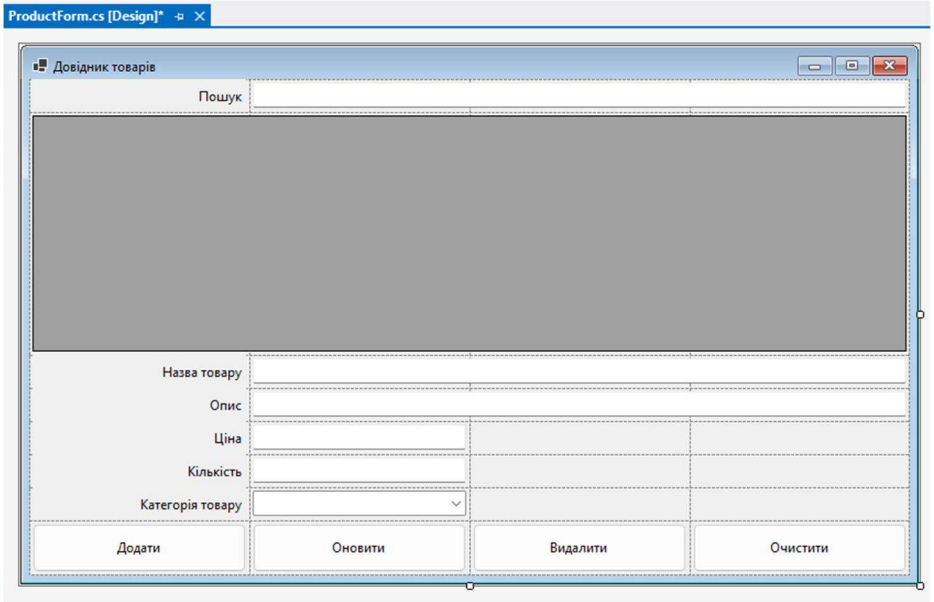

Рисунок 2.4.3 – Форма замовлень

Дозволяє додавати нові товари, редагувати існуючі та видаляти їх. Вибір товару в dgvProducts заповнює поля форми для редагування. Форма використовує ProductService і CategoryService для роботи з даними. Перевіряє коректність введених даних (наприклад, позитивна ціна та кількість) перед збереженням.

3. **CustomerForm** (Форма клієнтів) Управління базою клієнтів.

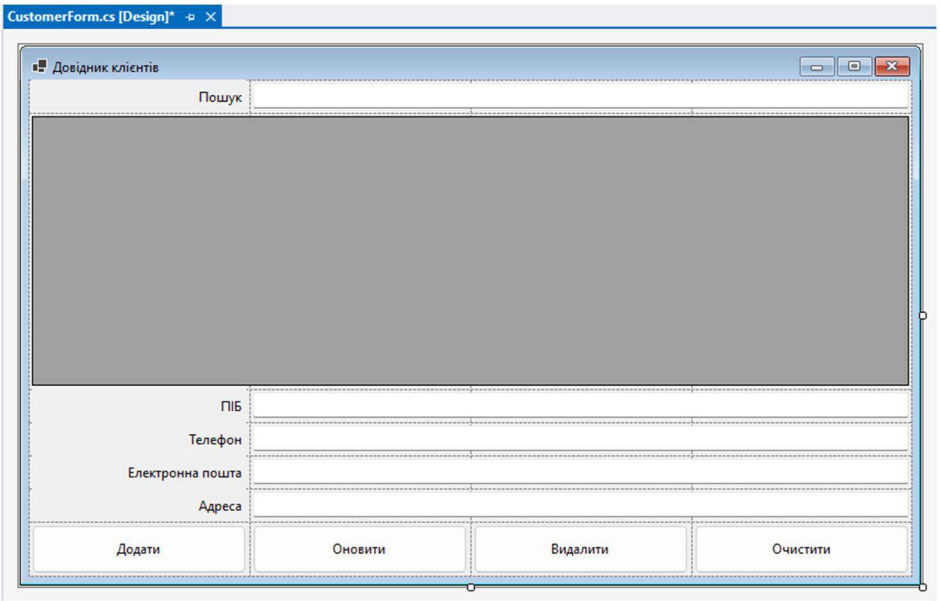

Рисунок 2.4.4 – Форма клієнтів

Надає можливість додавати, редагувати та видаляти клієнтів. Використовує CustomerService для доступу до даних.

Включає валідацію обов'язкових полів, таких як FullName і Phone.

4. **CategoryForm** (Форма категорій).Управління категоріями товарів.

