Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Залепи Владислава Володимировича*
(ПІБ)

академічної групи *121-20ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для автоматизації роботи посередницьких фірм зі складами на базі .Net Core з використанням бази даних MSSQL*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*ст. викладач Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20ск-1* *Залепи Владислава Володимировича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для автоматизації роботи посередницьких фірм зі складами на базі .Net Core з використанням бази даних MSSQL*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Спірінцев В.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Залепа В.В*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *14.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 84 с., 35 рис., 10 табл., 3 дод., 20 джерел.

Об'єкт розробки: десктоп-орієнтована система автоматизації ручних процесів.

Мета кваліфікаційної роботи: розробка програмного забезпечення для автоматизації роботи посередницьких фірм зі складами на базі .Net Core з використанням бази даних MSSQL.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку додатку автоматизації процесів, описана робота системи, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої автоматизованої системи, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні та налаштуванні додатку, що автоматизує роботу посередницьких фірм зі складами.

Актуальність розробки даної кваліфікаційної роботи зумовлена різким зростанням дрібних підприємств, що не завжди мають свої склади де зберігати товар, або мають проблеми з їх реалізацією, тому зростає попит на посередницькі фірми. Розроблений додаток допоможе автоматизувати управління внутрішніх процесів для облегшення діяльності фірм, що допоможе заощадити кошти та збільшити прибуток.

Список ключових слів: АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ВНУТРІШНІХ ПРОЦЕССІВ, ДЕСКТОП ДОДАТОК, БАЗА ДАНИХ, SQL, C#, ENTITY FRAMEWORK CORE, ADO NET, ASP NET CORE API, WIN FORMS.

ABSTRACT

Explanatory note: 84 p., 35 figs., 10 tabl., 3 appx., 20 source.

The object of development: a desktop-oriented system for automating manual processes

The purpose of the qualification work: development of software for automation of intermediary firms' work with warehouses based on .Net Core using the MSSQL database.

The introduction discusses the analysis and current state of the problem, specifies the purpose of the qualification work and its scope, provides a justification for the relevance of the topic and clarifies the task statement..

In the first chapter analyzes the subject area, determines the relevance of the task and the purpose of the development, formulates the task statement, and specifies the requirements for software implementation, technologies, and software tools.

In the second section analyzes existing solutions, selects a development platform, designs and develops a process automation application, describes the system operation, algorithm and structure of its functioning, as well as the application call and download, defines input and output data, and characterizes the composition of technical means parameters.

In the economic section determines the labor intensity of the developed automated system, calculates the cost of creating the application, and estimates the time required to create it.

The practical value lies in creating and configuring an application that automates the work of intermediary firms with warehouses.

The relevance of the development of this qualification work is due to the sharp growth of small businesses that do not always have their own warehouses to store goods or have problems with their sale, so the demand for intermediary firms is growing. The developed application will help automate the management of internal processes to facilitate the activities of firms, which will help save money and increase profits.

List of keywords: AUTOMATED INTERNAL PROCESS MANAGEMENT SYSTEM, DESKTOP APPLICATION, DATABASE, SQL, C#, ENTITY FRAMEWORK CORE, ADO NET, ASP NET CORE API, WIN FORMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

ПП Програмний продукт

ІС - Інформаційна система

ОС - Операційна система

БД - База даних

СУБД - Система управління базами даних

EF - Entity Framework Core

SQL - Structured Query Language

ВСТУП

Посередницькі фірми зі складами відіграють важливу роль в економічному житті не лише України, а й у всьому світі. Ці компанії надають послуги зі зберігання та управління товарами, що дозволяє виробникам та продавцям зосередитися на своїй основній діяльності. Значимість посередницьких фірм зі складами полягає у зменшенні витрат. Компанії, які займаються виробництвом та продажом товарів, не обов'язково повинні мати власні склади. Склади є дорогими, що потребують постійного обслуговування та інвестицій. Використання послуг посередницьких фірм зі складами дозволяє компаніям скоротити свої витрати підтримки складів, і навіть витрачати час і ресурси управління складськими процесами. Це спрощує процес управління запасами і допомагає клієнтам уникнути нестачі чи надлишку товарів на складі.

Десктопний додаток та сайт на базі API для посередницьких фірм зі складами можуть допомогти покращити ефективність керування запасами на 30-50%. Автоматизація процесів управління складом дозволяє зменшити час на обробку замовлень та прискорити процес доставки товарів. Це може знизити витрати на оплату робочого часу працівників складу. Додаток дозволяє контролювати рух товарів на складі в реальному часі, що забезпечує точність та надійність управління запасами. Це може знизити рівень втрат продуктів на складі, включаючи товари, що швидко псуються.

Нарешті, десктопне додаток та сайт на базі API можуть допомогти скоротити час на обробку замовлень клієнтів та збільшити швидкість доставки товарів. Вони дозволяють швидко отримувати інформацію про наявність товарів на складі та оформляти замовлення в режимі реального часу.

Загалом система дозволить керувати обігом договорів та завжди обробляти в найближчий термін, що допоможе побудувати надійні партнерські стосунки з клієнтами.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Посередницькі фірми між складами в Україні мають важливе значення для ефективної постачальної ланки. Малий та середній бізнес, що займається виготовленням товарів в більшості випадків не має великих складів для його зберігання, тому такі бізнеси співпрацюють з посередницькими фірмами, які знаходять де зберігати товар і кому його реалізувати. Оптимізація процесів в цій галузі може призвести до зниження витрат і покращення загальної продуктивності, як самих посередницьких фірм так і їх клієнтів.

Задачі, які виникають у посередницьких фірм зі складами і які вирішує дана кваліфікаційна робота включають:

1) Управління запасами: необхідно відстежувати наявність товарів на складах, контролювати рух товарів (прихід та видача), забезпечувати своєчасне поповнення запасів і зниження надлишків.

2) Оптимізація договорів: досягнення ефективності обробки договорів між складами та посередницькими фірмами. Договори завжди повинні бути заключенні, на всі товари, що зазначені на продаж.

3) Контроль якості: система, яка дозволяє перевіряти якість товарів під час поставок і зберігання на складах, виявляти товари терміні дії яких скоро закінчиться або уже закінчився.

4) Облік та звітність: ведення бухгалтерського обліку, завжди необхідно розуміти чи були всі договори успішно оплачені, контроль за боржниками.

Звісно така галузь не може бути без конкуренції і в 2013 році була розроблена система Warehouse Management System для автоматизації вище зазначених процесів. Система працює на ринку давно і має багато переваг в порівнянні з іншими системами, але додаток був побудований під автоматизацію процесів великих посередницьких фірм, які співпрацюють з одними з

найбільших складських мереж у світі, тому цей додаток не підходить для малого і середнього бізнесу.

Переваги розробленого кваліфікаційного додатку:

1) Вартість: додаток пропонує більш доступну ціну порівняно з існуючими рішеннями на ринку.

2) Масштабованість системи може бути адаптованою до різних розмірів і потреб малого і середнього бізнесу.

3) Функціональність: додаток пропонує повний спектр функцій, що включає управління запасами, автоматизацію обробки договорів, контроль якості та облік.

4) Налаштування: ваш додаток дозволяє користувачам налаштовувати його під свої вимоги та процеси.

1.2. Призначення розробки та галузь застосування

Розробка десктопного додатка для автоматизації роботи посередніх фірм зі складською діяльністю є важливим кроком у покращенні ефективності та точності робочих процесів. Галузь застосування таких додатків є широкою, оскільки багато компаній мають складську інфраструктуру та потребують автоматизованого рішення для ефективного управління товарами.

Призначення розробленого додатку:

1) Автоматизація процесів управління запасами. Він надає зручні інструменти для ведення обліку товарів, контролю їх руху та стану на складі, додаток може допомогти відстежувати кількість наявних товарів, контролювати терміни придатності, забезпечувати належне розміщення товарів та швидкий доступ до них.

2) Додаток сприяє покращенню процесів замовлення та постачання товарів. Він дозволяє автоматизувати процес створення та виконання замовлень, відстежувати їх статус, забезпечувати зв'язок з постачальниками, додаток може автоматично генерувати замовлення на підставі попередніх продажів або

попередніх замовлень, а також забезпечувати миттєвий оновлення інформації про поставки та доступність товарів.

3) Забезпечує точність, ефективність та зберігає час, дозволяючи фірмам просуватися на автоматизованих процесах, зменшувати помилки та покращувати загальну продуктивність роботи. Наприклад, додаток може аналізувати дані про попит на товари, історію продажів та рівень запасів, щоб автоматично генерувати замовлення на поповнення запасів відповідно до потреб компанії. Це дозволяє уникнути недостачі або перезапису товарів, забезпечуючи оптимальну кількість на складі і зменшуючи затрати.

4) Додаток забезпечує ведення віртуальної картотеки складу, де зазначена точна локація кожного товару. Додаток може збирати, обробляти та візуалізувати дані про продажі з картотеки, запаси, витрати та інші показники. Це дозволяє менеджерам та власникам фірм отримувати усвідомлені рішення на основі точних даних, а також відстежувати тенденції та прогнозувати результати.

5) За допомогою зручних звітів та графіків, десктопний додаток може представляти інформацію у зрозумілій формі, що дозволяє здійснювати аналіз та приймати стратегічні рішення. Окрім прикладів застосування, важливо також зазначити загальні вимоги до десктопних додатків.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
- завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення для автоматизації роботи посередницьких фірм зі складами на базі .Net Core з використанням бази даних MSSQL».

1.4. Постановка завдання

В даній кваліфікаційній роботі розглядається створення додатку для автоматизації роботи посередницьких фірм зі складами. Необхідно розробити систему, яка підвищить ефективність роботи фірм для збільшення прибутку. Також створений додаток повинен містити базу даних для зберігання всіх операцій в системі.

Розробити функціонал:

– можливість ведення обліку товарів, що поступили на склад і ті, що відвантажились;

– функціонал контролю якості товарів;

– функціонал оплати за поставлену і відвантажену продукцію повинна відображатися у звіті;

– функціонал укладання та контроль договорів на постачання товарів і встановлення посередницького відсотку. У договорі повинні міститися реквізити замовника і постачальника, предмет, необхідні відомості і терміни поставки;

– функціонал ведення звітності.

Створений додаток повинен містити наступні вимоги:

– постачальник зобов’язується поставити, а замовник прийняти та оплатити продукцію;

– кількість і асортимент продукції передбачаються сторонами в узгодженої специфікації, що є невід’ємною частиною договору;

– кількість товару вказується в специфікації на товари;

– перелік товару, вказується в специфікації на товари;

– покупець оплачує товари за цінами, що діють на момент покупки; визначення цілей створення системи та затвердження завдання на її розробку;

– можливість продавати або постачати товар термін дії яких не закінчився;

– покупець може вносити оплату частинами;

– покупець може замовити в рамках одного договору тільки однин товар.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставленої в роботі мети в додатку, що розробляється, повинні бути реалізовані наступні особливості:

– якісна та зручна можливість клієнтами отримати інформацію про товар, договори, постачальників, склади, покупців;

– сервер повинен видавати інформацію любого об’єму не більше ніж за 3 секунди;

– база даних повинна автоматично створювати реплікацію;

– дані, що потрапляють в додаток повинні проходити валідацію на кожному рівні;

– сервер не повинен обробляти запити неавторизованих користувачів, а в разі неавторизованого запиту видавати помилку;

– API повинні працювати за REST правилами;

– зручне управління додатку для користувачі додатку, що буде інтуїтивно зрозумілим і не займе багато часу при навчанні нового користувача;

– додаток повинен підтримувати протокол HTTP 2.0;

– передача запитів повинна шифруватися і мати SSN сертифікат.

1.5.2. Вимоги до інформаційної безпеки

Вимоги до безпеки додатка є надзвичайно важливими, оскільки забезпечення безпеки даних є критично важливою задачею. Варто зазначити деякі можливі проблеми, які можуть виникнути, а також шляхи їх усунення. Одна з основних проблем пов'язаних з безпекою - це загрози кібербезпеці. Наприклад, зловмисники можуть спробувати несанкціоновано отримати доступ до системи, викрасти конфіденційну інформацію або внести зміни в базу даних. Інша проблема - втрата даних. Наприклад, можуть виникнути проблеми з

жорстким диском або відбутися непередбачувана помилка, що призведе до втрати важливої інформації. Також важливо враховувати ризик внутрішньої загрози, тобто можливість виникнення проблем через недбалість або зловживання з боку власних працівників.

Для запобігання вищезазначених проблем в даній кваліфікаційній роботі необхідно буде:

– встановити ефективні механізми аутентифікації та авторизації користувачів, шифрування даних та застосування систем виявлення вторгнень;

– регулярно робити резервні копії даних та використовувати механізми відновлення системи;

– встановити обмеження доступу до конфіденційної інформації та вести контроль за використанням привілеїв користувачів.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для нормального функціонування системи, повинні виконуватися певні вимоги до технічних засобів, такі як:

– процесор з тактовою частотою не менше 1,8 ГГц;

– відеокарта з підтримкою браузера Google Chrome або Mozilla Firefox, які є рекомендованими браузерами для використання з відкритими API даної системи;

– операційна система повинна бути Windows 7 або новіше;

– наявність не менше 4 ГБ оперативної пам'яті;

– встановлений SQL Server 2026 або новішої версії;

– встановлений Windows Server 2012 або новішої версії;

– встановлений Microsoft IIS 2018 або новішої версії.

Для забезпечення безпеки даних необхідно використовувати антивірусне програмне забезпечення та забезпечити регулярні оновлення системи та додаткових програмних засобів.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати десктоп-орієнтована підсистема, відповідало наступним вимогам:

- операційна система Microsoft Windows XP/7/8/10/11;
- браузер Інтернет (Microsoft Internet Explorer, MozillaFireFox, Opera, Google Chrome);
- IIS сервер з підтримкою API.

Система має бути реалізовано на мові програмування C#, для роботи API використовувати ASP NET CORE, для взаємодії з БД використовувати СУБД ADO NET, для написання запитів в БД використовувати SQL, також для реалізацію користувацького інтерфейсу використовувати Win Forms.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути додаток з відкритими API для взаємодії з іншими сервісами на мові C# з застосуванням фреймворку ASP NET Core для автоматизації роботи посередницьких фірм зі складами.

Розроблений додаток повинен спростити роботу фірм їх взаємодію зі складами та автоматизувати ручні процеси, що покращить усі показники ефективності фірм.

Для реалізації поставлених цілей з покращенням роботи фірм та складів додаток буде містити наступний функціонал:

– функціонал створення фірм, складів, товару;
– можливість додавання мір вимірювання та варіантів оплати;
– можливість перегляду інформації про фірми, склади, товари;
– функціонал додавання, підписання та контролю договорів;
– можливість контролювати якість товару та слідкувати за терміном придатності товарів;
– функціонал оплати частинами за товар та ведення обліку боржників;
– можливість закривати договори, раніше зазначеного терміну;
– можливість створення звітів.

2.2. Опис застосованих математичних методів

Під час проектування та розробки даної програми використовувалися тільки арифметичні дії, такі як, додавання, віднімання, множення, ділення. Також були використані бібліотеки з алгоритмами, та готовими методами.

2.3. Опис використаної архітектури та шаблонів проектування

В якості основного архітектурного шаблону проектування додатку було обрано тришаровий архітектурний підхід. Проаналізувавши вимоги та поставленні цілі завдяки 3-layer архітектурі можливо розробити додаток, який буде виконувати поставлені цілі [1].

Тришарова архітектура - це підхід до розробки програмного забезпечення, який використовується для побудови складних додатків. Вона складається з трьох основних компонентів або шарів: представлення (інтерфейсного шару), логіки додатку та доступу до даних. Кожен шар виконує певні функції і має свої відповідальності (рис. 2.1).

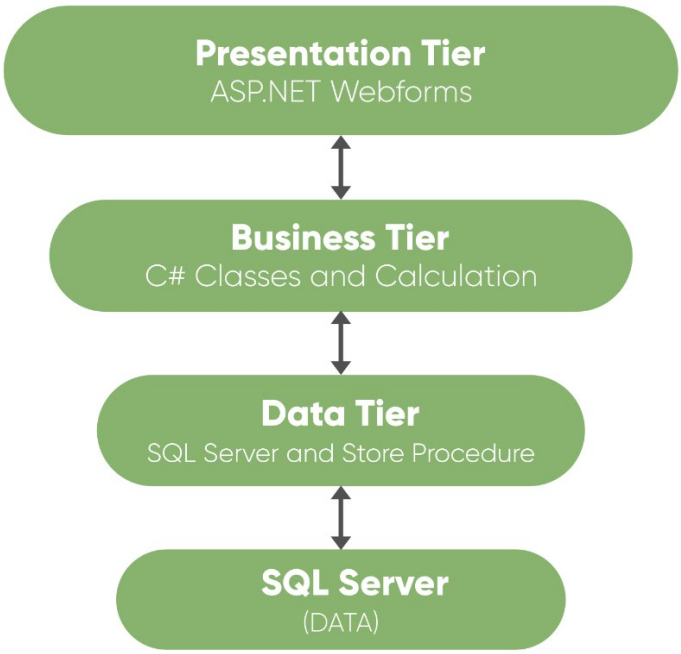


Рис. 2.1. Схема 3-Layer Architecture

Представлення – це шар, який відповідає за відображення інформації користувачеві і взаємодію з ним. Він складається з усіх елементів, які користувач може бачити і з якими може взаємодіяти, таких як графічний інтерфейс, кнопки,

форми, меню тощо. Основна перевага цього шару полягає в тому, що він відокремлений від решти додатку, тобто зміни в інтерфейсному шарі не впливають на логіку додатку або базу даних. Це дає можливість використовувати різні технології для розробки інтерфейсу, такі як HTML/CSS для веб-додатків або фреймворки для десктопних додатків, і змінювати їх незалежно від решти системи.

Логіка - цей шар містить основну функціональність додатку і виконує бізнес-логіку. Він обробляє запити користувача, здійснює обробку даних і приймає рішення щодо виконання конкретних дій. Логіка додатку може бути реалізована за допомогою різних мов програмування, таких як Java, Python, C# тощо, залежно в залежності від вимог і уподобань розробників. Основна перевага цього шару полягає в тому, що він відокремлений від інтерфейсного шару і шару доступу до даних. Це дозволяє змінювати логіку додатку без впливу на інтерфейс або базу даних. Крім того, це спрощує тестування, розширення та підтримку додатку, оскільки логіка зосереджена в одному місці і може бути легко модифікована без впливу на інші компоненти.

