**Міністерство освіти і науки України**
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету )
Кафедра комп'ютерних наук
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: "Розробка програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі"

Виконав: студент VI курсу, групи СТм-61
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

(підпис) Ралік І.Р. (прізвище та ініціали)

Керівник (підпис) Готович В.А. (прізвище та ініціали)

Нормоконтроль (підпис) Мацюк О.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Козак Р.О. (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 2022 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Ралік Ігор Романович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі

Керівник роботи Готович Володимир Анатолійович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_22_» _листопада__ 2022 року № _4/7-949___
2. Термін подання студентом завершеної роботи 23 грудня 2022р.
3. Вихідні дані до роботи Наукові публікації про ринок торгівлі товарами, бази даних, середовища розробки, про програмне забезпечення для обліку реалізації товарів в торгівлі

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз відомих рішень по розробці програмного забезпечення для обліку реалізації товарів в торгівлі. 2 Обгрунтування вибору засобів та технологій по розробці програмного забезпечення для обліку реалізації товарів в торгівлі. 3 Реалізація програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі. 4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Охорона праці | Мацюк О.В., доцент | | |
| Безпека в надзвичайних ситуаціях | Клепчик В.М., ст. викл. | | |

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Ознайомлення з завданням до кваліфікаційної роботи | | *Виконано* |
| 2. | Підбір літературних джерел про ринок торгівлі в | | *Виконано* |
| | Україні та світі, про програмні засоби для реалізації | | |
| | програмного забезпечення, бази даних. | | |
| 3. | Переклад та опрацювання наукових джерел про | | *Виконано* |
| | потреби ринку торгівлі в програмного забезпеченні | | |
| | для обліку реалізації товарів. | | |
| 4. | Виконання дослідження щодо ринку торгівлі, щодо | | *Виконано* |
| | відомих програмних рішень для обліку реалізації | | |
| | товарів в торгівлі та про засоби реалізації такого ПЗ | | |
| 5. | Оформлення розділу «Аналіз відомих рішень по | | *Виконано* |
| | розробці програмного забезпечення для обліку | | |
| | реалізації товарів в торгівлі» | | |
| 6. | Оформлення розділу «Обґрунтування вибору засобів | | *Виконано* |
| | та технологій по розробці програмного | | |
| | забезпечення для обліку товарів в торгівлі» | | |
| 7. | Оформлення розділу «Реалізація програмного забез | | *Виконано* |
| | печення на основі клієнт-серверної архітектури для | | |
| | обліку реалізації товарів в торгівлі» | | |
| 8. | Виконання завдання до підрозділу «Охорона праці» | | *Виконано* |
| 9. | Виконання завдання до підрозділу «Безпека в | | *Виконано* |
| | надзвичайних ситуаціях» | | |
| 10. | Оформлення кваліфікаційної роботи | | *Виконано* |
| 11. | Нормоконтроль | | *Виконано* |
| 12. | Перевірка на плагіат | | *Виконано* |
| 13. | Попередній захист кваліфікаційної роботи | | *Виконано* |
| 14. | Захист кваліфікаційної роботи | | |
| | | | |
| | | | |

Студент _____ Ралік І.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В.А.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Розробка програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі // Кваліфікаційна робота освітнього рівня «Магістр» // Ралік Ігор Романович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СТм-61 // Тернопіль, 2022 // С. 90, рис. – 29, табл. – 1, кресл. – 19, додат. – 4, бібліогр. – 58.

Ключові слова: програмне забезпечення для обліку, база даних, C#, Microsoft SQL Server, ADO.NET, клієнт-серверна архітектура.

Кваліфікаційна робота присвячена розробці програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі.

В першому розділі кваліфікаційної роботи проаналізовано та описано ринок торгівлі в Україні та світі, сформульовано актуальність задачі по розробці програмного забезпечення для обліку реалізації товарів в торгівлі, проведено огляд відомих рішень програмного забезпечення у цього сегменті.

В другому розділі кваліфікаційної роботи здійснено аналіз засобів та технологій розробки програмного забезпечення. Наведено аргументацію щодо вибору бази даних, архітектури, мови програмування та середовища розробки для реалізації поставленої задачі.

В третьому розділі кваліфікаційної роботи описано реалізацію програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі. Продемонстровано функціональні можливості розробленого програмного рішення.

# ANNOTATION

Software Development Based On Client-Server Architecture for Accounting of Products in Retail // Qualification work of the educational level "Master" // Ralik Ihor Romanovych // Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, STm-61 group // Ternopil, 2022 // P. 90, fig. – 29, tables – 1, chair. – 19, annexes – 4, references. – 58.

Key words: software for accounting, database, C#, Microsoft SQL Server, ADO.NET, client-server architecture.

Thesis is devoted to the development of software based on a client-server architecture for accounting for the sale of goods in retail.

The first section of the qualification work analyzed and described the retail market in Ukraine and the world, the relevance of the task of developing software for accounting for sales of goods in trade is formulated, an overview of well-known software solutions in this segment is carried out.

In the second section of the qualification work, the means and technologies of software development were analyzed. Arguments regarding the choice of a database, architecture, programming language and development environment for the implementation of the given task.

The third section of the qualification work describes the implementation of software based on the client-server architecture for accounting for the sale of goods in retail. Functional capabilities of the developed software solution are demonstrated.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЄСВ – єдиний соціальний внесок.

ООП – методологія програмування, заснована на представленні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, де класи утворюють ієрархію спадкування.

ПЗ – Програмне забезпечення.

ПК – Персональний комп’ютер.

СУБД – система управління базами даних, комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації.

API – опис способів взаємодії однієї комп'ютерної програми з іншими.

CRM (англ. Customer Relationship Management) – концепції, які використовуються компаніями для управління взаємовідносинами з клієнтами.

CSS – мова стилю сторінок в інтернеті.

C, C++, C# – мови програмування.

C# – об'єктно-орієнтована мова програмування.

Document/JSON – формат документів JSON.

HTML – мова тегів, засобами якої здійснюється розмітка веб-сторінок в мережі інтернет.

IDE – програмне рішення для розробки програмного забезпечення.

JavaScript – об'єктно-орієнтована мова програмування.

JSON – текстовий формат обміну даними між комп'ютерами.

Microsoft Visual Studio – середовище для створення програмного забезпечення.

.NET – платформа від Microsoft, яка дозволяє створювати програмні програми.

NoSQL – база даних, що забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць та відношень в реляційних базах даних.

OLTP – онлайнова обробка транзакцій, спосіб організації баз даних.

SQL (англ. Structured query language – мова структурованих запитів) – декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних.

TypeScript – мова прогрумування для створення веб застосунків.

XML – стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками.

XSLT – функціональна мова програмування, яка використовується для програмування переробки XML-документів.

# ЗМІСТ

# ВСТУП

**Актуальність теми.** На сьогоднішній день в Україні та світі функціонують тисячі компаній, які займаються торгівлею. Світовий досвід показав ефективність застосування програмного забезпечення, призначеного для обліку процесів реалізації товарів в електронній формі. Із певних причин, використання для цього популярних програних рішень іноземного виробництва в Україні є ускладненим. Тому, актуальною є задача по розробці програмного забезпечення для обліку реалізації товарів в торгівлі вітчизняного походження.

**Мета і задачі дослідження.** Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є створення якісного програмного забезпечення на основі клієнт-серверної архітектури, яке дозволить вести облік реалізації товарів у електронному форматі та автоматизує процеси в торгівлі. Для досягнення поставленої мети потрібно виконати наступні завдання:

– Проаналізувати наявні на сьогоднішній день потреби щодо програмного забезпечення з функціональними можливостями обліку реалізації товарів в торгівлі вітчизняного походження.

– Обгрунтувати вибір інструментальних засобів та технологій, які будуть оптимальними для розробки програмного забезпечення даного типу.

– Розробити програмне забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі.

**Об’єкт дослідження**. Автоматизація процесу вирішення задач обліку реалізації товарів в торгівлі.

**Предмет дослідження.** Програмне забезпечення на основі клієнт-серверної архітектури, що реалізує функціональні можливості по обліку реалізації товарів в торгівлі.

**Наукова новизна одержаних результатів** кваліфікаційної роботи полягає в розробці якісного програмного забезпечення вітчизняного

походження на основі сучасних інструментальних засобів та технологій, яке належним чином забезпечує потреби торгівлі в автоматизації процесів обліку реалізації товарів.

**Практичне значення одержаних результатів.** Створено програмне забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі.

**Апробація результатів магістерської роботи.** Основні результати проведених досліджень та отриманих результатів обговорювались на X Науково-технічній конференції «Інформаційні моделі, системи та технології» (м. Тернопіль, 7-8 грудня 2022 р.) та на XI Міжнародній науково-технічній конференції молодих учених та студентів «Актуальні задачі сучасних технологій» (м. Тернопіль, 7-8 грудня 2022 р.)

**Публікації.** Основні результати кваліфікаційної роботи опубліковано у двох працях конференцій (див. додатки А, Б).

**Структура й обсяг кваліфікаційної роботи.** Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 58 найменувань та 4 додатків. Загальний обсяг кваліфікаційної роботи без додатків складає 90 сторінок. Із них – 73 сторінок основного тексту, який містить 29 рисунків та 1 таблицю.

# 1 АНАЛІЗ ВІДОМИХ РІШЕНЬ ПО РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ РЕАЛІЗАЦІЇ ТОВАРІВ В ТОРГІВЛІ

## 1.1 Аналіз ринку торгівлі

На сьогоднішній день інформаційні технології широко застосовуються в торгівлі. Торгівлею вважається продаж продукції від підприємства (виробника) до клієнта (споживача). Торгові операції відбуваються в одній точці купівлі у формі звичайного магазину, веб-сайту, прямих продажів, тощо. Торгівля відрізняється від великомасштабних оптових угод тим, що вона спеціально спрямована на продаж населенню.

Перед створенням інформаційної системи для торгівлі необхідно розуміти, що більшість продавців не виробляють товарів, а лише продають їх. Роздрібні продавці закуповують у оптових постачальників великі обсяги продукції та продають її населенню поштучно. Іншими словами, роздрібний продавець є посередником у комерційному ланцюжку поставок між оптовиком і кінцевим споживачем.

З’ясуємо, що таке торгівля насправді, як вона функціонує та чому необхідні інформаційні системи для її обліку. Торгівля працює через ланцюг поставок, що складається з виробників, оптовиків, торговців і споживачів. Кожна частина ланцюга постачання має важливе значення для підтримки торгівлі.

Роздрібний ланцюжок поставок виглядає приблизно так:

1. Виробники виготовляють товари із сировини.

2. Оптові продавці закуповують товари у виробників за комерційними цінами.

3. Продавці закуповують товари у оптовиків у великих кількостях.

4. Продавці продають товари кінцевим споживачам у невеликих кількостях.

5. Споживачі купують товари для особистого користування.

Ланцюг постачання розподіляє продукцію від виробництва до споживання. На перший погляд може здатися, що ланцюг поставок спричиняє інфляцію цін, але насправді це знижує кінцеві ціни, оскільки виробникам не потрібно продавати власні товари, вони можуть підтримувати низькі витрати.

Існують такі типи магазинів в торгівлі: супермаркети, універмаги, інтернет-магазини, спеціалізовані магазини. Кожний магазин має різний тип товарів для закупівель, наприклад, хліб з пекарні або пара джинсів від модних брендів. Розглянемо кілька найпопулярніших типів магазинів в торгівлі:

1. Універмаг. Великі торговці, які продають товари зі свого асортименту разом із товарами інших компаній. Ці магазини поділені на відділи з широким асортиментом товарів на вибір. Від одягу до техніки та багато іншого, що ви можете собі уявити.

2. Інтернет-магазин. Ми всі знайомі з інтернет-магазинами, електронна комерція – це швидкозростаюче середовище торгівлі, популярне завдяки конкурентним цінам і зручним варіантам доставки. Такі гіганти онлайн торгівлі, як Amazon, можуть встановлювати ціни нижче роздрібної, укладаючи угоди з виробниками або продавцями.

3. Крамниця. Ці звичайні магазини є досить поширеним типом. Більшість міст мають магазини, де місцеві жителі можуть придбати товари першої необхідності: хліб, молоко, яйця, кондитерські вироби, тощо Ці магазини, як правило, подовжують години роботи для додаткової зручності.

4. Супермаркет. Супермаркети – гіганти торгівлі, і сьогодні більшість з них продають не тільки продукти. Зараз у багатьох великих супермаркетах світу є предмети домашнього вжитку, одяг, техніка, книги, тощо. Супермаркети мають додаткову цінність, урізноманітнивши свої послуги.

5. Склад магазину. Складські магазини масово продають величезний асортимент товарів за ціною, нижчою за ринкову. Ці магазини стають все більш популярними як спосіб заощадити гроші на першій необхідності. Клієнти можуть заощадити гроші, купуючи товари великими партіями.

6. Спеціалізований магазин. Спеціалізовані магазини спеціалізуються на одному конкретному типі товару. Зазвичай вони присутні на місцевих вулицях або в торгових центрах. Це може бути спеціалізований бакалійний магазин або магазин одягу. Все, що вам спадає на думку, для цього є спеціалізований магазин.

Торгівля це також поєднання видів діяльності, пов'язаних із продажем чи орендою споживчих товарів і послуг кінцевим споживачам для їх особистого використання. Окрім продажу, торгівля включає такі різноманітні види діяльності, як купівля, реклама, обробка даних чи підтримка запасів.

За словами Філіпа Котлера, торгівля включає всі види діяльності, пов'язані з продажем товарів або послуг споживачам для особистого використання [1]. Продавець або магазин – це будь-яке комерційне підприємство, обсяг продажів якого в основному відбувається за рахунок торгівлі у роздріб. Продавець може займатись продажем товару, наприклад, в продуктовому магазині чи книжковому магазині, або продавець може продавати послуги, для прикладу, це салон краси чи кінотеатр. Торговці - це комерційні фірми, які займаються реалізацією товарів і послуг споживачам. У більшості випадків роздрібні торгівельні точки займаються переважно продажем товарів. В основному такі підприємства продають окремі одиниці або невеличкі групи продуктів великій кількості клієнтів.

Деяка кількість торговців також отримує прибуток за рахунок оренди, а не прямого продажу товарів, наприклад, підприємства, які пропонують меблі чи садовий інвентар в оренду або поєднання продуктів і послуг, як у випадку магазину одягу, який пропонує безкоштовні заміни при покупці костюма. Таким чином, торгівля – це сукупність видів діяльності, пов'язаних із

продажем або орендою споживчих товарів або послуг кінцевим клієнтам для їх особистого або домашнього використання. Окрім продажу, торгівля включає такі різноманітні види діяльності, як купівля, реклама, обробка даних, управління робочою силою та обслуговування запасів.

Торгівля є одним з найбільших економічних секторів у світі та основним рушієм економіки країн. ЇЇ частка складає майже 10% ВВП більшості розвинених країн [2]. Галузь торгівлі переважає в розвинутих країнах, таких як США, Великобританія, тощо. Галузь також є основним роботодавцем у більшості економік країн світу: до 16% у США, 15% у Бразилії, 12% у Польщі, 7% у Китаї.

Торгівля також є одним із найбільших економічних секторів і в Україні. На рисунках 1.1 та 1.2 можна побачити, що роздрібна торгівля та оптова торгівля входять у кожний із топів видів діяльності популярних у 2018 році серед ФОП та юридичних осіб. Станом на теперішній час ця статистика майже не змінилась.

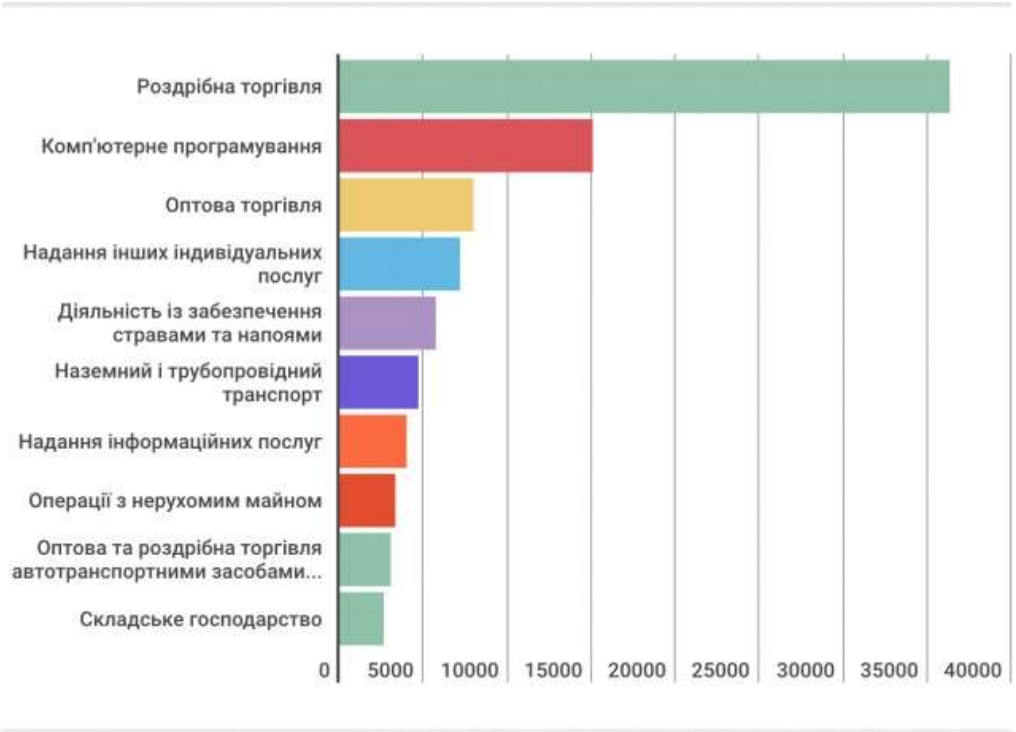



Рисунок 1.1 – Статистика видів діяльності серед фізичних осіб-підприємців

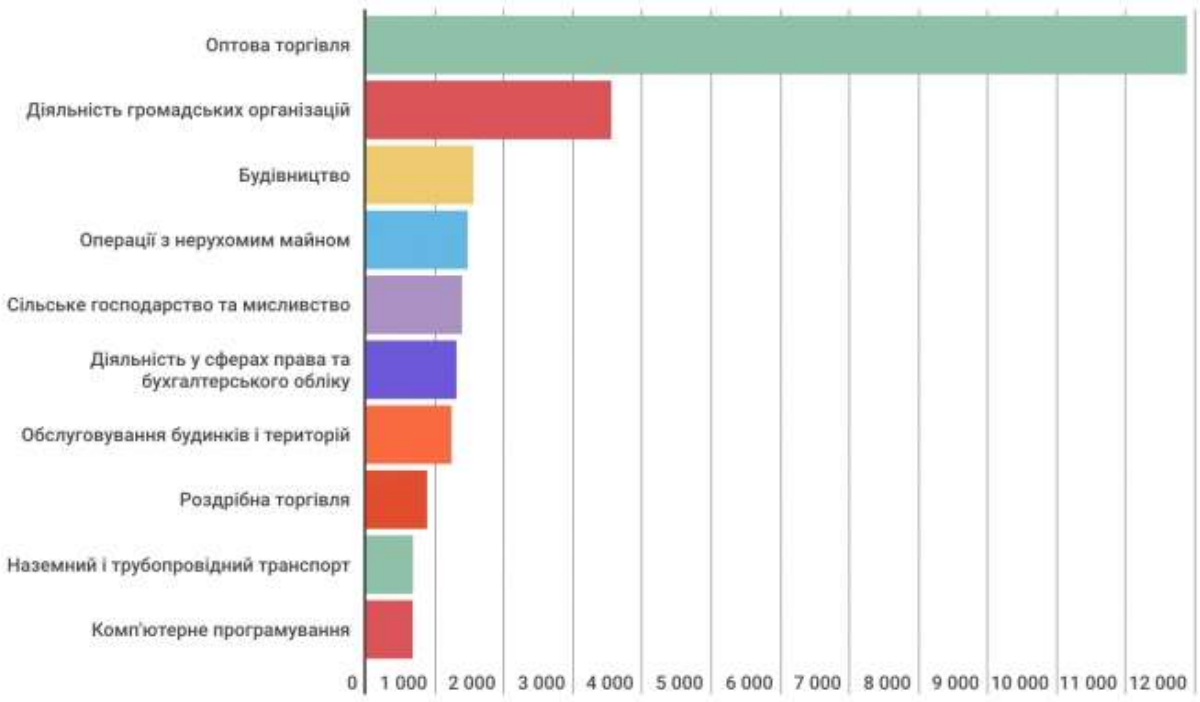



Рисунок 1.2 – Статистика видів діяльності серед юридичних осіб

У Європі спостерігається висока концентрація роздрібних продажів у таких країнах, як Великобританія і Німеччина. У Великобританії понад 80% роздрібних продажів продуктів харчування припадає на 5 великих торгівельних мереж. У Південній Європі та Україні ринок є більш фрагментованим, частіше зустрічаються традиційні невеликі сімейні магазини, які домінують на ринках.

## 1.2 Актуальність задачі по розробці програмного забезпечення для обліку реалізації товарів

Програмне забезпечення для торгівлі допомагає продавцю досягти успіху на висококонкурентному ринку. Перехід від традиційного методу обліку на папірцях до обліку в програмному забезпеченні на серверах є важливим кроком на шляху до того, щоб компанія розвинули свій бізнес.

Розглянемо переваги програмного забезпечення для обліку реалізації товарів:

- Детальна інформація. Дає вам загальне уявлення про рух грошових коштів, скільки продуктів було продано, скільки потрібно замовити, які є заборгованості, залишки товарів.
- Покращене управління запасами. Будь-якій компанії потрібен доступ до докладних звітів про запаси, щоб планувати, які продукти закупляти. Звіт про запаси показує тенденції продажів, історію продажів і товари, які розкуповуються дуже швидко або недостатньо швидко.
- Точність. Програмне забезпечення для торгівлі — це спосіб зменшити людські помилки під час обробки продажів за рахунок автоматизації в програмному забезпеченні.
- Безпека. Програмне забезпечення відстежує точний рівень запасів, тому співробітники будуть більш уважними, знаючи, що запаси відстежуються.
- Задоволеність клієнтів та постачальників. Швидкість має велике значення, коли мова йде про задоволеність клієнтів, а використання програмного забезпечення спрощує процес обліку. Замість того, щоб запитувати інформацію про клієнта чи постачальника щоразу, коли він купує чи постачає товар вашого магазину, його інформацію можна зберегти в довіднику.

Сфера торгівлі в Україні та світі є насиченою, тому конкуренція дуже висока, тому компанії мають докладати максимум зусиль для витримування конкуренції та утримання своїх позицій. Організацію обліку потрібно побудувати таким чином, щоб обліково–аналітична інформація могла бути отримана користувачем за потрібний йому проміжок часу. Для цього потрібне програмне забезпечення, яке дасть можливість створювати звіт по виконаних операціях.

Однією з складних задач для торгівельної компанії є точний та упорядкований облік товарів. При великому обороті товарних документів стає складним їх упорядкування. Багато компаній малого та середнього бізнесу дотепер при такому розвитку комп'ютерних технологій та програмного забезпечення не мають налагодженого обліку. Програмне забезпечення для обліку реалізації товарів повинне вирішувати дані проблеми та автоматизувати систему продажу.

Облік в торгівлі являє собою сукупність статистичного та бухгалтерського обліку [3]. Облік товарів є одним з найбільш трудомістких функцій з великою масовістю і однорідністю вихідних і вхідних даних.

Зі зростанням також електронної комерції, попиту клієнтів, конкуренції та високих показників продуктивності програмне забезпечення для обліку реалізації товарів в торгівлі є найкращим рішенням для максимального використання бізнес-потенціалу зі зниженням витрат та ручної праці. Програмне забезпечення відповідає сучасним технологічним вимогам [4], ефективно обраховуючи операції, організовуючи ключові процеси та покращуючи клієнтський досвід [5-6]. Користувачі можуть керувати такими функціями торгівлі, як інвентаризація, бухгалтерський облік, платежі, аналітика, тощо. Програмне забезпечення дозволяє ефективно керувати магазином за допомогою автоматизації, що економить час.

Програмне забезпечення для обліку реалізації товарів в торгівлі пропонує інструменти та функції, які економлять час та полегшують роботу користувача та покращують керування бізнесом. Його використовують супермаркети, універмаги, торгові точки, магазини, веб-сайти електронної комерції, аптеки, книжкові магазини, тощо. Програмне рішення забезпечує комбінацію функцій, включаючи управління запасами, управління клієнтами, аналіз даних, звіти. Програмні рішення пропонують динамічну платформу управління бізнесом для керування товарами, обробки платежів, обробки рахунків.

Програмне забезпечення пропонує швидкий огляд життєво важливих бізнес-аспектів [7], таких як тенденції продажів, найприбутковіші продукти, щоденні прибутки, тощо. Автоматизує такі операції магазину, як облік, аудит, виставлення рахунків і рахунків, обробку заробітної плати, ціноутворення тощо з мінімальним людським втручанням і отримуйте швидке бачення тенденцій запасів, фінансових операцій [8-9].