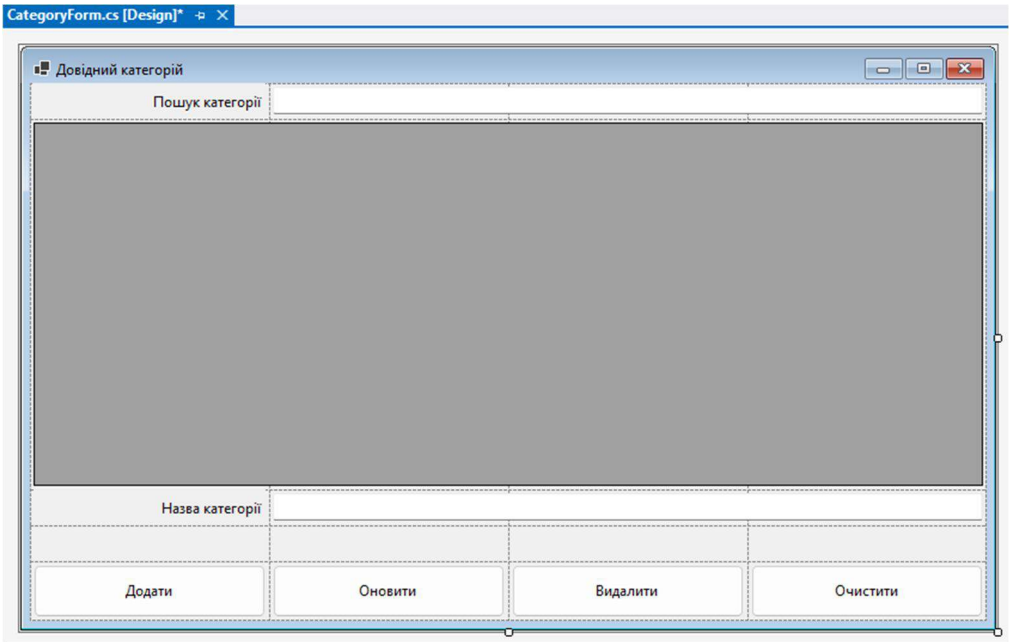

Рисунок 2.4.5 – Форма категорій

Дозволяє створювати, редагувати та видаляти категорії. Використовує CategoryService для роботи з даними. Перевіряє унікальність назви категорії перед збереженням.

5. **AboutForm** (Форма «Про програму»)

Відображення інформації про систему.

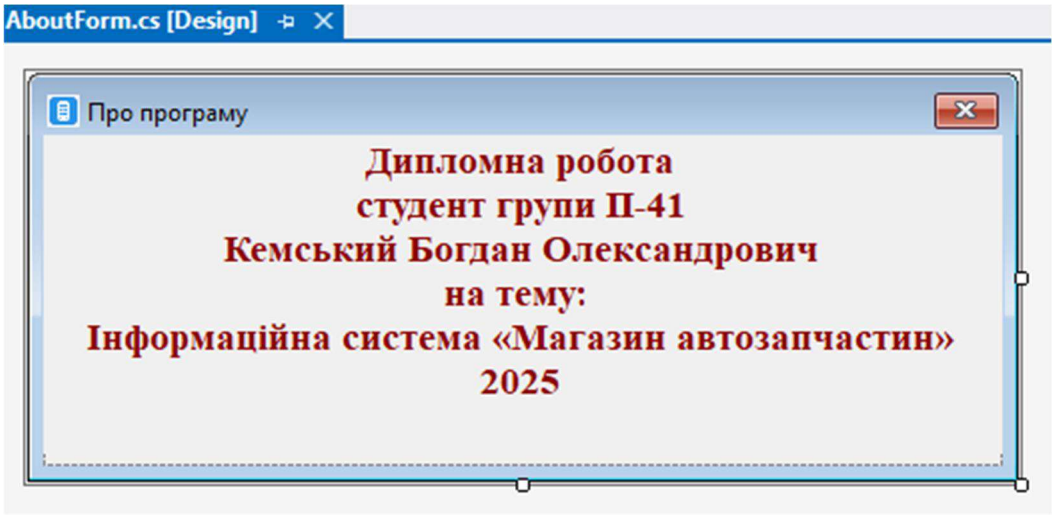

Рисунок 2.4.6 – Форма «Про програму»

Показує статичну інформацію про програму. Не взаємодіє з сервісами чи базою даних. Проста форма без складної логіки, відкривається через меню інших форм.

Реалізація форм ґрунтується на принципах модульності та повторного використання коду. Усі форми використовують однаковий стиль оформлення, реалізований через утиліту Service.SettingGrid, яка налаштовує таблиці DataGridView (автоматичне масштабування колонок, заборона редагування, вибір цілого рядка). Події, такі як SelectionChanged для таблиць і Click для кнопок, обробляються для забезпечення реактивного інтерфейсу. Наприклад, вибір замовлення в OrderForm автоматично оновлює dgvOrderItems і labelTotalPrice.

Для підвищення надійності використано обробку винятків у всіх операціях із базою даних. Наприклад, у разі помилки при створенні замовлення відображається повідомлення через MessageBox із деталями. Валідація введених даних (наприклад, перевірка позитивної кількості в txtQuantity) виконується перед збереженням, що запобігає некоректним операціям.

Реалізація форм оптимізована для швидкої роботи з невеликими обсягами даних (до кількох тисяч записів). Завантаження даних у таблиці та списки виконується асинхронно через сервіси, що мінімізує затримки. Використання транзакцій у OrderService забезпечує консистентність при складних операціях. Структура форм дозволяє легко додавати нові функції, наприклад, фільтри чи звіти, шляхом розширення існуючих класів або створення нових форм.

Програмна реалізація відповідає вимогам системи, забезпечуючи зручний інтерфейс, надійну обробку даних і можливість управління основними бізнес-процесами магазину автозапчастин.

Висновки до розділу 2

У процесі проєктування та розробки інформаційної системи «Магазин автозапчастин» було створено комплексне рішення для автоматизації бізнес-процесів невеликих магазинів автозапчастин. Вибір алгоритмів, таких як CRUD-операції, лінійний пошук із SQL-фільтрацією та транзакційна обробка, забезпечує достатню ефективність для роботи з невеликими обсягами даних, відповідаючи вимогам швидкодії та стабільності. Використання індексів у SQLite і простих алгоритмів із часовою складністю O(n) або O(log m) дозволяє досягти швидкого доступу до даних і обробки операцій.