Доступ до даних - цей шар відповідає за збереження та отримання даних з бази даних або інших джерел даних. Він забезпечує взаємодію з базою даних, виконує запити, зберігає та оновлює дані. Шар доступу до даних дозволяє відокремити роботу з даними від логіки додатку та інтерфейсу, що спрощує розробку і забезпечує більшу гнучкість. Це також дозволяє використовувати різні технології для роботи з базами даних, такі як SQL або NoSQL, і змінювати їх безпосередньо впливу на решту додатку.

Переваги використання такої тришарової архітектури для написання десктопного додатку включають:

1) Розділення відповідальностей дозволяє чітко розділити функціональні області додатку, такі як інтерфейс, логіка та доступ до даних. Це сприяє кращому організації коду, полегшує розробку, тестування і підтримку додатку.

2) Масштабованість для відокремлення шарів, можливе масштабування окремих компонентів додатку.

3) Підтримка багатоплатформенності при необхідності можливо розгорнути на різних платформах, таких як Windows, macOS або Linux, тришарова архітектура дозволяє зберегти логіку та доступ до даних незмінними, переносячи тільки інтерфейсний шар для кожної платформи. Це спрощує підтримку та розвиток додатку на різних операційних системах.

4) Забезпечення безпеки шарів дозволяє використовувати різні механізми забезпечення безпеки для кожного шару. Наприклад, можна встановити захист доступу до бази даних на рівні шару доступу до даних або забезпечити аутентифікацію та авторизацію на рівні логіки додатку. Це допомагає зберігати дані в безпеці і уникати можливих порушень безпеки.

5) Розділення обов'язків команди розробників: Тришарова архітектура дозволяє розділити роботу між різними командами розробників, кожна з яких може фокусуватись на своїй області відповідальності. Наприклад, команда дизайнерів може працювати над інтерфейсним шаром, команда програмістів - над логікою додатку, а команда бази даних - над шаром доступу до даних. Це підвищує продуктивність розробки і сприяє ефективному використанню ресурсів [2].

Узагальнюючи, використання тришарової архітектури для написання десктопного додатку надає багато переваг, таких як модульність, розширюваність, легкість тестування та підтримки. Ця архітектура дозволяє краще організувати розробку, забезпечуючи чітке розподіл функцій між різними компонентами додатку.

2.4. Опис використаних технологій та мов програмування

При створенні додатку використовувались технології:

1) ASP NET Core.
2) Win Forms.
3) ADO NEN.
4) Entity Framework Core.

5) MSSQL.

ASP.NET Core є відкритим фреймворком для розробки десктопних додатків, який має безліч переваг для розробників. Одна з ключових переваг ASP.NET Core полягає в його кросплатформенності. Це означає, що є можливість розробляти десктопні додатки, які працюватимуть як на Windows, так і на macOS або Linux. ASP.NET Core має свій певний життєвий цикл як це видно з рис. 2.1.

Рис. 2.2. Схема життєвого циклу ASP NET Core

ASP.NET Core також відомий своєю високою продуктивністю. Він має ефективну архітектуру та оптимізовані механізми, які дозволяють додаткам працювати швидко та ефективно навіть при великому обсязі даних та високих навантаженнях. Фреймворк також забезпечує високий рівень безпеки для десктопних додатків. Він має вбудовану підтримку аутентифікації та авторизації, що дозволяє контролювати доступ до функціональності додатку. Крім того, він

має механізми для захисту від типових атак, таких як міжсайтовий скриптинг або вставку SQL-запитів, також дозволяє використовувати сучасні технології розробки, такі як веб-сокети, веб-апі, мікрослужби та контейнеризація з можливістю легко інтегрувати фронтенд-фреймворки, такі як Angular, React або Vue.js, для створення сучасних та інтерактивних інтерфейсів користувача [3].

ASP.NET Core також підтримує розширений набір інструментів для тестування десктопних додатків, що дає можливість легко створювати автоматичні тести, інтегрувати їх з фреймворками тестування та забезпечувати якість та надійність вашого додатку.

Один з істотних аспектів ASP.NET Core - це його активна спільнота розробників та підтримка з боку Microsoft. Існує безліч онлайн-ресурсів, документації, форумів та блогів, де можливо знайти відповіді на питання, отримати допомогу та ділитися знаннями з іншими розробниками, а для розробки десктопних додатків є його масштабованість та стабільність. Він використовується в реальних проектах різних масштабів та складності, забезпечуючи надійну та стабільну роботу додатків навіть під високими навантаженнями. ASP.NET Core є потужним фреймворком для розробки десктопних додатків, який надає розробникам безліч переваг. Він є кросплатформеним, продуктивним, модульним, безпечним та підтримує сучасні технології розробки. З великою спільнотою розробників та підтримкою від Microsoft, ASP.NET Core є надійним та масштабованим вибором для написання десктопних додатків [4].

WinForms (Windows Forms) є одним з найпопулярніших фреймворків для розробки десктопних додатків під операційну систему Windows. Цей фреймворк надає розробникам безліч переваг і можливостей для написання десктопних додатків, які працюють в операційній системі Windows.

Одна з ключових переваг WinForms - його простота використання та розробки. Фреймворк надає розробникам інтуїтивно зрозумілий спосіб створення графічного інтерфейсу користувача за допомогою візуального редактора [5]. Він дозволяє просто перетягувати та розміщувати елементи

керування на формі, налаштовувати їх властивості та оброблювати події за допомогою простого коду. Це дозволяє швидко реалізувати ідеї та прототипи додатків. Ще однією перевагою WinForms є його широка підтримка функціональності та елементів керування. Фреймворк надає велику кількість вбудованих елементів керування, таких як кнопки, тексти, списки, таблиці, діалогові вікна тощо, що дозволяє легко створювати власні елементи керування або використовувати сторонні компоненти для додаткової функціональності. Це дозволяє створювати різноманітні та багатофункціональні десктопні додатки.

WinForms також відомий своєю високою продуктивністю та швидкістю виконання. Фреймворк працює безпосередньо з операційною системою Windows, що дозволяє використовувати всі переваги, які надає операційна система, та забезпечує ефективну роботу з ресурсами системи. Це особливо важливо для десктопних додатків, які можуть виконувати складні операції або працювати з великими обсягами даних. WinForms забезпечує ефективне використання ресурсів системи та мінімізує затримки та простої в роботі додатку [6].

ADO .NET - набір засобів і шарів, що дозволяють з додатком легко управляти і взаємодіяти зі своїм файловим або серверним сховищем даних (рис. 2.3).

Рис. 2.3. Схема взаємодії програми з БД за допомогою ADO.NET

DataSet, DataTable, Connection, Command, DataReader, DataAdapter - це компоненти, спеціально сконструйовані для обробки даних і швидкого, однопрохідного доступу до даних тільки для читання. Вони можуть забезпечити доступ до даних незалежно від джерела даних, обмін даними,

високопродуктивний потік даних з джерела, дозволяють звертатися до команд БД для повернення, зміни даних, виконання збережених процедур і передачі,отримання відомостей про параметри [7]. Схема ілюструє зв'язок між постачальником даних .NET Framework і DataSet (рис. 2.4).

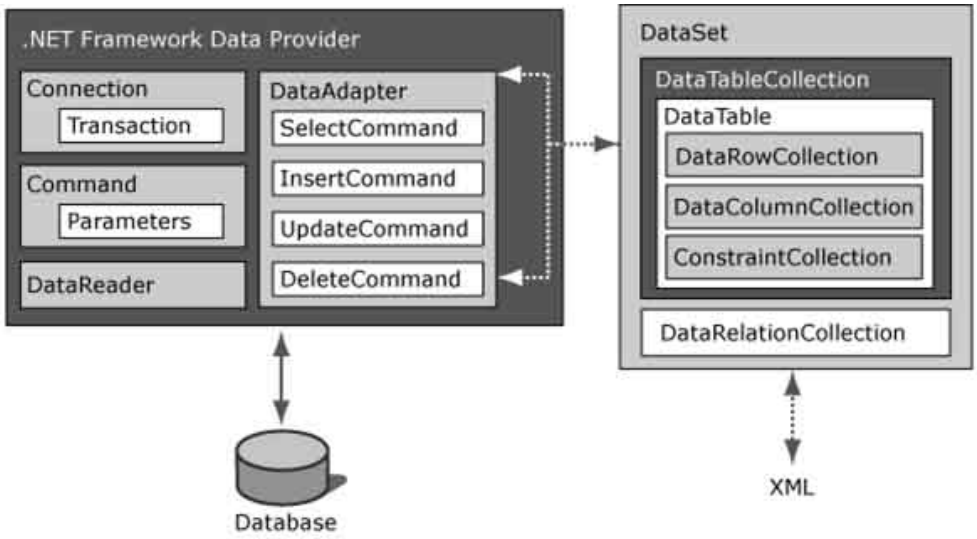


Рис. 2.4. Схема зв’язку між постачальником даних.NET Framework і DataSet.

З'єднання в додатку, створеному за допомогою ADO .NET, встановлюється лише на той час, який необхідно для проведення операції над БД. Це забезпечує автономний доступ до даних - не потрібно утримувати відкрите фізичне підключення до БД кожного окремого користувача або об'єкта. Додаток може мати відкрите підключення і спільно використовувати його з іншими користувачами тому в ADO .NET введений пул підключень. У ADO .NET реалізований потужний механізм підтримки високопродуктивних транзакцій БД, які відповідають за коректність змін в БД.

ADO.NET - це прогресивна технологія, в повній мірі задовольняє необхідність в інструментальних засобах при створенні програми, що працює з реляційної БД.

Для зберігання даних використовуються різні системи управління базами даних: MS SQL Server, Oracle, MySQL і так далі. І більшість великих додатків

так чи інакше використовують для зберігання даних ці системи управління базами даних. Однак щоб здійснювати зв'язок між базою даних і додатком на C# необхідний посередник [8]. І саме таким посередником є технологія ADO.NET (рис. 2.5).

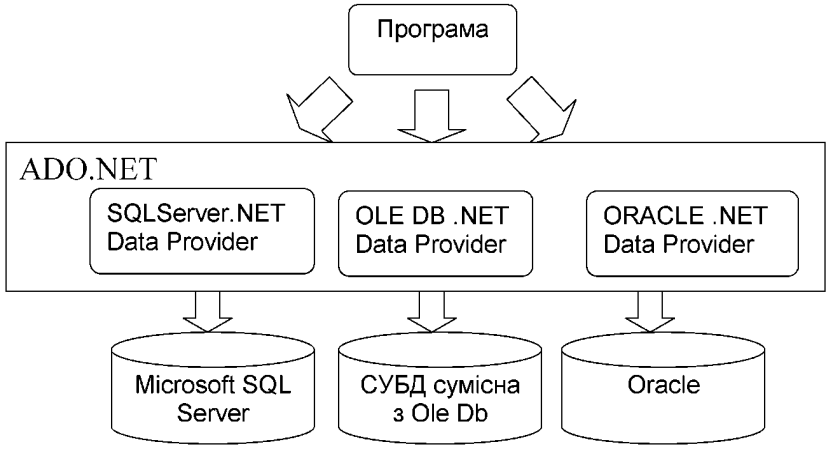


Рис. 2.5. Постачальники даних в ADO.NET

Entity Framework Core є сучасним ORM (Object-Relational Mapping) фреймворком для роботи з базами даних в десктопних додатках. Він надає розробникам безліч переваг та зручностей для написання десктопних додатків.

Однією з головних переваг Entity Framework Core є його спрощений підхід до взаємодії з базами даних. Фреймворк дозволяє розробникам працювати з базою даних за допомогою об'єктно-орієнтованого підходу, замість прямої роботи з SQL-запитами, що дозволяє створювати моделі об'єктів, які відповідають сутностям бази даних, і взаємодіяти з ними за допомогою простих методів та властивостей. Це спрощує розробку та підтримку бази даних в десктопних додатках [9].

Ще одною перевагою Entity Framework Core є його підтримка різних провайдерів баз даних. Фреймворк підтримує різні типи баз даних, такі як SQL Server, MySQL, PostgreSQL та інші. Це означає, що можливо використовувати

однаковий код для роботи з різними типами баз даних, що забезпечує гнучкість та переносимість вашого додатку.

Entity Framework Core також надає потужні інструменти для роботи з даними, такі як міграції бази даних, що дозволяє легко створювати та застосовувати міграції, що дозволяє автоматично оновлювати схему бази даних при зміні моделей об'єктів [10]. Це спрощує процес розвитку додатку та дозволяє ефективно керувати версіями бази даних (рис. 2.6).

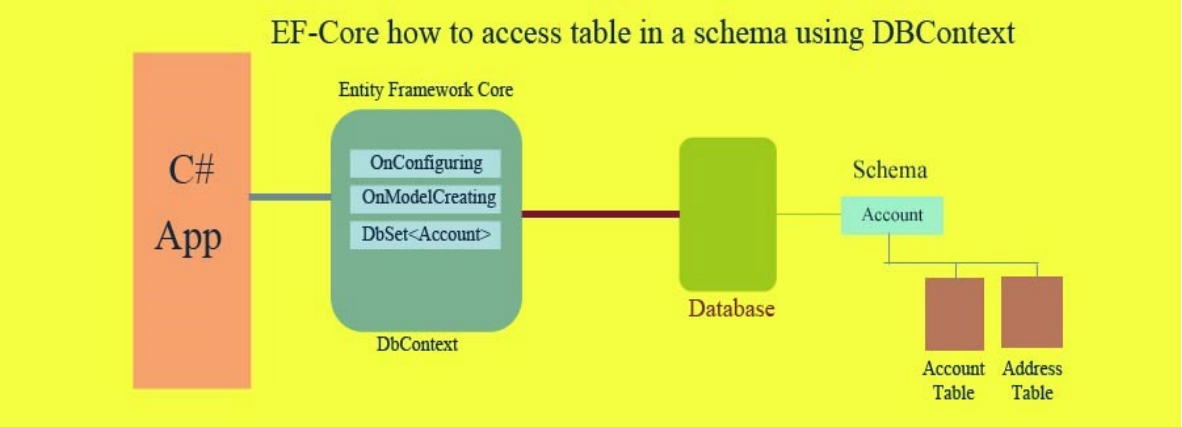


Рис. 2.6. Схема роботи Entity Framework Core

Microsoft SQL Server (MSSQL) є однією з найпопулярніших систем управління базами даних (СУБД) для розробки десктопних додатків. Ця реляційна база даних надає безліч переваг та зручностей для написання десктопних додатків.

Ключових переваг MSSQL є його висока надійність та стабільність. Вона розроблена компанією Microsoft, яка має багаторічний досвід у сфері розробки програмного забезпечення. MSSQL відома своєю стабільністю, оптимізацією та масштабованістю, що робить її відмінним вибором для десктопних додатків з великим обсягом даних та вимогами до продуктивності [11].

Ще одною перевагою MSSQL є його широкий функціонал та розширені можливості. Вона підтримує різні типи даних, запити SQL, транзакції, збережені процедури, функції та індекси, що дозволяє розробникам створювати потужні та

складні запити до бази даних. MSSQL також надає різноманітні інструменти для адміністрування, моніторингу та оптимізації бази даних, що спрощує роботу з нею та підвищує її продуктивність.

Іншою перевагою MSSQL є його інтеграція з екосистемою Microsoft. Вона нативно підтримується платформою .NET, що дозволяє розробникам легко взаємодіяти з базою даних за допомогою .NET-технологій, таких як ADO.NET або Entity Framework. MSSQL також інтегрується з іншими інструментами Microsoft, такими як Visual Studio, Azure, Power BI та інші [12].

2.5. Опис структури програми та алгоритмів її функціонування

Нижче приведена термінологія (сутності), яка використовується в даній предметній області:

– постачальник – перелік інформації про постачальників;

– продукти – перелік інформації про продукти;

– покупець – перелік інформації про покупців;

– договори – перелік інформації про договори на постачання та продаж;

– замовлення – перелік інформації про замовлення на постачання та продаж.

В ході нормалізації бази даних додались сутності:

– спосіб постачання – перелік інформації про способи постачання продукції;

– спосіб оплати – перелік інформації про способи оплати за замовлення продуктів;

– склад – перелік інформації про кількість товару на складі;

– одинця фізичної величини – інформація про одиниці фізичних величин.

При проектування використалась реляційна модель бази даних. Враховувалися аномалії видалення та додавання.

Всі данні зберігаються у таблицях бази даних. Таблиця – це об’єкт, призначений для збереження даних у вигляді записів (рядків) та полів (стовпців).

Звичайно кожна таблиця використовується для збереження відомостей по одному конкретному питанню, наприклад о предметах або спеціальностях.

Кожне поле таблиці має свій тип. При проектуванні таблиць використалися наступні типи даних та позначення:

1) A – символьне значення до 255 символів.

2) C – лічильник - автоматична вставка унікальних послідовних (що збільшуються на 1) або випадкових чисел при додаванні запису.

3) D – дата та час.

4) F – дійсне число.

При встановленні зв'язку між таблицями використалися наступні позначення:

1) PK – (Primary key) первинний ключ – унікальне поле.

2) FK – (Foreign key) зовнішній ключ – набор атрибутів одного відношення, яке є потенційним ключем другого відношення.

Нижче наведений опис сутностей у вигляді реляційних таблиць (таблиця 2.1- 2.10).