Також програмне забезпечення для обліку реалізації товарів в торгівлі дає такі переваги:

- Спрощує операції.
- Точний фінансовий менеджмент.
- Покращує утримання клієнтів.
- Економить час та гроші.
- Збільшує ефективність бізнесу

Програмне забезпечення для обліку реалізації товарів у торгівлі повинне охоплювати всі необхідні функції, які підійдуть для більшості видів товарів, також повинні бути відсутні баги, так як кожен баг чи недопрацювання може бути критичним в сфері торгівлі, інтерфейс програми повинен бути зручним на зрозумілим, а саме програмне забезпечення задіювати мінімум ресурсів комп‘ютера.

Існує безліч програм, які використовуються для автоматизації в сфері торгівлі, деякі компанії використовують просте рішення, таке як таблиці в Microsoft Excel. При використанні Microsoft Excel функціонал є обмеженим, тому найкращим рішенням для компаній буде спеціалізоване програмне забезпечення. Вивчення можливостей програмного забезпечення конкурентів є важливим перед створенням власного, тому у розділі 1.3 було проаналізовано та проведено огляд відомих програмних рішень для обліку реалізації товарів в торгівлі.

## 1.3 Огляд відомих рішень програмного забезпечення для обліку реалізації товарів в торгівлі

Одне з найпоширеніших рішень програмного забезпечення для обліку реалізації товарів в торгівлі був російський 1С. Проте починаючи з 2017 року рішенням Верховної ради була прийнята постанова, яка заборонила використання даного програмного забезпечення, але дана заборона не поширювалась на приватні підприємства, лише на державні. З початку повномасштабного вторгнення росії в Україну, яке розпочалось 24 лютого 2022 року, в українських користувачів «1С» деактивувалась ліцензія на ПЗ. Без оновлень функції ПЗ можуть не правильно працювати, оскільки не будуть виправлятись помилки, які існують в програма. При некоректній роботі програмного забезпечення можуть порушитись внутрішні процеси в компанії, в найгіршому випадку зупинитись повністю. Також може статись витік інформації.

Вивчення конкурентів є важливим кроком перед створенням власного програмного забезпечення даного типу.

Розглянемо найпопулярніші програмні рішення для обліку реалізації товарів та їх можливості.

Спершу розглянемо ISpro – це програмне забезпечення для обліку та управління компаніями та бюджетними організаціями, гнучка система в якій замовник має можливість вибрати потрібні йому модулі та автоматизувати вибрані робочі процеси. Функціонал програми дозволяє [10]:

- Вести бухгалтерський облік та податковий облік.
- Вести облік запасів товарів, автотранспорту, грошей, договорів, персоналу, праці, зарплати та комунальних послуг.
- Керувати закупівлями товару та збутом товару.
- Вести розрахунок з партнерами, планувати, аналізувати гроші компанії.

Ще дане програмне забезпечення дозволяє інтегруватися з ПЗ для подання звітності MEDoc. Ці обидва програмні продукти розробила українська компанія під назвою «Інтелект-Сервіс». ISpro орієнтований в основному на бюджетний сектор.

Наступним розглянемо «А5» – це програмний комплекс, який дозволяє вести бухгалтерський облік, а також облік зарплат та персоналу компаній. Програмне забезпечення розроблено українською компанією «А5 Systems».

Програмне забезпечення від «А5» складається з трьох продуктів [11]:

1. «А5: Бухгалтерія» – автоматизує бухгалтерський облік з адаптацією під українське законодавство.

2. «А5: Зарплата» – автоматизує розрахунок зарплати, виплат коштів та проведення звітності згідно з календарем.

3. «А5: Персонал» – охоплює цикл процесів компанії в управлінні персоналом. А також забезпечує кадровий документообіг у електронному форматі.

Також до особливостей цього програмного забезпечення належить те, що програма не встановлюється локально на пристрої, а дає доступ до серверів розробника. Перевагою є тим, що за допомогою цього можна отримати доступ до інформації з будь-якого пристрою в будь-який час. Це програмне забезпечення підходить переважно для бюджетного сектора. Також у ньому можна використовувати вбудований сервіс електронного документообігу державних структурах Megapolis.DocNet.

Ще одне відоме рішення ПЗ «Dilovod». Це український сервіс, де можна вести облік товарів та подавати звіти. Дане програмне забезпечення можуть використовувати ФОП з найманими працівниками, також інтернет-магазини, виробництва, тощо. Програмне забезпечення дозволяє вести такі види обліків [12]:

- Облік замовлень.
- Облік продажів.

- Облік закупівель.
- Облік витрат виробництва.

Також в ньому присутній програмний РРО, CRM-система, є можливість друкувати чеки та інші документи та вести кілька бізнесів різного типу в одній єдиній базі даних.

Далі розглянемо програмне забезпечення «MASTER: Бухгалтерія» Програма дозволяє вести та автоматизувати в торгівлі різні види обліків. Такі як [13]:

- Облік грошових коштів.
- Облік товару.
- Облік спецодягу.
- Облік витрат.
- Облік розрахунків з компаніями.
- Облік кредиторської заборгованості та дебіторської заборгованості.
- Облік податків.

Також є можливість підключити модуль «MASTER: Зарплата та Кадри». Це дозволить вести облік кадрів компанії, розрахунок заробітної плати, тощо в одній інформаційній базі.

«MASTER: Бухгалтерія» підійде як для державних установ, так і для приватних компаній різного типу. В програмі доступні модулі для обліку виробництва, автотранспорту, продуктів харчування, ліків, агропродуктів, також є підтримка повноцінного аналітичного обліку. В програмному забезпеченні також можна використовувати вбудований сервіс електронного документообігу Signy. З недоліків є неможливість використовувати сторонні системи.

Наступним для розгляду буде ПЗ «Дебет Плюс» – це українське програмне забезпечення, яке автоматизує бухгалтерський облік та фінансову звітність підприємств. Програмне забезпечення можна використовувати у

бюджетних організаціях, комунальних компаніях, закладах охорони здоров'я, відділах освіти, сільському господарстві, виробництві, торгівлі, будівництві, тощо.

Серед можливостей «Дебет Плюс» можна назвати [14]:

- Можливість використання на комп'ютерах із різними операційними системами.
- Підходить компаніям різних типів.
- Дозволяє автоматично створювати в електронному форматі звіти щодо ЄСВ та 1ДФ для програм MEDoc.

Часто програму використовують у закладах охорони здоров'я. Так як в ній дуже зручно вести облік товарів, медичних матеріалів, продуктів, автотранспорту. Також програмі можна автоматично нараховувати і виплачувати заробітню плату.

Останнім для огляду відомих рішень буде «HugeProfit» – це CRM-система для товарного бізнесу, яка включає в себе облік продажу, товару, створення та відстеження ТТН, робота з клієнтською базою.

В сервісі HugeProfit є можливість [15]:

- Проводити облік товарів на складах, а саме: кількість, ціна закупівлі, ціна продажу.
- Оформлювати постачання товарів в різних валютах.
- Зафіксовувати продажі у два кліки, як вручну, так і більш зручніше за допомогою сканера штрих-кодів.
- Друкувати фіскальні чеки, видаткові накладні.
- Сформувати список замовлень на відправку.
- Проводити відстежування відправок, кількості грошей у платежах, забрали чи відмовилися від поштової посилки.
- Добавляти витрати та додаткові прибутки, планувати витрати.
- Вести клієнтську базу з їх адресами та можливістю створювати програми лояльності.

HugeProfit надає робочій простір для продавців товарів, де підприємець має змогу контролювати залишки товарів на складі, добавляти та обробляти продажі, вести базу покупців та контролювати витрати.

### 1.4 Висновок до першого розділу

В першому розділі кваліфікаційної роботи описано та проаналізовано ринок торгівлі: як даний ринок працює, які бувають типи магазинів в торгівлі, перспектива даного ринку та сфери в цілому, поширеність у світі та Україні. Сфера торгівлі в Україні є однією з найбільш розвинених та популярних, конкуренція в цій сфері є дуже високою.

Сформульовано актуальність задачі по розробці програмного забезпечення для обліку реалізації товарів в торгівлі. Описано для чого потрібне програмне забезпечення такого типу та яку користь приносить для торгових точок. Серед переваг такого програмного забезпечення є те, що воно спрощує операції, надає точний фінансовий менеджмент, покращує утримання клієнтів, економить час та гроші, збільшує ефективність бізнесу, дозволяє вносити і отримувати детальна інформацію по бізнесу, створює покращене управління запасами, підвищує точність, покращує безпеку та підносить задоволеність клієнтів та постачальників.

Також наведено огляд відомих рішень програмного забезпечення даного функціонального призначення. Описано функціональні можливості кількох найпоширеніших програмних рішень, їх можливості, переваги та недоліки.

# 2 ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ПО РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ РЕАЛІЗАЦІЇ ТОВАРІВ В ТОРГІВЛІ

## 2.1 Вибір бази даних для реалізації програмного забезпечення

База даних – це організований набір логічно пов’язаних даних. Інформація перетворюється на корисні знання, структуровані та підтримувані відповідно до потреб користувача. Окрім зберігання самих даних, база даних також підтримує зв’язки між точками даних.

Бази даних є основою усіх сучасних інформаційних систем. Комп’ютери чи сервери зберігають безліч баз даних, дані можуть бути будь-якого розміру та будь-якої складності. Існує багато способів збору і впорядкування даних залежно від використання та типу даних.

А у ширшому розумінні база даних є  інтегрованим набором інформації про систему та процедури обслуговування та використання [16]. На відміну від електронних таблиць, кілька користувачів програми має змогу отримувати доступ до сховища одночасно.

Бази даних мають досить широкий спектр застосування. Для прикладу:

- Банківські системи зберігають бази даних про своїх клієнтів, банківські рахунки, кредити клієнтів, інформацію про транзакції, тощо. Операції з кредитними картками містять вкладки щодо платежів і генерують щомісячні звіти.
- Авіакомпанія зберігає інформацію про рейси, бронювання клієнтами квитків, тощо. Авіакомпанії були першим сектором, який використовує географічно розподілені бази даних.
- Університети використовують бази даних для запису даних про студентів, оцінки, курси, тощо.

- Телекомунікаційні компанії зберігають інформацію про дзвінки, формують щомісячні платіжки, відстежують лінії зв'язку, тощо.
- Фінансовий сектор відстежує продажі та покупки фінансових інструментів, таких як облігації чи акції.
- Підприємства торгівлі та електронної комерції зберігають дані про споживачів, товари.
- Виробничі підприємства управляють ланцюгами поставок, виробничими лініями, складами.
- Служба кадрів зберігає інформацію про працівників, їх зарплату, податки.

Розглянемо типи баз даних, щоб зрозуміти, яка найкраще підходить для виконання поставленої задачі створення ПЗ для обліку реалізації товарів в торгівлі на основі клієнт серверної архітектури.

Реляційні бази даних зберігають дані в табличних структурах у вигляді рядків і стовпців [17]. Цей тип баз даних зосереджений на зв’язках між цими даними, є найбільш широко використовуваним типом бази даних.

Об’єктно-орієнтовані бази даних поєднують принципи об’єктно-орієнтованого програмування із стандартами реляційних баз даних [18].

Розподілені бази даних поширюються на кілька сайтів та масштабуються.

Сховища даних послідовно інтегрують дані з різних джерел в єдину систему. Сховища обслуговують великі обсяги даних і зазвичай знаходяться на серверах з великою кількістю даних.

Бази даних OLTP зосереджені на коротких повсякденних транзакціях, підтримують велику базу даних із високою цілісністю даних і ефективністю одночасних запитів [19].

Самокеровані хмарні бази даних, їх ще називають автономні бази даних, використовують машинне навчання для автоматизації різних завдань у системах управління базами даних (СУБД).

Бази даних з відкритим вихідним кодом відкриті для змін і безкоштовні для використання. Можливість налаштування параметрів користувача та низька вартість роблять цей тип бази даних широко поширеним.

Хмарні бази даних мають усі звичні функції баз даних із гнучкістю хмарних обчислень.

Бази даних NoSQL структурно різноманітний тип баз даних з акцентом на високу доступність. Система NoSQL найкраще підходять для великих обсягів неструктурованих даних.

Графічні бази даних, тип бази даних NoSQL, яка зосереджена на зв'язках між певними точками даних. З топографічною мережевою структурою бази даних графів є найкращою системою для дослідження та виявлення зв'язків.

Багатомодельні бази даних забезпечують єдиний механізм для роботи з декількома моделями баз даних.

База даних Document/JSON – це система зберігання NoSQL, яка зберігає дані в документах JSON.

Бази даних складаються з таким компонентів:

- Дані. Це елементи, які зберігає база даних. Всі отримані дані потребують обробки, щоб набути значення та стати конкретною інформацією. Крім того, обробка витягує з даних важливі деталі та допомагає в процедурах.
- Процедури. Процедури бази даних включають всі функції, що працюють у сховищі. Наприклад, резервне копіювання, генерування звітів та інші операції, процедури є набором інструкцій, які виконуються в системі керування базами даних.
- Обладнання. Апаратне забезпечення охоплює фізичні пристрої, які з'єднують комп'ютери з клієнтами. Що стосується баз даних, серверів, дисків для зберігання та різних пристрої зберігання даних включають апаратне забезпечення, необхідне для запуску та заповнень бази даних.

- Програмне забезпечення. Програмне забезпечення включає набір програм, які використовуються для доступу, управління та контролю баз даних. На нижчих рівнях програмне забезпечення включає в себе операційні системи, на яких знаходяться бази даних, мережа для зв’язку з базами даних і програмне забезпечення для доступу до даних.
- Мова доступу до бази даних. Мова доступу до бази даних – це мова програмування, що використовується для вставлення, оновлення, видалення та зміни даних, що зберігаються в базі даних. Бази даних виконують запити безпосередньо мовою доступу до бази даних.

Система управління базами даних служить певним інтерфейсом між користувачем та базою даних. Програма забезпечує зв’язок із базою даних, дозволяючи операції з даними, таких як пошук, оновлення, оптимізацію та загальне керування інформацією, яка зберігається в базі даних [20].

Сервер бази даних – це виділений сервер, який надає послуги клієнту через програми баз даних. Одна частина сервера зберігає СУБД, а інша зберігає саму ж базу даних. Часто сервери баз даних мають великий обсяг пам’яті та велику кількість пам’яті.

Після вивчення основ бази даних, перейдемо до розгляду архітектур баз даних. Розпочнемо з архітектури «Файл-сервер» – це архітектура системи бази даних із мережевим доступом передбачає виділення однієї з машин у мережі в якості сервера. На ній відбувається зберігання спільно використовуваної централізованої бази даних. Усі інші машини у мережі виконують функцію робочих станцій за допомогою яких підтримується доступ системи до централізованої бази даних [21]. Файли бази даних у відповідності до призначених для користувача запитів передаються на робочі станції, де зазвичай проводиться обробка. При високій інтенсивності доступу до одних і тих же даних продуктивність інформаційної системи падає. Для виконання поставлених завдань даний тип архітектури не підходить, так як реалізація товару в торгівлі передбачає високу інтенсивність.

Далі розглянемо архітектуру «Клієнт-сервер» – ця архітектура побудована так, що крім зберігання централізованої бази даних сервер бази даних забезпечує виконання основного обсягу обробки даних [22]. Запит даних, який виконується клієнтом, запускає пошук і отримання даних на сервері. Отримані дані передаються по мережі від сервера до клієнта. Специфікою архітектури клієнт-сервер є використання мови запитів SQL.

Отже, провівши аналіз типів архітектур, найкращим рішенням буде клієнт-серверна архітектура, це буде перевагою для типу програмного забезпечення створюваного в кваліфікаційні роботі, детальніше про цей тип архітектури описано у розділі 2.2. Також для кваліфікаційної роботи було обрано мову запитів SQL. Розглянемо її детальніше.

SQL – це широко використовувана мова програмування, яка призначена для взаємодії з базами даних. Розробники та дослідники даних часто використовують SQL у своїй роботі. SQL функціонує як метод для отримання даних з бази даних [23], ця мова може взаємодіяти з кількома програмами та системами. SQL є популярною мовою для компаній, у тому числі для бізнес-адміністрування, оскільки вона забезпечує простий метод доступу та маніпулювання збереженими даними.

Є кілька різних переваг використання SQL на основі ваших конкретних цілей програмування або розробки [24]. Деякі з цих переваг можуть включати:

- Спільність. Однією із головних переваг використання SQL є спільність цієї мови. Це є перевагою в IT-системах, так як є можливість використовувати її з кількома іншими мовами. Спільність SQL також сприяє простоті застосування, що сприяє продуктивності та ефективності бізнесу.
- Простота. Ще однією перевагою використання SQL є простота мови. Команди SQL – це звичайні фрази англійською мовою, які дають можливість програмістам краще зрозуміти, що вони вимагають від мови.

- Інтеграція. SQL також корисний, так як його можна легко інтегрувати з іншими мовами програмування. Він добре працює зі всіма популярними мовами програмування. Використовуючи функцію інтеграції, ви можете легше маніпулювати даними та керувати базою даних, оскільки ви використовуєте ту саму мову кодування в усій системі. Аналітики даних, інженери або спеціалісти з розробки можуть використовувати цю функцію найбільше.
- Швидкість. SQL має здатність працювати з високою робочою швидкістю. Така висока швидкість може збільшити кількість даних, яку виконує. Також надає користувачам швидкий і ефективний спосіб отримувати, управляти або зберігати дані.

Для реалізації проекту обрано Microsoft SQL Server, так як він ідеально підходить для зберігання всієї потрібної інформації в реляційних базах даних, а також для керування такими даними завдяки продуманому візуальному інтерфейсу та наявним у нього параметрам та інструментам.

Для створення програмного забезпечення використання цього інструменту є важливим через можливості, які він пропонує, та утиліти, які він має у своєму розпорядженні. Якщо у вас є список клієнтів, каталог продуктів або навіть великий вибір мультимедійного вмісту, Microsoft SQL Server допомагає керувати абсолютно усім. Це важливо для належного функціонування програмного забезпечення.

Основним компонентом Microsoft SQL є наявність реляційного механізму, який відповідає за обробку команд, запитів, а також зберігання файлів, bb.dd., таблиць та буферів даних. Його вторинні рівні призначені для керування пам’яттю, програмування та адміністрування взаємодії запитів і відповідей із серверами, на яких розміщено бази даних.

Вивчимо функції та особливості Microsoft SQL Server. Деякі з основних функцій, які відрізняють Microsoft SQL Server поміж інших СУБД є різноманітні інструменти, спрямовані на керування та аналіз даних [25], а

також бізнес-аналітика, за допомогою якої можна отримати інформацію про свій бізнес і клієнтів за допомогою машинного навчання. Завдяки інтеграції з Azure AI, наприклад.

Microsoft SQL Server дозволяє легко інтегрувати дані в додатки та скористатися перевагами широкого набору когнітивних служб для використання штучного інтелекту у будь-якому масштабі даних, як в локальних, так і у хмарних середовищах.

Сервери SQL пропонують користувачеві високу доступність, щоб забезпечити швидші процеси перемикання. Його можливості в пам’яті забезпечують підвищену гнучкість і простоту використання, забезпечуючи повну інтеграцію із сімейством серверів Microsoft Server.

Базуючись на відкритому коді, доступ до нього дуже простий, і переважна більшість програмістів, які працюють у розробці, використовували Microsoft SQL Server у своїх проектах, а також дана СУБД є дуже поширена і має велику спільноту, яка пропонує підтримку іншим користувачам.

Підводячи підсумок, можна виділити такі особливості та переваги Microsoft SQL Server:

1. Інтелектуальна інформація про всі ваші дані за допомогою кластерів великих даних, можливість надсилати запити щодо усіх ваших даних із SQL Server.
2. Великий вибір мови та платформи: Windows, Linux.
3. Можливості інтелектуальної бази даних: підтримка постійної пам’яті, оптимізований tempbd у пам’яті.
4. Шифрування даних: система захисту даних, моніторингу та класифікації зробила її однією з найбезпечніших платформ за даними Національного інституту Бази стандартів і технологій за останні 10 років.

5. Масштабованість: дозволяє легко інтегрувати системи керування базами даних із будь-яким пристроєм і службами для кращої продуктивності та аналітичних можливостей щодо даних.

6. Підвищення безпеки даних: забезпечує безпеку бази даних, особливо за допомогою служби адміністратора бази даних MS SQL Server.

7. Простота налаштування: на відміну від інших систем керування базами даних встановлення та налаштування є простіші.

8. Оптимізоване зберігання даних: не потрібне інше зберігання даних для тієї самої бази даних при використанні іншого пристрою. Це також дозволяє легко виявляти несправності та підтримувати дані.

9. Підтримка відновлення даних: у разі припинення живлення або завершення роботи сервера дані можуть бути пошкоджені, тому Microsoft SQL Server усуває ризик втрати даних завдяки функції відновлення даних.

Тепер, проаналізувавши Microsoft SQL Server і для чого він використовується, ви знаємо переваги, які роблять його дуже цікавим для розробки програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі. Найбільш зрозумілим є те, що дана СУБД працює з реляційними базами даних, тобто використовує кілька взаємопов’язаних таблиць для зберігання інформації та правильної організації даних.

## 2.2 Клієнт-серверна архітектура та її переваги

Клієнт-серверна архітектура – це архітектура додатків, що розділяє завдання між клієнтами та серверами [26], які знаходяться в одній системі або ж передають дані через мережу інтернет. Щоб отримати доступ до послуги, що надається сервером, клієнт надсилає запит, який запускає відповідну процедуру та повертає відповідь.

В цій архітектурі сервер діє як постачальник, а клієнт — як споживач. Сервер надає послуги, що потребують обчислення клієнту за вимогою. Ці послуги можуть включати доступ до програм, даних.

Щоб краще зрозуміти архітектуру клієнт-сервер спочатку розглянемо, що таке клієнт і сервер.

Клієнт – клієнтом може бути будь-який комп'ютер, який запитує щось у сервера [27]. Наприклад, надати йому дані, які товари є на складі.

Сервер – це комп'ютер, призначений для обслуговування запитів клієнта [28]. Для прикладу з клієнтом, сервер обробляє запит та видає всі дані, які знайшов на сервері, клієнт побачить у програмі, які товари є на складі.

Архітектура клієнт-сервер класифікується на різні типи на основі логіки [29-30], яка реалізована для запиту, що обробляється між клієнтом та сервером.

У однорівневій клієнт-серверній архітектурі все, що стосується програми, групується і використовується як єдиний пакет для доставки даних [31]. Уся програмна логіка, пов'язана з інтерфейсом користувача, логікою бази даних, базою даних, згрупована в одну сутність.

Однорівнева архітектура пропонує різні служби, які роблять її надійною, проте інколи можуть виникати проблеми з керування нею. Розбіжність даних є основною проблемою в однорівневій клієнт-серверній архітектурі. Однорівнева архітектура включає в себе кілька рівнів, включаючи презентаційний рівень, бізнес-рівень і рівень даних, які поєднуються за допомогою програмного пакета [32-33]. Цей рівень, як правило, зберігає дані в локальних системах або на спільному диску.

У двохрівневій клієнт-серверній архітектурі вся логіка програми розділена на два рівні. База даних діє як окрема сутність в цій архітектурі. Бази даних розробляються окремо, а програма має всю логіку, пов'язану з інтерфейсом користувача [34-35], а також бізнес-логіку та логіку бази даних

для зв’язку із базою даних. Ця архітектура має краще середовище, ніж однорівнева архітектура.

У порівнянні із однорівневими архітектурами дворівневі архітектури є більш швидшими, так як між клієнтом та сервером немає посередника. Двохрівнева архітектура допомагає уникнути плутанини всередині клієнта [36]. Найкращим прикладом для двохрівневої клієнт-серверної архітектури є сервіс онлайн-бронювання.

Ще один тип це трьохрівнева клієнт-серверна архітектура. На відміну від двохрівневої архітектури, яка не включає проміжного програмного забезпечення [37-38], трьохрівнева система має проміжне програмне забезпечення між клієнтом та сервером. Коли клієнт робить запит інформації від сервера, запит спочатку передається в проміжне програмне забезпечення. Аж тоді запит надсилається на сервер для обробки. В свою чергу так само сервер надішле відповідь клієнту.

У трьохрівневій архітектурі є три рівні, а саме: презентаційний рівень, прикладний рівень і рівень бази даних. Кожен із рівнів контролюють один одного [39]. Клієнтські пристрої контролюють рівень презентації, тоді як проміжне програмне забезпечення і сервер контролюють прикладний рівень і рівень бази даних відповідно. Наявність третього рівня, який забезпечує контроль даних, робить трьохрівневу архітектуру більш безпечною і забезпечує цілісність даних.

Розглянемо компоненти клієнт-серверної архітектури. Є три компоненти: робочі станції, сервери та мережеві пристрої. Розглянемо їх докладніше.