База даних, спроєктована на основі SQLite, відповідає принципам третьої нормальної форми, що усуває надмірність і забезпечує цілісність даних. Структура з п’яти таблиць (Products, Customers, Orders, OrderItems, Categories) ефективно підтримує зберігання інформації про товари, клієнтів, замовлення та категорії. Використання транзакцій і індексів підвищує надійність і швидкодію операцій, а локальне зберігання даних робить систему економічною та портативною.

Класи програми, поділені на моделі даних, репозиторії, сервіси та форми, реалізовано з урахуванням принципів модульності та поділу відповідальностей. Моделі даних відображають сутності системи, репозиторії ізолюють логіку доступу до бази, сервіси координують бізнес-логіку, а форми забезпечують зручний інтерфейс. Така архітектура полегшує підтримку та розширення системи.

Програмна реалізація включає п’ять форм Windows Forms (OrderForm, ProductForm, CustomerForm, CategoryForm, AboutForm), які забезпечують управління всіма аспектами роботи магазину. Форми мають інтуїтивно зрозумілий інтерфейс, підтримують валідацію даних і обробку винятків, що підвищує надійність системи. Структура проєкту, побудована на .NET 8, дозволяє легко інтегрувати нові функції чи мігрувати на іншу базу даних у разі потреби.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для коректної роботи інформаційної системи «Магазин автозапчастин» необхідно забезпечити відповідність апаратного та програмного забезпечення мінімальним вимогам. Система розроблена як настільний застосунок для операційної системи Windows, що робить її сумісною з більшістю сучасних комп’ютерів, використовуваних у роздрібній торгівлі.

Щодо апаратного забезпечення, комп’ютер має бути оснащений процесором із тактовою частотою не нижче 1 ГГц, оперативною пам’яттю обсягом від 2 ГБ і вільним місцем на диску щонайменше 500 МБ для встановлення програми та зберігання бази даних SQLite. Для комфортної роботи рекомендується монітор із роздільною здатністю 1280x720 пікселів або вище, щоб забезпечити зручне відображення форм інтерфейсу. Система не потребує потужного графічного процесора, оскільки використовує стандартні компоненти Windows Forms.

Програмне забезпечення має включати операційну систему Windows 10 або новішу версію (64-бітна архітектура рекомендована, але 32-бітна також підтримується). Для роботи програми необхідно встановити .NET 8 Runtime, який забезпечує виконання застосунку. Додаткове програмне забезпечення, таке як сервер бази даних, не потрібне, оскільки SQLite працює з локальним файлом бази даних, що спрощує розгортання. Для встановлення чи оновлення програми може знадобитися доступ до інтернету, щоб завантажити .NET 8 Runtime або бібліотеки Microsoft.Data.Sqlite, якщо вони не включені в інсталяційний пакет.

Ці вимоги забезпечують доступність системи для використання на бюджетних комп’ютерах, які зазвичай застосовуються в невеликих магазинах автозапчастин, і мінімізують витрати на апаратне забезпечення та ліцензування програмного забезпечення.

3.2. Інтерфейс користувача

Інтерфейс інформаційної системи «Магазин автозапчастин» розроблено з використанням технології Windows Forms, що забезпечує простоту, інтуїтивність і зручність взаємодії для користувачів із мінімальним досвідом роботи з подібними програмами. Система складається з п’яти основних форм: «Замовлення», «Товари», «Клієнти», «Категорії» та «Про програму». Кожна форма має єдиний стиль оформлення, логічне розташування елементів і навігаційне меню для швидкого переходу між модулями. Нижче описано інтерфейс кожної форми та їх функціональні особливості.

Форма «Замовлення»

Форма «Замовлення» є головним інтерфейсом системи, призначеним для створення, редагування, видалення та перегляду замовлень. У верхній частині форми розташовано меню з пунктами «Клієнти», «Товари», «Категорії» та «Про програму», що дозволяють відкривати відповідні форми. Основна область форми поділена на кілька секцій.