Таблиця 2.1

Схема відносин постачальник (Supplier)

Зміст поля	Ім'я поля	Тип, довжина	Примітки
Ідентифікатор постачальника	IDSupplier	N(1000)	Первинний ключ
Ім'я	Name	C(15)	Обов'язкове поле
По-батькові	Patronymic	C(15)	Обов'язкове поле
Прізвище	Surname	C(20)	Обов'язкове поле
Місто	City	C(30)	Обов'язкове поле
Вулиця	Street	C(30)	Обов'язкове поле
Будинок	House	N	Обов'язкове поле

Таблиця 2.2

Схема відносин договір на постачання (DeliveryContract)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор договору на поставку	IDDeliveryContrct	N	Первинний ключ
Ідентифікатор постачальника	IDSupplier	N	Зовнішній ключ
Дата договору на постачання	Date	Date	Обов’язкове поле
Спосіб доставки	IDDeliveryMethod	N	Зовнішній ключ
Тип оплати	IDPayment	N	Зовнішній ключ
% посередника	Percentage	F(18,1)	Обов’язкове поле

Таблиця 2.3

Схема відносин замовлення-постачання на постачання (OrderDelivery)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Номер замовлення	IDOrderDelivery	N	Первинний ключ
Номер договору на постачання	IDDeliveryContrct	N	Зовнішній ключ
Ідентифікатор товару	IDProduct	N	Зовнішній ключ
Кількість	Quantity	N	Обов’язкове поле
Сума договору	MoneyContract	Money	Обов’язкове поле

Таблиця 2.4

Схема відносин товар на постачання (Product)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор товару	IDProduct	N	Первинний ключ
Назва товару	NameProduct	C(20)	Обов’язкове поле
Закупочна ціна за одиницю	UnitPrice	Money	Обов’язкове поле
Номер складу	IDWarehouse	N	Зовнішній ключ
Одиниця фізичної величини	PhysicaQuantity	N	Обов’язкове поле

Таблиця 2.5

Схема відносин замовлення-продаж на постачання (OrderSale)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Номер замовлення	IDOrderSale	N	Первинний ключ
Номер договору на продаж	IDSalesContract	N	Зовнішній ключ
Ідентифікатор товару	IDProduct	N	Зовнішній ключ
Кількість	Quantity	N	Обов’язкове поле
Сума договору	MoneyContract	Money	Обов’язкове поле

Таблиця 2.6

Схема відносин договір на продаж (SalesContract)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор договору на продаж	IDSalesContract	N	Первинний ключ
Ідентифікатор замовника	IDCustomer	N	Зовнішній ключ
Дата договору на продаж	Date	Date	Обов’язкове поле
Спосіб доставки	IDDeliveryMethod	N	Зовнішній ключ
Тип оплати	IDPayment	N	Зовнішній ключ

Таблиця 2.7

Схема відносин замовник (Customer)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор замовника	IDCustomer	N	Первинний ключ
Ім’я	Name	C(15)	Обов’язкове поле
По-батькові	Patronymic	C(15)	Обов’язкове поле
Прізвище	Surname	C(20)	Обов’язкове поле
Місто	City	C(30)	Обов’язкове поле
Вулиця	Street	C(30)	Обов’язкове поле
Будинок	House	N	Обов’язкове поле
Номер_країни	CountryNumber	N	Обов’язкове поле
Рахунок	Score	C(20)	Обов’язкове поле
МФО	MFO	N	Обов’язкове поле

Таблиці, що з’явилися після нормалізації:

Таблиця 2.8

Схема відносин спосіб доставки (DeliveryMethod)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор способу	IDDeliveryMethod	N	Первинний ключ
Назва	NameMethod	C(15)	Обов’язкове поле

Таблиця 2.9

Схема відносин спосіб оплати (TypeOfPayment)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор способу	IDTypeOfPayment	N	Первинний ключ
Назва	NamePay	C(15)	Обов’язкове поле

Таблиця 2.10

Схема відносин склад (Warehouse)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор складу	ID Warehouse	N	Первинний ключ
Назва	Name Warehouse	C(15)	Обов’язкове поле
Товар	NameProd	C(20)	Обов’язкове поле
Кількість	Quantity	N	Обов’язкове поле

Для відображення формалізації інформації о сутностях та їх відношення застосовують діаграми "сутність – зв'язок" (Entity-Relationship Diagrams, ERD).

ER-діаграма містить інформацію про сутності системи та способи їхньої взаємодії, включає ідентифікацію об'єктів, важливих для предметної області (сутностей), властивостей цих об'єктів (атрибутів) і їхніх відносин з іншими об'єктами (зв'язків), тобто модель база даних відображає її логічну структуру, показує поля кожної таблиці, ключі, зв’язки між таблицями (рис. 2.7).

Ключ (ключове поле) – це поле таблиці бази даних, яке однозначно ідентифікує запис таблиці. на схемі ключові поля відмічені як PK (Primary key - первинний ключ) та FK (Foreign key - зовнішній ключ) [13].

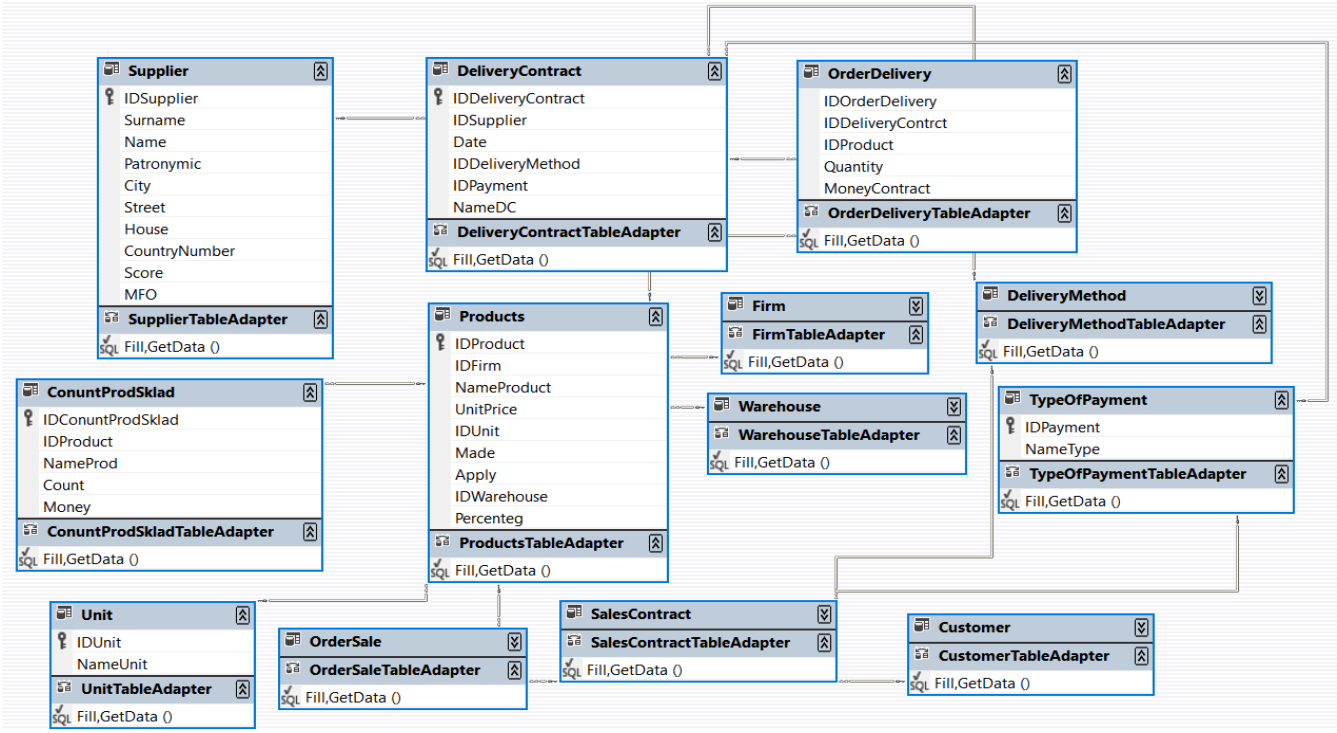


Рис. 2.7. ER-діаграма

Стрілки на схемі, які ідуть від поля одної таблиці до другої, означає зв'язок один-до-багатьох між батьківської та дочірньою таблицями (одному запису батьківської таблиці може відповідати декількох записів дочірньої таблиці). Для зручної обробки даних таблиць бази даних створюємо ескізи віконних форм з відповідними елементами керування та відображення БД у інструментальному

середовищі візуального програмування MS Visual Studio 2019 та технології Windows Forms (рис. 2.8):

1) Фірма (main) – головна форма програми. Виклик другорядних вікон програмиж.

2) Підключення (conection) – завантаження файлу бази даних відповідного формату.

3) Постачальник (suppler) – форма перегляду списку постачальників, перегляд та корегування даних.

4) Договір на постачання (DeliveryContract) – форма для перегляду списку договорів на постачання товару.

5) Замовлення на постачання (OrderDelivery) – форма для перегляду списку замовлення на постачання товару.

6) Товар (products) – форма перегляду списку продуктів.

7) Замовлення на купівлю (SaleDelivery) – форма для перегляду списку замовлення на покупку товару.

8) Договір на купівлю (SaleContract) – форма для перегляду списку договорів на покупку товару.

9) Покупець (customer) – форма для відображення списку покупців.

Рис. 2.8. Головне меню

Також створена файлова структура згідно архітектурному шаблону проектування додатків 3 layers (рис. 2.9). Кожен шар відповідає за виконання певної зони відповідальності і інкапсулює свою роботу від інших.

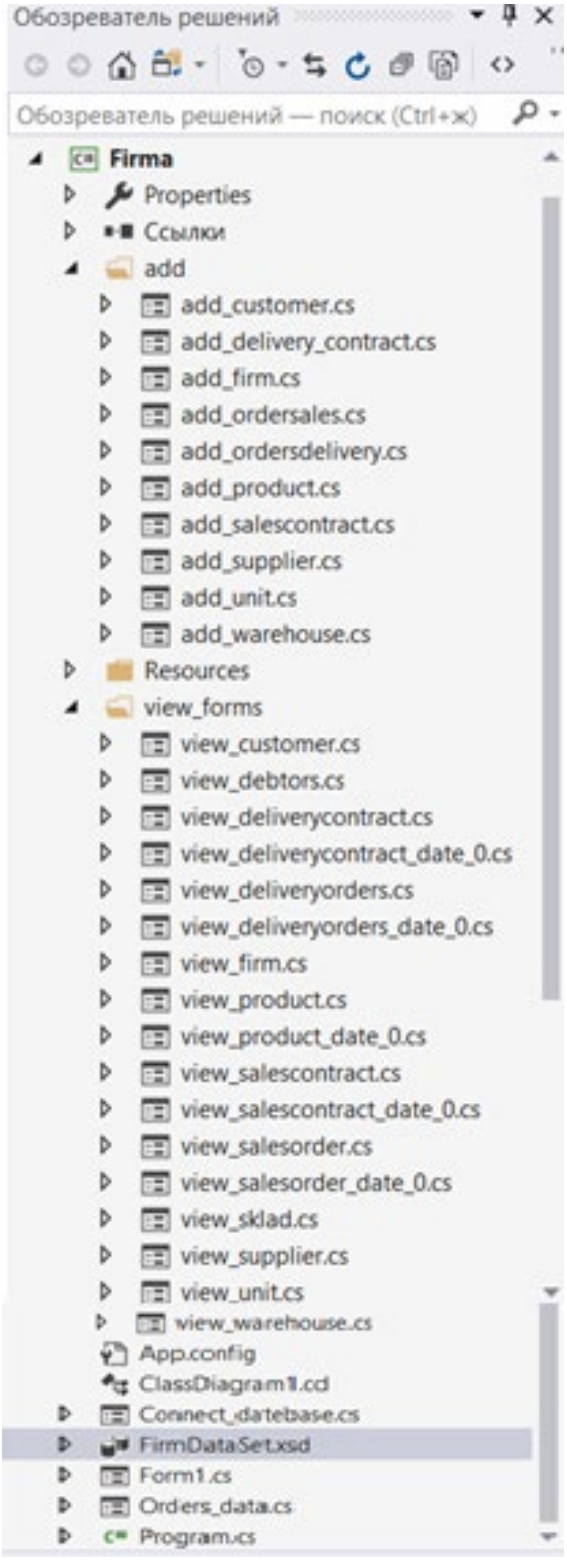


Рис. 2.9. Файлова структура проекту

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними додатку є фірми, товари, склади та покупці які користувач повинен заповнити для повноцінної роботи програми. Також вхідними даними можуть буди одиниці вимірювання, що слугують при додаванні товару, спосіб оплати, спосіб доставки і тд.

Вихідними даними є укладені договори між постачальником та складом, або між складом та покупцем; сформовані звіти на остачу товару на складах; інформація про товари з обмеженим терміном придатності, товари які закінчилися і їх потрібно поповнювати.

Так як додаток має відкриті API для взаємодії з іншими сервісами для обміну між ними використовується REST підхід, а в якості передачі даних між серверами використовується JSON формат.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для повноцінної роботи програми необхідний персональний комп’ютер або ноутбук з операційною системою Windows v.7/v.8.1/10/11. Конфігурація комп’ютеру мінімально повинна бути такою:

- процесор Pentium з частотою 1.6 ГГц або швидший;
- не менш 2ГБ оперативної пам'яті;
- не менше 750 МБ вільного місця на жорсткому диску;
- не менше 250 МБ відеопам'яті;
- клавіатура, миша Microsoft Mouse чи сумісний вказівний пристрій;
- монітор з розподільною здатністю мінімально1024x768 px.

2.7.2. Використані програмні засоби

При розробці даного додатку були використані такі програмні застосунки:

1) Visual Studio.
2) SQL Server Management Studio.
3) Git.

Visual Studio - це інтегроване середовище розробки (Integrated Development Environment, IDE), розроблене компанією Microsoft. Це потужний інструмент, який дозволяє програмістам створювати, редагувати та відлагоджувати програмне забезпечення на різних мовах програмування [14].

Visual Studio надає багато функцій та можливостей для ефективної розробки програм. Воно підтримує широкий спектр мов програмування, таких як C++, C# та багато інших.

У Visual Studio є інтегрована система керування версіями, що дозволяє відстежувати зміни в коді, спілкуватися з іншими розробниками та об'єднувати їхні зусилля [15]. Воно також надає засоби для відлагодження програми, профілювання її продуктивності та автоматизованого тестування (рис. 2.10).

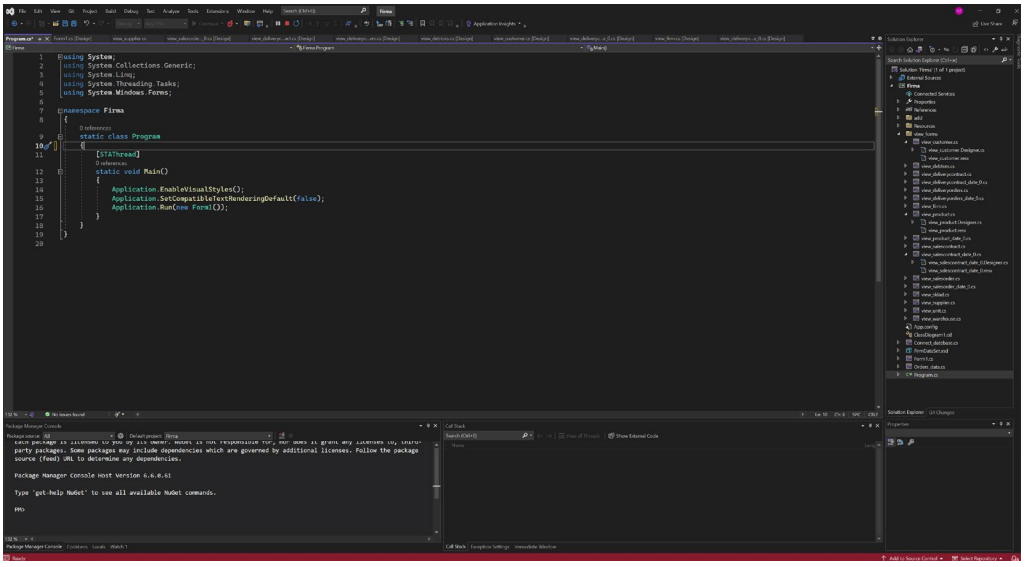


Рис. 2.10. Головне меню Visual Studio

SQL Server Management Studio (SSMS) - це інтегроване середовище керування базами даних, розроблене компанією Microsoft. Воно призначене для роботи з СУБД Microsoft SQL Server і надає розширені функції для управління, адміністрування та розробки баз даних [16].

SSMS дозволяє розробникам та адміністраторам зручно працювати з базами даних. Воно надає можливість створювати нові бази даних, таблиці, збережені процедури, функції та інші об'єкти баз даних. Також воно дозволяє виконувати запити до баз даних, редагувати дані, виконувати операції з резервним копіюванням та відновленням баз даних.

SSMS забезпечує потужні засоби для управління безпекою баз даних, дозволяючи налаштовувати доступ до об'єктів баз даних для користувачів та ролей. Воно також надає можливість моніторити продуктивність баз даних, аналізувати запити та виявляти проблеми з продуктивністю [17].

Інтерфейс SSMS простий у використанні, зручний та функціональний. SSMS має розширені можливості для налаштування та персоналізації, що дозволяє користувачам адаптувати середовище до своїх потреб (рис. 2.11).

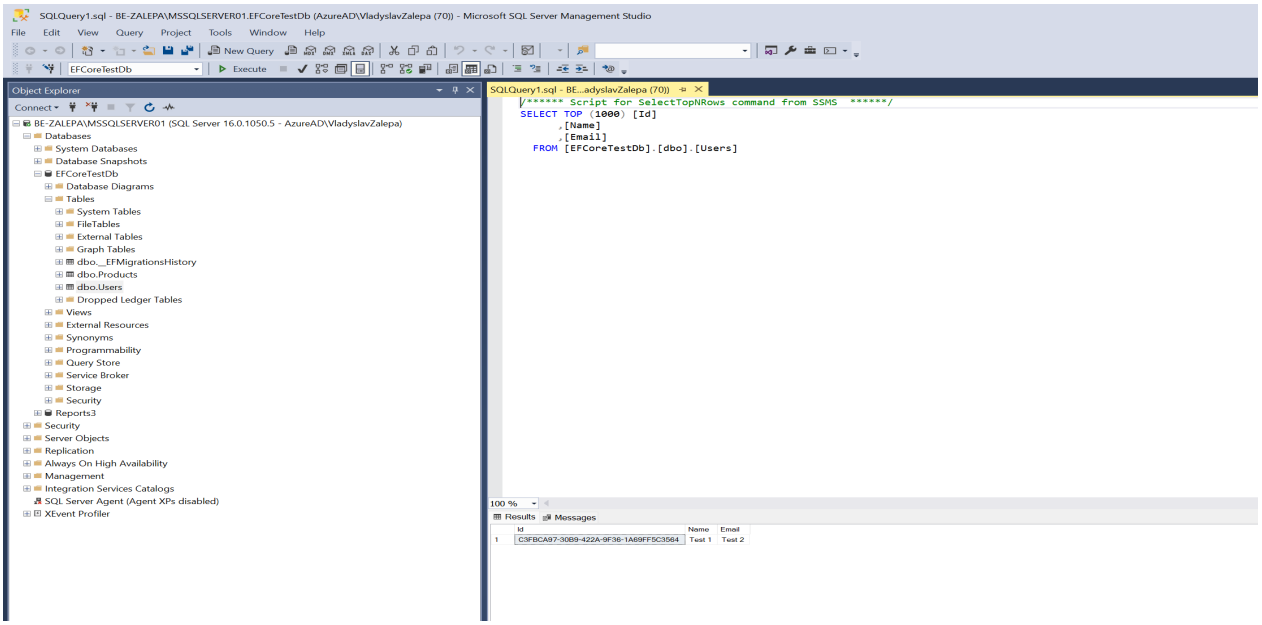


Рис. 2.11. Головне меню SQL Server Management Studio

Git - це розподілена система керування версіями, яка використовується для відстеження змін у програмному коді та спільної роботи розробників. Вона дозволяє ефективно керувати різними версіями проекту та координувати зусилля команди розробників [18].