Робочі станції, також їх називають клієнтські комп’ютери, вони є залежними від серверів та надсилають їм запити. Частина серверного програмного забезпечення та комп’ютерного обладнання запитує доступ до баз даних та спільних сайтів. Клієнтські обчислення класифікуються як:

- Товстий клієнт – товстий клієнт надає велику функціональність, оскільки він найменше залежить від сервера і здатний виконувати більшість завдань обробки даних самостійно.
- Тонкий клієнт – це так званий «легкий комп'ютер», який значною мірою залежить від ресурсів сервера. Сервер програмного забезпечення виконує всі завдання обробки даних.
- Гібридний клієнт – поєднання характеристик тонкого і товстого клієнта. Гібридний клієнт покладається в основному на сервер для всіх централізованих даних, але здатний до обробки локально.

Сервери – це пристрої обробки, які діють як централізоване сховище програмного забезпечення, баз даних чи мережевих файлів [40]. Вони мають величезний простір для зберігання і надійні системи для обробки запитів клієнтів щодо використання ресурсів і розподілу роботи. Сервери можуть виконувати кілька ролей:

- Сервер додатків – сервери розміщують веб-додатки в мережі без потреби у копії.
- Сервер бази даних – підтримує і надає доступ до бази даних для комп'ютерних програм, щоб завантажувати дані у разі потреби, для прикладу, електронні таблиці або бухгалтерське програмне забезпечення.
- Обчислювальний сервер – сервер, який спільно використовує величезну кількість комп'ютерних ресурсів для під'єднаних комп'ютерів, яким для ПК потрібно більше оперативної пам'яті або потужностей процесора.
- Веб-сервер – сервери, на яких розміщені веб-сторінки та засоби, доступні у мережі інтернет.

Мережеві пристрої. Середовище, що з'єднує клієнта і сервери в архітектурі клієнт-сервер, називають мережевим пристроєм [41]. Крім того, вони використовуються для виконання інших операцій у мережі. Для прикладу, ретранслятор використовується для ефективної передачі даних

поміж двома мостам, хаб з'єднує сервер з кількома робочими станціями, мости допомагають ізолювати сегментацію мережі.

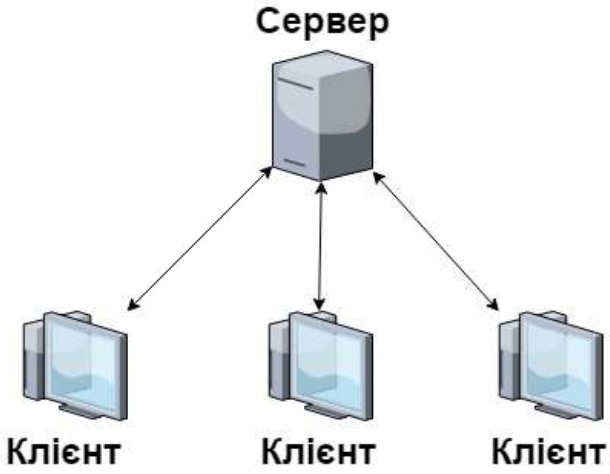



Рисунок 2.1 – Принцип роботи клієнт-серверної архітектури

Отож, підсумуємо переваги клієнт-серверної архітектурі в розробці програмного забезпечення:

- Легко керувати – керувати файлами досить легко, так як вони зберігаються на одному сервері. Мережі клієнт-сервер мають найкраще керування для відстеження та пошуку записів необхідних файлів, що дуже суттєвою перевагою для створення програмного забезпечення для обліку реалізації товарів в торгівлі.
- Легкий доступ – клієнти можуть входити в систему незалежно від свого місцезнаходження чи вибраної платформи, що дозволяє кожному співробітнику отримувати доступ до своєї корпоративної інформації без використання режиму терміналу.
- Масштабованість серверів – клієнт-серверна архітектура має високу масштабованість. Кожного разу, коли користувач потребує більше потужностей, є можливість додати більше ресурсів, таких як клієнти чи сервери, таким чином збільшуючи розмір сервера. Через те, що сервер є централізованим, дозвіл на доступ до ресурсів не є проблемою.

- Централізоване керування – архітектура клієнт-сервер має перевагу централізованого керування, оскільки вся інформація зберігається в одному місці. Адміністратор мережі має повний контроль над керуванням і адмініструванням. У результаті проблеми, які виникають у всій мережі, можна вирішувати з одного місця. У результаті оновлювати дані та ресурси стає набагато легше.
- Безпека – завдяки централізованій архітектурі клієнт-сервер дані добре захищенні. Одну резервну копію можна використати для відновлення усіх файлів у разі втрати даних, для прикладу, нав’язування облікових даних, таких як ім’я користувача та пароль.

З недоліків клієнт-серверної архітектури можна виділити те, що серверам потрібне регулярне технічне обслуговування так як зазвичай вони працюватимуть безперервно. Їх потрібно належним чином обслуговувати. При виникненні проблем, їх необхідно негайно усувати. Для цього може знадобитись менеджер мережі для обслуговування сервера. Вимагає витрат – у мережі клієнт-сервер вартість налаштування та обслуговування сервера, як правило, висока. Оскільки сервера мають бути потужні, їх придбання може бути дорогим. Таким чином, не всі користувачі зможуть ними скористатися. Перевантаженість мережі, якщо занадто багато клієнтів звертаються до одного сервера, це може призвести до уповільнення або збою з’єднання [42]. Перевантажений сервер створює багато проблем при доступі до інформації.

Клієнт-серверна архітектура дозволяє оновлювати спільну базу даних від кількох користувачів через графічний інтерфейс користувача. Усі компанії, великі чи малі, використовують переваги сервера, щоб розвивати та оцифровувати свій бізнес, продавати свою продукцію та отримувати інформацію про новини та події у своїх сферах діяльності. Згідно проведених досліджень було обрано клієнт-серверну архітектуру з використанням тонкого клієнту.

## 2.3 Технологія доступу до даних ADO.NET

Для взаємодії з сховищем даних оберемо технологію ADO.NET. Вона є набором певних бібліотек, які постачаються з Microsoft .NET Framework та призначені для взаємодії із різними сховищами даних із застосунками на базі .NET. ADO.NET є частиною фреймоворка .NET, який надає доступ до різних типів джерел даних. ADO.NET включає підтримку реляційнійних даних, даних XML та даних додатків. ADO.NET застосовується на рівні між джерелами даних та клієнтськими програмами, а також дозволяє клієнтам отримувати, оновлювати та керувати даними через об’єктну модель ADO.NET.

ADO.NET представили у десятій версії фрейморка .NET, архітектура була представлена як метод вирішення двох способів обробки доступу до даних та керування. Один з способів заключається в тому, щоб отримати доступ до даних та повторити збір даних за один раз, де з’єднання із джерелом даних залишається відкритим. Іншим же способом є доступ до даних у відключеному режимі, так як це дозволяє керувати даними, отримати доступ до них без відкритого підключення до джерела цих даних. ADO.NET використовує багаторівневу архітектуру, яка побудована навколо трьох основних концепцій:

1. Connection.
2. Command.
3. DataSet.

Технологія ADO.NET використовує класи для роботи з даними. Архітектура включає такі класи, як: Command, Connection, Parameter, DataReader, Transaction, ParameterCollection. Дані класи надають доступ для читання джерела даних та дають можливість виконувати команди з даними. Також є такі класи як: DataTable, DataColumn, DataSet, DataRow, DataRelationship і DataView, Constraint. Дані класи дозволяють отримувати

дані із джерела даних, працювати із даними у пам’яті та оновлювати базу даних новими даними. ADO.NET представляє підключений клас під назвою DataAdapter, який працює як місток між джерелом даних та представленням даних, схема роботи архітектури зображена на рисунку 2.2.

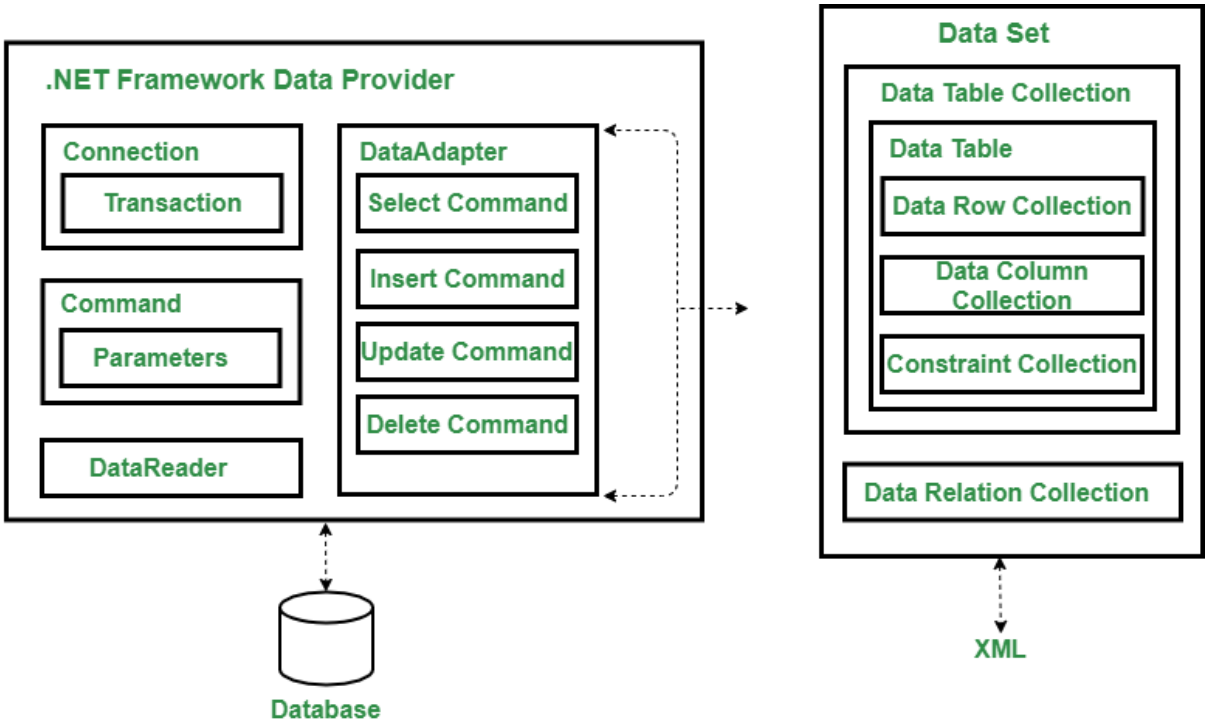



Рисунок 2.2 – Схема роботи технології ADO.NET

Розглянемо основні класи технології ADO.NET, які потрібні для виконання поставленої задачі з розробки програмного забезпечення для обліку реалізації товарів в торгівлі.

DataReader є підключеним класом, тому для його використання потрібне відкрите підключення до бази даних. Поки з’єднання DataReader з базою даних відкрито, підключення використовується виключно цим об’єктом DataReader, важливо після виконання закрити зчитувач та закрити з’єднання.

DataTable є представлення даних у пам’яті та схоже на таблиці у SQL. Це центральний об’єкт використовуваний у ADO.NET, який можливо використовувати як окремо так і з іншим об’єктом, для прикладу, це може

бути DataSet або DataView. Як місток від бази даних до об’єкту DataTable використовується об’єкт DataAdapter. Для того, щоб заповнити об’єкт DataTable даними із бази даних, використовується метод Fill із об’єкта DataAdapter. Для того, щоб оновити базу даних та внести, які зберігаються в об’єкті DataTable потрібно викликати метод Update об’єкта DataAdapter.

DataAdapter використовується для отримання даних із сховища даних та для заповнення таблиць в DataSet. Data Adapter використовує об’єкт Connection від постачальника даних фреймворку .NET для підключення до сховища даних, а також використовує об’єкт Command для отримання даних з сховища даних та внесення усіх змін.

DataSet входить у колекцію DataTable. Він заповнюється схожим чином, як і DataTable. DataSet може містити у собі кілька таблиць, для того щоб заповнити кілька таблиць одночасно, необхідно виконати кілька запитів Select одночас.

Всі функції, які були описані вище у технології ADO.NET, ідеально підходять для роботи із потоком даних, які використовуються у торгівлі. Підсумуємо переваги технології ADO.NET, через яку було вибрано саме її. Багаторівнева архітектура ADO.NET дозволяє створювати логіку додатків на окремому рівні, це дозволяє зменшити ризики та спрощує додавання нових функцій. Даними можна обмінюватись від і до сховища даних, а також архітектура надає внутрішні представлення даних в форматі XML. Дані є із строгою типізацією та підтримкою IntelliSense, підтримка Microsoft Visual Studio спрощує написання функцій. Архітектура надає висока масштабованість програми, яка дозволяє завдяки відключеному режиму працювати з даними не зберігаючи відкритого з’єднання. Це дозволяє зберігати ресурси, а також надає можливість кільком користувачам отримувати доступ до даних одночасно. Підключені і відключені класи оптимізовані для підвищення продуктивності обміну, а правильне використання даних класів дозволяє підтримувати продуктивність програми

на високому рівні. Також дана архітектура є простою та легкою у використанні та відкритою для оптимізації.

## 2.4 Вибір мови програмування та середовища розробки програмного забезпечення

Для створення об’єктно-орієнтованих програм на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі найкращим рішенням буде використати середовище розробки Microsoft Visual Studio (рис. 2.3). Розглянемо це середовище та його переваги.

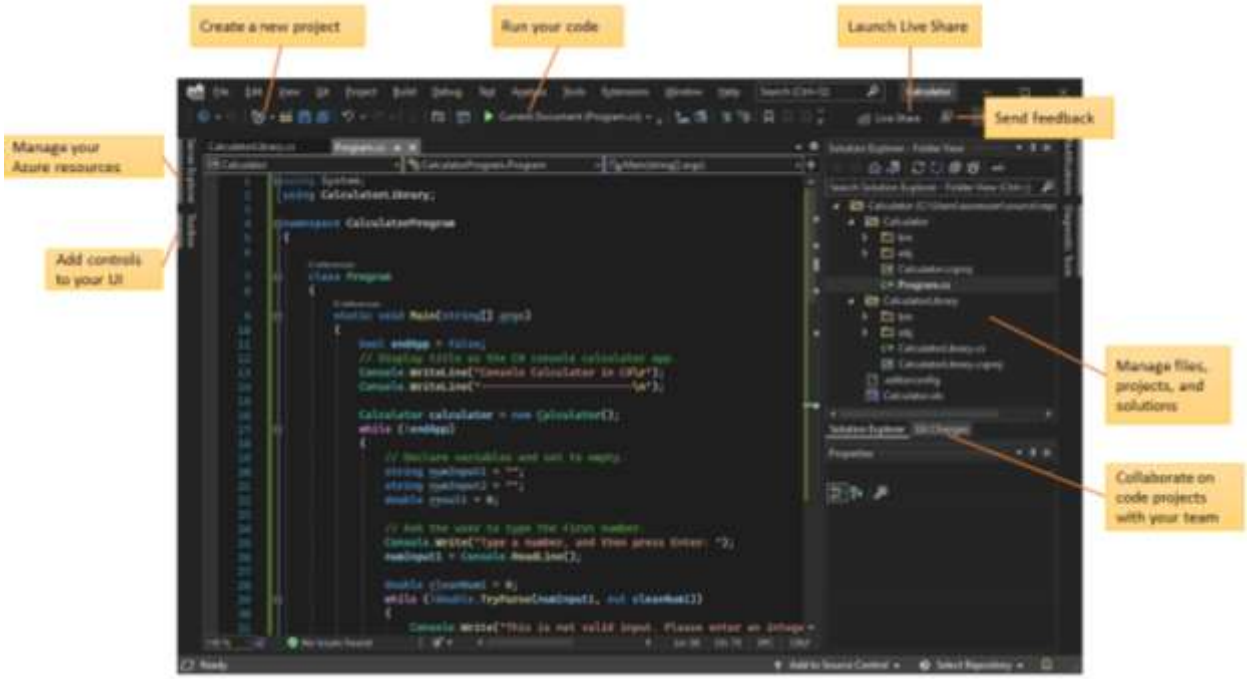



Рисунок 2.3 – Середовище Microsoft Visual Studio

Інтегроване середовище розробки – це багатофункціональна програма, яка має підтримку багатьох аспектів розробки програмного забезпечення [43]. Visual Studio IDE – творчий майданчик, який можна використовувати для редагування, налагодження та створення коду [44], а пізніше і для публікації програми. Окрім стандартного редактора та налагоджувача, які пропонують більшість IDE, Visual Studio містить

компілятори, інструменти завершення коду, графічні дизайнери та багато інакших функцій для покращення процесу розробки потрібного програмного забезпечення [45].

Microsoft Visual Studio є інтегрованим середовищем розробки [46]. По суті це програма, яка дозволяє розробляти, писати чи редагувати код.

Visual Studio робить більше, ніж більшість аналогів, дозволяючи використовувати майже будь-яку мову програмування, зокрема:

- C, C++, C#;
- .NET;
- JavaScript;
- TypeScript;
- XML;
- XSLT;
- HTML;
- CSS.

Такі мови як Python, Ruby, Node.JS доступні через додаткові плагіни.

Наведений список показує, що у вашому розпорядженні є широкий вибір варіантів розробки. Microsoft Visual Studio є комплексним для потреб у розробці програмного забезпечення, так як працює на чотирьох рівнях: розробка, налагодження, тестування та співпраця.

Щоб допомогти розробникам швидко визначити помилки в коді, Visual Studio пропонує надійний інструмент налагодження коду. Розробники можуть із впевненістю розмістити програму на сервері, оскільки програмне забезпечення показує чи усунули все, що могло призвести до проблем із продуктивністю. Незалежно від того яку мову програмування використовують розробники, користувачі Visual Studio має змогу отримати підтримку програмування. Для швидшої розробки середовище розробки пропонує опцію автозаповнення [47]. Вбудована інтелектуальна система пропонує описи та поради для API [48].

За допомогою Visual Studio IDE можна співпрацювати із колегами у одному проекті. Ця IDE допомагає розробникам ділитися кодом та проектами із своєю командою розробників. Кожен користувач Visual Studio має можливість детального налаштування. Середовище надає можливість додавати функції відповідно до своїх потреб [49]. Для прикладу, вони можуть завантажувати доповнення та встановлювати розширення у своїй IDE.

Підтримка різних мов програмування у Visual Studio додається за допомогою спеціального VSPackage, відомого також як Language Service [50]. При інсталяції Visual Studio, функціональні можливості, закодовані як VSPackage, будуть доступні як служба. Microsoft Visual Studio також надає три різні типи служб:

- Служба SVsSolution використовується для надання функціональних можливостей для перерахування проектів у Visual Studio.
- Сервіс SVsUIShell використовується для надання функцій інтерфейсу користувача, серед яких панелі інструментів, вкладки, тощо.
- Служба SvsShell використовується для реєстрації VSPackages.

Середовище розробки Microsoft Visual Studio є найкращим серед аналогів, та найбільш підходящим для створення програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі. До його переваг можна віднести:

- Краще кодування. Visual Studio IDE надає користувачам допомогу в кодуванні у реальному часі незалежно від мови програмування, яку вони використовують. Вбудований у платформу IntelliSense пропонує підказки та описи API, а також може автоматично заповнювати рядки для більшої зручності. Окрім того, Visual Studio IDE гарантує, що розробник не втратить місце останнього перебування, досліджуючи решту коду.
- Швидке налагодження. Пошук чи діагностика помилок інколи може бути складним завданням, але Visual Studio IDE має безліч інструментів, які полегшують це. Платформа підтримує налагодження для

усіх підтримуваних мов. Процес також можна виконувати локально, віддалено чи в середині середовища. Це дозволяє розробнику розгортати програми на робочому столі або емулятори на мобільних пристроях, а також використовувати інакші методи налагодження.

- Тестування. Visual Studio IDE має платформу тестування програм, яка дозволяє розробникам переконатися чи все виконано вірно. Це зменшує затрати сил на часу на ручне тестування.
- Командна співпраця. Зазвичай розробників буває не один, а декілька, саме тому платформа має можливості для співпраці, які підвищують продуктивність команди. Ці інструменти інтегровані в життєвий цикл розробки. Окрім того, Visual Studio IDE добре працює в режимі спільної роботи незалежно від платформ кожного члена команди.
- Налаштування. Visual Studio IDE має параметри налаштування для кожного користувача. Користувачі можуть розширювати функціональні можливості платформи за допомогою розширень і доповнень, доступних у Visual Studio Marketplace. Розробники навіть можуть публікувати також власні розширення.

Після вибору середовища розробки потрібно обрати мову програмування, тут вибір впав на мову програмування C#, розглянемо цю мову та її переваги.

Мова C# є популярною мовою із різних причин, але головним перевагає, що ця мова є універсальною, досить легкою для вивчення та є об’єктно-орієнтованою. C# – це сучасна мова програмування загального призначення, яку можна використовувати для виконання досиит широкого кола завдань та цілей, які охоплюють різні професії. C# як правило використовується в середовищі Windows.NET [51], хоча його можна застосувати для платформ з відкритим кодом.

Так як і інші мови програмування загального призначення C# можна використовувати для створення різних програм: мобільних додатків,

настільних додатків, хмарних служб, веб-сайтів, корпоративного програмного забезпечення та комп’ютерних ігор.

Мову C# було створено Microsoft для Microsoft, тому стає зрозуміло чому він найчастіше використовується для розробки настільних програм Windows [52]. Щоб програми C# працювали найкраще, потрібна платформа Windows.NET, тому найкращим варіантом використання цієї мови є розробка додатків і програм, які відповідають архітектурі платформи Microsoft [53], якраз така яка потрібна для виконання завдання по створення програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі.

Мова програмування C# масштабована та проста в обслуговуванні [54]. Через сувору природу написання статичних кодів, програми на мові C# є надійно узгодженими, що робить їх набагато легшими для налаштування та обслуговування, ніж програми, написані з використанням інших мов.

Якщо вам знадобиться повернутися до старого проекту, написаного на C#, ваші процеси, можливо, змінилися з часом, проте стек C# залишився незмінним у всьому.

У програмуванні є важливим спільнота, на яку можна покластися. Мови програмування – це не лише платформа чи служба зі зручною підтримкою. Програмісти повинні покладатися на підтримку інших в тій сфері, в якій вони працюють. Одну з таких спільнот експертів з програмування на мові C# можна знайти на форумі StackOverflow [55], де можна дізнатись відповідь на потрібне питання.

Мова програмування C# є повністю об'єктно-орієнтованою, що є рідкісною характеристикою для мови програмування. Багато з поширеніших мов певною мірою включають об’єктну орієнтованість [56], але невелика кількість з них досягли масштабу C#. Парадигма об’єктно-орієнтованого програмування (ООП) має безліч різних переваг, зокрема ефективність та гнучкість.

## 2.5 Висновок до другого розділу

В другому розділі кваліфікаційної роботи проаналізовано та досліджено бази даних, Microsoft SQL Server, клієнт-серверну архітектуру, технологію доступу до даних ADO.NET, середовище розробки Microsoft Visual Studio, об’єктно-орієнтовану мову програмування, та обрано оптимальні засоби та технології для створення програмного забезпечення для обліку реалізації товарів в торгівлі.

Зокрема, для розробки програмного забезпечення для обліку реалізації товарів в торгівлі обрано: середовище розробки Microsoft Visual Studio, мову програмування C#, технологію ADO.NET, клієнт-серверну архітектуру з використанням тонкого клієнту, базу даних Microsoft SQL Server.

Також проаналізувано, які поєднання технологій найкраще використовувати для створення об’єктно-орієнтованого програмного забезпечення на основі клієнт-серверної архітектури.

Використання клієнт-серверної архітектури дозволить використовувати програмне забезпечення на слабких комп‘ютерах, що буде значною перевагою розроблюваного програмного рішення.

# 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ ДЛЯ ОБЛІКУ РЕАЛІЗАЦІЇ ТОВАРІВ В ТОРГІВЛІ

## 3.1 Робота з базою даних

Розроблене та представлене в даній роботі програмне забезпечення створено на мові програмування C# та з використанням СУБД Microsoft SQL Server. Програмне забезпечення можна було описати без використання Microsoft SQL Server, описавши таблиці в програмі. Але всі обчислення відбуваються на сервері, так як це спрощує роботу на стороні клієнта та дозволяє зберігати великі об’єми інформації на сервері. Для цієї взаємодії з сервером було використано класи «sqlDataAdapter», «sqlConnection», «sqlCommand», які є певним мостом між C# та MS SQL. За допомогою методів цього класу, реалізовано роботу між програмним забезпеченням та базою даних.

В програмі доступний функціонал додавання, редагування та видалення даних у таблицях, які змінюються у режимі реального часу.

В базі даних, у таблицях якої зберігаються дані, за допомогою збережених процедур реалізовано «товстий» сервер із усією бізнес-логікою. База даних отримує запити від клієнта, обробляє їх та надсилає відповідь клієнту за допомогою локального мережевого з'єднання. Довідники у програмі взаємодіють із базами даних на стороні сервера, на якому прив’язані процедури. Серверна частина описана в таблицях та процедурах. Певні таблиці для клієнтської частини було створено віртуально за допомогою конструкцій Left Join, Select та Inner Join мови запитів SQL.

Всього для програмного забезпечення для обліку реалізації товарів в торгівлі було створено десять таблиць. Зі зв’язками між таблиця та полями таблиць можна ознайомитись на рисунку 3.1.