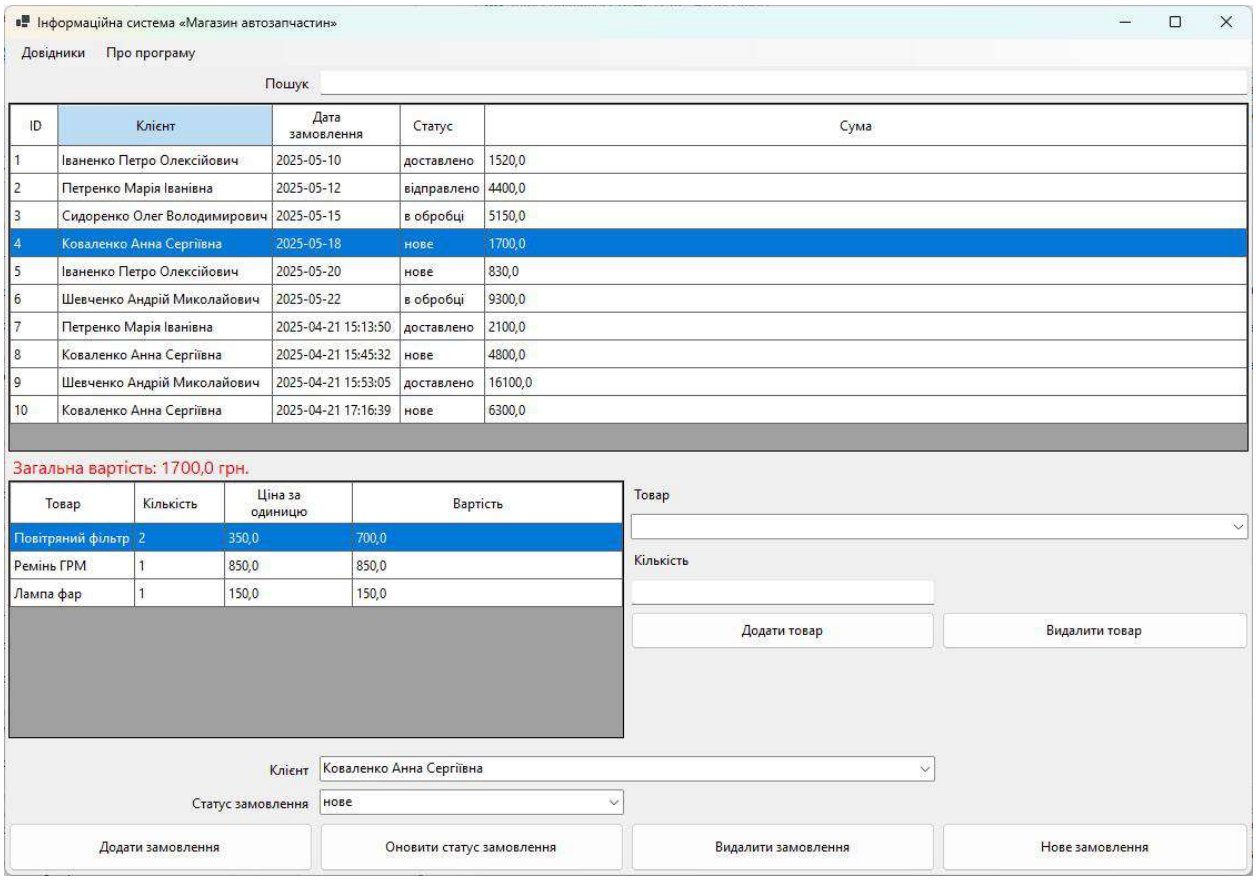

Рисунок 3.2.1 – Форма «Замовлення»

Ліва верхня секція містить таблицю, яка відображає список замовлень із колонками: ідентифікатор, ім’я клієнта, дата замовлення, статус і загальна сума. Під таблицею розташовано поле пошуку, що дозволяє фільтрувати замовлення за ідентифікатором клієнта або статусом. Права верхня секція включає поля для вибору клієнта, статусу замовлення та кнопки для додавання, оновлення, видалення чи очищення замовлення. У нижній частині форми розміщена таблиця позицій замовлення з колонками: назва товару, кількість, ціна за одиницю та загальна вартість. Поруч із таблицею знаходяться елементи для додавання товару до замовлення: список товарів, поле для введення кількості, кнопки для додавання та видалення позицій. Під таблицею позицій відображається мітка із загальною вартістю замовлення у форматі «Загальна вартість: X грн».

Елементи керування логічно згруповано за допомогою панелі таблиць, що забезпечує чітке розташування. Таблиці автоматично масштабуються для заповнення доступного простору, а вибір рядка в таблиці замовлень оновлює деталі в інших полях, що робить інтерфейс реактивним і зручним.

Форма «Товари»

Форма «Товари» призначена для управління асортиментом автозапчастин. У верхній частині розташовано те саме навігаційне меню для доступу до інших модулів. Основна область форми поділена на таблицю та панель введення даних. Таблиця відображає список товарів із колонками: ідентифікатор, назва, опис, ціна, кількість і назва категорії. Панель введення містить текстові поля для назви, опису, ціни та кількості товару, а також список для вибору категорії. Кнопки для додавання, оновлення, видалення та очищення даних розміщено поруч із полями введення. Вибір товару в таблиці автоматично заповнює поля для редагування, що спрощує внесення змін. Інтерфейс мінімалістичний, з акцентом на швидкий доступ до функцій управління товарами.