Git дозволяє зберігати повну історію змін у проекті та відстежувати, як код змінюється з часом. Кожна зміна, відома як "коміт", містить інформацію про внесені зміни, автора та дату зміни. Це дозволяє розробникам легко переходити між різними версіями коду та відновлювати попередні стани проекту [19].

Одним з основних переваг Git є можливість розгалуження та злиття гілок. Розгалуження дозволяє створювати окремі гілки, де можна працювати над окремими функціями або виправленнями без впливу на основну гілку розробки. Після завершення роботи над функціоналом гілки можна злити з основною гілкою, об'єднавши зміни [20]. Схема роботи з Git називається Gitflow (рис. 2.12).

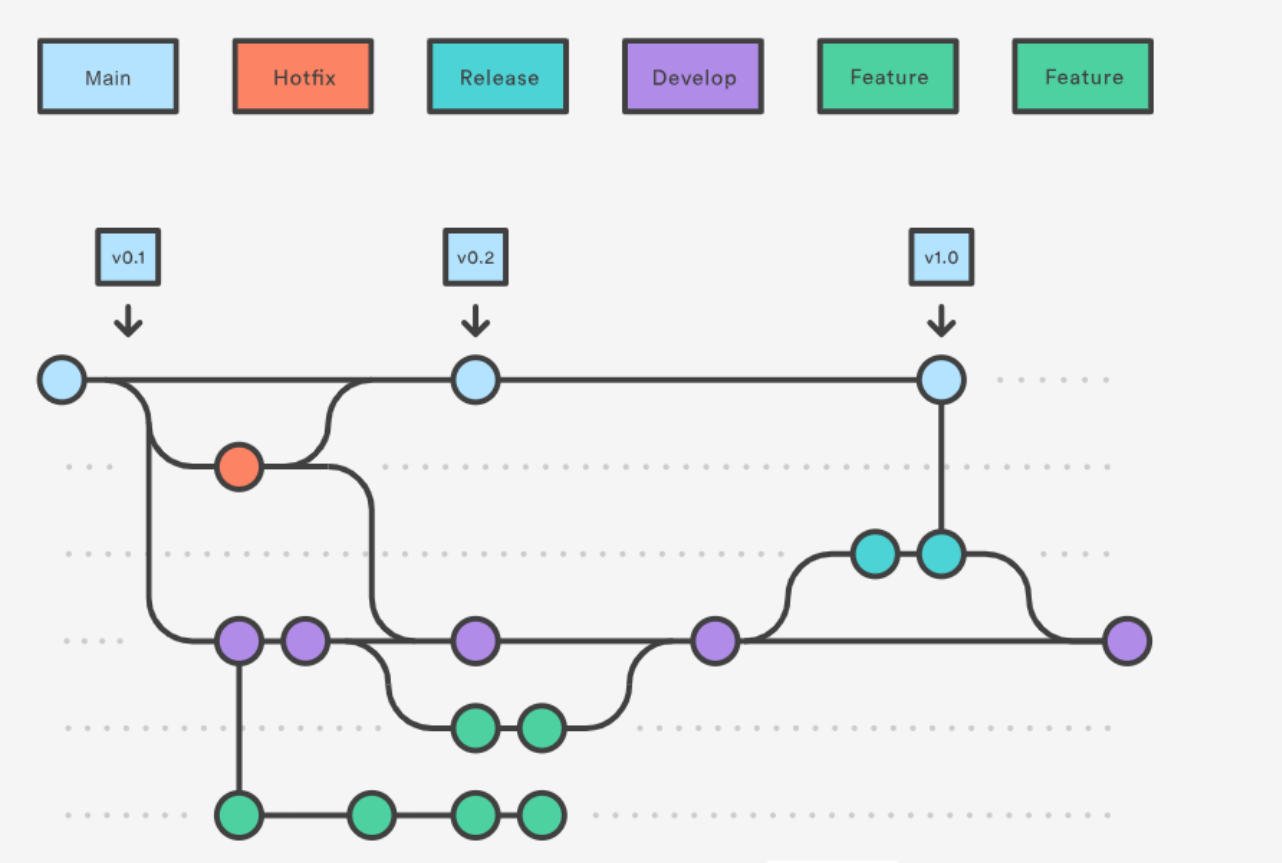


Рис. 2.12. Схема роботи Gitflow

2.7.3. Виклик та завантаження програми

Розроблений додаток завантажується після запуску файлу Firma.exe. Так як додаток розроблений з використанням .NET Standard, то він гарантовано буде встановлюватися та запускатися на всіх системах, що працюють на операційній системі Windows. Для взаємодії з відкритими API буде достатньо використовувати такі браузери, як Google Chrome, Firefox, Brave.

Програма гарантовано підтримується усіма версіями операційної системи з Windows 8 версії та новіші.

База даних буде автоматично налаштована на системі де буде встановлений додаток та налаштує SQL Service системи для роботи з даним додатком.

2.7.4. Опис інтерфейсу користувача

Після запуску додатку користувач попадає на головне меню (рис. 2.13)

Рис. 2.13. Головне меню додатку

Перед початком роботи потрібно приєднати базу даних. У відкритому вікні натиснути на кнопку  і вибрати потрібну базу даних. Після цього потрібно натиснути ✓ (рис. 2.14).

Рис. 2.14. Форма з’єднання за базою даних

Головне вікно програми має 5 варіантів меню.

Варіант «Постачальники». Після обрання цього варіанту відкриється вкладка, що відображає відомості про постачальників продуктів, та відомості про договори (рис. 2.15).

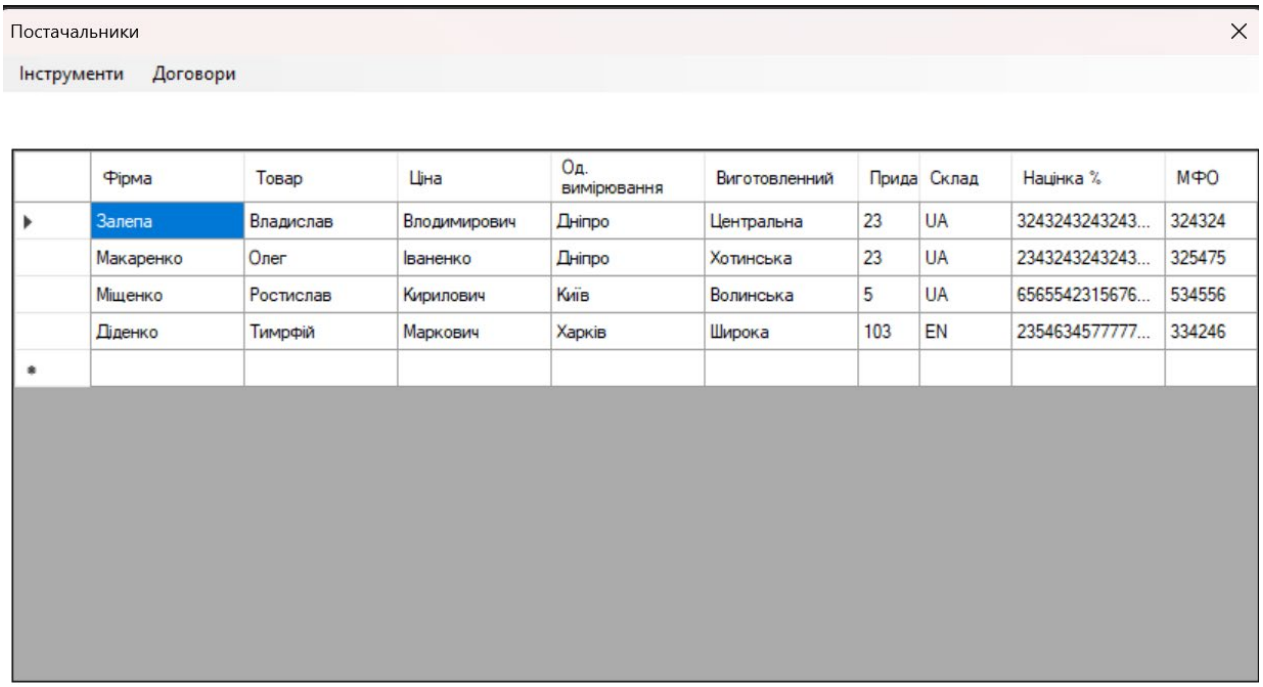


Рис. 2.15. Форма постачальників

При виборі пункту меню «Інструменти» відкриється вкладка з двома варіантами додати та видалити. При виборі першого варіанту відкриється вікно додавання постачальника (рис. 2.16).

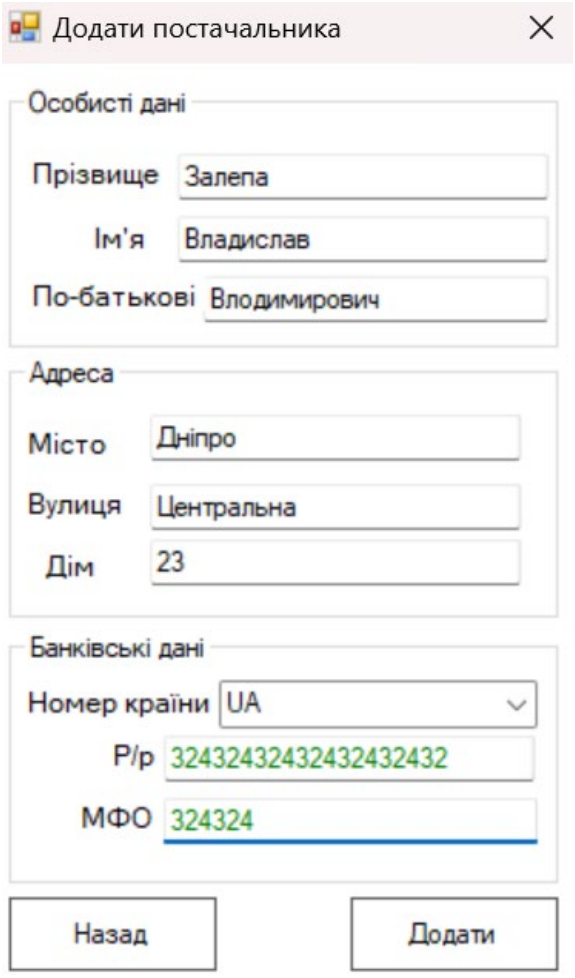


Рис. 2.16. Форма створення постачальників

При натисканні кнопки «Додати» спрацює перевірка на:

– чи довжина Р/с не менше 20 символі;
– чи довжина МФО не менше 6 символів;
– чи відсутні пробіли;
– чи правильно введені дані.

При виборі поля в таблиці і другого пункту меню виведеться повідомлення. Якщо користувач натисне «Yes», то поле буде видалене (рис. 2.17).

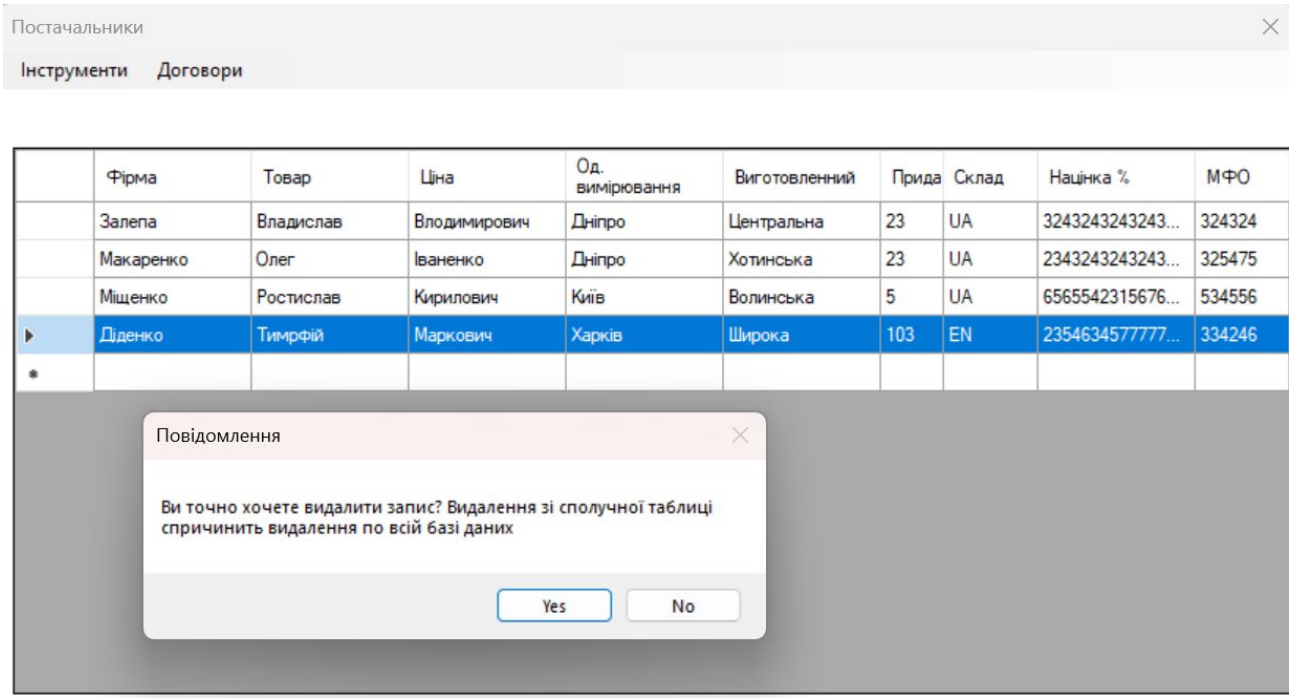


Рис. 2.17. Форма видалення постачальника

При виборі пункту «Договори» відкриється вкладка, що буде містити данні про договори термін дії, яких не закінчився (рис. 2.18).

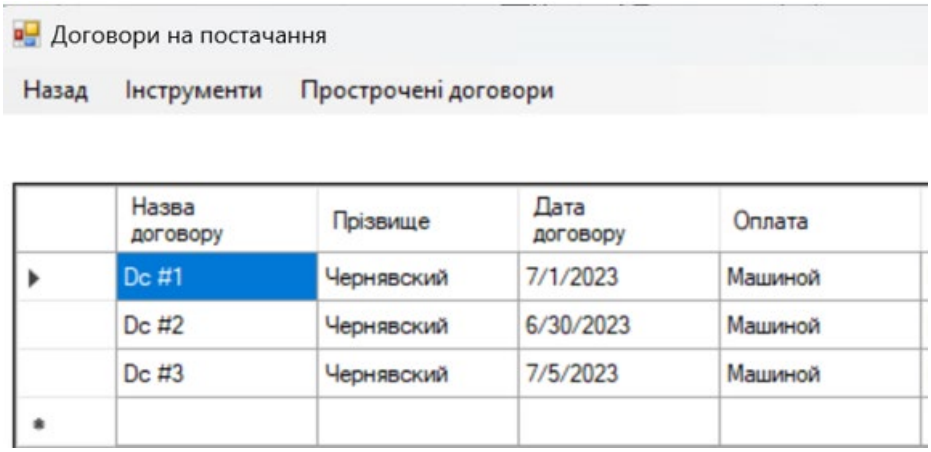


Рис. 2.18. Форма договорів на постачання

При виборі пункту «Інструменти» відкриється вкладка з двома варіантами. При виборі першого варіанту відкриється вікно додавання договору (рис. 2.19).

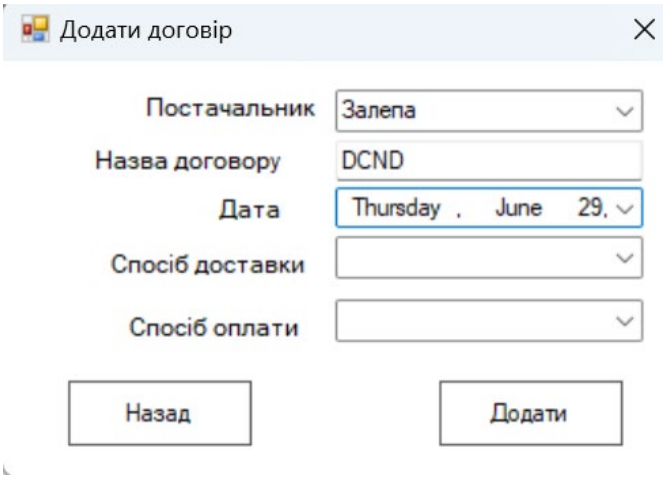


Рис. 2.19. Форма додавання договору на постачання

При натисканні кнопки «Добавить» спрацює перевірка на:

– чи дата договору не менше і не дорівнює поточній даті;
– чи всі у всі поля внесли данні;
– чи відсутні пробіли;
– чи правильно введені дані.

При виборі пункту меню «Неактуальні договори» відкриється вікно, що відображає договори термін дії яких менше і не дорівнюють поточній даті (рис. 2. 20).