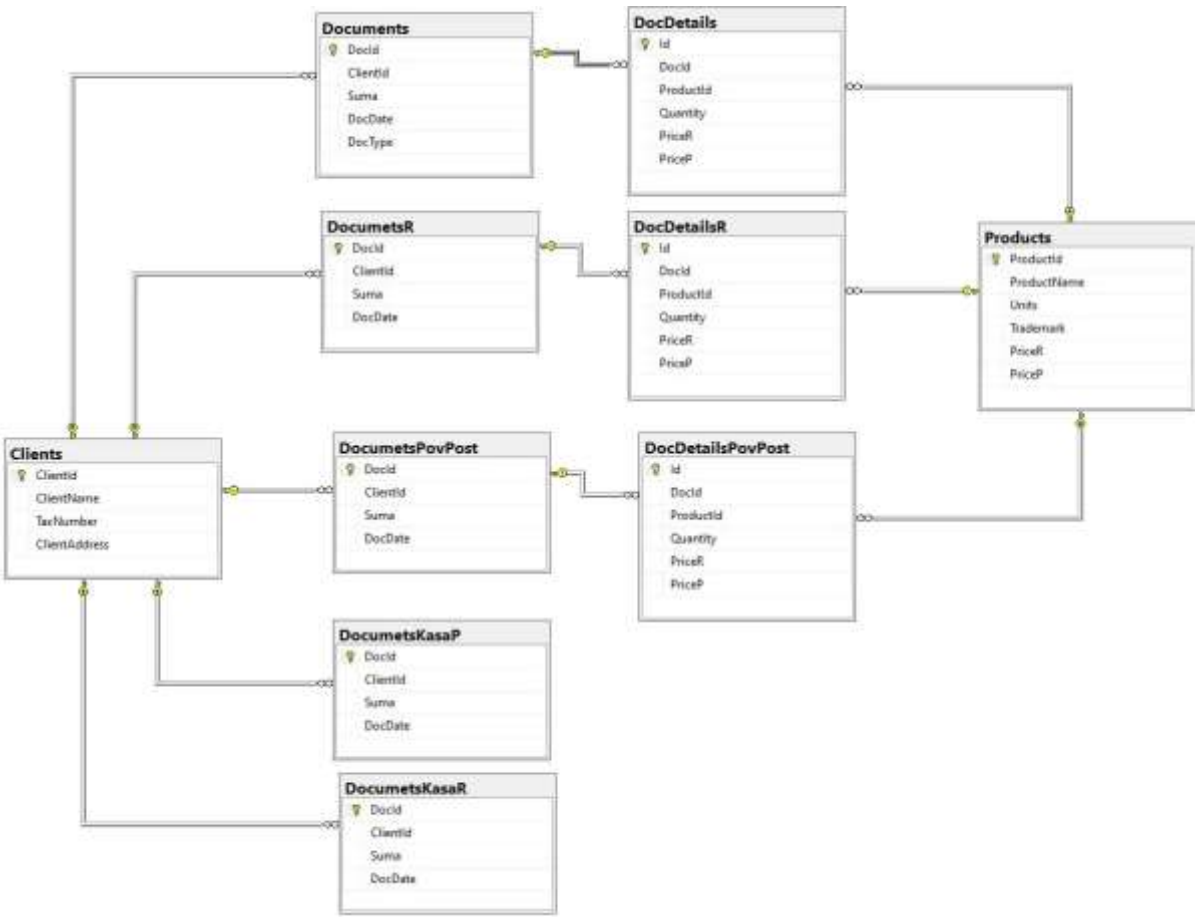



Рисунок 3.1 – Схема таблиць бази даних

Призначення таблиць наступне:

- Таблиця “Documents” – зберігаються дані прихідних документів.
- Таблиця “DocDetails” – зберігаються детальні дані прихідних документів.
- Таблиця “DocumentsR” – зберігаються дані розхідних документів.
- Таблиця “DocDetailsR” – зберігаються детальні дані розхідних документів.
- Таблиця “Products” – зберігаються дані товарів.
- Таблиця “Clients” – зберігаються дані клієнтів.
- Таблиця “DocumentsPovPost” – зберігаються дані документів повернень постачальнику.
- Таблиця “DocDetailsPovPost” – зберігаються детальні дані документів повернень постачальнику.

- Таблиця “DocumentsKasaP” – зберігаються прибуткові касові ордени.
- Таблиця “DocumentsKasaR” – зберігаються видаткові касові ордени.

Отже, зв’язок між клієнтом та базою даних організований за допомогою класу «sqldataAdapter». Він представляє набір команд даних і підключення до бази даних, які використовуються для заповнення та оновлення бази даних Microsoft SQL Server.

### 3.2 Розробка класів клієнтської частини програмного забезпечення

Клієнтська частина була реалізована в середовищі розробки Microsoft Visual Studio з використанням мови програмування C#. Для з’єднування клієнта із сервером було використано архітектуру ADO.NET, двома основними компонентами якої, для доступу до даних і управління ними є постачальники даних платформа .NET Framework і DataSet. ADO.NET складається із набору класів, що використовуються для підключення до бази даних та надання доступу до реляційних даних, даних додатків, а також для отримання результатів. Постачальники даних ADO.NET містять класи, які представляють об’єкти постачальника Connection, Command, DataAdapter.

ADO.NET забезпечує певний місток між елементами керування та серверною базою даних. Об’єкти ADO.NET інкапсулюють операції доступу до потрібних даних, елементи керування взаємодіють з цими об’єктами для відображення потрібних даних, забезпечуючи таким чином безпеку, приховуючи подробиці переміщення даних. Дана технологія дозволяє отримати доступ до даних за один раз, а також переглянути всю колекцію даних у одному екземплярі. Також вона представляє певну відключену архітектури в якому можна захопити колекцію даних, та використовувати дані окремо від самого сховища даних.

Розглянемо класи клієнтської частини створеного програмного забезпечення на основі клієнт–серверної архітектури для обліку реалізації товарів в торгівлі. На рисунку 3.2 представлено зв’язки між класами в клієнтській частині програмного забезпечення.

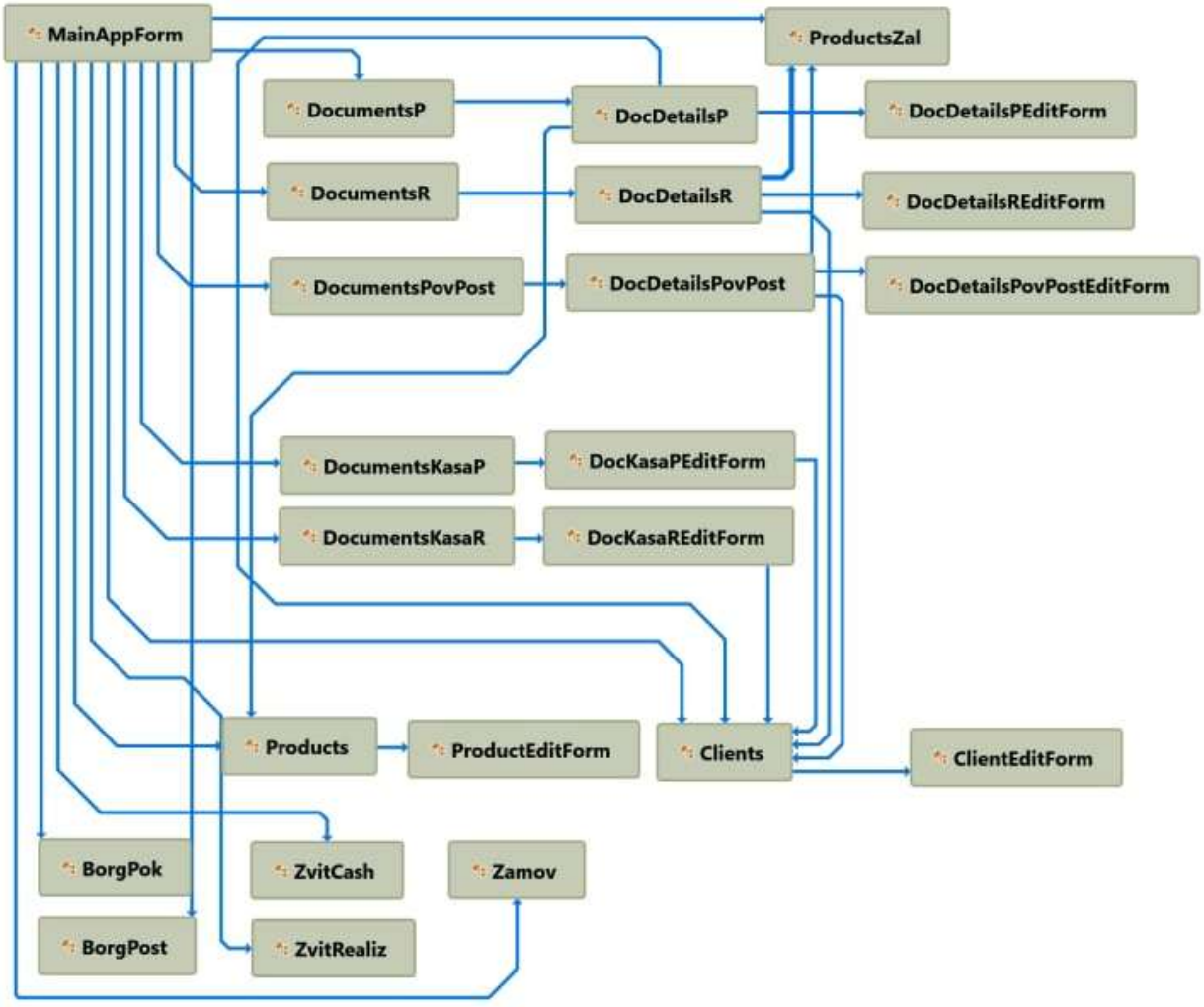



Рисунок 3.2 – Зв’язки між класами в клієнтській частині ПЗ

В кожному з класів зв'язок з базою даних відбувається за тим же принципом. За допомогою архітектури ADO.NET з використанням об’єкта DataSet та його команд дані передають у таблицю. В таблиці 3.1 подано відомості про те, за що відповідає кожен із класів.

Таблиця 3.1 – Класи клієнтської частини ПЗ

| **Клас** | **Призначення класу** |
|---|---|
| MainAppForm | Форма стартового меню |
| DocumentsP | Форма прихідних документів |
| DocDetailsP | Форма детальних даних прихідного документа |
| DocDetailsPEditForm | Форма для змін в прихідному документі |
| DocumentsR | Форма розхідних документів |
| DocDetailsR | Форма детальних даних розхідного документа |
| DocDetailsREditForm | Форма для змін в прихідному документі |
| DocumentsPovPost | Форма повернення товарів постачальнику |
| DocDetailsPovPost | Форма детальних даних повернення постачальнику |
| DocDetailsPovPostEditForm | Форма для внесення змін в деталі повернення |
| ProductZal | Форма залишків товару |
| DocumentsKasaP | Форма прибуткових касових ордерів |
| DocumentsKasaPEditForm | Форма для внесення змін в прибутковий ордер |
| DocumentsKasaR | Форма видаткових касових ордерів |
| DocumentsKasaREditForm | Форма для внесення змін в видатковий ордер |
| Products | Форма товарів |
| ProductsEditForm | Форма внесення змін в товари |
| Clients | Форма клієнтів |
| ClientsEditForm | Форма внесення змін даних клієнтів |
| BorgPok | Форма боргів покупців |
| BorgPost | Форма боргів постачальникам |
| ZvitCash | Форма звіту по руху коштів |
| ZvitRealiz | Форма звіту по реалізації товарів |
| Zamov | Форма замовлення товарів постачальнику |

На рисунках 3.3 – 3.6 зображено поля та методи класів клієнтської частини створеного програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі.

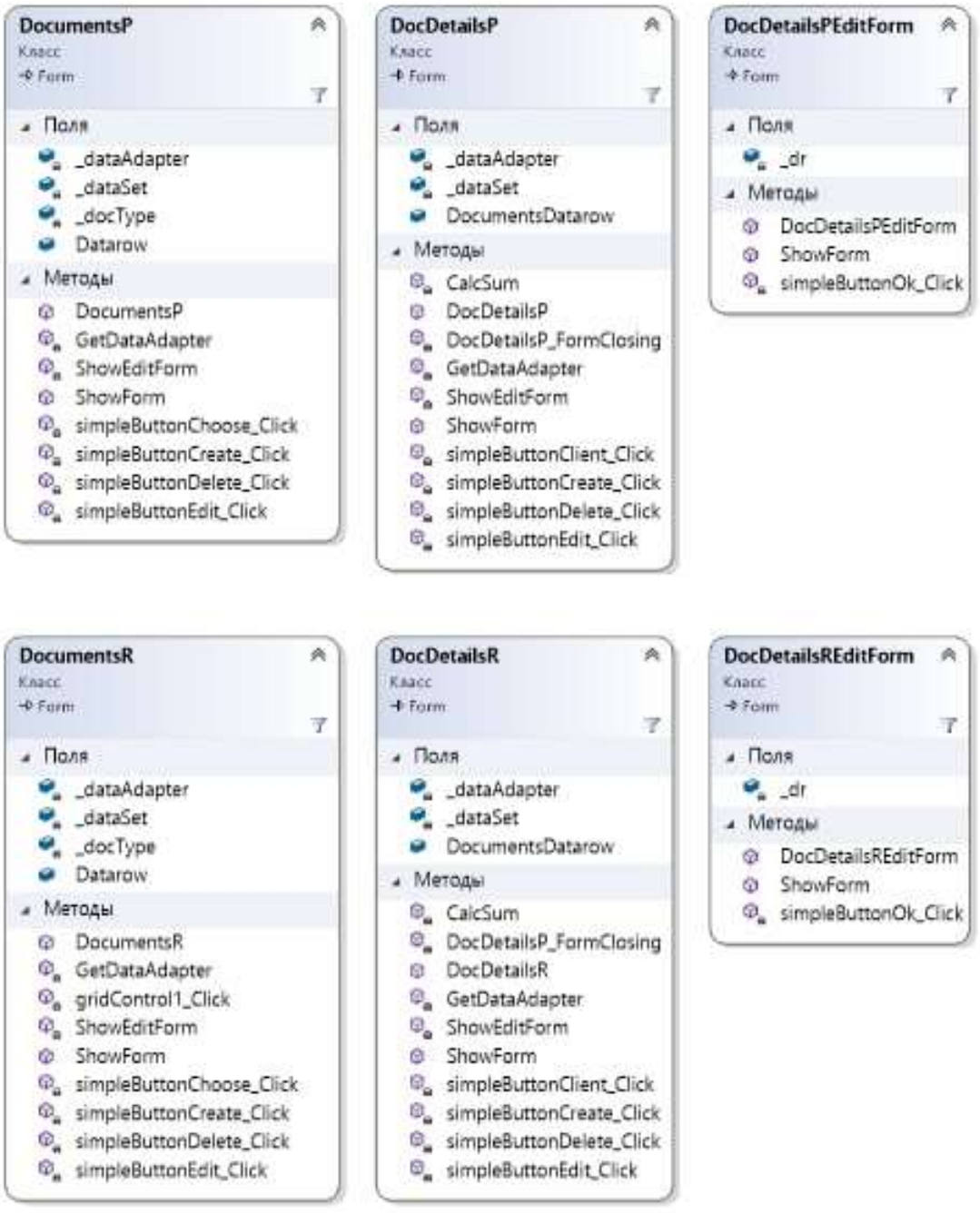



Рисунок 3.3 – Поля та методи класів: прихідних, розхідних документів та форми для редагування цих документів

На рисунку 3.3 зображено класи прихідних документів та детальних даних до них, розхідних документів та детальних даних для них, а також два класи, які закінчуються «EditForm» призначені для редагування цих документів.

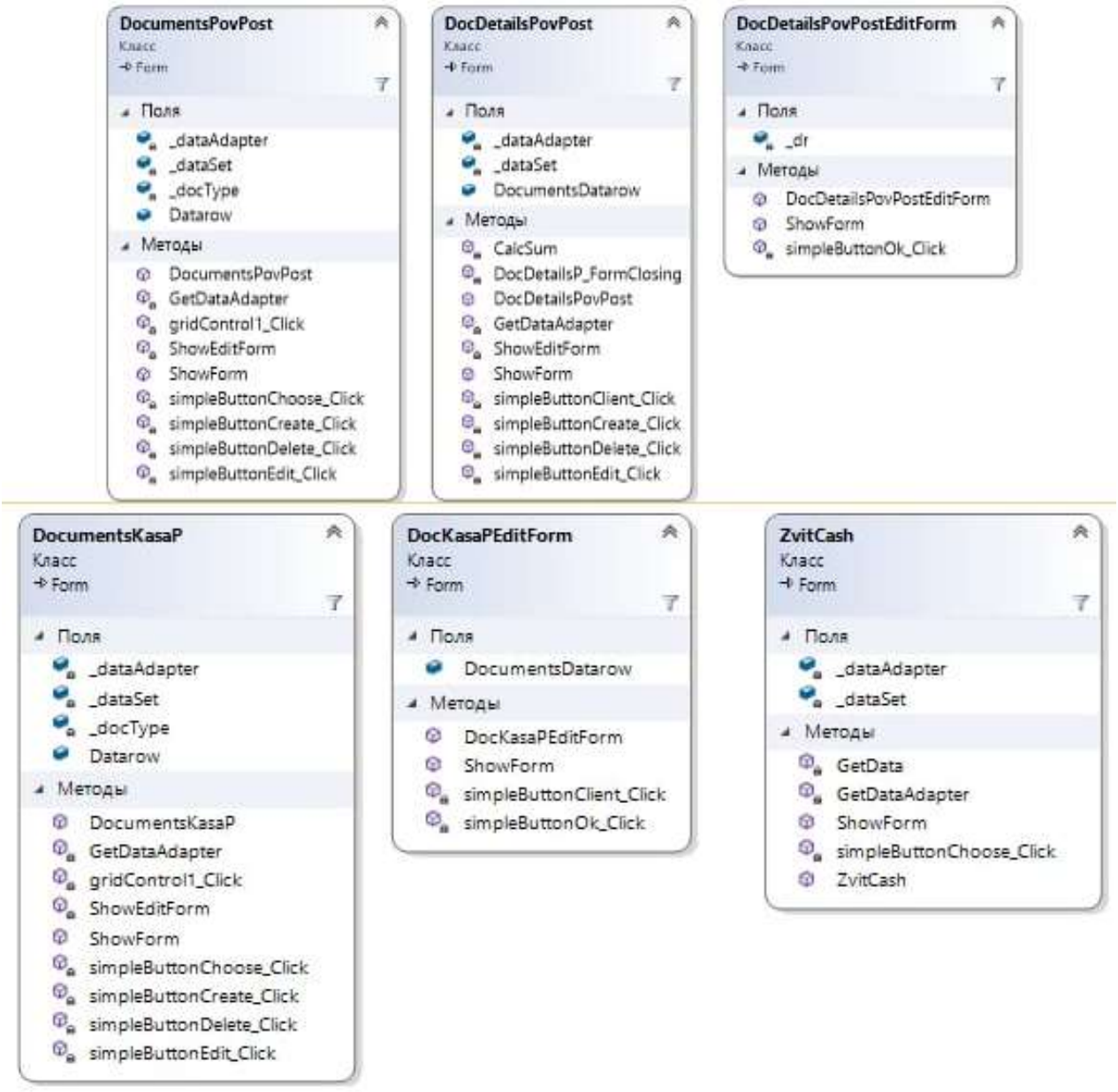



Рисунок 3.4 – Поля та методи класів: повернення постачальнику, прибуткових касових ордерів, руху грошових коштів

На рисунку 3.4 зображено класи: повернення постачальнику, детальні дані повернення постачальнику, які включать дані про товари, які йдуть на

повернення, прибуткових касових ордерів, руху грошових коштів, а також два класи, які закінчуються «EditForm» призначені для редагування цих документів.

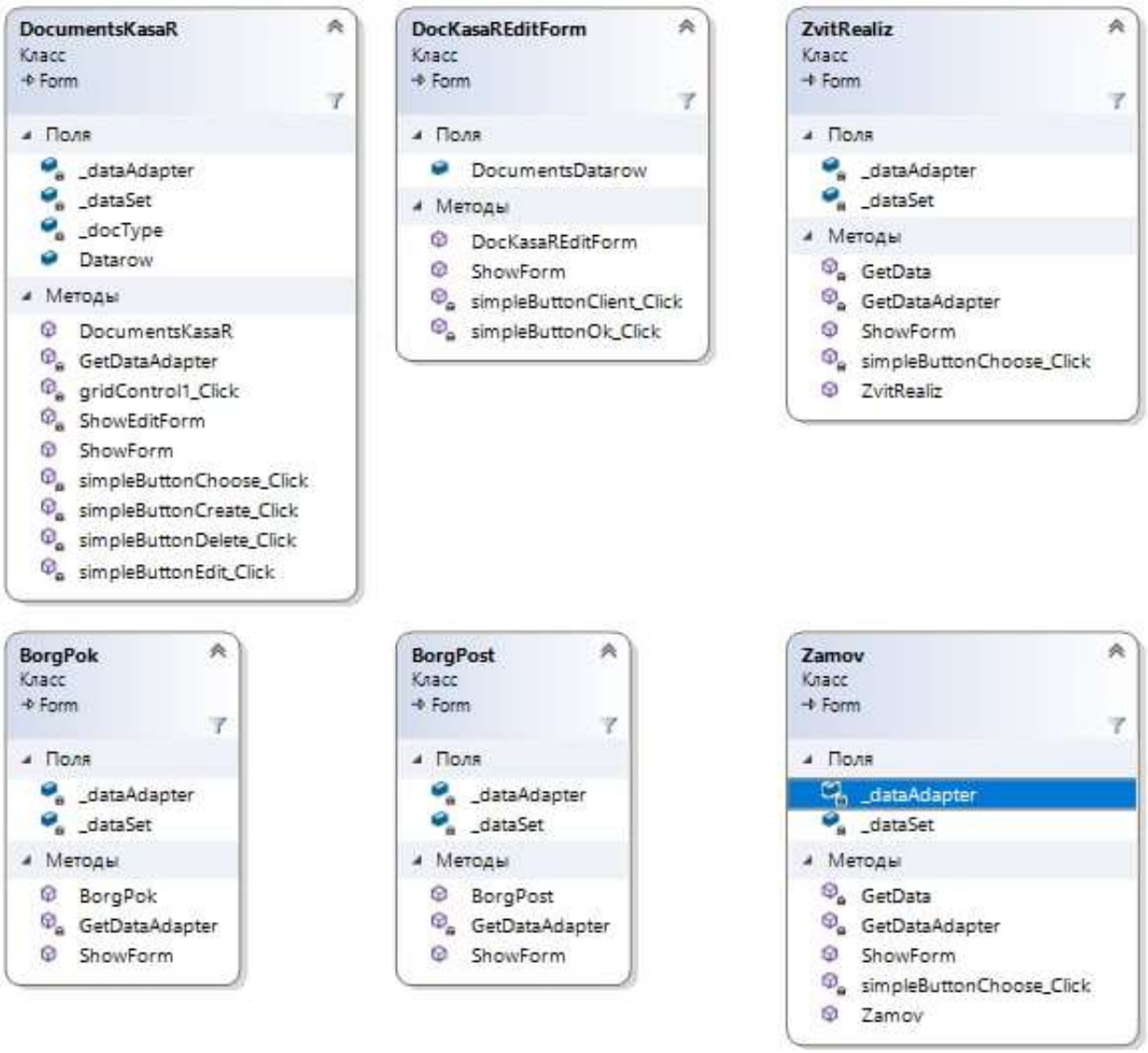



Рисунок 3.5 – Поля та методи класів: видаткових касових орденів, звіту про реалізацію товару, борги покупців, борги постачальникам, замовлення товару

На рисунку 3.5 зображено класи: видаткових касових орденів, який містить реєтр цих орденів, звіту про реалізацію товару, борги покупців, борги постачальникам, замовлення товару по кожній позиції, а також клас під назвою «DocKasaREditForm», який призначений для редагування видаткових

касових орденів. Видатковий касовий ордер є обов'язковим документом при наявності  каси у підприємства, а також якщо торгова компанія видає готівку на торгові потреби або є якщо передає гроші до банку.