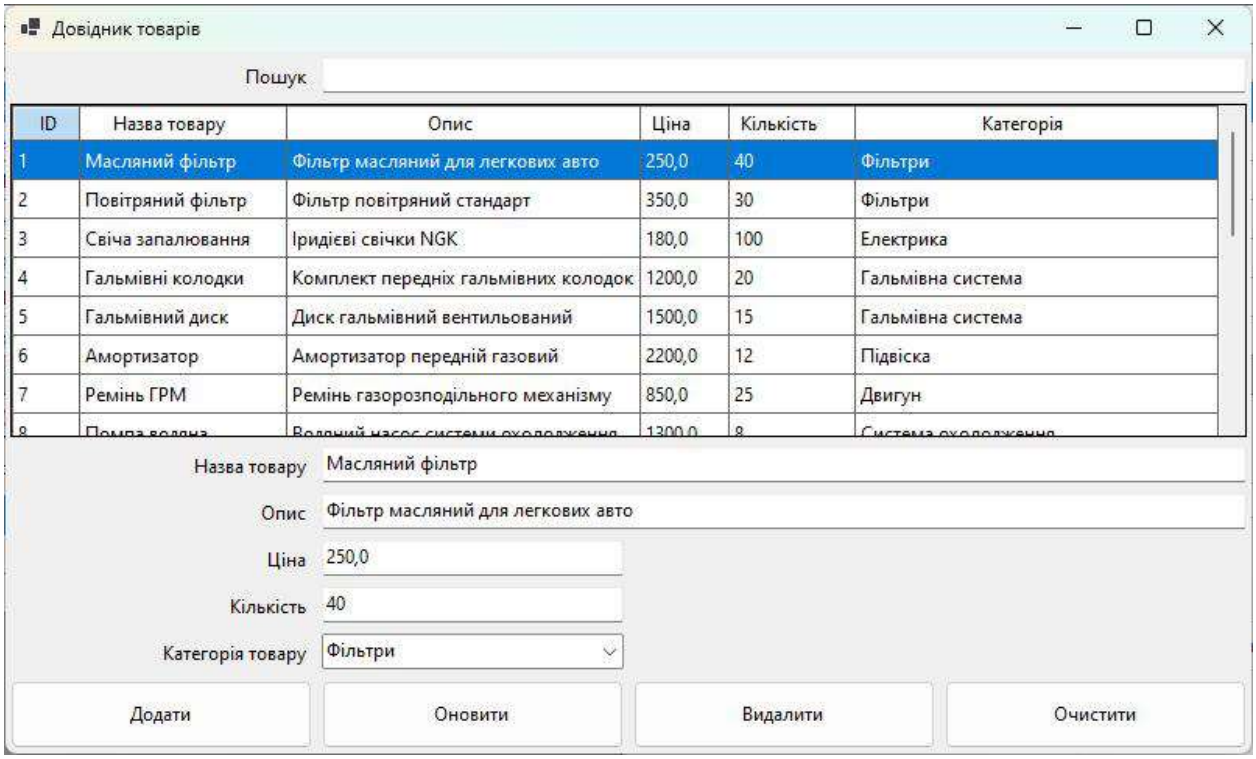

Рисунок 3.2.2 – Форма «Товари»

Форма «Клієнти»

Форма «Клієнти» забезпечує управління базою клієнтів. Вона містить навігаційне меню у верхній частині та дві основні секції. Таблиця клієнтів відображає дані з колонками: ідентифікатор, ПІБ, номер телефону, електронна пошта та адреса. Панель введення включає текстові поля для введення ПІБ, телефону, електронної пошти та адреси, а також кнопки для додавання, оновлення, видалення та очищення форми. Логічне розташування елементів і автоматичне заповнення полів при виборі клієнта в таблиці забезпечують зручність роботи з клієнтською базою.

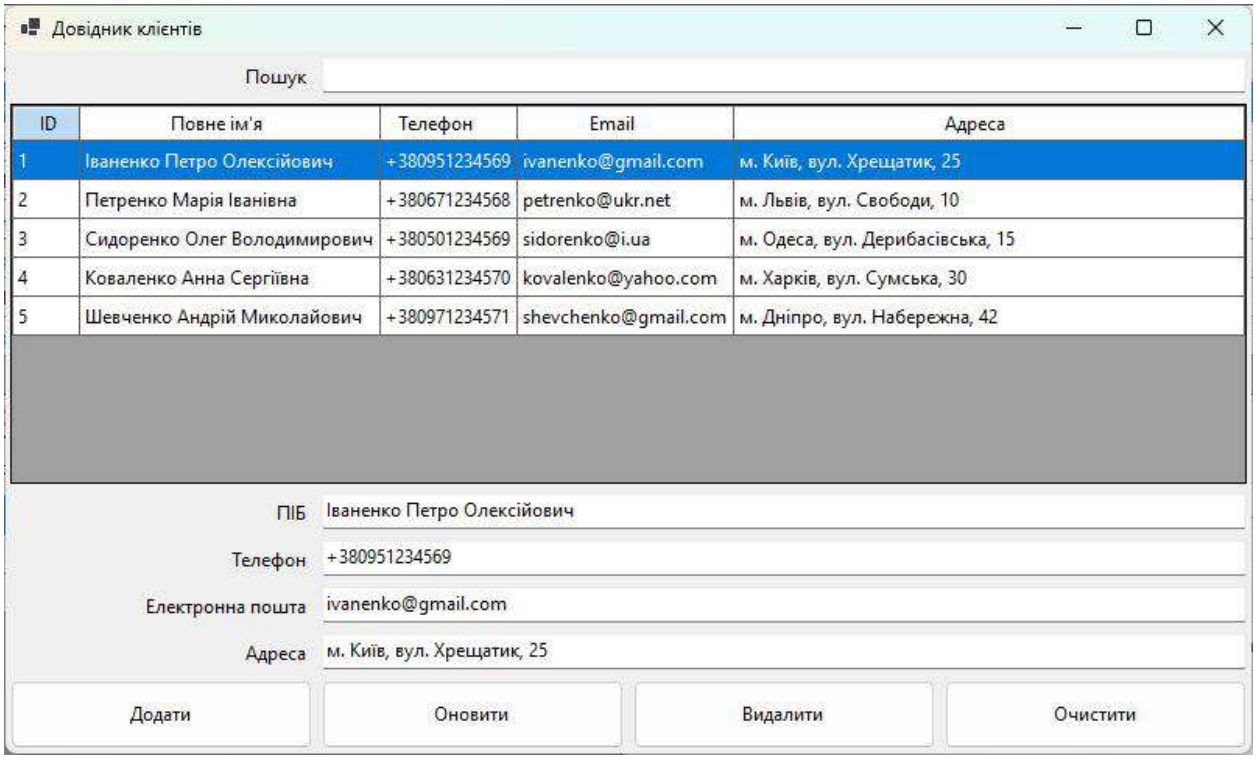

Рисунок 3.2.3 – Форма «Клієнти»