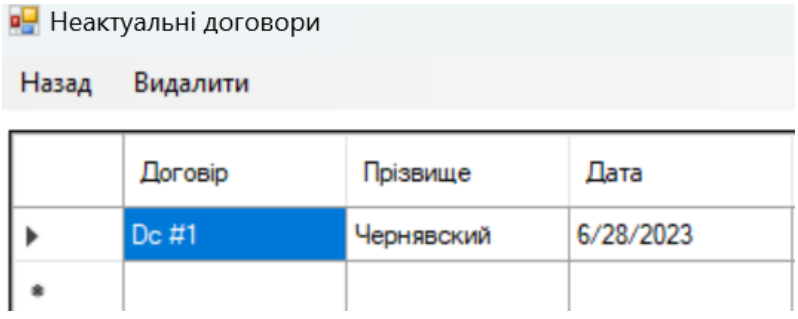
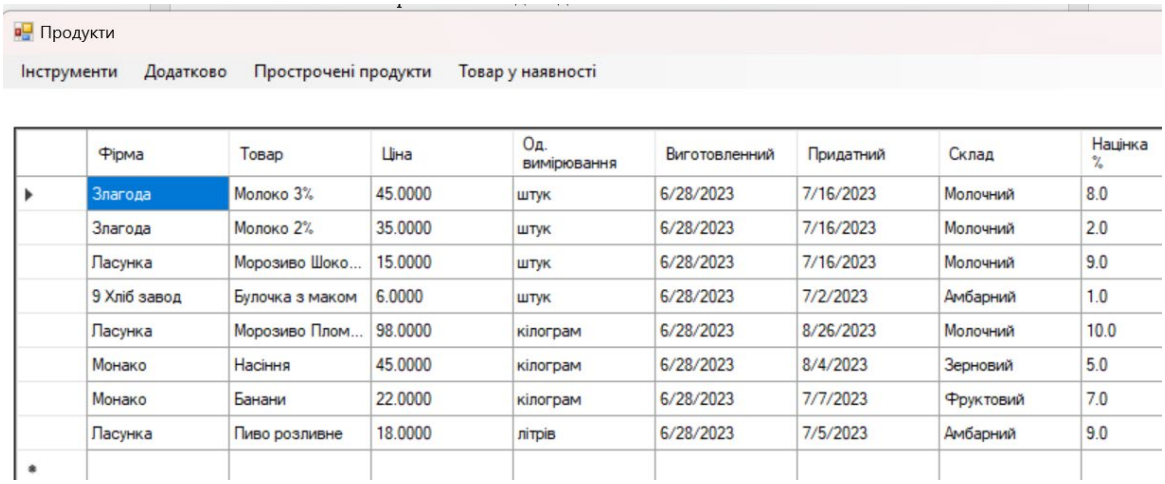


Рис. 2.20. Форма неактуальних договорів

Неактуальний договір можна видалити натиснувши «Видалити»

При виборі «Товар» відкриється вікно, що містить данні про товар термін придатності якого більше поточної дати (рис. 2.21).



	Фірма	Товар	Ціна	Од. вимірювання	Виготовленний	Придатний	Склад	Націнка %
▶	Злагода	Молоко 3%	45.0000	штук	6/28/2023	7/16/2023	Молочний	8.0
	Злагода	Молоко 2%	35.0000	штук	6/28/2023	7/16/2023	Молочний	2.0
	Ласунка	Морозиво Шоко...	15.0000	штук	6/28/2023	7/16/2023	Молочний	9.0
	9 Хліб завод	Булочка з маком	6.0000	штук	6/28/2023	7/2/2023	Амбарний	1.0
	Ласунка	Морозиво Плом...	98.0000	кілограм	6/28/2023	8/26/2023	Молочний	10.0
	Монако	Насіння	45.0000	кілограм	6/28/2023	8/4/2023	Зерновий	5.0
	Монако	Банани	22.0000	кілограм	6/28/2023	7/7/2023	Фруктовий	7.0
	Ласунка	Пиво розливне	18.0000	літрів	6/28/2023	7/5/2023	Амбарний	9.0
*								

Рис. 2.21. Форма товарів

При виборі пункту інструменти відкриється підменю з двома варіантами додати та видалити При виборі першого пункту відкриється вікно додавання продукту (рис. 2.22).

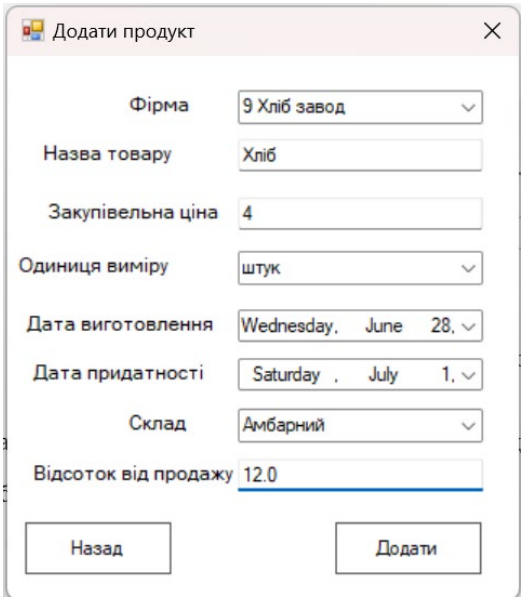


Рис. 2.22. Форма додавання товару

Після натискання кнопки «Додати» спрацює перевірка на:

– чи немає товару з внесеною назвою в конкретній фірмі;

– чи всі у всі поля внесли данні;

– чи відсутні пробіли;

– чи правильно введені дані;

– чи дата створення не менше поточної дати;

– чи дата створення не більше дати придатності;

– чи дата природності не менше поточної дати і не менше дати створення продукту.

При виборі пункту меню «Додатково» відкривається підменю з варіантами Склад, Одниця вимірювання, Виробник при виборі пунктів меню відкриються вікна, що будуть відображати інформацію про «Склад», «Одиниця вимірювання», «Виробник» з можливістю додавати та видаляти інформацію (рис. 2.23).

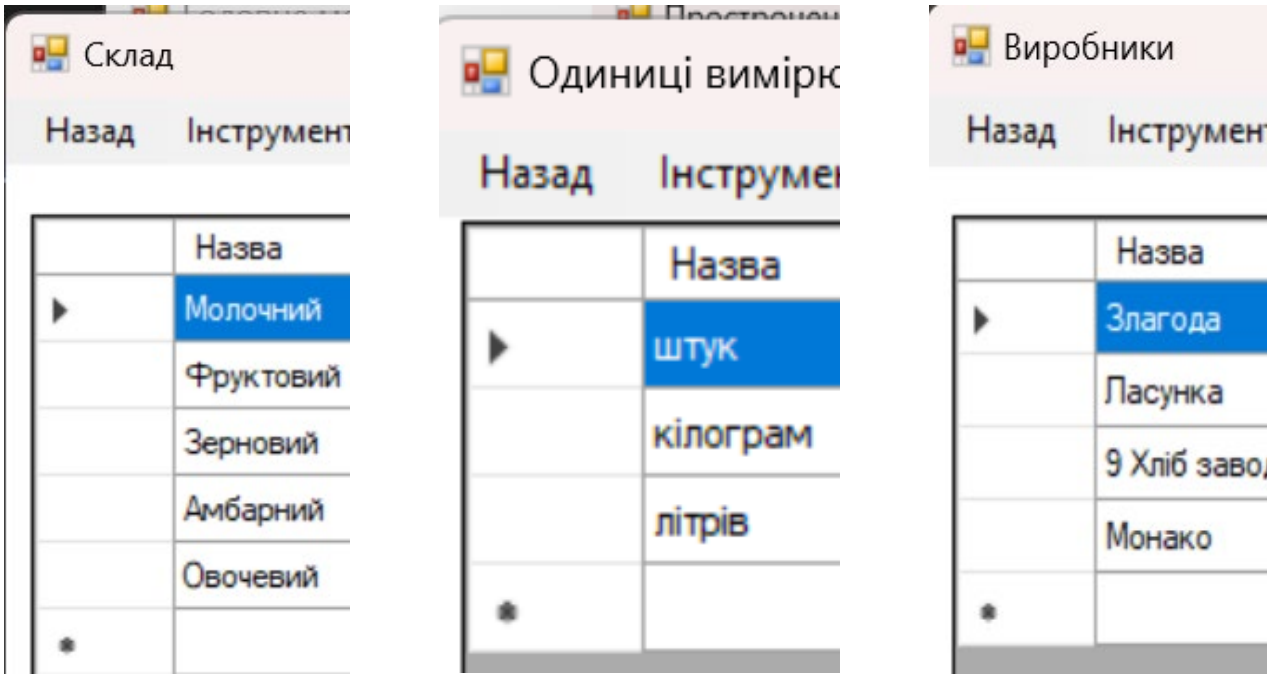


Рис. 2.23. Форми додавання складів, одиниць вимірювання та виробників

При виборі пункту меню «Прострочені продукти» відкривається вікно, що відображає продукти термін придатності яких менше поточної дати (рис. 2.24).

Рис. 2.24. Форми додавання складів, одиниць вимірювання та виробників

При виборі пункту меню «Товар у наявності» відкриється вікно, що буде відображати інформацію про наявність товару. Також можна в випадаючому списку обрати: «Товар який треба поповнювати», «Товар, який не закуповується», «Кількість прострочених продуктів на складі», таким чином працюють звіти в додатку (рис. 2.25).

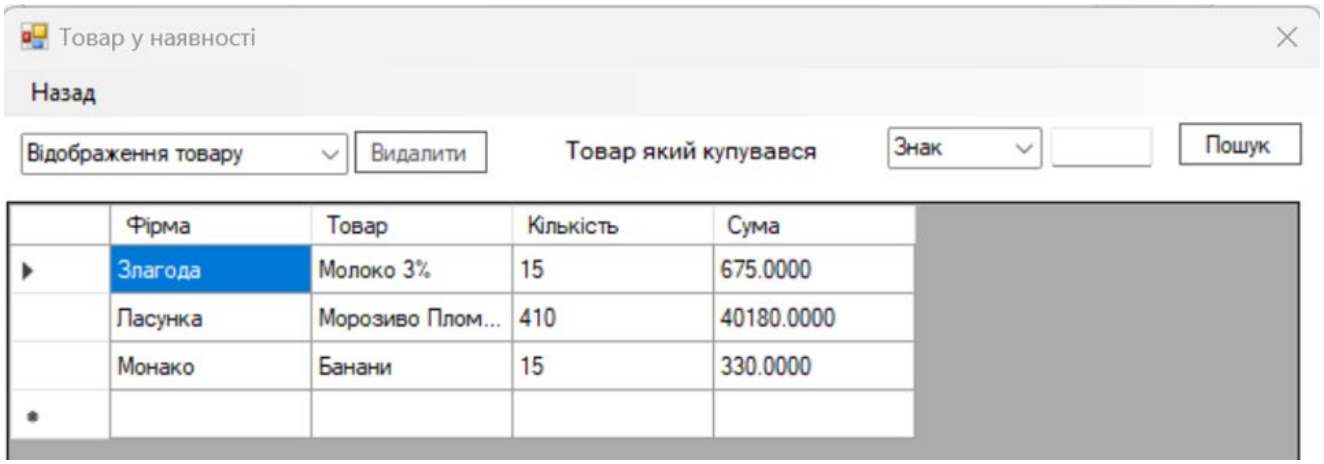


Рис. 2.25. Форма товару у наявності та інша фільтрація

При виборі пункту «Покупці» відкриється вікно, що буде містити відомості про покупців (рис. 2.26).

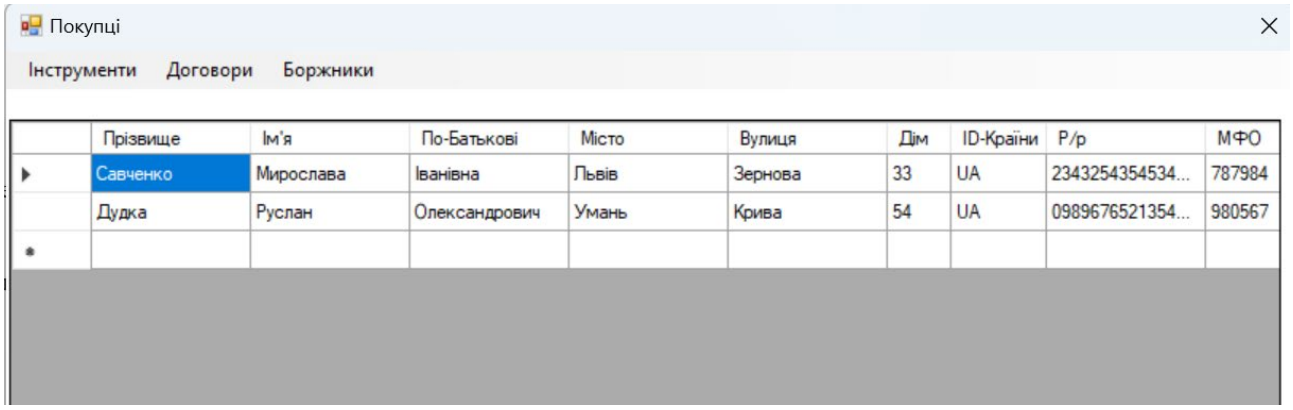


Рис. 2.26. Форма покупців

При виборі пункту «Інструменти» відкриється меню. додати та видалити При виборі першого пункту відкриється вікно додавання нового покупця (рис. 2.27).

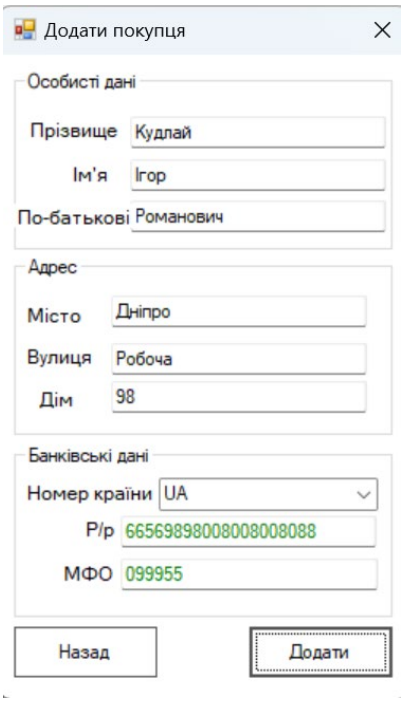


Рис. 2.27. Форма додавання покупців

При натисканні кнопки «Додати» спрацює перевірка на:

– чи довжина Р/с не менше 20 символі;

– чи довжина МФО не менше 6 символів;

– чи відсутні пробіли;

– чи правильно введені дані.

При виборі пункту «Договори» відкриється вікно, що буде відображати інформацію про договори покупців (рис. 2.28).

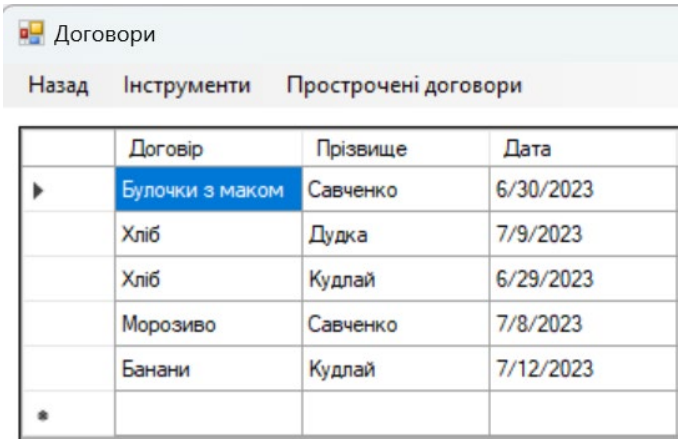


Рис. 2.28. Форма договорів покупців

При виборі пункту «Інструменти» відкриється вкладка з двома варіантами. При виборі першого варіанту відкриється вікно додавання договору (рис. 2.29).

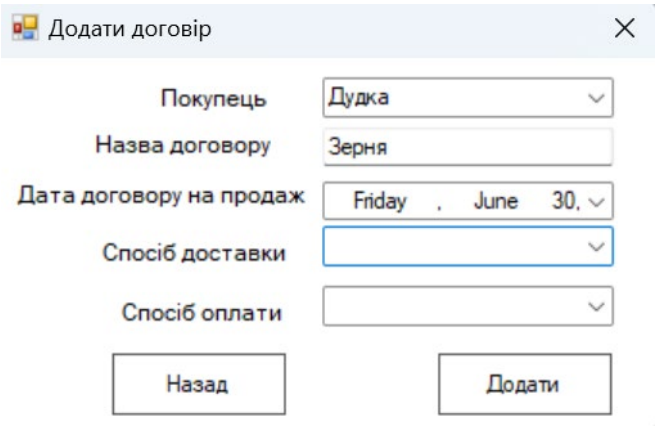


Рис. 2.29. Форма додавання договорів покупців

При натисканні кнопки «Добавить» спрацює перевірка на:

– чи дата договору не менше і не дорівнює поточній даті;
– чи всі у всі поля внесли данні;
– чи відсутні пробіли;
– чи правильно введені дані.

При виборі пункту меню «Прострочені договори» відкриється вікно, що відображає договори термін дії яких менше і не дорівнюють поточній даті.

При виборі «Боржники» відкриється вікно, що містить данні про заборгованості покупці (рис. 2.30).

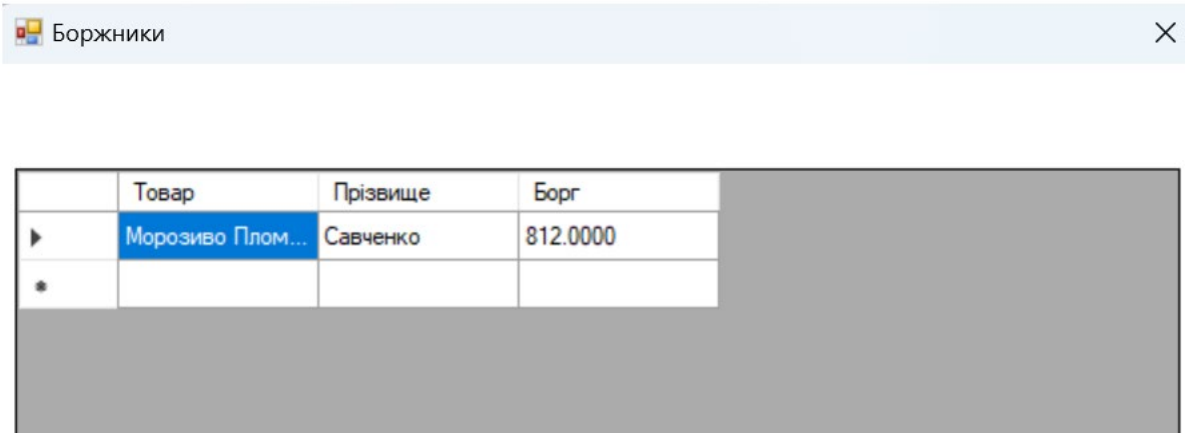


Рис. 2.30. Форма додавання договорів покупців

При виборі «Замовлення на поставку» відкриється форма вибору (рис. 2.31).