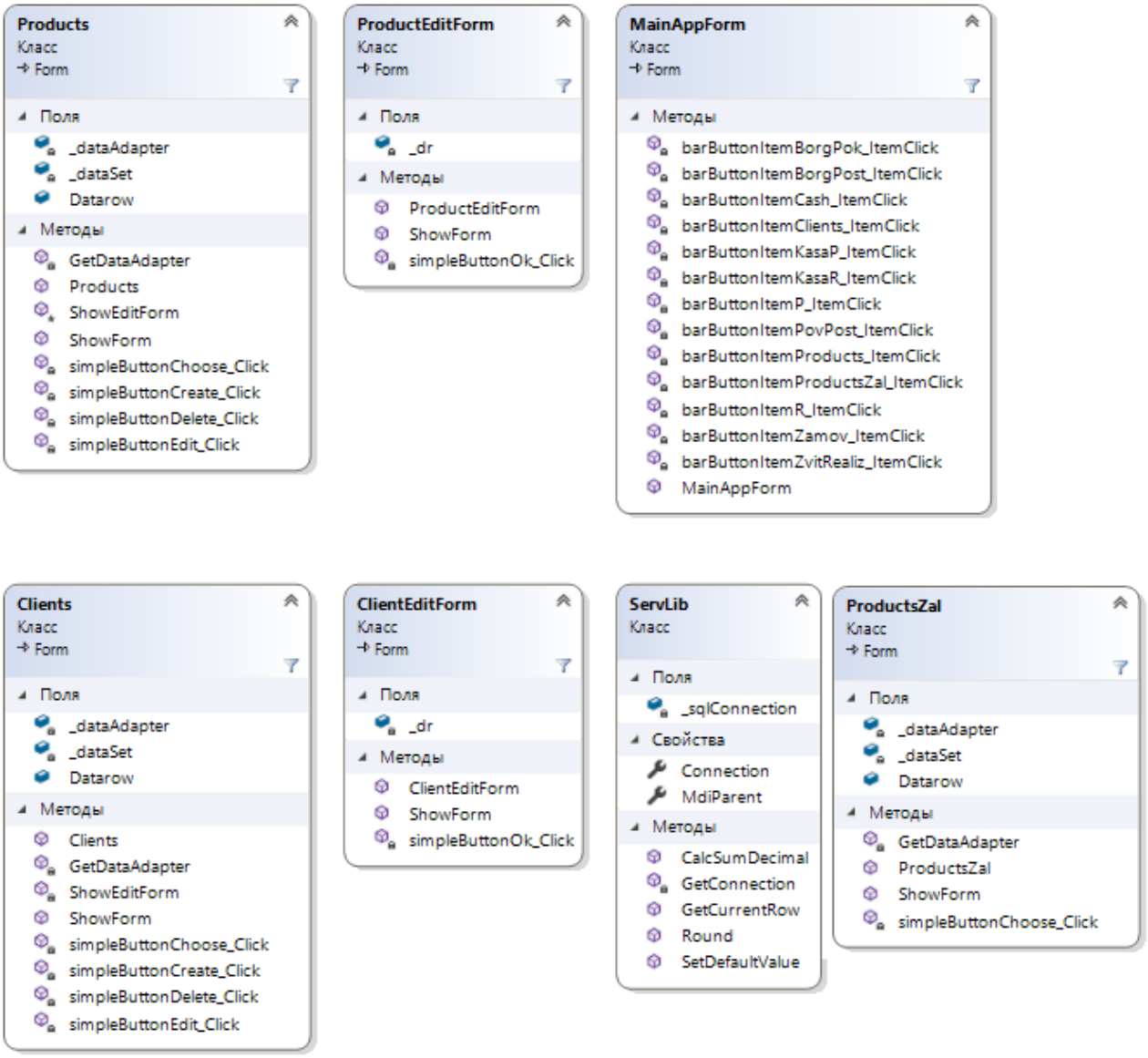



Рисунок 3.6 – Поля та методи класів: товарів, стартового меню, клієнтів, залишків товару, серверної бібліотеки

Як бачимо на рисунках 3.3 – 3.6, кожен з класів має схожі поля, вони надають доступ до об'єктів, які служать для зберігання даних в клієнті та обміну інформацією з базою даних. Методи ж в усіх класах відповідають за дію кнопок (створити, редагувати, видалити, обрати) та взаємодією між

базою даних та клієнтом. Загалом було створено 24 класи, з призначенням яких можна ознайомитись в таблиці 3.1, із взаємозв'язком між них можна ознайомитись на рисунку 3.2, з полями та методами на рисунках 3.3 – 3.6.

### 3.3 Реалізація стартового меню програми-клієнта

Стартове меню включає в себе три вкладки, вкладка документи включає такі функції: прихід товарів, реалізація товарів, повернення товарів постачальнику. Вкладка довідники включає: клієнти, товари. Вкладка звіти включає: поточні залишки товарів, звіт по реалізації товарів, замовлення товарів, борги постачальникам, борги покупців, рух грошових коштів. Дані вкладки були графічно реалізовані, ознайомитись з ними можна на рисунках 3.7 – 3.9.

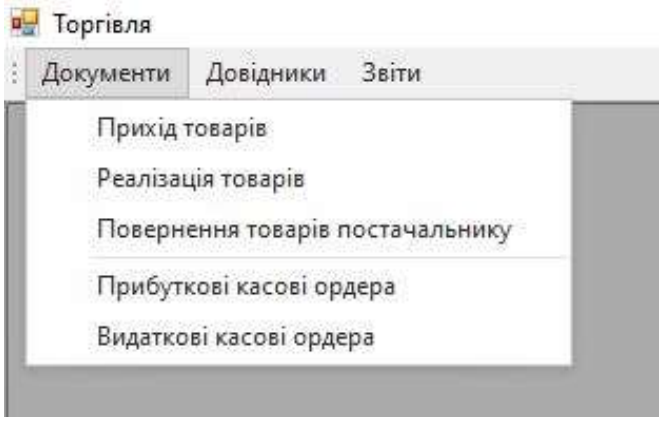



Рисунок 3.7 – Вкладка «Документи»

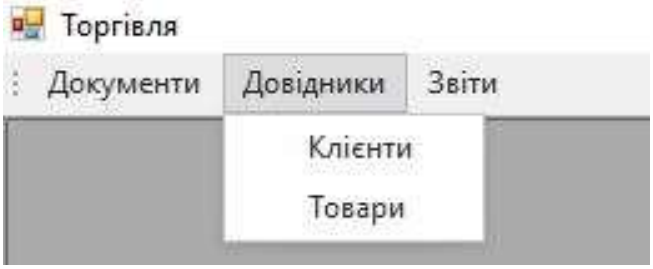



Рисунок 3.8 – Вкладка «Довідники»


Рисунок 3.9 – Вкладка «Звіти»

Загалом стартове меню програмного забезпечення включає в себе 3 вкладки та 13 функцій в них.

### 3.4 Реалізація довідника клієнтів та товарів

Довідник клієнтів – реєстр для накопичення карток із постачальниками із даними такими як: внутрішній код клієнта, назва клієнта, податковий номер, адреса.

Довідник товарів – реєстр даних в якому накопичуються картки із товарами з вказаними характеристиками товару. Довідник містить такі функції: створення нової картки товару, внесення змін в існуючу картку товару і видалення картки з товаром.

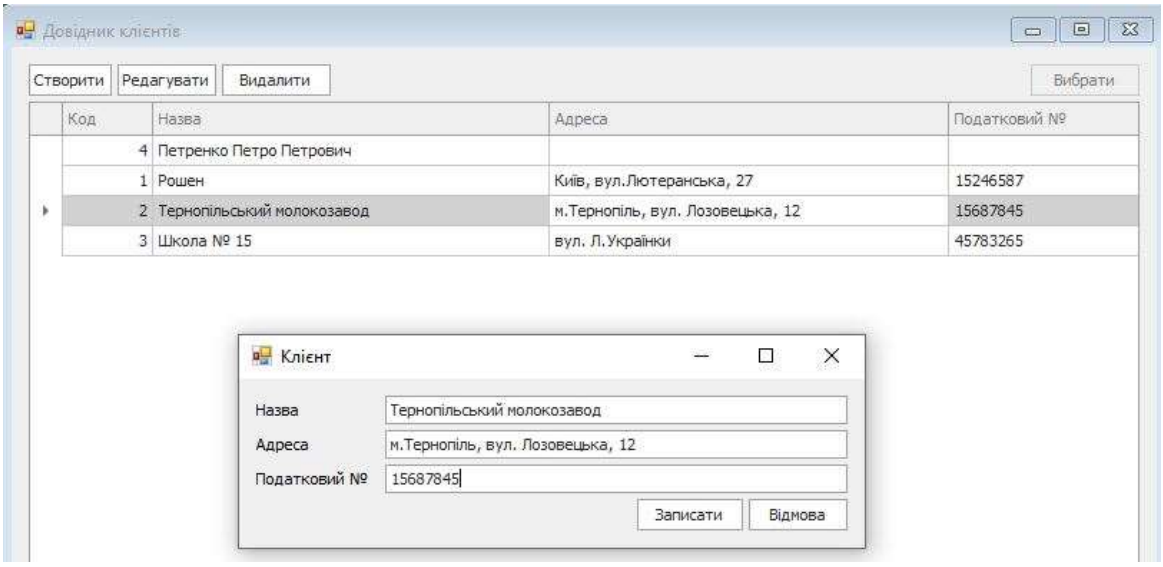



Рисунок 3.10 – Вкладка довідника клієнтів

На рисунку 3.10 зображено вкладку довідника клієнтів. Довідник клієнтів містить такі функції, як створення, редагування, видалення та вибирання. Вкладка запису клієнтів містить такі поля: назва, адреса та податковий номер.

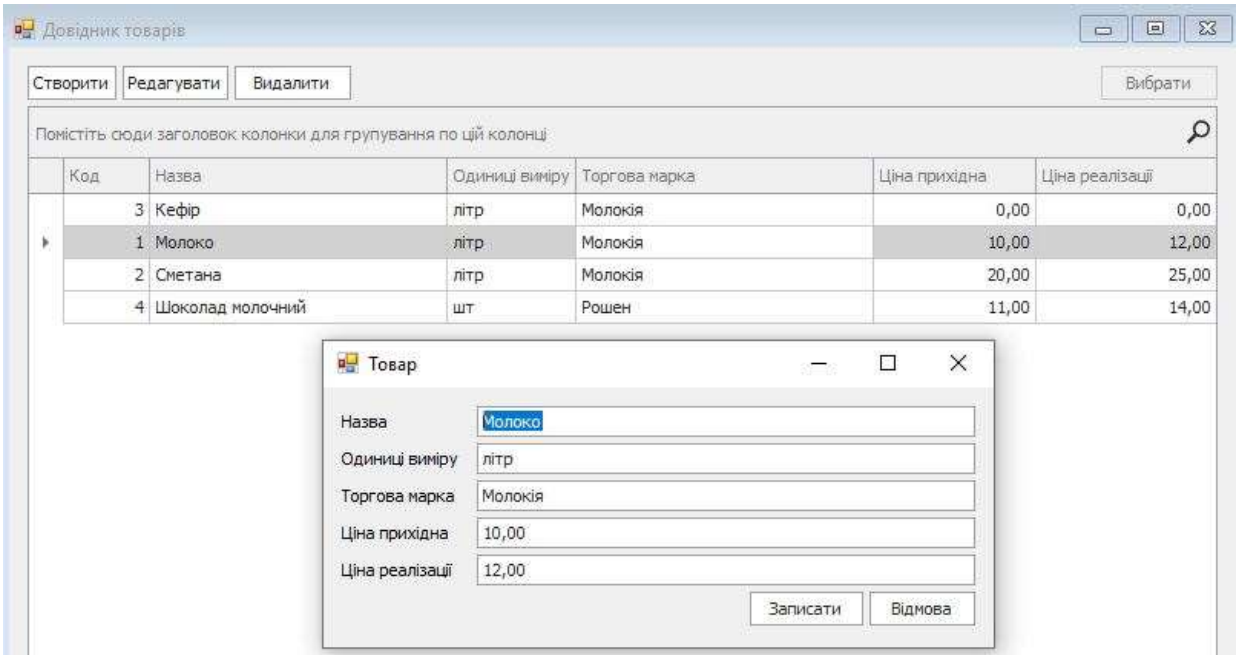



Рисунок 3.11 – Вкладка довідника товарів

Функціями вікна клієнта на рисунку 3.11 є створення нової картки товару, внесення змін у створену картку, видалення картки клієнта. У вкладці з додаванням товару є можливість вказати: назву товару, одиниці виміру, торгову марку, прихідну ціну та ціну реалізіції товару.

Для роботи з даними довідника товарів, а також аналогічно для довідника клієнтів, було написано 3 збережені процедури: реалізована функція добавлення Insert та оновлення Update даних; реалізована функція відбору даних Select; реалізована функція знищення Delete.

Функція Insert описана у процедурі Update, і реалізовано функцією If. При отриманні значення id=0, створюється новий запис, при отриманні значення id=1 запускається функція Update, яка дозволить відредагувати вже

введені значення в таблицях. Лістинг функції добавлення і оновлення інформації в довідники зображений у лістингах 3.1-3.2.

Лістинг 3.1 – Реалізована функція добавлення і оновлення даних в довіднику продуктів

```
GO
CREATE PROCEDURE [dbo].[ProductsUpdate]
@ProductName varchar(100), @Trademark varchar(100), @ProductId
int output,  @Units varchar(20), @PriceP numeric (9,2), @PriceR
numeric (9,2)
AS
BEGIN
     if @ProductId = 0        begin
               insert   into   Products   (Trademark,   PriceR,
ProductName, Units,  PriceP)
                    values  (@Trademark,  @PriceR,  @ProductName,
@Units, @PriceP)
               select @ProductId = SCOPE_IDENTITY()
          end
     else
          begin
               update Products set Units = @Units, ProductName =
@ProductName, Trademark = @Trademark, PriceP = @PriceP, PriceR =
@PriceR,
where ProductId = @ProductId
          end
END
```

Лістинг 3.2 – Реалізована функція добавлення і оновлення даних в довіднику клієнтів

```
     if @ClientId = 0
          begin
               insert  into  Clients  (TaxNumber,  ClientAddress,
ClientName) values (@TaxNumber, @ClientAddress, @ClientName)
               select @ClientId = SCOPE_IDENTITY()
          end
     else
          begin
               update   Clients   set   TaxNumber   =   @TaxNumber,
ClientAddress = @ClientAddress, ClientName = @ClientName,
          where ClientId = @ClientId
          end
```

Функція ж видалення у двох довідниках працює таким чином, що по ідентифікаційному номеру товару в базі данних знаходиться необіхдний запис та видаляється.

Функція відбору SelectAll реалізована так, що відбирає необхідні нам дані з бази даних по назві продукту.

Загалом для довідника товарів та клієнтів  написано по 3 збереженій процедури на кожний довідник.

### 3.5 Реалізація реєстру розхідних та прихідних документів, залишків товару та звіту по реалізації товарів

Спочатку розглянемо як реалізовані реєстри розхідних та прихідних документів. У полі «DocType» в даній таблиці реєструється тип документу, а саме 1 – це прихідні документи, 2 – розхідні.

В реєстрах прихідних та розхідних документів є таблиці і функціональні кнопки, такі як: створити документ, редагувати документ, видалити документ.

Коли створюється новий документ чи редагується поточний, відкривається вікно, в якому описано товар, його кількість, його ціна придбання чи продажі, у випадку розхідного документу. При добавленні нового продукту до списку, з’являється вікно «Довідник товарів», з якого є можливість обрати потрібну товари, а натиснувши кнопку «Вибрати», відбувається відкриття вікна редагування, в якому є можливість записати яка кількість товару та по якій ціні була придбана. При створенні нового об'єкту у журналі розхідних документів, програмне забезпечення відкриє таблицю залишків, щоб була можливість обрати товар, серед тих які є на складі.

Формування ж прихідних накладних реалізується за такою послідовністю:

1. Створюється бланк накладної

2. Відбувається добавляння однієї або більше позиції товару по ціні закупівлі та вказується кількість товару.

3. Відбувається розрахунок суми приходу товарів.

4. Вибирається постачальник із довідника постачальників, або ж є можливість у даному вікні створити нову картку постачальника.

Програмне забезпечення також дає можливість користувача спочатку здійснити продаж товарів та утворити від'ємну кількість товару в залишках, а потім за допомогою прихідної накладної, кількість дійне до фактичної кількості.

Ознайомитись з виглядом вікон та тестовими позиціями прихідного та та розхідного документу можна на рисунках 3.12 – 3.15.

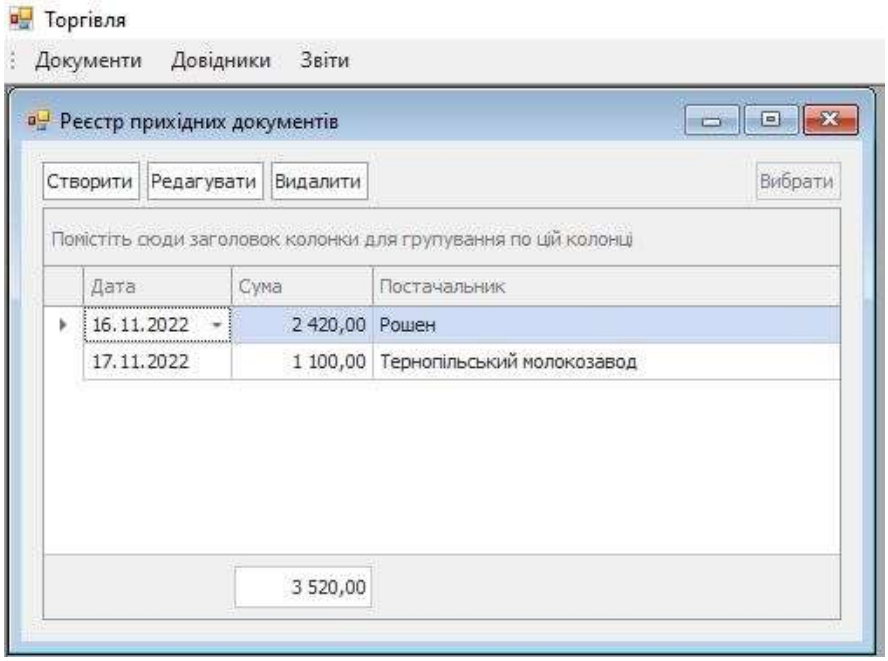



Рисунок 3.12 – Реєстр прихідних документів

На рисунку 3.12 зображено реєстр прихідних документів. В цей реєстр потрапляють всі створені прихідні документи. Реєстр показує дату приходу товару, загальну суму товару який поступив, а також постачальника. Користувач має змогу створити, редагувати, видалити та вибрати у цьому вікні відповідний прихідний документ.

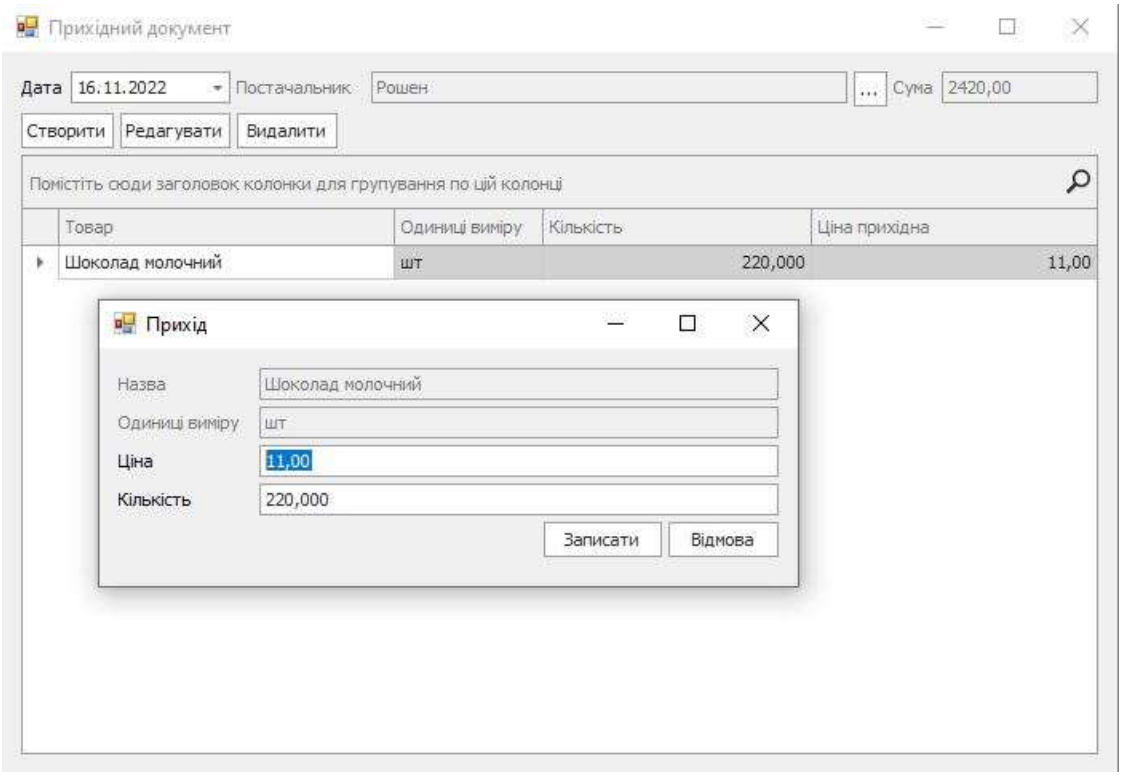



Рисунок 3.13 – Вікно прихідних документів

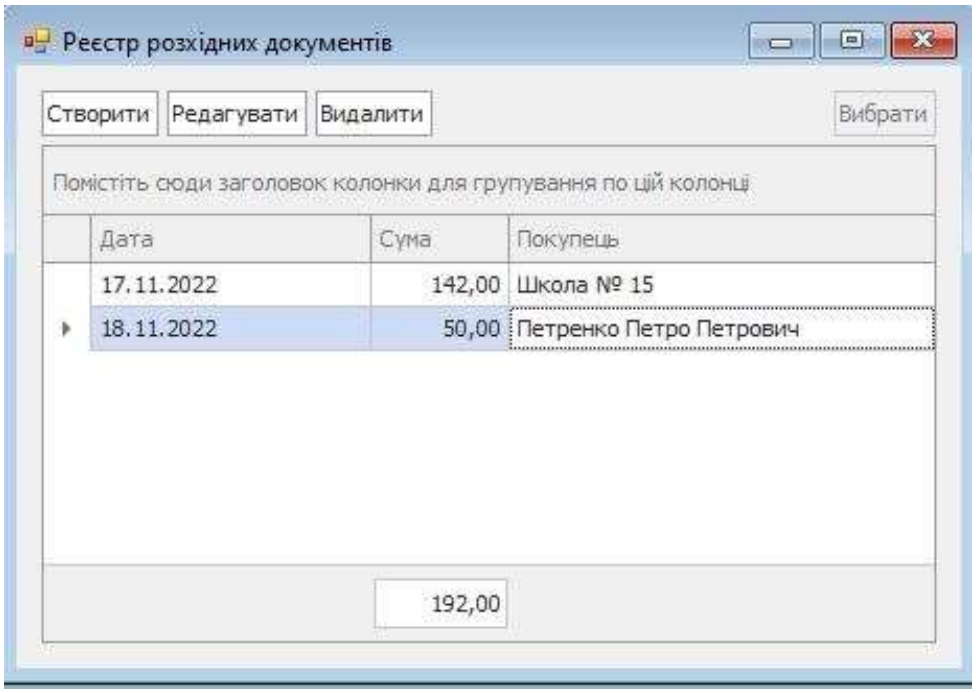



Рисунок 3.14 – Реєстр прихідних документів

На рисунку 3.14 зображений реєстр розхідних документів, тут відображається дата, сума, а також покупець, який здійснив купівлю товару.