Форма «Категорії»

Форма «Категорії» призначена для управління категоріями товарів. Вона має навігаційне меню для переходу до інших форм. Основна область поділена на таблицю з колонками ідентифікатор і назва категорії та панель із текстовим полем для введення назви категорії. Кнопки для додавання, оновлення, видалення та очищення розміщено поруч із полем. Інтерфейс є максимально спрощеним, оскільки категорії мають лише одне ключове поле — назву, що робить форму зручною для швидкого редагування.

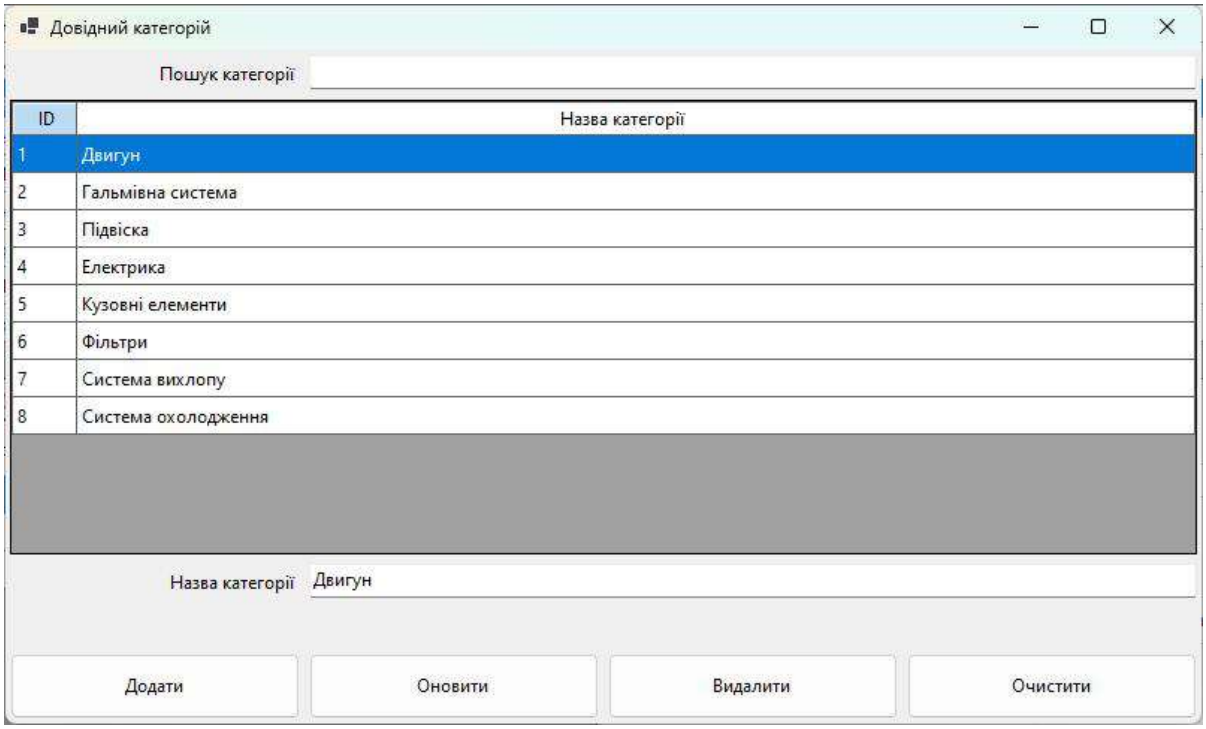

Рисунок 3.2.4 – Форма «Категорії»

Форма «Про програму»

Форма «Про програму» відображає статичну інформацію про систему. Вона не містить навігаційного меню, оскільки є інформаційною. У центрі форми розташовано текстову мітку з назвою програми, автором, роком створення та коротким описом. Єдина кнопка «Закрити» дозволяє повернутися до попередньої форми. Інтерфейс мінімалістичний, з акцентом на чітке подання інформації.

Рисунок 3.2.5 – Форма «Про програму»

Усі форми мають єдиний стиль оформлення: світлу кольорову палітру, чіткі написи та однаковий шрифт для елементів керування. Таблиці в усіх

формах налаштовано для автоматичного масштабування колонок, заборони прямого редагування та вибору цілого рядка, що підвищує зручність і запобігає випадковим змінам. Навігаційне меню, присутнє в усіх формах, крім «Про програму», забезпечує швидкий доступ до будь-якого модуля системи. Обробка винятків реалізована через спливні повідомлення, які інформують користувача про помилки, наприклад, некоректне введення даних чи проблеми з базою даних.

Інтерфейс системи розроблено з урахуванням потреб працівників магазинів автозапчастин, які можуть мати обмежений досвід роботи з програмним забезпеченням. Логічна структура форм, чітке групування елементів і реактивність інтерфейсу забезпечують швидке освоєння та ефективне виконання завдань, таких як створення замовлень, управління товарами чи клієнтською базою.