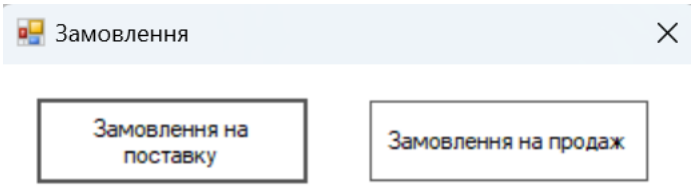


Рис. 2.31. Форма вибору замовлень

При виборі «Замовлення на поставку» відкриється форма, що зберігає відомості про замовлення на постачання термін, яких менше поточної дати (рис. 2.32).

Рис. 2.32. Форма замовлень на поставку

При виборі додати договір відкриється форма додавання замовлення (рис. 2.33).

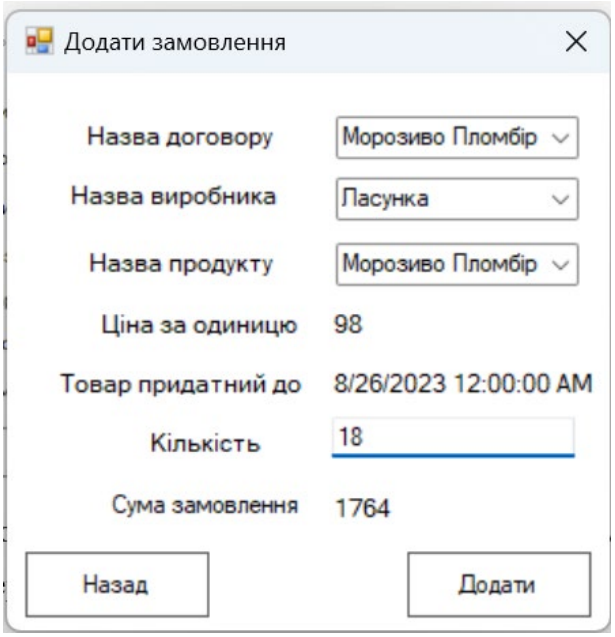


Рис. 2.33. Форма додавання замовлення на поставку

При виборі «Неактуальні замовлення» відкриється форма, що відображає замовлення термін договору яких менше поточної дати.

При виборі «Замовлення на продаж» відкриється форма, що зберігає відомості про закази на продажу термін, яких менше поточної дати (рис. 2.34).

Рис. 2.34. Форма замовлень на продаж

При виборі додати договір відкриється форма додавання замовлення (рис. 2.35).

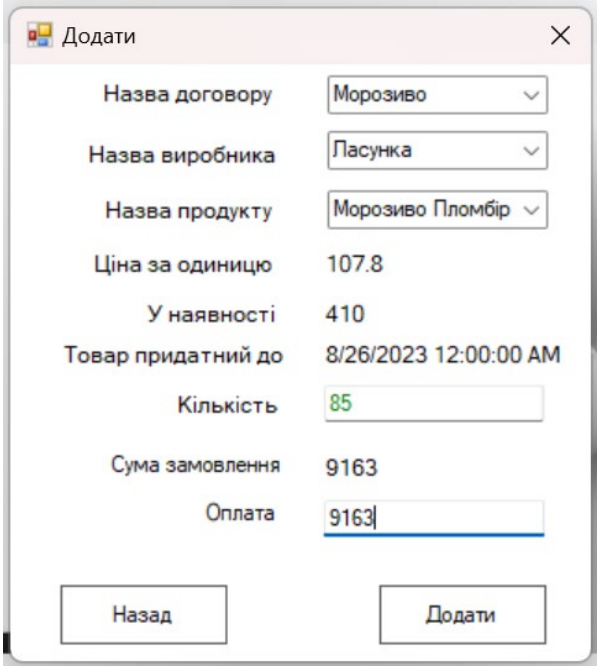


Рис. 2.35. Форма додавання замовлень на продаж

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми – 1200.
2. Коефіцієнт складності програми – 1,3.
3. Коефіцієнт корекції програми в ході її розробки – 0,08.
4. Годинна заробітна плата програміста – 65 грн/год.
5. Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3.
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,3.
7. Вартість машино-години ЕОМ – 20 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p) Q = q \cdot C \cdot (1 + p), \quad (3.2)$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1200 \cdot 1{,}3 \cdot (1 + 0{,}08) = 1685$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K} t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_u = \frac{1685 \cdot 1{,}3}{85 \cdot 1{,}3} = 19{,}82 t_u = \frac{1685 \cdot 1{,}3}{85 \cdot 1{,}3} = 19{,}82$$ людино-годин.

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot K} t_a = \frac{Q}{(20 \ldots 25) \cdot K}, \quad (3.4)$$

$$t_a = \frac{1685}{23 \cdot 1{,}3} = 56,35 t_a = \frac{1685}{23 \cdot 1{,}3} = 56{,}35 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot K} t_n = \frac{Q}{(20...25) \cdot K}, \quad (3.5)$$

$$t_a = \frac{1685}{22 \cdot 1{,}3} = 58,6 t_n = \frac{1685}{22 \cdot 1{,}3} = 58{,}6 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5) \cdot K} t_{отл} = \frac{Q}{(4...5) \cdot K}, \quad (3.6)$$

$$t_a = \frac{1685}{4 \cdot 1{,}3} = 324 t_n = \frac{1685}{4 \cdot 1{,}3} = 324 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл} t^{к}_{отл} = 1{,}2 \cdot t_{отл}, \quad (3.7)$$

$$t^{к}_{отл} = 1{,}2 \cdot 324 = 388.8 t^{к}_{отл} = 1{,}2 \cdot 324 = 388.8 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{дp} + t_{до} t_д = t_{дp} + t_{до}, \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_д = \frac{Q}{(15...20)\cdot K} t_д = \frac{Q}{(15...20)\cdot K} \quad , \tag{3.9}$$

$$t_a = \frac{1685}{18 \cdot 1{,}3} = 72 t_д = \frac{1685}{18 \cdot 1{,}3} = 72 \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др} \quad , \tag{3.10}$$

$$t_{до} = \ 0{,}75 \ \cdot 56{,}35 = 42{,}26 t_{до} = \ 0{,}75 \ \cdot 56{,}35 = 42{,}26 \text{ людино-годин.}$$

$$t_д = \ 56{,}35 + 42{,}26 \ = 98{,}61 t_д = \ 56{,}35 + 42{,}26 \ = 98{,}61 \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + \ 19{,}82 + \ 56{,}35 + \ 58{,}6 + 388.8 + \ 98{,}61 = 672{,}18 \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно близько 672,18 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ включають витрати на заробітну плату виконавця програми З/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ} \quad \text{грн.,} \tag{3.11}$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР} \quad \text{грн.,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 672{,}18 \cdot 65 = 43691{,}7 З_{ЗП} = 672{,}18 \cdot 65 = 43691{,}7 \text{ грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М} \quad \text{грн.,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

СМЧ – вартість машино-години ЕОМ, грн/год.

$$З_{ЗМ} = 388.8 \cdot 20 = 7776 З_{ЗМ} = 388.8 \cdot 20 = 7776 \text{ грн.}$$

$$К_{ПО} = 43691{,}7 + 7776 = 51467{,}7 К_{ПО} = 43691{,}7 + 7776 = 51467{,}7 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \quad \text{міс.,} \qquad (3.14)$$

де $В_k$- число виконавців;

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{672{,}18}{1 \cdot 176} = 3{,}82 T = \frac{672{,}18}{1 \cdot 176} = 3{,}82 \text{ міс.}$$

Висновки. На розробку даного програмного забезпечення піде близько 672,18 людино-години. Тобто, ймовірна очікувана тривалість розробки складатиме 3,82 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 51467,7 грн.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи був розроблений додаток для автоматизації роботи посередницьких фірм зі складами, були використані сучасні підходи роботи з фреймворком ASP NET Core на мові програмування C# з ORM ADO NET та інструменти Visual Studio та MSSQL Server.

Додаток буде легко розширяти та покращувати в майбутньому, так як сучасні підходи в проектуванні архітектури систем дозволяють контролювати версійність програм та легко додавати новий функціонал, за рахунок використовування абстракцій між модулями системи.

Система дозволить автоматизувати ручні процеси, що дозволить всім сторонам заощадити значні кошти, які втрачаються за рахунок людського фактору.

Програма допоможе ефективніше контролювати договори, які укладаються на постачання і на покупку товарів, що допоможе реагувати на кожний із них і мінімізувати ризик ігнорування якогось договору.

В результаті був розроблений додаток, який має практичну цінність, тому, що допомагає бізнесу вирішити проблеми такі як заощадження часу та коштів.

Під час виконання кваліфікаційної роботи були виконані пройдені наступні етапи створення програмного продукту:

− аналіз предметної області задачі, що розв’язується;

− обрання архітектури та технології створення додатку;

− написання програмного коду веб-додатку;

− розробка рекомендацій щодо використання застосунку.

Під час виконання кваліфікаційної роботи також було визначено трудомісткість розробленого програмного продукту (672,18 людино-годин), проведений підрахунок вартості роботи по створенню програми (51467,7 грн.) та розраховано час на його створення (3,82 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тришарова архітектура [Електронний ресурс] URL: https://medium.com/@deanrubin/the-three-layered-architecture-fe30cb0e4a6 (дата звернення: 25.06.2023).
2. Що таке трирівнева архітектура? [Електронний ресурс] URL: https://www.ibm.com/topics/three-tier-architecture (дата звернення: 25.06.2023).
3. Загальні відомості про ASP.NET Core [Електронний ресурс] URL: https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0 (дата звернення: 26.06.2023).
4. ASP.NET Core Web API Найкращі практики [Електронний ресурс] URL: https://code-maze.com/aspnetcore-webapi-best-practices/ (дата звернення: 26.06.2023).
5. C# і Windows Forms [Електронний ресурс] URL: https://metanit.com/sharp/windowsforms/1.1.php (дата звернення: 27.06.2023).
6. Переваги та недоліки Windows Forms [Електронний ресурс] URL: https://www.bytehide.com/blog/wpf-vs-winforms (дата звернення: 27.06.2023).
7. ADO NET від Microsoft [Електронний ресурс] URL: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ (дата звернення: 28.06.2023).
8. Інформація про ADO NET [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/ADO.NET (дата звернення: 28.06.2023).
9. Адам Фріман. Entity Framework Core 2 for ASP.NET. Apress, 2018. – 178с.
10. Entity Framework Core Tutorials [Електронний ресурс] URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx (дата звернення: 29.06.2023).
11. Загальні відомості про Microsoft SQL Server [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення: 29.06.2023).
12. Microsoft SQL Server [Електронний ресурс] URL: https://habr.com/ru/hub/mssql/ (дата звернення: 29.06.2023).

13. ER-діаграми [Електронний ресурс] URL: https://lucidchart.zendesk.com/hc/ru/articles/207299756-ER-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B (дата звернення: 30.06.2023).

14. Джеффрі Ріхтер. CLR via C#. Microsoft Press, 2012. – 644с.

15. Visual Studio Roadmapr [Електронний ресурс] URL: https://learn.microsoft.com/en-us/visualstudio/productinfo/vs-roadmap (дата звернення: 30.06.2023).

16. SQL Server Tutorial [Електронний ресурс] URL: https://www.sqlservertutorial.net/ (дата звернення: 30.06.2023).

17. SQL Server Management Studio [Електронний ресурс] URL: https://ru.wikipedia.org/wiki/SQL_Server_Management_Studio (дата звернення: 30.06.2023).

18. Робочий процес Gitflow Workflow [Електронний ресурс] URL: https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow (дата звернення: 30.06.2023).

19. Git на практиці [Електронний ресурс] URL: https://habr.com/ru/articles/342116/ (дата звернення: 30.06.2023).

20. Найкращі практики Git [Електронний ресурс] URL: https://www.freecodecamp.org/news/how-to-use-git-best-practices-for-beginners/ (дата звернення: 30.06.2023).

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

Main.cs

```
public partial class view_product : Form
  {
    public view_product()
    {
      InitializeComponent();
    }
    void ClearDataGridView()
    {
      while (dataGridView1.Rows.Count > 1)
        for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
          dataGridView1.Rows.Remove(dataGridView1.Rows[i]);
    }
    string ConnectionString;
    private void view_product_Load(object sender, EventArgs e)
    {
      ConnectionString = Properties.Settings.Default.FirmConnectionString;
      ClearDataGridView();
      int x = 0;
      using (var connection = new SqlConnection(ConnectionString))
      {
        using (var command = new SqlCommand("Select_Products", connection))
        {
          DataSet ds = new DataSet();
          BindingSource source = new BindingSource();
          connection.Open();
          SqlDataReader reader = command.ExecuteReader();
          while (reader.Read()) x++; reader.Close();
          SqlDataAdapter house = new SqlDataAdapter(command);
          SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
          house.Fill(ds, "Products");
          source.DataSource = ds;
          source.DataMember = "Products";
          dataGridView1.DataSource = source;
          dataGridView1.Columns[0].Visible = false;
          dataGridView1.Columns[8].Width = 60;

          connection.Close();
        }
      }
    }

    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
      this.Close();
      add.add_product product = new add.add_product();
      product.Show();
    }

    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
      this.Close();
      view_warehouse warehouse = new view_warehouse();
```

```
        warehouse.Show();
    }

    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        DialogResult result;
        MessageBoxButtons but = MessageBoxButtons.YesNo;
        result = MessageBox.Show("Ви точно хочете видалити запис? Видалення зі сполучної таблиці спричинить видалення по всій базі даних  ", "Повідомлення", but);
        if (result == DialogResult.Yes)
        {

            try
            {
                SqlConnection conn = new SqlConnection(ConnectionString);
                conn.Open();
                int id = Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value);
                string sql = string.Format("DELETE FROM Products WHERE IDProduct='{0}'", id);
                using (SqlCommand cnd = new SqlCommand(sql, conn))
                {
                    try
                    {
                        cnd.ExecuteNonQuery();
                        MessageBox.Show("Видалено!");
                    }
                    catch (SqlException ex)
                    {
                        Exception error = new Exception("Помилка видалення", ex);
                        throw error;
                    }

                }
                ClearDataGridView();
                view_product_Load(sender, e);
            }
            catch { MessageBox.Show("Помилка видалення", "Повідомлення"); }
        }
    }

    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.Close();
        view_unit unit = new view_unit();
        unit.Show();
    }

    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.Close();
        view_firm firm = new view_firm();
        firm.Show();
    }

    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.Close();
        view_product_date_0 view_Product_Date_0 = new view_product_date_0();
        view_Product_Date_0.Show();
    }

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

```
            this.Close();
            view_sklad sklad = new view_sklad();
            sklad.Show();
        }
    }


public partial class add_customer : Form
    {
        public add_customer()
        {
            InitializeComponent();
        }
        bool Check_Null(string s1, string s2, string s3, string s4, string s5, string s6, string s7, string s8, string s9)
        {
            if (s1 == null || s1.Length == 0) return true;
            if (s2 == null || s2.Length == 0) return true;
            if (s3 == null || s3.Length == 0) return true;
            if (s4 == null || s4.Length == 0) return true;
            if (s5 == null || s5.Length == 0) return true;
            if (s6 == null || s6.Length == 0) return true;
            if (s7 == null || s6.Length == 0) return true;
            if (s8 == null || s6.Length == 0) return true;
            if (s9 == null || s6.Length == 0) return true;
            return false;
        }
        bool Check_Space(string s1, string s2, string s3, string s4, string s5, string s6, string s7)
        {

            for (int i = 0; i < s1.Length; i++) if (char.IsWhiteSpace(s1[i])) return true;
            for (int i = 0; i < s2.Length; i++) if (char.IsWhiteSpace(s2[i])) return true;
            for (int i = 0; i < s3.Length; i++) if (char.IsWhiteSpace(s3[i])) return true;
            for (int i = 0; i < s4.Length; i++) if (char.IsWhiteSpace(s4[i])) return true;
            for (int i = 0; i < s6.Length; i++) if (char.IsWhiteSpace(s6[i])) return true;
            for (int i = 0; i < s7.Length; i++) if (char.IsWhiteSpace(s6[i])) return true;
            return false;
        }
        string ConnectionString;
        private void button1_Click(object sender, EventArgs e)
        {
            bool chek_null = Check_Null(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text,
textBox6.Text, comboBox1.Text, textBox8.Text, textBox9.Text);
            bool chek_space = Check_Space(textBox1.Text, textBox2.Text, textBox3.Text, textBox6.Text, comboBox1.Text,
textBox8.Text, textBox9.Text);
            if (chek_null == false)
            {

                if (chek_space == false)
                {
                    if (comboBox1.Text != "оберіть")
                    {
                        if (textBox8.Text.Length == 20 && textBox9.Text.Length == 6)
                        {
                            using (var connection = new SqlConnection(ConnectionString))
                            {
                                using (var command = new SqlCommand("Inset_Customer", connection))
                                {
                                    try
                                    {
                                        connection.Open();
                                        command.CommandType = CommandType.StoredProcedure;
                                        command.Parameters.AddWithValue("@Surname", textBox1.Text);
```

```
                    command.Parameters.AddWithValue("@Name", textBox2.Text);
                    command.Parameters.AddWithValue("@Patronymic", textBox3.Text);
                    command.Parameters.AddWithValue("@City", textBox4.Text);
                    command.Parameters.AddWithValue("@Street", textBox5.Text);
                    command.Parameters.AddWithValue("@House", textBox6.Text);
                    command.Parameters.AddWithValue("@CountryNumber", comboBox1.Text);
                    command.Parameters.AddWithValue("@Score", textBox8.Text);
                    command.Parameters.AddWithValue("@MFO", textBox9.Text);
                    int result = command.ExecuteNonQuery();
                    if (result != 0)
                        MessageBox.Show("Покупця додано");
                    else MessageBox.Show("Виникла помилка!");
                    connection.Close();
                    this.Close();
                    view_forms.view_customer customer = new view_forms.view_customer();
                    customer.Show();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Неправильне введення даних!");
                }
                finally
                {
                    connection.Close();
                }
            }
        }
        else MessageBox.Show("Довжина розрахункового рахунку має бути 20 символів, а MFO 6");
    }

    else MessageBox.Show("Виберіть код країни");
  }
  else MessageBox.Show("Пробіли допустимі тільки в назві міста та вулиці");
 }
 else MessageBox.Show("Усі поля мають бути заповнені");
}

private void add_customer_Load(object sender, EventArgs e)
{
 ConnectionString = Properties.Settings.Default.FirmConnectionString;
}

private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
 if (Char.IsDigit(e.KeyChar) != true) return;
 e.Handled = true;
 return;
}

private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
{
 if (Char.IsDigit(e.KeyChar) != true) return;
 e.Handled = true;
 return;
}

private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
{
 if (Char.IsDigit(e.KeyChar) != true) return;
 e.Handled = true;
 return;
```