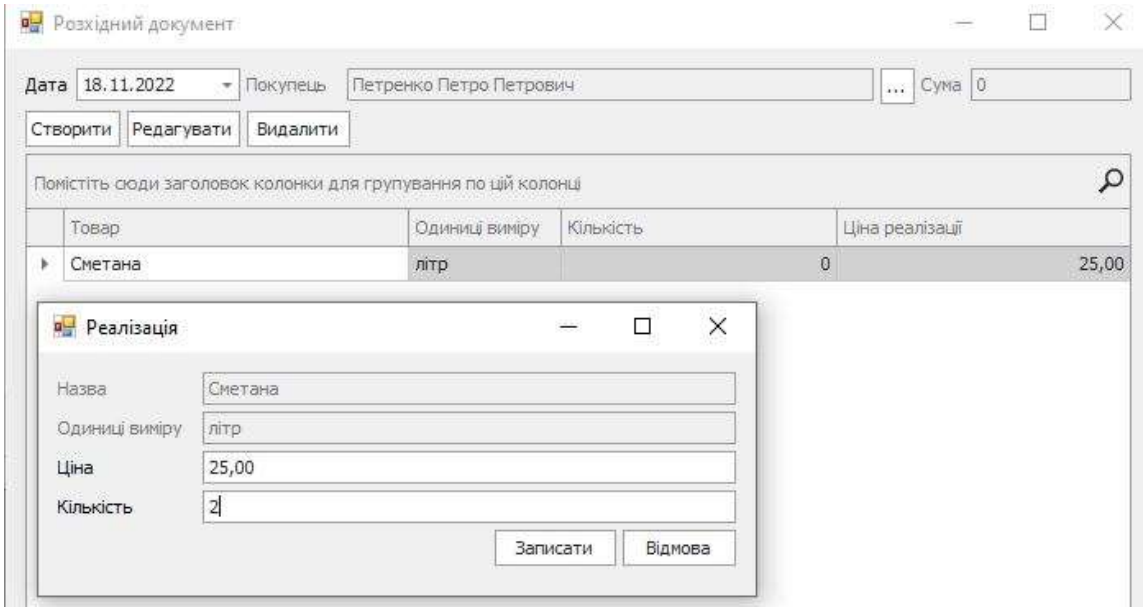


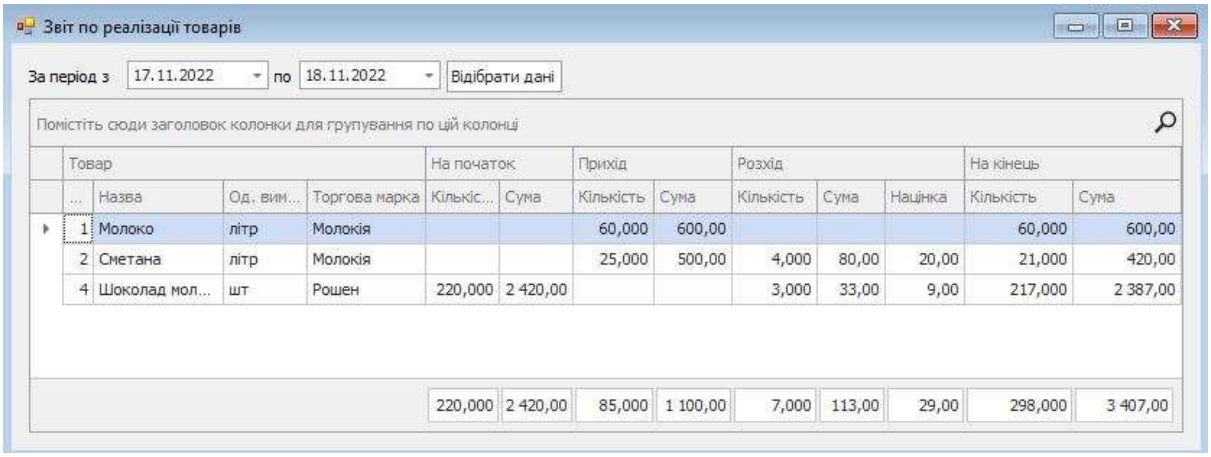

Рисунок 3.15 – Вікно розхідних документів

Поточні залишки товарів генеруються у реальному часі та демонструють кількість і назву товару, яка присутня у даний час на складі (рис. 3.16).

Вікно звіту по продажу товару включає в себе таблицю та кнопки вибору дати для отримання звіту по продажу товарів за певний період часу (рис. 3.17).


Рисунок 3.16 – Вікно поточних залишків

У вікні поточних залишків можна отримати інформацію про товар, який зараз наявний на складі, а також такі його параметри як: назва, одиниця вимірювання, торгова марка, кількість, ціна прихідна та реалізації.




| | Товар | | | На початок | | Прихід | | Розхід | | | На кінець | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | Назва | Од. вим... | Торгова марка | Кількіс... | Сума | Кількість | Сума | Кількість | Сума | Націнка | Кількість | Сума |
| 1 | Молоко | літр | Молокія | | | 60,000 | 600,00 | | | | 60,000 | 600,00 |
| 2 | Сметана | літр | Молокія | | | 25,000 | 500,00 | 4,000 | 80,00 | 20,00 | 21,000 | 420,00 |
| 4 | Шоколад мол... | шт | Рошен | 220,000 | 2 420,00 | | | 3,000 | 33,00 | 9,00 | 217,000 | 2 387,00 |
| | | | | 220,000 | 2 420,00 | 85,000 | 1 100,00 | 7,000 | 113,00 | 29,00 | 298,000 | 3 407,00 |

Рисунок 3.17 – Вікно звіту по реалізації товарів

Для роботи з даними таблиць документів була реалізована функція добавляння Insert, оновлення даних Update, функція відбору даних Select, а також реалізована функція знищення Delete.

Функція добавляння Insert описана у процедурі Update та виконана функцією If. При значення id=0 сторюється новий запис, при отриманні id=1 запускається функція Update, яка відредагує вже введені раніше значення в таблицях.

Лістинг 3.3 – Реалізація добавлення та оновлення даних у таблиці з документами

```
CREATE PROCEDURE [dbo].[DocumentsUpdate]
@Suma numeric (18,2), @DocType smallint, @DocId int output,
@DocDate datetime, @ClientId int
AS
BEGIN
     if @DocId = 0
          begin
               insert into Documents (DocDate, ClientId, Suma,
DocType)
                    values (@DocDate, @ClientId, @Suma, @DocType)
               select @DocId = SCOPE_IDENTITY()
          end
     else
          begin
               update Documents set DocDate = @DocDate, ClientId
= @ClientId, Suma = @Suma, DocType = @DocType
                    where DocId = @DocId
          end
```

Функція знищення працює так, що по ідентифікаційному номеру документа у базі даних знаходиться потрібний запис і проводиться видалення з двох таблиць документів прихідних та розхідних.

Функція відбору даних з таблиці документів так, що дані беруться з різних таблиць та об’єднуються за допомогою функції Left Join.

Лістинг 3.4 – Реалізація добавляння та оновлення даних у таблиці прихідних та розхідних документів

```
CREATE PROCEDURE [dbo].[DocDetailsUpdate]
@Quantity numeric (18,3), @PriceR numeric (9,2), @Id int output,
@DocId int, @ProductId int, @PriceP numeric (9,2)
AS
BEGIN
     if @Id = 0
          begin
               insert into DocDetails (Quantity, PriceR, DocId,
PriceP, ProductId)
                    values (@Quantity, @PriceR, @DocId, @PriceP,
@ProductId)
               select @Id = SCOPE_IDENTITY()
          end
     else
          begin
               update DocDetails set Quantity = @Quantity, PriceR
= @PriceR, DocId = @DocId, ProductId = @ProductId, PriceP =
@PriceP
                    where Id = @Id
          end
```

Функція знищення працює тут працює таким чином: по ідентифікаційному номеру у базі даних знаходиться запис і видаляється.

Функція відбору SelectAll відбирає необхідні дані із різних таблиць та з’єднує в одну.

Для прихідних та розхідних документів в SQL було створено в загальному 6 збережених процедур, повний лістинг яких знаходиться у додатку Б.

Для звіту по реалізації товарів за певний період часу була описана функція Select. Реалізація цієї функції показана у лістингу 3.5.

Лістингу 3.5 – Реалізація функція звіту про реалізацію товарів

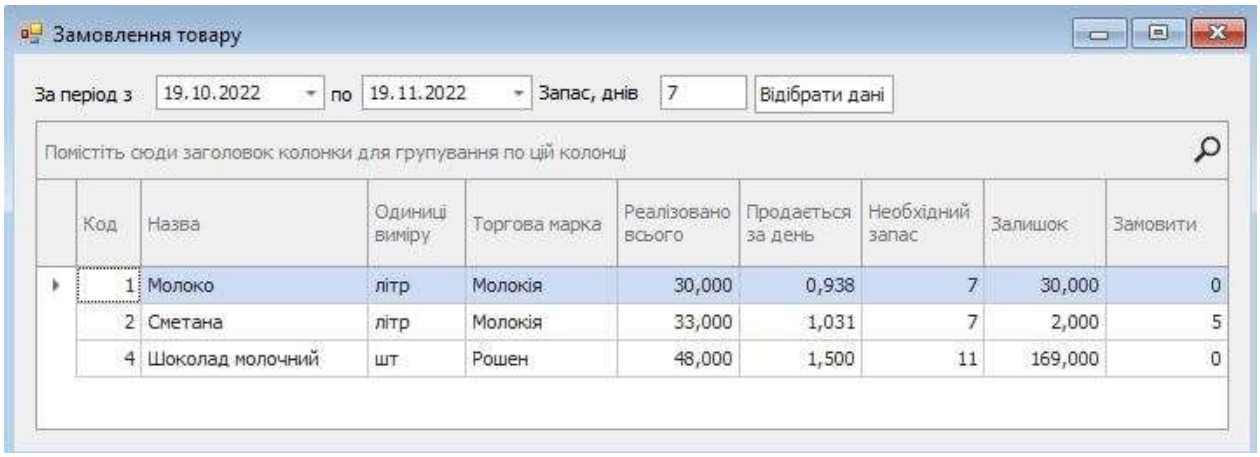
```
CREATE PROCEDURE [dbo].[SelectZalProducts]
AS
BEGIN
     SET NOCOUNT ON;

     SELECT Quantity = sum(Quantity), ProductId
          into #temp
          from DocDetails
          where DocId in (select DocId from Documents where
DocType = 1)

     insert into #temp (Quantity, ProductId)

SELECT Quantity = -1*sum(Quantity), ProductId
               from DocDetails where DocId in (select DocId from
Documents where DocType = 2)

     SELECT Quantity = sum(Quantity, ProductId)
          into #temp2
          from #temp
          group by ProductId


     SELECT t2.ProductName, t2.Units, t1.ProductId, t1.Quantity,
t2.PriceR t2.Trademark
          from #temp2 t1 inner join Products t2 on t1.ProductId =
t2.ProductId

          where t1.Quantity != 0
```

Звіт про реалізацію товару за певний проміжок часу дозволить компанії, яка користуватиметься створеним програмним забезпеченням отримуватиме інформацію, скільки товару була продано за період часу та дохід від цього.

### 3.6 Реалізація функції замовлень і повернень товарів постачальнику

У торгівлі, щоб торгувати товарами потрібно замовити товар у постачальника, проте інколи товар може не підійти, тому появляється потреба в поверненні товарів.

Функція замовлень товарів у створеному програмному забезпечення для обліку реалізації товарів в торгівлі (рис. 3.18), реалізована таким чином, що обраховує середньому кількість покупок певно товару за введений проміжок часу, та визначає скільки потрібно докупити товару, щоб не вичерпати запас, для цього потрібно ввести в таблицю скільки днів запасу товару потрібно.




| Код | Назва | Одиниці виміру | Торгова марка | Реалізовано всього | Продається за день | Необхідний запас | Залишок | Замовити |
|---|---|---|---|---|---|---|---|---|
| 1 | Молоко | літр | Молокія | 30,000 | 0,938 | 7 | 30,000 | 0 |
| 2 | Сметана | літр | Молокія | 33,000 | 1,031 | 7 | 2,000 | 5 |
| 4 | Шоколад молочний | шт | Рошен | 48,000 | 1,500 | 11 | 169,000 | 0 |

Рисунок 3.18 – Вікно замовлення товару

Ця функції дозволяє не вичерпати запас товару та спрогнозувати потрібну кількість замовлення товару. Лістинг даної функції продемонстрований у лістингу 3.6.

Лістинг 3.6 – Реалізації функції замовлень товару

```
GO
CREATE PROCEDURE [dbo].[SelectZvitZamov]
@date1 datetime, @date2 datetime, @countDays int
AS
BEGIN
     SET NOCOUNT ON;
     /*
     declare @date1 datetime, @date2 datetime
     select @date1 = convert(datetime, '25.09.2022', 104)
     select @date2 = convert(datetime, '25.10.2022', 104)
     exec SelectZvitZamov @date1, @date2, 7
     */
     declare @DaysRealiz int
     select @DaysRealiz = datediff(dd, @date1, @date2) + 1
```

```
     -- розхід товару
     select ProductId, SoldAll = sum(Quantity)
          into #t0
          from DocDetails
          where DocId in (select DocId from Documents where
DocType = 2 and DocDate between @date1 and @date2)
          group by ProductId

     select t1.*, Zal = isnull(t2.Quantity, 0)
          into #t1
          from #t0 t1 left join dbo.ZalProducts() t2 on
t1.ProductId = t2.ProductId
     alter table #t1 add
          SoldDay numeric(18,3) not null default 0,
          Need numeric(18,0) not null default 0,
          Ord numeric(18,0) not null default 0

     update #t1 set SoldDay = round(SoldAll / @DaysRealiz, 3)
     update #t1 set Need = round(@countDays * SoldDay, 0)
     update #t1 set Ord = case when Zal >= Need then 0 else
round(Need - Zal, 0) end
     select t1.*,
               t2.ProductName, t2.Units, t2.Trademark --
додаткові характеристики з довідника товарів
          from #t1 t1 inner join Products t2 on t1.ProductId =
t2.ProductId
          order by t2.ProductName
     drop table #t0
     drop table #t1
END
```

Функції повернень товарів постачальнику потрібна, якщо товар, який був внесений у програму і по певним причинам не підійшов чи містить брак та йде на повернення постачальнику. Функція дозволяє внести дату повернення товарів постачальну, постачальника, якому повертається товар, сума повернення, назва товару, одиниці виміру, кількість товару, ціна кожного товару окремо.

Повернення товару постачальнику звична річ у торгівлі, наприклад, у продуктових магазинах зазвичай, у більшості випадків, товар у яких вийшов термін придатності йде на повернення постачальнику. У торгівлі не продуктовими товари, наприклад електронікою, часто бувають випадки браку, тоді цей товар повертається постачальнику.

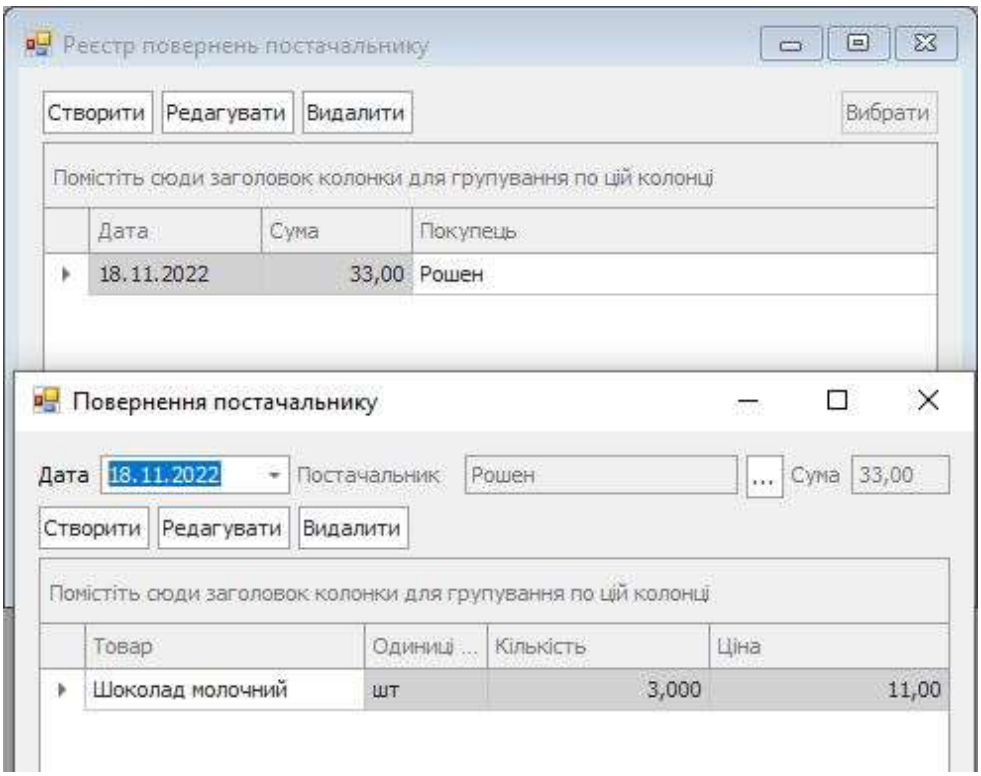



Рисунок 3.19 – Вікно повернення товарів постачальнику

Дана функція зображена на рисунку 3.19 відмінусовує товари із залишків на складі та створює запис в реєстрі з поверненням.

### 3.7 Реалізації реєстру прибуткових, видаткових касових ордерів та руху грошових коштів

Торгівля товарами завжди супроводжується рухом коштів, прибуткових, коли гроші добавляються у касу, та видаткових, коли гроші забираються з каси, тому є важливим контроль за цими процесами, для цьому був створений реєстр для цього. Дані реєстри показують такі дані як: дата, суму та партнер з яким йде обмін коштами. Дані реєстри дозволяються створити, редагувати, видаляти, а також вибирати конкретні позиції з реєстру касових орденів. З виглядом вікон реєстру прибуткових та видаткових ордерів та тестовими позиціями можна ознайомитись на рисунках 3.20 та 3.21.

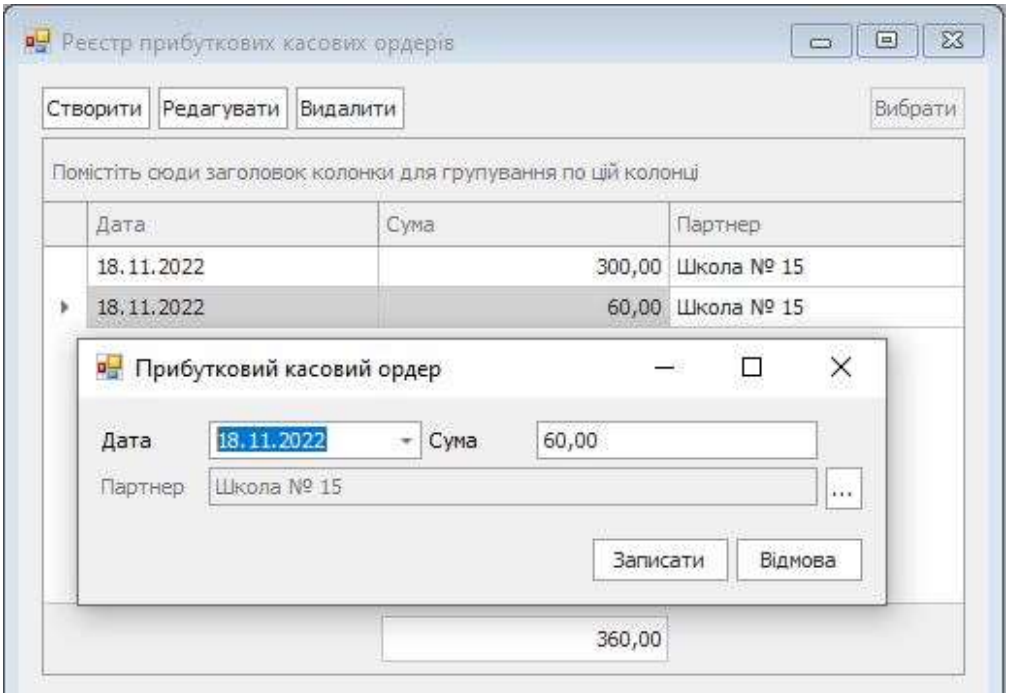



Рисунок 3.20 – Вікно реєстру прибуткового касового ордеру

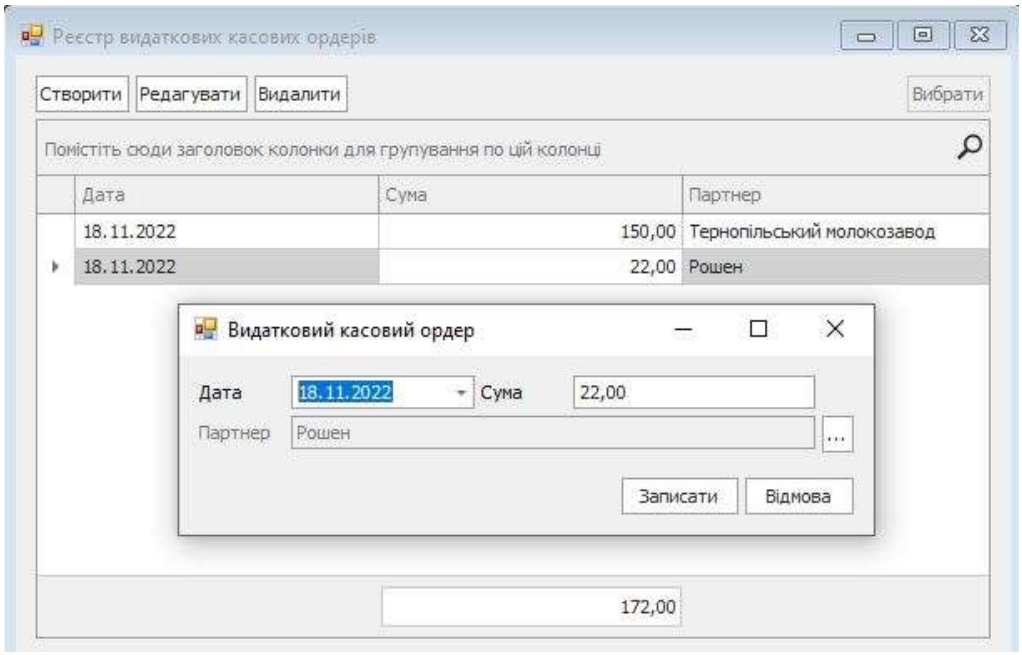



Рисунок 3.21 – Вікно реєстру видаткового касового ордеру

Як результат прибуткових та видаткових касових ордерів, отримуємо постійний рух коштів кожного дня. Тому було реалізовано вікно руху грошових коштів, де можна побачити дохід та розхід по кожному окремому

дню та маємо змогу вибрати потрібний період часу (рис. 3.22). Реалізацію даної функції можна побачити у лістингу 3.7.

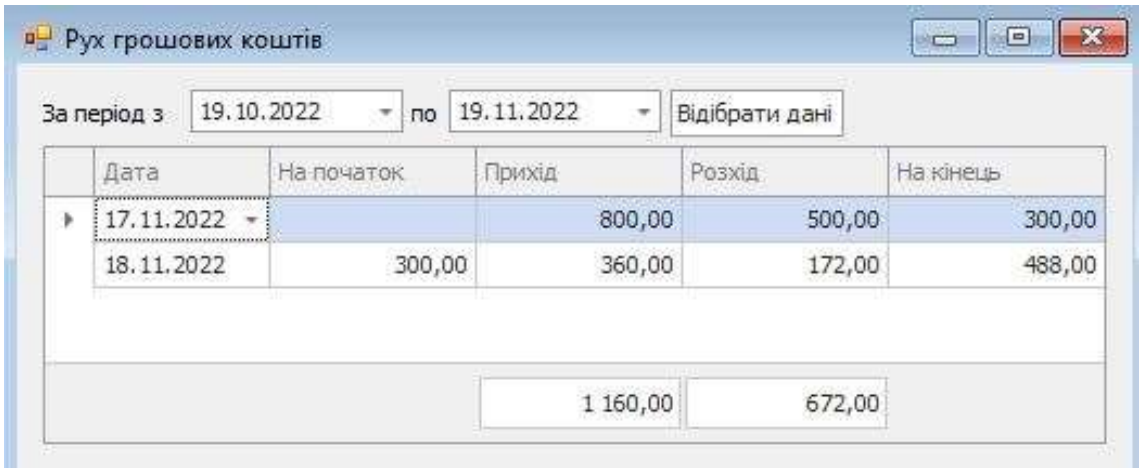



Рисунок 3.22 – Вікно руху грошових коштів

Лістинг 3.7 – Реалізації руху грошових коштів

```
declare @SumaBeg numeric (18,2)

     select @SumaBeg = sum(case when DocType = 4 then Suma else -
1*Suma end)
          from Documents
          where DocType in (4, 5) and DocDate < @date1
     if @SumaBeg is null select @SumaBeg = 0

     select dbo.setmintime(DocDate) as DocDate, SumaP = sum(Suma)
          into #tp
          from Documents
          where DocType = 4 -- Прихід
               and  dbo.setmintime(DocDate)  between  @date1  and
@date2
          group by dbo.setmintime(DocDate)

     select dbo.setmintime(DocDate) as DocDate, SumaR = sum(Suma)
          into #tr
          from Documents
          where DocType = 5 -- Розхід
               and  dbo.setmintime(DocDate)  between  @date1  and
@date2
          group by dbo.setmintime(DocDate)
     select    isnull(tp.DocDate,    tr.DocDate)    as    DocDate,
isnull(tp.SumaP, 0) as SumaP, isnull(tr.SumaR, 0) as SumaR
          into #t1
          from #tp tp full join #tr tr on tp.DocDate = tr.DocDate
```

```
     alter table #t1 add
               SumaBeg numeric(18,2) not null default 0,
               SumaEnd numeric(18,2) not null default 0
     ;WITH q AS ( SELECT TOP 100000 *
                          FROM #t1
                          order by DocDate
)

          update q set @SumaBeg = SumaEnd = @SumaBeg + SumaP -
SumaR,

                        SumaBeg = @SumaBeg - SumaP + SumaR
     select * from #t1 order by DocDate
```

Як результат створення даної функції в програмному забезпеченні для обліку реалізації товарів в торгівлі маємо змогу контролювати рух грошових коштів, який відбувається у торговця, що є важливим процесом у обліку реалізації товарів.

### 3.8 Реалізація реєстру боргів постачальникам та покупців

Зазвичай постачальники товарів та продукції, які часто працюють з торговою компанію не проводять грошові розрахунки зразу після отримання товару, а також можуть надавати товари в розстрочку. Тому було створено реєстр боргів постачальникам. У ситуації з покупцями, бувають випадки, коли відбувається продаж із післяплатою з відправкою поштою, коли фактично товар проданий, а грошові кошти не отримані. Також бувають випадки, коли систематичні покупці, які закуповують товар виплачують грошові кошти наперед. Саме тому створено реєстр боргів покупців. Функція дозволяє вести контроль за боргами в торгівлі перед постачальниками чи клієнтами. Програмне забезпечення дозволяє бачити назву постачальника чи клієнта, який заборгував гроші, суму боргу, а також загальну суму боргу. Ще було реалізовано пошук по реєстру боргів. З виглядом цих реєстрів можна ознайомитись на рисунках 3.23 та 3.24. З програмний кодом у лістингу 3.8.