3.3. Інструкція для роботи з програмою

Інформаційна система «Магазин автозапчастин» розроблена для автоматизації управління замовленнями, товарами, клієнтами та категоріями товарів у невеликих магазинах автозапчастин. Програма має інтуїтивно зрозумілий інтерфейс, побудований на базі Windows Forms, і дозволяє швидко виконувати основні операції. Ця інструкція описує покрокові дії для роботи з програмою, включаючи запуск, навігацію та використання основних функцій.

Запуск програми

Для запуску програми необхідно переконатися, що на комп’ютері встановлено .NET 8 Runtime і операційну систему Windows 10 або новішу. Після встановлення програми виконайте такі дії: знайдіть виконуваний файл AutoPartsStore.exe у папці встановлення, двічі клацніть на нього, щоб відкрити програму. При першому запуску створюється локальна база даних SQLite, якщо вона ще не існує. Головна форма «Замовлення» відкривається автоматично.

Навігація між формами

У верхній частині кожної форми, крім «Про програму», розташовано меню з пунктами «Клієнти», «Товари», «Категорії» та «Про програму». Клацніть на відповідний пункт, щоб відкрити потрібну форму. Наприклад, вибір «Товари» відкриває форму для управління асортиментом. Форми відкриваються в модальному режимі, що дозволяє зосередитися на поточному завданні. Щоб повернутися до попередньої форми, закрийте поточну або виконайте потрібні дії.

Форма «Замовлення» є основним інтерфейсом для створення та управління замовленнями. Щоб створити нове замовлення, виконайте такі дії: у списку клієнтів виберіть потрібного клієнта, у списку товарів оберіть товар, введіть кількість у відповідне поле, натисніть кнопку «Додати позицію». Повторіть цей крок для всіх потрібних товарів. Загальна вартість замовлення відобразиться внизу форми. Після додавання всіх позицій натисніть «Додати замовлення», щоб зберегти його. Для редагування статусу замовлення виберіть замовлення в таблиці, змініть статус у списку та натисніть «Оновити». Щоб видалити замовлення, виберіть його та натисніть «Видалити», підтвердивши дію у спливному вікні. Для пошуку замовлення введіть ідентифікатор клієнта або частину статусу в поле пошуку — таблиця автоматично оновиться. Кнопка «Очистити» скидає всі поля та вибір.

Форма «Товари» дозволяє додавати, редагувати та видаляти автозапчастини. Щоб додати новий товар, введіть назву, опис, ціну та кількість у відповідні поля, виберіть категорію зі списку та натисніть «Додати». Для редагування виберіть товар у таблиці, змініть потрібні поля та натисніть «Оновити». Щоб видалити товар, виберіть його та натисніть «Видалити», підтвердивши дію. Кнопка «Очистити» скидає поля для введення нового товару. Таблиця товарів відображає всі записи з можливістю швидкого перегляду.

Форма «Клієнти» призначена для управління клієнтською базою. Щоб додати клієнта, введіть ПІБ, номер телефону, електронну пошту та адресу в

поля, потім натисніть «Додати». Для редагування виберіть клієнта в таблиці, змініть дані та натисніть «Оновити». Видалення клієнта виконується через вибір у таблиці та натискання «Видалити» з подальшим підтвердженням. Кнопка «Очистити» скидає поля для введення нового клієнта. Усі поля, крім електронної пошти, є обов’язковими.

Форма «Категорії» дозволяє управляти категоріями товарів. Для створення категорії введіть її назву в поле та натисніть «Додати». Щоб змінити назву, виберіть категорію в таблиці, введіть нову назву та натисніть «Оновити». Видалення категорії виконується через вибір і натискання «Видалити» з підтвердженням. Кнопка «Очистити» скидає поле введення. Назва категорії має бути унікальною.

Щоб переглянути інформацію про програму, виберіть у меню пункт «Про програму». Відкриється форма з назвою програми, автором, роком створення та коротким описом. Натисніть «Закрити», щоб повернутися до попередньої форми.

Перед виконанням дій, таких як видалення замовлення чи товару, уважно перевіряйте вибір у таблиці, оскільки ці операції незворотні. Якщо виникає помилка, наприклад, некоректне введення кількості чи відсутність клієнта, програма відобразить спливне повідомлення з описом проблеми. Для швидкого пошуку використовуйте поля пошуку в формі «Замовлення». Регулярно створюйте резервні копії файлу бази даних, який розташовано в папці програми, щоб уникнути втрати даних.

Програма розроблена для простого освоєння, тому більшість операцій інтуїтивно зрозумілі. У разі виникнення труднощів звертайтеся до цієї інструкції або до адміністратора системи. Система забезпечує швидке виконання основних завдань магазину автозапчастин, таких як оброблення замовлень і управління асортиментом, із мінімальними зусиллями з боку користувача.

Висновки до розділу 3

Розділ містить детальне керівництво для використання інформаційної системи «Магазин автозапчастин», що забезпечує ефективну взаємодію користувачів із програмою. Визначено мінімальні вимоги до системи, які включають бюджетне апаратне забезпечення та операційну систему Windows 10 або новішу з .NET 8 Runtime. Локальна база даних SQLite усуває потребу в серверному програмному забезпеченні, що робить систему доступною для невеликих магазинів автозапчастин із обмеженими ресурсами.