```
    }

    private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
    {
      if (Char.IsDigit(e.KeyChar) != true) return;
      e.Handled = true;
      return;
    }

    private void textBox5_KeyPress(object sender, KeyPressEventArgs e)
    {
      if (Char.IsDigit(e.KeyChar) != true) return;
      e.Handled = true;
      return;
    }

    private void textBox6_KeyPress(object sender, KeyPressEventArgs e)
    {
      if (!Char.IsDigit(e.KeyChar) != true) return;
      e.Handled = true;
      return;
    }

    private void textBox8_KeyPress(object sender, KeyPressEventArgs e)
    {
      if (!Char.IsDigit(e.KeyChar) != true) return;
      e.Handled = true;
      return;
    }

    private void textBox9_KeyPress(object sender, KeyPressEventArgs e)
    {
      if (!Char.IsDigit(e.KeyChar) != true) return;
      e.Handled = true;
      return;
    }

    private void textBox8_KeyUp(object sender, KeyEventArgs e)
    {
      if (textBox8.Text.Length != 20) textBox8.ForeColor = Color.Red;
      else textBox8.ForeColor = Color.Green;
    }

    private void textBox9_KeyUp(object sender, KeyEventArgs e)
    {
      if (textBox9.Text.Length != 6) textBox9.ForeColor = Color.Red; else textBox9.ForeColor = Color.Green;
    }

    private void button2_Click(object sender, EventArgs e)
    {
      this.Close();
      view_forms.view_customer customer = new view_forms.view_customer();
      customer.Show();
    }
  }


public partial class Connect_datebase : Form
  {
    public Connect_datebase()
    {
      InitializeComponent();
```

```
    }
    string ConnectionString;
    public string filename;
    private void button1_Click(object sender, EventArgs e)
    {

      openFileDialog1.Filter = "mdf files (*.mdf)|*.mdf|All files (*.*)|*.*"; ;
      if (openFileDialog1.ShowDialog() == DialogResult.OK)
      {
        filename = openFileDialog1.FileName;
        textBox1.Text = filename;
      }
    }

    private void button3_Click(object sender, EventArgs e)
    {
      if (textBox1.Text != "")
      {
        Form1 main_Form = new Form1();
        ConnectionString = @"Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename =""" +
        filename + @""";Integrated Security = True; Connect Timeout = 30";
        Properties.Settings.Default["FirmConnectionString"] = ConnectionString;
        MessageBox.Show("Підключення пройшло успішно!");
        this.Close();
        main_Form.Indicator = true;
        main_Form.Show();

      }
      else MessageBox.Show("Не знайдено шлях до бази!");
    }

    private void button2_Click(object sender, EventArgs e)
    {
      this.Close();
      Form1 main_Form = new Form1();
      main_Form.Show();
    }
  }

public partial class view_customer : Form
  {
    public view_customer()
    {
      InitializeComponent();
    }
    void ClearDataGridView()
    {
      while (dataGridView1.Rows.Count > 1)
        for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
          dataGridView1.Rows.Remove(dataGridView1.Rows[i]);
    }
    string ConnectionString;
    private void view_customer_Load(object sender, EventArgs e)
    {
      ConnectionString = Properties.Settings.Default.FirmConnectionString;
      ClearDataGridView();
      int x = 0;
      using (var connection = new SqlConnection(ConnectionString))
      {
        using (var command = new SqlCommand("Select_Customer", connection))
        {
          DataSet ds = new DataSet();
```

```
            BindingSource source = new BindingSource();
            connection.Open();
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read()) x++; reader.Close();
            SqlDataAdapter house = new SqlDataAdapter(command);
            SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
            house.Fill(ds, "Customer");
            source.DataSource = ds;
            source.DataMember = "Customer";
            dataGridView1.DataSource = source;
            dataGridView1.Columns[0].Visible = false;
            dataGridView1.Columns[6].Width = 40;
            dataGridView1.Columns[7].Width = 60;
            dataGridView1.Columns[9].Width = 60;
            connection.Close();
        }
    }
}

private void ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
    add.add_customer customer = new add.add_customer();
    customer.Show();
}

private void ToolStripMenuItem_Click(object sender, EventArgs e)
{
    DialogResult result;
    MessageBoxButtons but = MessageBoxButtons.YesNo;
    result = MessageBox.Show("Ви точно хочете видалити запис? Видалення зі сполучної таблиці спричинить
видалення по всій базі даних ", "Повідомлення", but);
    if (result == DialogResult.Yes)
    {

        try
        {
            SqlConnection conn = new SqlConnection(ConnectionString);
            conn.Open();
            int id = Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value);
            string sql = string.Format("DELETE FROM Customer WHERE IDCustomer='{0}'", id);
            using (SqlCommand cnd = new SqlCommand(sql, conn))
            {
                try
                {
                    cnd.ExecuteNonQuery();
                    MessageBox.Show("Видалено!");
                }
                catch (SqlException ex)
                {
                    Exception error = new Exception("Помилка видалення", ex);
                    throw error;
                }

            }
            ClearDataGridView();
            view_customer_Load(sender, e);
        }
        catch { MessageBox.Show("Помилка видалення", "Повідомлення"); }
    }
}
```

```
    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
      this.Close();
      view_salescontract salescontract = new view_salescontract();
      salescontract.Show();
    }

    private void ToolStripMenuItem_Click(object sender, EventArgs e)
    {
      this.Close();
      view_debtors debtors = new view_debtors();
      debtors.Show();
    }
  }

public partial class add_product : Form
  {
    public add_product()
    {
      InitializeComponent();
    }
    string ConnectionString;
    private void add_product_Load(object sender, EventArgs e)
    {
      ConnectionString = Properties.Settings.Default.FirmConnectionString;
      using (var connection = new SqlConnection(ConnectionString))
      {
        dateTimePicker2.Value = DateTime.Now.AddDays(1);
        connection.Open();
        using (var command = new SqlCommand("SELECT * FROM Firm", connection))
        {
          BindingSource bindingSource = new BindingSource();
          DataSet ds = new DataSet();
          SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
          dataAdapter.Fill(ds, "Firm");
          bindingSource.DataSource = ds;
          bindingSource.DataMember = "Firm";
          comboBox4.DataSource = bindingSource;
          comboBox4.ValueMember = "IDFirm";
          comboBox4.DisplayMember = "Name_Firm";
          comboBox4.SelectedIndex = -1;
          connection.Close();
        }
        using (var command = new SqlCommand("SELECT * FROM Unit", connection))
        {
          BindingSource bindingSource = new BindingSource();
          DataSet ds = new DataSet();
          SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
          dataAdapter.Fill(ds, "Unit");
          bindingSource.DataSource = ds;
          bindingSource.DataMember = "Unit";
          comboBox1.DataSource = bindingSource;
          comboBox1.ValueMember = "IDUnit";
          comboBox1.DisplayMember = "NameUnit";
          comboBox1.SelectedIndex = -1;
          connection.Close();
        }
        using (var command = new SqlCommand("SELECT * FROM Warehouse", connection))
        {
          BindingSource bindingSource = new BindingSource();
          DataSet ds = new DataSet();
          SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
```

```
            dataAdapter.Fill(ds, "Warehouse");
            bindingSource.DataSource = ds;
            bindingSource.DataMember = "Warehouse";
            comboBox3.DataSource = bindingSource;
            comboBox3.ValueMember = "IDWarehouse";
            comboBox3.DisplayMember = "NameWarehouse";
            comboBox3.SelectedIndex = -1;
            connection.Close();
        }
    }
}

private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
    if (dateTimePicker1.Value < DateTime.Today) { dateTimePicker1.Value = DateTime.Today; MessageBox.Show("Дата виготовлення має бути більшою за поточну дату"); }
    if (dateTimePicker1.Value.Date >= dateTimePicker2.Value.Date) { MessageBox.Show("Дата виготовлення має бути меншою за дату придатності"); dateTimePicker1.Value = DateTime.Today; }
}

private void dateTimePicker2_ValueChanged(object sender, EventArgs e)
{
    if (dateTimePicker2.Value.Date < dateTimePicker1.Value.Date) { MessageBox.Show("Термін придатності має бути більшим за дату виготовлення"); dateTimePicker2.Value = DateTime.Now.AddDays(1); }
    if (dateTimePicker2.Value.Date <= DateTime.Now.Date) { MessageBox.Show("Дата придатності має бути більшою за поточну дату"); dateTimePicker2.Value = DateTime.Now.AddDays(1); }
}

private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text.Length != 0 && maskedTextBox1.Text.Length!=0 && comboBox1.Text.Length !=0 && textBox3.Text.Length !=0 && comboBox3.Text.Length!=0 && comboBox4.Text.Length!=0)
    {
        int x = 0;
        using (var connection = new SqlConnection(ConnectionString))
        {
            using (var command = new SqlCommand("Check_Table_Product", connection))
            {
                connection.Open();
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@IDFirm", comboBox4.SelectedValue);
                command.Parameters.AddWithValue("@NameProduct", textBox3.Text);
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read()) x++; reader.Close();
                connection.Close();
            }
            if (x == 0)
            {
                using (var command = new SqlCommand("Insert_Product", connection))
                {
                    try
                    {
                        connection.Open();
                        command.CommandType = CommandType.StoredProcedure;
                        command.Parameters.AddWithValue("@IDFirm", comboBox4.SelectedValue);
                        command.Parameters.AddWithValue("@NameProduct", textBox3.Text);
                        command.Parameters.AddWithValue("@UnitPrice", textBox1.Text);
                        command.Parameters.AddWithValue("@IDUnit", comboBox1.SelectedValue);
                        command.Parameters.AddWithValue("@Made", dateTimePicker1.Value);
                        command.Parameters.AddWithValue("@Apply", dateTimePicker2.Value);
                        command.Parameters.AddWithValue("@IDWarehouse", comboBox3.SelectedValue);
                        command.Parameters.AddWithValue("@Percenteg", maskedTextBox1.Text);
```

```
                int result = command.ExecuteNonQuery();
                if (result != 0)
                    MessageBox.Show("Продукт додано");
                else MessageBox.Show("Виникла помилка!");
                connection.Close();
                this.Close();
                view_forms.view_product product = new view_forms.view_product();
                product.Show();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Неправильне введення даних");
            }
            finally
            {
                connection.Close();
            }
        }
    }
    else MessageBox.Show("Такий товар вже існує!");
  }

  }
  else MessageBox.Show("Усі поля мають бути заповнені");
}

private void comboBox4_KeyPress(object sender, KeyPressEventArgs e)
{
  if (Char.IsDigit(e.KeyChar) != true) return;
  e.Handled = true;
  return;
}

private void comboBox1_KeyPress(object sender, KeyPressEventArgs e)
{
  if (Char.IsDigit(e.KeyChar) != true) return;
  e.Handled = true;
  return;
}

private void comboBox3_KeyPress(object sender, KeyPressEventArgs e)
{
  if (Char.IsDigit(e.KeyChar) != true) return;
  e.Handled = true;
  return;
}

private void button2_Click(object sender, EventArgs e)
{
  this.Close();
  view_forms.view_product product = new view_forms.view_product();
  product.Show();
}

private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
  if (!Char.IsDigit(e.KeyChar) != true) return;
  e.Handled = true;
  return;
}
}
```

```
public partial class add_warehouse : Form
  {
    public add_warehouse()
    {
      InitializeComponent();
    }
    string ConnectionString;
    private void button1_Click(object sender, EventArgs e)
    {
      if (textBox1.Text.Length != 0)
      {
        int x = 0;
        using (var connection = new SqlConnection(ConnectionString))
        {
          using (var command = new SqlCommand("Check_Table_Warehouse", connection))
          {
            connection.Open();
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@NameWarehouse", textBox1.Text);
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read()) x++; reader.Close();
            connection.Close();
          }
          if (x == 0)
          {
            using (var command = new SqlCommand("Insert_Warehouse", connection))
            {
              try
              {
                connection.Open();
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@NameWarehouse", textBox1.Text);
                int result = command.ExecuteNonQuery();
                if (result != 0)
                  MessageBox.Show("Склад додано");
                else MessageBox.Show("Виникла помилка!");
                connection.Close();
                this.Close();
                view_forms.view_warehouse warehouse = new view_forms.view_warehouse();
                warehouse.Show();
              }
              catch (Exception ex)
              {
                MessageBox.Show("Неправильне введення даних");
              }
              finally
              {
                connection.Close();
              }
            }
          }
          else MessageBox.Show("Такий склад уже існує!");
        }
      }else MessageBox.Show("Усі поля мають бути заповнені");
    }

    private void add_warehouse_Load(object sender, EventArgs e)
    {
      ConnectionString = Properties.Settings.Default.FirmConnectionString;
    }

    private void button2_Click(object sender, EventArgs e)
```

```
    {
        this.Close();
        view_forms.view_warehouse warehouse = new view_forms.view_warehouse();
        warehouse.Show();
    }

    private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (Char.IsDigit(e.KeyChar) != true) return;
        e.Handled = true;
        return;
    }
}

public partial class add_salescontract : Form
{
    public add_salescontract()
    {
        InitializeComponent();
    }
    string ConnectionString;

    private void button2_Click(object sender, EventArgs e)
    {
        this.Close();
        view_forms.view_salescontract salescontract = new view_forms.view_salescontract();
        salescontract.Show();
    }

    private void add_salescontract_Load(object sender, EventArgs e)
    {
        ConnectionString = Properties.Settings.Default.FirmConnectionString;
        dateTimePicker1.Value = DateTime.Today.AddDays(1);
        using (var connection = new SqlConnection(ConnectionString))
        {
            using (var command = new SqlCommand("SELECT * FROM Customer", connection))
            {
                BindingSource bindingSource = new BindingSource();
                DataSet ds = new DataSet();
                SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                dataAdapter.Fill(ds, "Customer");
                bindingSource.DataSource = ds;
                bindingSource.DataMember = "Customer";
                comboBox3.DataSource = bindingSource;
                comboBox3.ValueMember = "IDCustomer";
                comboBox3.DisplayMember = "Surname";
                comboBox3.SelectedIndex = -1;
            }
            using (var command = new SqlCommand("SELECT * FROM DeliveryMethod", connection))
            {
                BindingSource bindingSource = new BindingSource();
                DataSet ds = new DataSet();
                SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                dataAdapter.Fill(ds, "DeliveryMethod");
                bindingSource.DataSource = ds;
                bindingSource.DataMember = "DeliveryMethod";
                comboBox4.DataSource = bindingSource;
                comboBox4.ValueMember = "IDDeliveryMethod";
                comboBox4.DisplayMember = "NameMethod";
                comboBox4.SelectedIndex = -1;
            }
            using (var command = new SqlCommand("SELECT * FROM TypeOfPayment", connection))
```

```
            {
                BindingSource bindingSource = new BindingSource();
                DataSet ds = new DataSet();
                SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                dataAdapter.Fill(ds, "TypeOfPayment");
                bindingSource.DataSource = ds;
                bindingSource.DataMember = "TypeOfPayment";
                comboBox2.DataSource = bindingSource;
                comboBox2.ValueMember = "IDPayment";
                comboBox2.DisplayMember = "NameType";
                comboBox2.SelectedIndex = -1;
            }
        }
    }

    private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
    {
        if (DateTime.Today >= dateTimePicker1.Value) { MessageBox.Show("Дата договору не може бути меншою або дорівнювати поточній даті!"); dateTimePicker1.Value = DateTime.Today.AddDays(1); }
    }
    private bool Check_Null(string s1, string s2, string s3, string s4)
    {
        if (s1 == null || s1.Length == 0) return true;
        if (s2 == null || s2.Length == 0) return true;
        if (s3 == null || s3.Length == 0) return true;
        if (s4 == null || s4.Length == 0) return true;
        return false;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        bool flag = Check_Null(textBox1.Text, comboBox2.Text, comboBox3.Text, comboBox4.Text);
        if (flag == false)
        {
            using (var connection = new SqlConnection(ConnectionString))
            {
                using (var command = new SqlCommand("Insert_Sales_Contract", connection))
                {
                    try
                    {
                        connection.Open();
                        command.CommandType = CommandType.StoredProcedure;
                        command.Parameters.AddWithValue("@IDCustomer", comboBox3.SelectedValue);
                        command.Parameters.AddWithValue("@Date", dateTimePicker1.Value);
                        command.Parameters.AddWithValue("@IDDeliveryMethod", comboBox4.SelectedValue);
                        command.Parameters.AddWithValue("@IDPayment", comboBox2.SelectedValue);
                        command.Parameters.AddWithValue("@NameSC", textBox1.Text);
                        int result = command.ExecuteNonQuery();
                        if (result != 0)
                            MessageBox.Show("Договір додано");
                        else MessageBox.Show("Виникла помилка!");
                        connection.Close();
                        this.Close();
                        view_forms.view_salescontract salescontract = new view_forms.view_salescontract();
                        salescontract.Show();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Неправильне введення даних!");
                    }
                    finally
                    {
                        connection.Close();
```