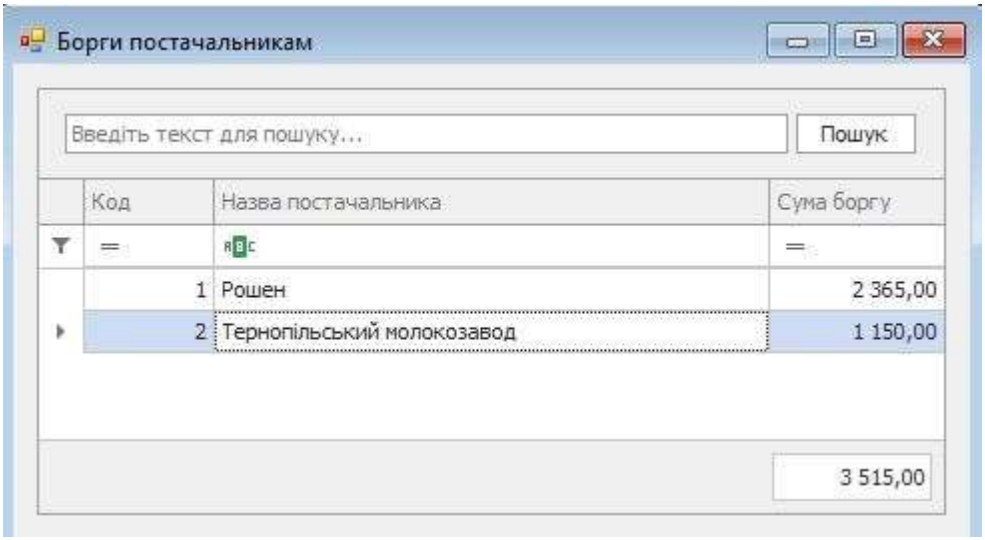



Рисунок 3.23 – Вікно боргів постачальникам

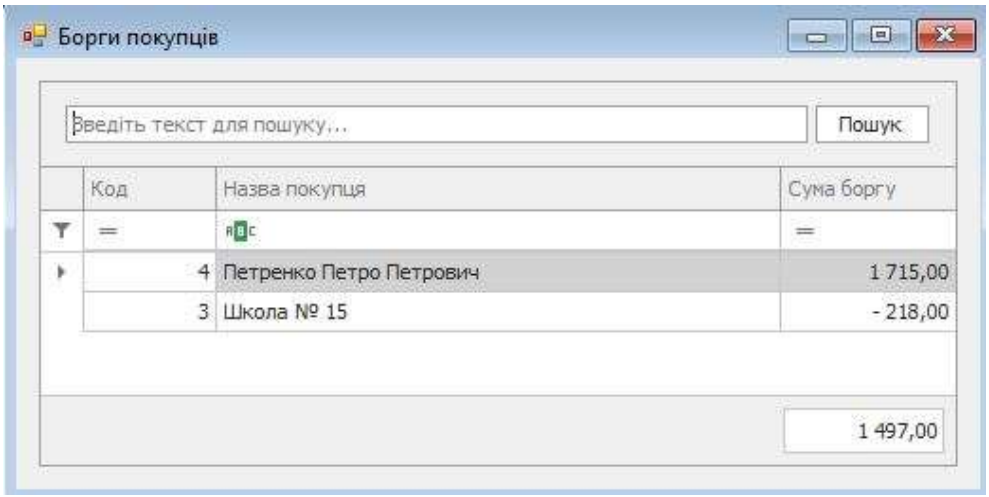



Рисунок 3.24 – Вікно боргів покупців

Лістинг 3.8 – Реалізація боргів постачальникам та покупців

```
GO
CREATE PROCEDURE [dbo].[SelectBorgPok]
AS
BEGIN
     SET NOCOUNT ON;
     select ClientId, Suma = sum(case when DocType = 2 then Suma
else -1*Suma end)
          into #tmpBorg
          from Documents
          where DocType in (2, 4)
          group by ClientId

     select isnull(t2.ClientName, '') as ClientName, t1.*
```

```
          from #tmpBorg t1 left join Clients t2 on t1.ClientId =
t2.ClientId
          order by 1
     drop table #tmpBorg

END
GO
CREATE PROCEDURE [dbo].[SelectBorgPost]
AS
BEGIN
     SET NOCOUNT ON;
     select ClientId, Suma = sum(case when DocType = 1 then Suma
else -1*Suma end)
          into #tmpBorg
          from Documents
          where DocType in (1, 3, 5)
          group by ClientId

     select isnull(t2.ClientName, '') as ClientName, t1.*
          from #tmpBorg t1 left join Clients t2 on t1.ClientId =
t2.ClientId
          order by 1
     drop table #tmpBorg

END
```

Реєстр боргів є важливим процесом для торгової компанії в обліку реалізації товарів в торгівлі, потрібно вести контроль боргів. Борги перед клієнтами та постачальниками є поширеним явищем, особливо серед магазинів роздрібної торгівлі.

### 3.9 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи описано розроблене програмне забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі.

Спроєктовано архітектуру програмного забезпечення. Подано опис функцій програмного забезпечення та як було реалізовано дані функції. Зокрема, реалізовані такі функції: довідник клієнтів, довідник товарів, прихід товарів, реалізація товарів, повернення товарів постачальну, прибуткові

касові ордера, видаткові касові ордера, поточні залишки товарів, звіт по реалізації товарів, замовлення товарів, борги постачальникам, борги покупців, рух грошових коштів.

В результаті отримано програмне забезпечення, яке повністю переводить облік товару в торгівлі у електронну форму та автоматизує відповідні процеси. Використання цифрових технологій та автоматизація процесів є важливими кроками для кожного бізнесу, який планує розвиватись.

В розділі подана лише найважливіша частина програмного коду, повний лістинг серверної частини, яка є основною в створеному програмному забезпечені знаходиться у додатку Б. Частина основних функцій клієнтської частини програмного забезпечення знаходиться у додатку В, так як інша частина реалізована схожим чином.

# 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## 4.1 Організація робочого місця користувача комп'ютера стосовно нормування площі та об'єму приміщення, необхідних для розташування робочих місць.

Тема кваліфікаційної роботи освітнього рівня «Магістр» присвячена розробці програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі, в якій кінцевий користувач користується програмою за допомогою персонального комп'ютера. Організація робочого місця користувача нормується законами. Кожен має право на належні, безпечні і здорові умови праці. Це гарантує Конституція України (ч. 4 ст. 43). відповідності до вимог ст. 153 Кодексу законів про працю України та ст. 6 Закону України «Про охорону праці» [57] на всіх підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці. Забезпечення безпечних і нешкідливих умов праці покладається на власника або уповноважений ним орган.

Згідно з частиною 1 статті 13 Закону України «Про охорону праці» [57] роботодавець зобов'язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці. Робочі місця офісних працівників, обладнані персональними комп'ютерами повинні відповідати вимогам «Правил охорони праці під час експлуатації електронно-обчислювальних машин», затверджених Наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 року № 65 та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно–обчислювальних машин», затверджених постановою Головного державного санітарного лікаря України.

Будівлі та приміщення, де розміщені робочі місця, повинні відповідати вимогам нормативно-технічної та експлуатаційної документації виробника персональних комп'ютерів. Будівлі та приміщення, де розміщені робочі місця операторів, мають бути не нижче другого ступеня вогнестійкості. Для всіх будівель і приміщень, де знаходяться робочі місця, повинно бути визначено клас зони згідно з НПАОП 40.1-1.01-97 [58]. Відповідне позначення повинно бути нанесено на вхідних дверях кожного приміщення. Не дозволяється розташування приміщень з робочими місцями у підвалах і цокольних поверхах. Неприпустимим є розташування приміщень категорій А і Б, а також виробництв з мокрими технологічними процесами поряд з приміщеннями, де розташовуються робочі місця, а також над ними чи під ними. При цьому площа приміщення має бути не менше 6,0 кв. м. із розрахунку на одне робоче місце, а об'єм – не менше 20,0 куб. м.

Віконні прорізи приміщень для роботи з персональними комп'ютерами мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки. Для внутрішнього оздоблення приміщень з персональними комп'ютерами слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5. Поверхня підлоги має бути рівною, неслизькою, з антистатичними властивостями. Забороняється для оздоблення інтер'єру приміщень з персональними комп'ютерами застосовувати полімерні матеріали (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини.

При розміщенні робочих столів з персональними комп'ютерами слід дотримувати відстань між бічними поверхнями персональних комп'ютерів 1,2 м, а відстань від тильної поверхні одного персонального комп'ютера до екрана іншого – 2,5 м.

За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2 м. Конструкція робочого місця користувача персонального комп'ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має регулюватися в межах 680–800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600–1400 мм, глибина – 800–1000 мм).

Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим. Регулювання за кожним із параметрів має здійснюватися незалежно, легко і надійно фіксуватися. Шаг регулювання елементів стільця має становити: для лінійних розмірів – 15-20мм, для кутових – 2-5 градусів. Зусилля регулювання має не перевищувати 20Н. Висота поверхні сидіння має регулюватися в межах 400-500мм, а ширина і глибина становити не менше ніж 400мм. Кут нахилу сидіння – до 15 градусів вперед і до 5 градусів назад. Висота спинки стільця має становити (300+-20) мм, ширина не менше ніж 380 мм, радіус кривизни горизонтальної площини – 400мм. Кут нахилу спинки має регулюватися в межах 1–30 градусів від вертикального положення. Відстань від спинки до переднього краю сидіння має регулюватися в межах 260–400мм. Для зниження статичного напруження м'язів верхніх кінцівок слід використовувати стаціонарні або змінні

підлокітники завдовжки не менше ніж 250мм, завширшки 50–70мм, що регулюються за висотою над сидінням у межах 230–260мм і відстанню між підлокітниками в межах 350–500мм.

Поверхня сидіння і спинки стільця має бути напівм'якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується. Робоче місце має бути обладнане підставкою для ніг завширшки не менше ніж 300мм, завглибшки не менше ніж 400мм, що регулюється за висотою в межах до 150мм і за кутом нахилу опорної поверхні підставки до 20 градусів. Підставка повинна мати рифлену поверхню і бортик по передньому краю заввишки 10мм. Робочі місця слід розташовувати відносно світових прорізів так, щоб природне світло падало переважно з лівого боку.

Монітор має розташовуватися на оптимальній відстані від очей користувача, що становить 600–700мм, але не ближче ніж за 600мм з урахуванням розміру літерно-цифрових знаків і символів. Розташування екрана монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 градусів до нормальної лінії погляду працівника. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, звернутого до працюючого.

У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-15 градусів. Висота середнього рядка клавіш має не перевищувати 30мм. Поверхня клавіатури має бути матовою з коефіцієнтом відбиття 0,4. Розташування пристрою введення – виведення інформації має забезпечувати добру видимість монітору, зручність ручного керування в зоні досяжності моторного поля і за висотою – 900–1300мм, за шириною 400–500мм.

## 4.2 Підвищення стійкості роботи об’єктів торгівлі у воєнний час

Одним із основних завдань цивільної оборони під час воєнного часу є підвищення стійкості роботи об'єктів торгівлі, оскільки торгівля є одним з найпоширеніших та найприбутковіших сфер для економіки України. Під стійкістю об’єкта торгівлі в цілому розуміється його здатність в умовах надзвичайних ситуацій мирного і воєнного часу забезпечувати всі галузі необхідною промисловою і сільськогосподарською продукцією.

Під стійкістю роботи об’єкта торгівлі розуміють здатність його в умовах надзвичайних ситуацій мирного і воєнного часу випускати продукцію в запланованому обсязі, а при одержанні слабких чи середніх руйнувань, порушенні зв’язків по кооперації та постачанням відновлювати виробництво в мінімальний термін.

Здатність об’єкта торгівлі випускати чи продавати продукцію залежить від захисту та нормального функціонування чотирьох основних елементів сучасного виробництва, якими є:

- виробничий персонал;
- система постачання енергією, водою, паливом, устаткуванням та ремонтною базою;
- система виробничих та кооперативних зв’язків з іншими об’єктами;
- будинки і споруди із технологічним устаткуванням.

Для підвищення стійкості роботи на кожному об'єкті торгівлі необхідно завчасно організовувати та провести великий обсяг робіт, спрямованих на підвищення стійкості його роботи в умовах застосування зброї масового ураження (ракет, дронів-камікадзе тощо). До таких робіт відносяться організаційні заходи, технологічні та інженерно-технічні.

Організаційними заходами передбачається завчасна розробка та планування дій особового складу торгової компанії, служб та формувань

цивільної оборони об'єкта в умовах застосування противником зброї масового ураження.

Технологічними заходами здійснюється підвищення стійкості шляхом зміни технологічного режиму, що виключає можливість виникнення вторинних уражаючих факторів, викликаних впливом різного виду зброї.

Інженерно-технічними заходами забезпечується підвищення стійкості промислових будівель, споруд, обладнання та комунікацій підприємства до впливу вражаючих факторів.

З усього комплексу заходів, які підвищують стійку роботу об'єктів економіки у воєнний час, особливо важливе значення має саме проведення інженерно-технічних заходів.

До таких заходів належать:

- підвищення стійкості постачання електроенергією, газом, парою, водою і роботою мереж комунального господарства;
- захист об'єктів від пожеж та інших вторинних факторів ураження;
- забезпечення захисту робітників і службовців від зброї масового ураження;
- підготовка до відновлення порушеного виробництва;
- підвищення стійкості управління об'єкту;
- захист устаткування;
- підвищення стійкості матеріально-технічного постачання.

Розглянемо детальніше кілька заходів забезпечення стійкості детальніше. Отож, забезпечення захисту робітників від зброї масового ураження, основним способом захисту робітників підприємства торгівлі є створення або організація укриття у захисних спорудах, це може бути сховище або укриття. Для захисту персоналу, який обслуговує агрегати, зупинка яких внаслідок особливості процесу виробництва торгової продукції неможлива навіть при оголошенні сигналу повітряної тривоги, в таких

випадках доцільно зводити спеціальні захисні споруди. Для захисту зміни, яка відпочиває у заміській зоні з виникненням загрози атаки противника будуються протирадіаційні укриття. Будівництво їх зазвичай планується у мирний час.

Підвищення стійкості управління об'єктом, під цим заходом мається на увазі управління об’єктом торгівлі, яке складає основу діяльності начальника цивільної оборони об'єкта та його штабу, полягає в здійсненні постійного керівництва робітниками, формуваннями цивільної оборони об'єкта на всіх етапах ведення діяльності. У цих умовах повинна бути розроблена схема оповіщення та зв'язку, яка є складовою частиною загального плану цивільної оборони об'єкта. Управління має бути постійним на всіх етапах: при загрозі нападу, в умовах проведення розосередження і евакуації, так і при проведенні рятувальних та інших невідкладних робіт.

На важливих об'єктах торгівлі при загрозі атаки противника створюються дві групи управління: одна безпосередньо на підприємстві, друга в заміській зоні, в районі розосередження робітників.

Обладнання на об’єктах торгівлі не є дешевим: сервери, комп’ютери, монітори, безперебійних, різного роду периферія, принтери. Тому необхідно надійно захистити обладнання від впливу ударної хвилі, захистити все обладнання практично неможливо. Завдання захисту обладнання полягає в тому, щоб звести до мінімуму небезпеку руйнування і пошкодження обладнання торгової компанії. Захист обладнання та готової продукції може здійснюватися шляхом розміщення деяких видів найціннішого обладнання в заглиблених приміщеннях і використання для цього захисних пристроїв, таких як: шатри, камери, кожухи, парасольки тощо. Крім застосування захисних пристроїв велике значення має міцне кріплення робочих поверхнонь на фундаментах, які підвищують їх стійкість до перекидання.

Велика більшість об’єктів в торгівлі є критично залежна від стабільного електропостачання, також важливим для об’єктів даного типу є

газопостачання, так як це гарантує комфорт працівниів та підвищує їх роботоспроможність. Підвищення стійкості системи електропостачання досягається базуванням підприємства на двох і більше джерелах, віддалених на таку відстань, щоб виключалася можливість руйнування їх ударом по одній і же тій точці чи поблизу неї. При відсутності можливості отримання електроенергії від двох джерел на випадок виходу з ладу основного джерела електропостачання готується резервний автономне джерело, це може бути генератор, акумулятори чи блоки безперебійного живлення. Доцільно також провести заходи щодо захисту існуючих, а також будівництва резервних підстанцій, а розподільну апаратуру і прилади розмістити в захисних спорудах. Електропостачання найкраще перевести з повітряної на підземно-кабельне. Для запобігання виходу з ладу електричних мереж слід встановлювати пристрої автоматичного відключення їх при утворенні перенапруг, які можуть бути створені електромагнітними полями, що виникають при стрибках напруги.

На багатьох об'єктах торгівлі газ може використовуватися для виробництва продукції, але здебільшого в роздрібній торгівлі газ потрібен для опалювання приміщень, що підвищує комфорт працівників, а відповідно їх продуктивність. При руйнуванні газових мереж, газ може бути причиною вторинних вражаючих факторів. На випадок пошкодження джерел газопостачання на великих підприємствах рекомендується мати підземні ємності.. Газ під великим тиском закачується в підземні ємності та служить резервом. Крім того, необхідно готувати підприємство до роботи на різних видах палива та створювати їхні запаси. На газопроводах слід встановити запірну арматуру та крани з дистанційним управлінням, це дозволяє автоматично перемикати потік газу при розриві труби.

Також на об’єкті торгівлі слід зайнятись підвищення стійкості мереж комунального господарства. Теплову мережу доцільно будувати з використанням кільцевої системи та прокладати труби опалювальної системи

в спеціальних каналах під землею. Для підвищення стійкості системи каналізації слід будувати окремі системи каналізації: одна для зливових, інша для промислових і господарських вод.

На об’єктах торгівлі організовується також захист об'єктів від вторинних факторів ураження. Для захисту об'єктів від вторинних факторів ураження передбачаються наступні заходи:

- якщо торгівля відбувається вибухонебезпечною продукцію, організовується будівництво сховищ для нафти, бензину, зрідженого газу, мазуту, отрутохімікатів, обов’язково за межами території об'єкта у безпечному місці;
- підвищення вогнестійкості дерев'яних конструкцій (вогнезахисне фарбування, побілка тощо);
- спорудження водоймищ для гасіння пожеж.

Для стійкості об’єкта торгівлі важливе підвищення матеріально-технічного постачання об'єкта. Щоб виробництво велося безперебійно, потрібно забезпечити його потрібною продукцією, це може бути сировина, різного роду матеріали, паливо, електроенергія, інструменти. Резервний запас всіх матеріалів повинен зберігатися розосереджено в місцях, де він менше всього може піддатися знищення під час атаки противника. Об'єкт повинен підготуватися до роботи в різних видах палива.

Для запезпечення стійкості об’єкта торгівлі є важливим підготовка до відновлення порушеного виробництва. Для кожного варіанту повинен бути розроблений план можливого ураження, розробляється план відновлення об'єкта. При цьому складаються розрахунки потрібних матеріалів, механізмів та сил. В основу планів та проектів відновлення має бути закладена вимога – якомога швидше відновити випуск продукції. Тому в проектах відновлення допустимі відступи від прийнятих технічних, будівельних, чи інших норм.

## 4.3 Висновок до четвертого розділу

Будь-яке електроустаткування вимагає особливої уваги через те, що в цих пристроях міститься безліч різних елементів контактування з якими може серйозно травмувати людину. З цієї причини є дуже важливим дотримуватися правил техніки безпеки під час експлуатації електрообладнання.

В даному розділі було описано як потрібно організовувати робоче місця користувача комп'ютера стосовно нормування площі та об'єму приміщення, необхідних для розташування робочих місць, а також описано як забезпечити підвищення стійкості роботи об'єктів торгівлі у воєнний час.

Значення економіки в забезпеченні обороноздатності країни є важливим фактором, а об'єкти торгівлі є одним з найбільших платників податків, також вони займаються закупівлею та реалізацією торгівельної продукції, яка необхідна для обороноздатності країни. Так як для кожного підприємства існує реальна загроза руйнування із застосування сучасних засобів атаки, основна задача у воєнний час є підвищення стабільної роботи об'єктів торгівлі у воєнний час. Виконання цих задача покладається на сили цивільної захисту, а підвищення стійкості роботи досягається проведенням ряду технічних, організаційних та інженерно-технічних заходів, що враховують вимоги цивільного захисту та безпеки життєдіяльності. Оцінка стійкості роботи об'єкта спрямована на забезпечення безперебійності виробничого процесу у надзвичайній ситуації та допомагає досягти максимального зниження можливих втрат чи руйнувань.

Інформування користувачів програмного забезпечення на рахунок охорони праці та безпеки в надзвичаних ситуаціях є важливим, оскільки працівник повинен знати та дотримуватися техніки безпеки, щоб уникнути небажаних ситуацій.

# ВИСНОВКИ

В першому розділі кваліфікаційної роботи:

– описано ринок торгівлі в Україні та світі;

– висвітлено потреби програмного забезпечення для обліку реалізації товарів;

– розглянуто відомі програмні рішення для обліку реалізації товарів в торгівлі;

– проаналізовано публікації в області торгівлі товарами.

В другому розділі кваліфікаційної роботи:

– проаналізовано бази даних, клієнт–серверну архітектуру, мову програмування C# та середовище розробки Microsoft Visual Studio;

– наведено обґрунтування вибору інструментальних засобів та технологій реалізації програмного забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі.

В третьому розділі кваліфікаційної роботи:

– описано створене програмне забезпечення на основі клієнт–серверної архітектури для обліку реалізації товарів в торгівлі;

– наведено реалізацію взаємодії клієнта з сервером;

– наведено огляд реалізованих класів клієнтської частини;

– подано детальний аналіз реалізованих функцій програмного забезпечення на основі клієнт–серверної архітектури для обліку реалізації товарів в торгівлі.

– описано створені класи, поля та методи цих класів.

– описано створені таблиці в СУБД, їх призначення та поля, подано рисунок таблиць та зв’язків між ними.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» проаналізовано організацію робочого місця користувача комп’ютера, описано засоби підвищення стійкості роботи об’єктів торгівлі у воєнний час.

# ПЕРЕЛІК ДЖЕРЕЛ

1. Силкіна Ю. О. Проблеми та перспективи розвитку ринку роздрібних торговельних мереж в Україні / Ю. О. Силкіна. – Київ: Ранок, 2017. – 150-152 с.

2. Мєшкова В. В. Статистичний аналіз ринку роздрібної торгівлі України / В. В. Мєшкова, Н. С. Пащенко. – Львів: Наш формат, 2018. – 148 с.

3. Карнаушенко А. С. Проблеми автоматизації обліку торгових підприємств / Алла Сергіївна Карнаушенко. – Харків: Комора, 2019. – 120 с.

4. Бойчук А. А. Проблеми організації автоматизації обліку на підприємствах торгівлі / А. А. Бойчук, В. О. Бойчук, Л. О. Моцна. – Київ: Фабула, 2017. – 49 с.

5. Безус А. М. Перспективи інноваційного розвитку роздрібної торгівлі в Україні / А. М. Безус, Б. М. Шевчун, П. І. Безус. – Полтава: Астра, 2019. – 19-21 с.

6. Карнаушенко А. С. Особливості автоматизації обліку торговельних підприємств / Алла Сергіївна Карнаушенко. – Дніпро: Піраміда, 2019. – 76 с.

7. Дмитрів О. Р. Цифровізація економіки та автоматизація виробництва: проблеми та шляхи їх вирішення." Тези доповідей міжнародної науково-практичної конференції „Цифрова економіка як фактор інноваційного розвитку суспільства / О. Р. Дмитрів, В. В. Семеген. – Київ: Фоліо, 2020. – 12 с.

8. Завгородній О. Ю. Системи та програми автоматизації на підприємствах торгівлі: вимоги до функціоналу та їх реалізація / О. Ю. Завгородній. – Харків: Теза, 2021. – 190 с.

9. Демянова Ю. О. Автоматизовані системи управління як невід’ємний елемент розвитку глобальних торговельно-роздрібних мереж / Ю. О. Демянова. – Київ: Човен, 2018. – 168-169 с.

10. Сайт ISpro [Електронний ресурс] – Режим доступу до ресурсу: https://ispro.ua/.

11. Сайт A5 [Електронний ресурс] – Режим доступу до ресурсу: https://a5erp.solutions/.

12. Сайт Dilovod [Електронний ресурс] – Режим доступу до ресурсу: https://dilovod.ua/ru/

13. Сайт Master: Бухгалтерія [Електронний ресурс] – Режим доступу до ресурсу: https://masterbuh.com/.

14. Сайт Дебет Плюс [Електронний ресурс] – Режим доступу до ресурсу: https://debet.com.ua/.