Інтерфейс користувача, побудований на основі Windows Forms, характеризується простотою та інтуїтивністю. П’ять основних форм — «Замовлення», «Товари», «Клієнти», «Категорії» та «Про програму» — забезпечують зручне управління всіма аспектами роботи магазину. Єдиний стиль оформлення, логічне розташування елементів і навігаційне меню сприяють швидкому освоєнню програми навіть для користувачів із мінімальним досвідом. Реактивність інтерфейсу та обробка винятків підвищують зручність і надійність роботи.

Інструкція для роботи з програмою описує покрокові дії для запуску, навігації та виконання основних операцій, таких як створення замовлень, управління товарами, клієнтами та категоріями. Вона враховує потреби працівників магазинів, надаючи чіткі вказівки та рекомендації для уникнення помилок. Система забезпечує швидке виконання бізнес-процесів, мінімізуючи час на навчання та підвищуючи ефективність роботи магазину.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено інформаційну систему «Магазин автозапчастин» на базі технологій Windows Forms, .NET 8 і SQLite, що забезпечує автоматизацію ключових бізнес-процесів невеликих магазинів автозапчастин. Система дозволяє ефективно управляти асортиментом товарів, обробляти замовлення, вести базу клієнтів і категорій, а також контролювати складські запаси.

На етапі аналізу предметної області визначено основні вимоги до системи, проведено огляд аналогічних рішень, таких як «1С:Торгівля і склад», «AutoIntellect» і «Торгсофт», та виявлено їхні недоліки. Розроблена система усуває ці недоліки, пропонуючи економічне, просте у використанні рішення з локальним зберіганням даних.

Проєктування системи включало вибір ефективних алгоритмів для CRUD-операцій, пошуку та обробки замовлень, які забезпечують швидкодію для невеликих обсягів даних. База даних SQLite, спроєктована за принципами третьої нормальної форми, гарантує цілісність і компактність даних. Архітектура програми, побудована на модульних класах (моделі, репозиторії, сервіси, форми), забезпечує гнучкість і можливість розширення функціоналу.

Програмна реалізація включає п’ять форм Windows Forms, які надають зручний інтерфейс для управління всіма аспектами роботи магазину. Інтерфейс є інтуїтивно зрозумілим, що мінімізує час на навчання персоналу.

Розроблена система відповідає поставленій меті, забезпечуючи автоматизацію бізнес-процесів, підвищення ефективності роботи магазину та зниження ймовірності помилок. Вона має практичну цінність для невеликих магазинів автозапчастин і може бути адаптована до інших сфер роздрібної торгівлі шляхом додавання нових функцій, таких як звіти чи інтеграція з зовнішніми системами. Подальший розвиток системи може включати впровадження веб-інтерфейсу або міграцію на серверну базу даних для підтримки більших обсягів даних.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Алфьоров О. М. Основи програмування на C#: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2020. 320 с.
2. Бойко В. В. Проектування баз даних: навч. посіб. Львів: НУ «Львівська політехніка», 2018. 256 с.
3. Гаврилов О. В., Кравець П. О. Розробка програмного забезпечення з використанням .NET: навч. посіб. Київ: НТУУ «КПІ», 2019. 288 с.
4. Гарнаєв А. Ю. SQLite: проектування та використання баз даних. Харків: Фоліо, 2021. 200 с.
5. Григор'єв А. О. Технології Windows Forms для створення настільних застосунків: навч. посіб. Одеса: ОНУ ім. І. І. Мечникова, 2020. 240 с.
6. Дейт К. Дж. Вступ до систем баз даних. 8-е вид. Київ: ДіаСофт, 2019. 896 с.
7. Еванс Дж. Програмування на C# для початківців. Київ: Видавнича група BHV, 2022. 432 с.
8. Зубенко В. В. Сучасні інформаційні системи: теорія та практика: монографія. Дніпро: ДНУ, 2020. 360 с.
9. Іванов С. М. Розробка клієнт-серверних додатків на платформі .NET: навч. посіб. Харків: ХНУ ім. В. Н. Каразіна, 2021. 312 с.
10. Коваленко О. О. Об'єктно-орієнтоване програмування: навч. посіб. Київ: КНУ ім. Тараса Шевченка, 2019. 280 с.
11. Ліберті Дж. Вивчаємо C# через розробку додатків. Київ: Діалектика, 2020. 672 с.
12. Мак-Дональд М. Windows Forms у .NET: практичний посібник. Львів: Видавництво «Магнолія», 2021. 512 с.
13. Нейгл К., Ев'єн Б. Професійне програмування на C# 8 та .NET Core 3. Київ: Видавнича група BHV, 2020. 1248 с.
14. Петров О. П. Алгоритми та структури даних у програмуванні: навч. посіб. Запоріжжя: ЗНУ, 2018. 224 с.

15. Сільбершатц А., Корт Г., Судершан С. Концепції баз даних. 7-е вид. Харків: Книжковий клуб, 2020. 784 с.
16. Таненбаум Е. Сучасні операційні системи. 4-е вид. Київ: ДіаСофт, 2019. 1120 с.
17. Троелсен Е., Джепікс Ф. Мова програмування C# 8 і платформа .NET Core: довідник. Київ: Видавництво «Вільямс», 2021. 1376 с.
18. Фаулер М. Шаблони проєктування програмного забезпечення. Київ: Фактор, 2020. 424 с.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace AutoPartsStore.Forms.Order
{
    partial class OrderForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            txtQuantity = new TextBox();
```