```
                }
            }
        }
    }
    else MessageBox.Show("Усі поля мають бути заповнені!");
  }
}

public partial class view_sklad : Form
  {
    public view_sklad()
    {
      InitializeComponent();
    }
    string ConnectionString; void ClearDataGridView()
    {
      while (dataGridView1.Rows.Count > 1)
        for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
          dataGridView1.Rows.Remove(dataGridView1.Rows[i]);
    }
    private void назадToolStripMenuItem_Click(object sender, EventArgs e)
    {
      this.Close();
      view_product product = new view_product();
      product.Show();
    }

    private void view_sklad_Load(object sender, EventArgs e)
    {
      ConnectionString = Properties.Settings.Default.FirmConnectionString;
      ClearDataGridView();
      int x = 0;
      using (var connection = new SqlConnection(ConnectionString))
      {
        using (var command = new SqlCommand("Select_ConuntProdSklad", connection))
        {
          DataSet ds = new DataSet();
          BindingSource source = new BindingSource();
          connection.Open();
          SqlDataReader reader = command.ExecuteReader();
          while (reader.Read()) x++; reader.Close();
          SqlDataAdapter house = new SqlDataAdapter(command);
          SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
          house.Fill(ds, "ConuntProdSklad");
          source.DataSource = ds;
          source.DataMember = "ConuntProdSklad";
          dataGridView1.DataSource = source;
          dataGridView1.Columns[0].Visible = false;

          connection.Close();
        }
      }
    }

    private void comboBox1_SelectionChangeCommitted(object sender, EventArgs e)
    {
      if(comboBox1.SelectedIndex==0)
      {
        button2.Enabled = false;
        ClearDataGridView();
        view_sklad_Load(sender,e);
      }
```

```
else if(comboBox1.SelectedIndex==1)
{
   ClearDataGridView();
   using (var connection = new SqlConnection(ConnectionString))
   {
      button2.Enabled = false;
      int x = 0;
      using (var command = new SqlCommand("Select_ConuntProdSklad_date_0", connection))
      {
         DataSet ds = new DataSet();
         BindingSource source = new BindingSource();
         connection.Open();
         SqlDataReader reader = command.ExecuteReader();
         while (reader.Read()) x++; reader.Close();
         SqlDataAdapter house = new SqlDataAdapter(command);
         SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
         house.Fill(ds, "ConuntProdSklad");
         source.DataSource = ds;
         source.DataMember = "ConuntProdSklad";
         dataGridView1.DataSource = source;
         dataGridView1.Columns[0].Visible = false;
         connection.Close();
      }
   }
} else if(comboBox1.SelectedIndex ==2)
{
   button2.Enabled = false;
   ClearDataGridView();
   using (var connection = new SqlConnection(ConnectionString))
   {
      int x = 0;
      using (var command = new SqlCommand("Func2", connection))
      {
         DataSet ds = new DataSet();
         BindingSource source = new BindingSource();
         connection.Open();
         SqlDataReader reader = command.ExecuteReader();
         while (reader.Read()) x++; reader.Close();
         SqlDataAdapter house = new SqlDataAdapter(command);
         SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
         house.Fill(ds, "ConuntProdSklad");
         source.DataSource = ds;
         source.DataMember = "ConuntProdSklad";
         dataGridView1.DataSource = source;
         dataGridView1.Columns[0].Visible = false;
         connection.Close();
      }
   }
}
else if(comboBox1.SelectedIndex==3)
{
   button2.Enabled = true;
   ClearDataGridView();
   using (var connection = new SqlConnection(ConnectionString))
   {
      int x = 0;
      using (var command = new SqlCommand("Func4", connection))
      {
         DataSet ds = new DataSet();
         BindingSource source = new BindingSource();
         connection.Open();
         SqlDataReader reader = command.ExecuteReader();
```

```
            while (reader.Read()) x++; reader.Close();
            SqlDataAdapter house = new SqlDataAdapter(command);
            SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
            house.Fill(ds, "Products");
            source.DataSource = ds;
            source.DataMember = "Products";
            dataGridView1.DataSource = source;
            dataGridView1.Columns[0].Visible = false;
            connection.Close();
        }
    }
  }
}

private void button1_Click(object sender, EventArgs e)
{
  button2.Enabled = false;
  if (comboBox2.Text.Length != 0 && textBox1.Text.Length != 0)
  {
    try
    {
      ClearDataGridView();
      using (var connection = new SqlConnection(ConnectionString))
      {
        int x = 0;
        if (comboBox2.SelectedIndex == 0)
        {
          using (var command = new SqlCommand("Func2_1", connection))
          {
            DataSet ds = new DataSet();
            BindingSource source = new BindingSource();
            connection.Open();
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@Count", textBox1.Text);
            SqlDataAdapter house = new SqlDataAdapter(command);
            SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
            house.Fill(ds, "ConuntProdSklad");
            source.DataSource = ds;
            source.DataMember = "ConuntProdSklad";
            dataGridView1.DataSource = source;
            dataGridView1.Columns[0].Visible = false;
            connection.Close();
          }
        }
        else if (comboBox2.SelectedIndex == 1)
        {
          using (var command = new SqlCommand("Func2_2", connection))
          {
            DataSet ds = new DataSet();
            BindingSource source = new BindingSource();
            connection.Open();
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@Count", textBox1.Text);
            SqlDataAdapter house = new SqlDataAdapter(command);
            SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
            house.Fill(ds, "ConuntProdSklad");
            source.DataSource = ds;
            source.DataMember = "ConuntProdSklad";
            dataGridView1.DataSource = source;
            dataGridView1.Columns[0].Visible = false;
            connection.Close();
          }
```

```
                }
                else if (comboBox2.SelectedIndex == 2)
                {
                    using (var command = new SqlCommand("Func2_3", connection))
                    {
                        DataSet ds = new DataSet();
                        BindingSource source = new BindingSource();
                        connection.Open();
                        command.CommandType = CommandType.StoredProcedure;
                        command.Parameters.AddWithValue("@Count", textBox1.Text);
                        SqlDataAdapter house = new SqlDataAdapter(command);
                        SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
                        house.Fill(ds, "ConuntProdSklad");
                        source.DataSource = ds;
                        source.DataMember = "ConuntProdSklad";
                        dataGridView1.DataSource = source;
                        dataGridView1.Columns[0].Visible = false;
                        connection.Close();
                    }
                }
            }
            catch { MessageBox.Show("Неправильне введення даних!"); }
        }
        else MessageBox.Show("Усі поля мають бути заповнені!");
    }

    private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!Char.IsDigit(e.KeyChar) != true) return;
        e.Handled = true;
        return;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        button2.Enabled = false;
        DialogResult result;
        MessageBoxButtons but = MessageBoxButtons.YesNo;
        result = MessageBox.Show("Ви точно хочете видалити запис? Видалення зі сполучної таблиці спричинить
видалення по всій базі даних ", "Повідомлення", but);
        if (result == DialogResult.Yes)
        {

            try
            {
                SqlConnection conn = new SqlConnection(ConnectionString);
                conn.Open();
                int id = Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value);
                string sql = string.Format("DELETE FROM Products WHERE IDProduct='{0}'", id);
                using (SqlCommand cnd = new SqlCommand(sql, conn))
                {
                    try
                    {
                        cnd.ExecuteNonQuery();
                        MessageBox.Show("Видалено!");
                    }
                    catch (SqlException ex)
                    {
                        Exception error = new Exception("Помилка видалення", ex);
                        throw error;
                    }
```

```
            }
            ClearDataGridView();
            view_sklad_Load(sender, e);
          }
          catch { MessageBox.Show("Помилка видалення", "Повідомлення"); }
        }
      }
    }

public partial class view_deliveryorders : Form
    {
      public view_deliveryorders()
      {
        InitializeComponent();
      }
      string ConnectionString;
      private void назадToolStripMenuItem_Click(object sender, EventArgs e)
      {
        this.Close();
        Orders_data orders_Data = new Orders_data();
        orders_Data.Show();
      }
      void ClearDataGridView()
      {
        while (dataGridView1.Rows.Count > 1)
          for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
            dataGridView1.Rows.Remove(dataGridView1.Rows[i]);
      }
      private void view_deliveryorders_Load(object sender, EventArgs e)
      {
        ConnectionString = Properties.Settings.Default.FirmConnectionString;
        ClearDataGridView();
        int x = 0;
        using (var connection = new SqlConnection(ConnectionString))
        {
          using (var command = new SqlCommand("Select_OrderDelivery", connection))
          {
            DataSet ds = new DataSet();
            BindingSource source = new BindingSource();
            connection.Open();
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read()) x++; reader.Close();
            SqlDataAdapter house = new SqlDataAdapter(command);
            SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(house);
            house.Fill(ds, "OrderDelivery");
            source.DataSource = ds;
            source.DataMember = "OrderDelivery";
            dataGridView1.DataSource = source;
            dataGridView1.Columns[0].Visible = false;

            connection.Close();
          }
        }
      }

      private void добавитьToolStripMenuItem_Click(object sender, EventArgs e)
      {
        this.Close();
        add.add_ordersdelivery ordersdelivery = new add.add_ordersdelivery();
        ordersdelivery.Show();
      }
```

```
    private void неактульныеЗаказыToolStripMenuItem_Click(object sender, EventArgs e)
    {
      this.Close();
      view_deliveryorders_date_0 view_ = new view_deliveryorders_date_0();
      view_.Show();
    }
  }
public partial class add_ordersales : Form
  {
    string ConnectionString;
    public add_ordersales()
    {
      InitializeComponent();
    }

    private void add_ordersales_Load(object sender, EventArgs e)
    {
      ConnectionString = Properties.Settings.Default.FirmConnectionString;
      using (var connection = new SqlConnection(ConnectionString))
      {
        connection.Open();
        using (var command = new SqlCommand("SELECT * FROM SalesContract as D Where GETDATE()<=D.Date",
connection))
        {
          BindingSource bindingSource = new BindingSource();
          DataSet ds = new DataSet();
          SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
          dataAdapter.Fill(ds, "SalesContract");
          bindingSource.DataSource = ds;
          bindingSource.DataMember = "SalesContract";
          comboBox1.DataSource = bindingSource;
          comboBox1.ValueMember = "IDSalesContract";
          comboBox1.DisplayMember = "NameSC";
          comboBox1.SelectedIndex = -1;
          connection.Close();
        }
        using (var command = new SqlCommand("SELECT * FROM Firm", connection))
        {
          BindingSource bindingSource = new BindingSource();
          DataSet ds = new DataSet();
          SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
          dataAdapter.Fill(ds, "Firm");
          bindingSource.DataSource = ds;
          bindingSource.DataMember = "Firm";
          comboBox2.DataSource = bindingSource;
          comboBox2.ValueMember = "IDFirm";
          comboBox2.DisplayMember = "Name_Firm";
          comboBox2.SelectedIndex = -1;
          connection.Close();
        }

      }
    }

    private void comboBox2_SelectionChangeCommitted(object sender, EventArgs e)
    {
      label12.Text = "0";
      label5.Text = "0";
      label9.Text = "0";
      label8.Text = "0"; textBox1.Text = "0";
      int x = 0;
```

```
        using (var connection = new SqlConnection(ConnectionString))
        {
            connection.Open();
            using (var command = new SqlCommand("SELECT * FROM Products as P,ConuntProdSklad as C Where P.IDFirm =
@IDFirm and P.Apply > GETDATE() and P.IDProduct = C.IDProduct and C.Count>0", connection))
            {
                BindingSource bindingSource = new BindingSource();
                DataSet ds = new DataSet();
                command.Parameters.AddWithValue("@IDFirm", comboBox2.SelectedValue);
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read()) x++;
                reader.Close();
                SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                dataAdapter.Fill(ds, "Products");
                bindingSource.DataSource = ds;
                bindingSource.DataMember = "Products";
                comboBox3.DataSource = bindingSource;
                comboBox3.ValueMember = "IDProduct";
                comboBox3.DisplayMember = "NameProduct";
                comboBox3.SelectedIndex = -1;
                connection.Close();
            }
        }
    }

    private void comboBox3_SelectionChangeCommitted(object sender, EventArgs e)
    {
        label12.Text = "0";
        label8.Text = "0"; textBox1.Text = "0";
        using (var connection = new SqlConnection(ConnectionString))
        {
            using (var command = new SqlCommand("Out_Product_Procent", connection))
            {
                try
                {
                    command.CommandType = CommandType.StoredProcedure;
                    command.Parameters.AddWithValue("@IDProduct", comboBox3.SelectedValue);
                    SqlParameter money = new SqlParameter
                    {
                        DbType = DbType.Double,
                        ParameterName = "@UnitPrice",
                        Direction = ParameterDirection.Output,
                    };
                    SqlParameter date = new SqlParameter
                    {
                        DbType = DbType.Date,
                        ParameterName = "@Apply",
                        Direction = ParameterDirection.Output,
                    };
                    SqlParameter sklad = new SqlParameter
                    {
                        DbType = DbType.Int32,
                        ParameterName = "@Count",
                        Direction = ParameterDirection.Output,
                    };
                    command.Parameters.Add(money);
                    command.Parameters.Add(date);
                    command.Parameters.Add(sklad);
                    connection.Open();
                    int result_procedure = command.ExecuteNonQuery();
                    string s = command.Parameters["@UnitPrice"].Value.ToString();
                    label5.Text = (Math.Round(float.Parse(s), 1)).ToString();
```

```
                label9.Text = command.Parameters["@Apply"].Value.ToString();
                label12.Text = command.Parameters["@Count"].Value.ToString();

                connection.Close();

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                connection.Close();
            }
        }
    }
}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
    view_forms.view_salesorder salesorder = new view_forms.view_salesorder();
    salesorder.Show();
}

private void textBox1_KeyUp(object sender, KeyEventArgs e)
{
    if (textBox1.Text.Length != 0)
    {
        if (int.Parse(textBox1.Text) > int.Parse(label12.Text) || int.Parse(textBox1.Text) < 0) { textBox1.ForeColor =
Color.Red; label8.Text = "0"; }
        else
        {
            textBox1.ForeColor = Color.Green;

            label8.Text = (int.Parse(textBox1.Text) * double.Parse(label5.Text)).ToString();
        }
    }
    else textBox1.Text = "0";

}

private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text.Length != 0 && comboBox1.Text.Length != 0 && comboBox2.Text.Length != 0 &&
comboBox3.Text.Length != 0)
    {
        int x = 0;
        using (var connection = new SqlConnection(ConnectionString))
        {

            using (var command = new SqlCommand("Insert_OrderSales", connection))
            {
                try
                {
                    connection.Open();
                    command.CommandType = CommandType.StoredProcedure;
                    command.Parameters.AddWithValue("@IDSalesContract", comboBox1.SelectedValue);
                    command.Parameters.AddWithValue("@IDProduct", comboBox3.SelectedValue);
                    command.Parameters.AddWithValue("@Quantity", textBox1.Text);
                    command.Parameters.AddWithValue("@MoneyContract", label8.Text);
                    int result = command.ExecuteNonQuery();
```

```
                if (result != 0)
                    MessageBox.Show("Замовлення додано");
                else MessageBox.Show("Виникла помилка!");
                connection.Close();

            }
            catch (Exception ex)
            {
                MessageBox.Show("Неправильне введення даних");
            }
            finally
            {
                connection.Close();
            }
        }
    }
    if (double.Parse(textBox2.Text) < double.Parse(label8.Text))
    {
        int sum_dolg = int.Parse(label8.Text) - int.Parse(textBox2.Text);

        using (var connection = new SqlConnection(ConnectionString))
        {

            using (var command = new SqlCommand("Insert_Debtors", connection))
            {
                connection.Open();
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@IDSalesContract", comboBox1.SelectedValue);
                command.Parameters.AddWithValue("@IDProduct", comboBox3.SelectedValue);
                command.Parameters.AddWithValue("@debt", sum_dolg);
                int result = command.ExecuteNonQuery();
                if (result != 0)
                    MessageBox.Show("Вам нараховано борг! Сума боргу= " + sum_dolg);
                else MessageBox.Show("Виникла помилка!");
                this.Close();
                view_forms.view_salesorder salesorder = new view_forms.view_salesorder();
                salesorder.Show();
                connection.Close();
            }
        }
    }
    else
    {
        this.Close();
        view_forms.view_salesorder salesorder = new view_forms.view_salesorder();
        salesorder.Show();
    }

  }
  else MessageBox.Show("Всі поля мають бути заповнені!");

}

  private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
  {
    if (!Char.IsDigit(e.KeyChar) != true) return;
    e.Handled = true;
    return;
  }

  private void comboBox1_KeyPress(object sender, KeyPressEventArgs e)
  {
```

```
        if (Char.IsDigit(e.KeyChar) != true) return;
        e.Handled = true;
        return;
      }

      private void comboBox2_KeyPress(object sender, KeyPressEventArgs e)
      {
        if (Char.IsDigit(e.KeyChar) != true) return;
        e.Handled = true;
        return;
      }

      private void comboBox3_KeyPress(object sender, KeyPressEventArgs e)
      {
        if (Char.IsDigit(e.KeyChar) != true) return;
        e.Handled = true;
        return;
      }
    }

GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Insert_Product]
 @IDFirm int,
 @NameProduct nvarchar(50),
 @UnitPrice money,
 @IDUnit int,
 @Made date,
 @Apply date,
 @IDWarehouse int,
 @Percenteg numeric(18,1)
AS
  INSERT INTO Products VALUES
(@IDFirm,@NameProduct,@UnitPrice,@IDUnit,@Made,@Apply,@IDWarehouse,@Percenteg)
RETURN 0

GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Func2]
AS
       SELECT C.IDConuntProdSklad, F.Name_Firm as 'Фірма',P.NameProduct as 'Товар',C.Count
as 'Кількість',C.Money as 'Сума'
       From ConuntProdSklad as C, Products as P, Firm as F
       Where C.IDProduct = P.IDProduct and P.IDFirm = F.IDFirm and P.Apply > GETDATE() and
C.Count = 0
RETURN 0

GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Func2_1]
       @Count int
as
       SELECT P.IDProduct, F.Name_Firm as 'Фірма',P.NameProduct as 'Товар',
count(P.NameProduct) as Замовлення'
       From ConuntProdSklad as C, Products as P, Firm as F, OrderSale as O
       Where C.IDProduct = P.IDProduct and P.IDFirm = F.IDFirm and P.Apply > GETDATE() and
P.IDProduct = O.IDProduct
        group by P.IDProduct, F.Name_Firm,P.NameProduct
  having COUNT(P.NameProduct) > @Count
RETURN 0
GO
```

```
SET QUOTED_IDENTIFIER ON

GO
ALTER PROCEDURE [dbo].[Func4]
AS
       SELECT P.IDProduct, F.Name_Firm as 'Виробник', P.NameProduct as 'Товар'
       From Products as P, Firm as F
       Where Not EXISTS(Select * From OrderSale as O Where O.IDProduct = P.IDProduct ) and
P.Apply > GETDATE() and P.IDFirm =F.IDFirm
RETURN 0
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна робота_Залепа.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота _ Залепа.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Program.rar	Архів. Містить коди програми і скомпільований додаток.
Презентація	
Презентація_ Залепа.pptx	Презентація кваліфікаційної роботи.