15. Сайт HugeProfit [Електронний ресурс] – Режим доступу до ресурсу: https://h-profit.com/.

16. Husni M. Improved Information Retrieval Performance on SQL Database Using Data Adapter / M. Husni. – Madrid: Grupo Planeta, 2018. – 127 c. – (Materials Science and Engineering).

17. Noaman A. eCloudDB: A Unified API for Secure SQL and NoSQL Cloud Databases. / A. Noaman, E. Noaman, A. Noaman. – Washington: Wiley, 2019. – 124 c. – (Proceedings of the 2019 3rd International Conference on Cloud and Big Data Computing).

18. Mukherjee S. Indexes in Microsoft SQL Server / Sourav Mukherjee. – New-York: HarperCollins, 2019. – 53 c.

19. Efficiency test of Microsoft SQL Server 2016 / T.Balla, T. Radvanyi, S. Kiraly, R. Kiraly. –Amsterdam: Wolters Kluwer, 2018. – 192 c.

20. Dejan S. SQL Server 2016 Developer's Guide / S. Dejan, M. Radivojevic, W. Durkin. – Zagreb: Cornelsen, 2017. – 77 c.

21. William A. SQL server 2017 administration inside out / Assaf William. – Washington: RELX Group, 2018. – 36 c.

22. Mukherjee S. Popular SQL server database encryption choices / Sourav Mukherjee. – New-York: HarperCollins, 2019. – 37 c.

23. Chmel M. SQL Server 2017 Administrator's Guide: One stop solution for DBAs to monitor, manage, and maintain enterprise databases / M. Chmel, V. Muzny. – Warsaw: Klett, 2017. – 52 с.

24. Mukherjee S. SQL Server Development Best Practices / Sourav Mukherjee. – New-York: HarperCollins, 2019. – 41 с.

25. Johnson B. Using Git in Visual Studio 2019 / Bruce Johnson. – New-York: Apress, 2020. – 141 с.

26. Alian M. Ncap: Network-driven, packet context-aware power management for client-server architecture / Mohammad Alian. – Washington: Scholastic, 2017. – 99 с.

27. Посвістак В.С., Демківська Т.І. Клієнт-серверна архітектура та її використання при розробці програмного забезпечення. / Інформаційні технології в науці, виробництві та підприємництві – 2020. – 14 с.

28. Висоцький А.В. Поняття клієнт-серверної архітектури. / Тези доп. міжнародної науково-практичної інтернет-конференції. – Дніпро, 2020. – 8 с.

29. Михайленко А.М. Клієнт-серверна система управління задачами. / КПІ ім. Ігоря Сікорського. – Київ, 2019. – 7 с.

30. Войтко В., Денисюк П.. Особливості розробки серверних додатків клієнт-серверної архітектури. – 2017. – 44 с.

31. Божок Р.Ю. Клієнт–серверна система підтримки учбового процесу. / КПІ ім. Ігоря Сікорського. – Київ, 2020. – 55 с.

32. Geary Nigel. The client-server architecture in a mixed database environment. / Data Distribution: Managing the Environment. – Routledge, 2018. – 77 с.

33. Langa S. Management aspects of client/server computing. – 2019. – 13 с.

34. Kumar Santosh. A Review on Client-Server based applications and research opportunity. / International Journal of Recent Scientific Research. – 2019. – 24 с.

35. Madalina Eleonora. Providing Security for Client-Server Applications. – 2019. – 8 с.

36. Rabin Steven. Developing Workstation-Based Client/Server Applications. – Auerbach Publications, 2018. – 44 с.

37. Sahoo Niranjan, Rajashree Shukla. Use of Client Monitoring Client-Server Data Security and Health Management. – 2019. – 25 с.

38. Romanov E.L., Troshina G.V., Menzhulin S.A. Client-server application framework based on an object-oriented network model. – IOP Publishing, 2020. – 35 с.

39. Радченко Д. Ю., Колодний В.В.. Технологія та моделі архітектури клієнт-серверної взаємодії. – ВНТУ, 2019. – 17 с.

40. Висоцький А.В. Поняття клієнт-серверної архітектури. – Дніпро, 2020. – 98 с.

41. Вакуленко С.. Розробка клієнт-серверного застосунку на C++. – 2021. – 5 с.

42. Снопок А.І. Багатопотоковий клієнт-серверний додаток. / КПІ ім. Ігоря Сікорського. – Київ, 2020. – 29 с.

43. Sharp John. Microsoft visual C# step by step. – Microsoft Press, 2018. – 33 с.

44. Bouna Praveenkumar. Visual Studio Code for C# Developers. – 2022. – 11 с.

45. Ilic Milos. The difference between ADO .NET and Entity Framework in software. – 2019. – 14 с.

46. Strauss D. Working with Visual Studio 2019. Getting Started with Visual Studio 2019. / Dirk Strauss. – New-York: Apress, 2020. – 201-202 с.

47. Snell P. Microsoft Visual Studio 2015 / P. Snell, L. Snell, M. Snell. – New-York: HarperCollins, 2019. – 109 с.

48. Bhandari P. Consuming Microsoft Cognitive APIs. / P. Bhandari, N. Bhandari, A. Bhandari. – Berkeley: Californial, 2018. – 116 с.

49. Development of Dissolved Gas Analysis Analyzing Program using Visual Studio Program. / P.Kunagonniyomrattana, P. Kunagonniyomrattana, T. Kunagonniyomrattana, C. Supakit. – Tokyo: Shuesisha, 2019. – 178 с.

50. Buananno E. Functional programming in C#. Manning / Enrico Buananno. – San-Jose: HarperCollins, 2018. – 111 с.

51. Kunal C. Mastering Visual Studio 2017 / Clowdhury Kunal. – Los-Angeles: RELX Group, 2017. – 69 с

52. Sharp J. Microsoft Visual C# 2013 Step by Step. Pearson Education / John Sharp. – Boston: McGraw, 2017. – 41 с.

53. Schrotenboer S. Illustrated C# 7: The C# Language Presented Clearly, Concisely, and Visually / S. Schrotenboer, D. Schrotenboer, C. Schrotenboer. – New-York: Apress, 2018. – 108 с.

54. Guerin B. ASP. NET con C# en Visual Studio 2017: diseño y desarrollo de aplicaciones Web / Brice-Arnaud Guerin. – Madrid: Grupo Santillana, 2018. – 50 с.

55. Watts G. Developing a Declarative Analysis Language: LINQToROOT. / Gordon Watts. – Washington: Apress, 2019. – 21 с. – (EPJ Web of Conferences).

56. Muller, Nathan J. Client-server Architecture and Implementation. - Auberbach Publication, 2020. – 186 с.

57. Кодекс законів про працю України [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/322-08.

58. Про затвердження Правил безпечної експлуатації електроустановок [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0011-98.

# ДОДАТКИ

**Додаток А**

**Тези конференцій**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

**М А Т Е Р І А Л И**

**X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ**

# «ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

**7–8 грудня 2022 року**

**ТЕРНОПІЛЬ**
**2022**

УДК 004.4
**І. Ралік**
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

# ЗАДАЧА РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ РЕАЛІЗАЦІЇ ТОВАРІВ В ТОРГІВЛІ

UDC 004.4
**I. Ralik**

# THE TASK OF SOFTWARE DEVELOPMENT FOR THE GOODS SALE ACCOUNTING IN RETAIL

Актуальність розробки програмного забезпечення для обліку реалізації товарів зумовлена тим, що на сьогоднішній день галузь торгівлі є однією із найбільш розвинених в економіці України. Кількість компаній, які займаються торгівлею товарами, збільшується з кожним роком. Разом із цим зростає потреба в обліку реалізації товарів в торгівлі у електронному форматі, оскільки це дає змогу автоматизувати та пришвидшити значну кількість процесів.

До відомих програмних рішень в даній предметній області належать: 1С, ISpro, A5, Master: Бухгалтерія, Dilovod. Дебет Плюс, HugeProfit. Загальною їх характеристикою є те, що вони пропонують широкий набір функціональних можливостей, які підійдуть в основному представникам середнього та великого бізнесу. Компанії такого масштабу мають можливість витрачати більше коштів на закупівлю відповідного програмного забезпечення та зробити більше інвестицій в навчання персоналу.

На сьогоднішній день внаслідок сукупності декількох причин використання програмних продуктів сімейства 1С є обмеженим. Однак, фактично багато компаній продовжують нею користуватися. Тому є важливим створення якісного українського продукту аналогічного призначення.

Для розробки програмного рішення обліку реалізації товарів в торгівлі пропонується обрати мову програмування C# та СУБД Microsoft SQL Server. Дане поєднання технологій вже зарекомендувало себе з позитивної сторони чином при реалізації проектів різного призначення та масштабів. На мові C# буде розроблено клієнтське програмне забезпечення, тоді як SQL Server відповідатиме за організацію роботи із даними.

Проект пропонується реалізувати на основі клієнт-серверної архітектури. Основними перевагами даної архітектури є безпека, централізований доступ до даних та простота [1, 2]. В склад програмного забезпечення входитимуть:

1) сервер, на якому будуть оброблятись запити до бази даних та проводитимуться обчислення.

2) клієнтське програмне забезпечення (тонкий клієнт), яке дозволить вносити та отримувати дані з сервера.

Пропоноване програмне забезпечення повинне забезпечити реалізацію наступних функцій: довідник клієнтів, довідник товарів, облік приходу товарів, облік реалізації товарів, повернення товарів постачальну, прибуткові касові ордера, видаткові касові ордера, поточні залишки товарів, звіт по реалізації товарів, замовлення товарів, борги постачальникам, борги покупців, рух грошових коштів тощо.

**Література**

1. Nathan Muller. Client-server Architecture and Implementation. Auberbach Publication. 2020. P. 189–190.
2. J.D. Meier, David Hill, Alex Homer, Jason Taylor, Prashant Bansode, Lonnie Wall, Rob Boucher Jr., Akshay Bogawat. «Руководство Microsoft по проектированию архитектуры приложений». 2009.

**Додаток Б**

**Тези конференцій**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Тернопільський національний технічний університет імені Івана Пулюя (Україна)**
**Університет імені П'єра і Марії Кюрі (Франція)**
**Маріборський університет (Словенія)**
**Технічний університет у Кошице (Словаччина )**
**Вільнюський технічний університет ім. Гедимінаса (Литва)**
**Міжнародний університет цивільної авіації (Марокко)**
**Наукове товариство ім. Т.Шевченка**

# АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

## Збірник
тез доповідей

**XI Міжнародної науково-практичної конференції молодих учених та студентів**
7-8 грудня 2022 року

**УКРАЇНА**
**ТЕРНОПІЛЬ – 2022**

**УДК 004.62**

**В.А. Готович, к.т.н., І.Р. Ралік**
Тернопільський національний технічний університет імені Івана Пулюя, Україна

## ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ ДЛЯ ОБЛІКУ РЕАЛІЗАЦІЇ ТОВАРІВ В ТОРГІВЛІ

**V.A. Hotovych, Ph.D., I.R. Ralik**

## SOFTWARE BASED ON CLIENT-SERVER ARCHITECTURE FOR ACCOUNTING FOR THE SALE OF GOODS IN RETAIL

На сьогоднішній день задачі обліку в торгівлі вирішуються за допомогою спеціалізованих апаратно-програмних засобів. Труднощі при використанні програмних засобів іноземного походження в Україні зумовлюють актуальність розробки відповідного вітчизняного програмного забезпечення (ПЗ).

Для вирішення задач обліку реалізації товарів в торгівлі пропонується проект ПЗ. Даний проект реалізовано на основі клієнт-серверної архітектури з наступною структурою [1]:

1) програма-клієнт, з графічним інтерфейсом (“десктоп”), яка реалізує функції “тонкого” клієнта;

2) база даних (БД), в таблицях якої зберігаються дані і в якій за допомогою збережених процедур реалізовано “товстий” сервер з усією бізнес-логікою. БД отримує запити від клієнта, обробляє їх та надсилає відповідь клієнту за допомогою локального мережевого з'єднання.

Пропонований програмний продукт розроблено на основі використання стеку технологій від Microsoft. Зокрема, використано платформу .Net, мову програмування C# та систему управління базами даних (СУБД) під управлінням SQL Server.

Програма-клієнт надає графічний інтерфейс, за допомогою якого відбувається введення даних і передача їх в базу даних. Інтерфейс програми-клієнта включає в себе 3 вкладки: Документи (функціонал по приходу товарів, реалізації товарів, повернення товарів постачальнику, прибуткові касові ордера, видаткові касові ордера), Довідники (функціонал по обліку клієнтів і товарів), та Звіти (функціонал по обліку поточних залишків товарів, звіти по реалізації товарів, замовлення товарів, борги постачальникам, борги покупців, рух грошових коштів).

Для взаємодії клієнта з базою даних використано архітектуру ADO.NET, основними компонентами якого для доступу до даних і управління ними є постачальники даних платформи .NET Framework та DataSet. Зв’язок між набором даних та базою даних реалізований через класи sqlDataAdapter, sqlCommand та sqlConnection. Для захисту даних використовуються системи захисту СУБД Microsoft SQL Server [1], а саме перевірка автентифікації SQL. Резервне копіювання відбувається стандартними засобами та методами СУБД. Один раз на добу створюється повна копія даних, кожних 15 хвилин створюється бекап транзакцій.

СУБД в архітектурі проекту слугує “товстим” сервером та є мозковим центром створеного програмного продукту. Значну роль у виконанні обчислень та формуванні звітів покладено на збережені процедури.

До перспектив подальших досліджень та розробок належать: розширення функціональних можливостей програмного продукту, автоматизація процесів тестування та розгортання, реалізація роботи із сканером штрих-кодів.

**Література**

1. Nathan Muller. Client-server Architecture and Implementation. - Auberbach Publication, 2020. - С. 192-193.

**Додаток В**

## Скрипт збережених процедур бази даних

```
USE [torg]
GO
/****** Object:  UserDefinedFunction [dbo].[setmintime] /
SET ANSI_NULLS OFF
GO
SET QUOTED_IDENTIFIER ON
GO
create FUNCTION [dbo].[setmintime] (@dat datetime)
RETURNS datetime AS
BEGIN

declare @str1 varchar(25), @date datetime
set                                          @str1=convert(char(2),
datepart(dd,@dat))+'.'+convert(char(2),datepart(mm,@dat))+'.'+co
nvert(char(4),datepart(yyyy,@dat)) +' 00:00:00.00'
set @date=convert(datetime, @str1, 104)

return @date

END

GO
/****** Object:  UserDefinedFunction [dbo].[ZalProducts]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION [dbo].[ZalProducts]()
RETURNS @RtnValue table ( ProductId int, Quantity numeric(18,3)
)

AS

BEGIN
     declare   @SaldAll   table   (   ProductId   int,   Quantity
numeric(18,3) )

     insert into @SaldAll ( ProductId, Quantity)
          select ProductId, Quantity = sum(Quantity)
               from DocDetails
               where DocId in (select DocId from Documents where
DocType = 1) -- Прихід товару
               group by ProductId

     insert into @SaldAll ( ProductId, Quantity)
          select ProductId, Quantity = -1*sum(Quantity)
               from DocDetails
```

```
               where DocId in (select DocId from Documents where
DocType = 2) -- Реалізація товару
               group by ProductId

     insert into @SaldAll ( ProductId, Quantity)
          select ProductId, Quantity = -1*sum(Quantity)
               from DocDetails
               where DocId in (select DocId from Documents where
DocType = 3) -- Повернення постачальнику
               group by ProductId

     insert into @RtnValue ( ProductId, Quantity)
          select ProductId, Quantity = sum(Quantity)
               from @SaldAll
               group by ProductId
     RETURN
END

GO
/****** Object:  StoredProcedure [dbo].[ClientsDelete]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[ClientsDelete]
@ClientId int

AS
BEGIN
     delete from Clients where ClientId = @ClientId
END
GO
/****** Object:  StoredProcedure [dbo].[ClientsSelectAll]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

create PROCEDURE [dbo].[ClientsSelectAll]

AS
BEGIN
     select * from Clients order by ClientName
END
GO
/****** Object:  StoredProcedure [dbo].[ClientsUpdate]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[ClientsUpdate]
```

```
@ClientId int output, @ClientName varchar(100), @TaxNumber
varchar(12), @ClientAddress varchar(200)

AS
BEGIN
if @ClientId = 0
          begin
               insert into Clients (TaxNumber, ClientAddress,
ClientName) values (@TaxNumber, @ClientAddress, @ClientName)
               select @ClientId = SCOPE_IDENTITY()
          end
     else
          begin
               update Clients set TaxNumber = @TaxNumber,
ClientAddress = @ClientAddress, ClientName = @ClientName,
          where ClientId = @ClientId
          end
GO
/****** Object:  StoredProcedure [dbo].[DocDetailsDelete]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[DocDetailsDelete]
@Id int

AS
BEGIN
     delete from DocDetails where Id = @Id
END
GO
/******            Object:                         StoredProcedure
[dbo].[DocDetailsSelectByDocId]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[DocDetailsSelectByDocId]
@DocId int

AS
BEGIN
     select t1.*, ProductName = isnull(t2.ProductName, ''), Units
= isnull(t2.Units, '')
          from DocDetails t1 left join Products t2 on
t1.ProductId = t2.ProductId
          where t1.DocId = @DocId
          order by t1.Id
END
GO
```

```
/****** Object:  StoredProcedure [dbo].[DocDetailsUpdate]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[DocDetailsUpdate]
@Quantity numeric (18,3), @PriceR numeric (9,2), @Id int output,
@DocId int, @ProductId int, @PriceP numeric (9,2)
AS
BEGIN
     if @Id = 0
          begin
               insert into DocDetails (Quantity, PriceR, DocId,
PriceP, ProductId)
                    values (@Quantity, @PriceR, @DocId, @PriceP,
@ProductId)
               select @Id = SCOPE_IDENTITY()
          end
     else
          begin
               update DocDetails set Quantity = @Quantity, PriceR
= @PriceR, DocId = @DocId, ProductId = @ProductId, PriceP =
@PriceP
                    where Id = @Id
          end

END
GO
/****** Object:  StoredProcedure [dbo].[DocumentsDelete]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[DocumentsDelete]
@DocId int

AS
BEGIN
     delete from DocDetails where DocId = @DocId
     delete from Documents where DocId = @DocId
END
GO
/****** Object:  StoredProcedure [dbo].[DocumentsSelectAll]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[DocumentsSelectAll]
@DocType smallint
```

```
AS
BEGIN
     select t1.*, ClientName = isnull(t2.ClientName, '')
          from Documents t1 left join Clients t2 on t1.ClientId =
t2.ClientId
          where t1.DocType = @DocType
          order by t1.DocDate
END
GO
/****** Object:  StoredProcedure [dbo].[DocumentsUpdate]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[DocumentsUpdate]
@Suma numeric (18,2), @DocType smallint, @DocId int output,
@DocDate datetime, @ClientId int
AS
BEGIN
     if @DocId = 0
          begin
               insert into Documents (DocDate, ClientId, Suma,
DocType)
                    values (@DocDate, @ClientId, @Suma, @DocType)
               select @DocId = SCOPE_IDENTITY()
          end
     else
          begin
               update Documents set DocDate = @DocDate, ClientId
= @ClientId, Suma = @Suma, DocType = @DocType
                    where DocId = @DocId
          end

END
GO
/****** Object:  StoredProcedure [dbo].[ProductsDelete]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

create PROCEDURE [dbo].[ProductsDelete]
@ProductIdId int

AS
BEGIN
     delete from Products where ProductId = @ProductIdId
END
GO
/****** Object:  StoredProcedure [dbo].[ProductsSelectAll]/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

create PROCEDURE [dbo].[ProductsSelectAll]

AS
BEGIN
     select * from Products order by ProductName
END
GO
/****** Object:  StoredProcedure [dbo].[ProductsUpdate]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

GO
CREATE PROCEDURE [dbo].[ProductsUpdate]
@ProductName varchar(100), @Trademark varchar(100), @ProductId
int output,  @Units varchar(20), @PriceP numeric (9,2), @PriceR
numeric (9,2)
AS
BEGIN
     if @ProductId = 0        begin
               insert   into   Products   (Trademark,   PriceR,
ProductName, Units,  PriceP)
                    values  (@Trademark,  @PriceR,  @ProductName,
@Units, @PriceP)
               select @ProductId = SCOPE_IDENTITY()
          end
     else
          begin
               update Products set Units = @Units, ProductName =
@ProductName, Trademark = @Trademark, PriceP = @PriceP, PriceR =
@PriceR,
where ProductId = @ProductId
          end
END
GO
/****** Object:  StoredProcedure [dbo].[SelectBorgPok]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectBorgPok]

AS
BEGIN
     SET NOCOUNT ON;
```

```
     select ClientId, Suma = sum(case when DocType = 2 then Suma
else -1*Suma end)
          into #tmpBorg
          from Documents
          where DocType in (2, 4)
          group by ClientId

     select isnull(t2.ClientName, '') as ClientName, t1.*
          from #tmpBorg t1 left join Clients t2 on t1.ClientId =
t2.ClientId
          order by 1

     drop table #tmpBorg

END
GO
/****** Object:  StoredProcedure [dbo].[SelectBorgPost]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectBorgPost]

AS
BEGIN
     SET NOCOUNT ON;

     select ClientId, Suma = sum(case when DocType = 1 then Suma
else -1*Suma end)
          into #tmpBorg
          from Documents
          where DocType in (1, 3, 5)
          group by ClientId

     select isnull(t2.ClientName, '') as ClientName, t1.*
          from #tmpBorg t1 left join Clients t2 on t1.ClientId =
t2.ClientId
          order by 1

     drop table #tmpBorg

END
GO
/****** Object:  StoredProcedure [dbo].[SelectZalCash]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectZalCash]
@date1 datetime, @date2 datetime
AS
BEGIN
```

```
     SET NOCOUNT ON;

     /*
     declare @date1 datetime, @date2 datetime
     select @date1 = convert(datetime, '21.10.2022', 104)
     select @date2 = convert(datetime, '25.10.2022', 104)
     exec SelectZalCash @date1, @date2
     */

     declare @SumaBeg numeric (18,2)

     select @SumaBeg = sum(case when DocType = 4 then Suma else -
1*Suma end)
          from Documents
          where DocType in (4, 5) and DocDate < @date1
     if @SumaBeg is null select @SumaBeg = 0

     select dbo.setmintime(DocDate) as DocDate, SumaP = sum(Suma)
          into #tp
          from Documents
          where DocType = 4 -- Прихід
               and  dbo.setmintime(DocDate)  between  @date1  and
@date2
          group by dbo.setmintime(DocDate)

     select dbo.setmintime(DocDate) as DocDate, SumaR = sum(Suma)
          into #tr
          from Documents
          where DocType = 5 -- Розхід
               and  dbo.setmintime(DocDate)  between  @date1  and
@date2
          group by dbo.setmintime(DocDate)

     select    isnull(tp.DocDate,    tr.DocDate)    as    DocDate,
isnull(tp.SumaP, 0) as SumaP, isnull(tr.SumaR, 0) as SumaR
          into #t1
          from #tp tp full join #tr tr on tp.DocDate = tr.DocDate

     alter table #t1 add
               SumaBeg numeric(18,2) not null default 0,
               SumaEnd numeric(18,2) not null default 0

     ;WITH q AS ( SELECT TOP 100000 *
                         FROM #t1
                         order by DocDate
                    )
          update  q  set  @SumaBeg  =  SumaEnd  =  @SumaBeg  +  SumaP  -
SumaR,
                    SumaBeg = @SumaBeg - SumaP + SumaR
     select * from #t1 order by DocDate

     drop table #t1
```

```
     drop table #tp
     drop table #tr

END
GO
/****** Object:  StoredProcedure [dbo].[SelectZalProducts]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectZalProducts]

AS
BEGIN
     SET NOCOUNT ON;

     SELECT Quantity = sum(Quantity), ProductId
          into #temp
          from DocDetails
          where DocId in (select DocId from Documents where
DocType = 1)
     insert into #temp (Quantity, ProductId)
          SELECT Quantity = -1*sum(Quantity), ProductId
               from DocDetails where DocId in (select DocId from
Documents where DocType = 2)

     SELECT Quantity = sum(Quantity, ProductId)
          into #temp2
          from #temp
          group by ProductId


     SELECT t2.ProductName, t2.Units, t1.ProductId, t1.Quantity,
t2.PriceR t2.Trademark
          from #temp2 t1 inner join Products t2 on t1.ProductId =
t2.ProductId
          where t1.Quantity != 0
GO
/****** Object:  StoredProcedure [dbo].[SelectZalProductsOnDate]
/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectZalProductsOnDate]
@date datetime

AS
BEGIN
     SET NOCOUNT ON;
```

```
     select ProductId, Quantity = sum(Quantity), SumaP =
sum(round(Quantity*PriceP, 2))
          into #temp
          from DocDetails
          where DocId in (select DocId from Documents where
DocType = 1 and DocDate < @date) -- Прихід товару
          group by ProductId

     insert into #temp (ProductId, Quantity, SumaP)
          select ProductId, Quantity = -1*sum(Quantity), SumaP =
-1*sum(round(Quantity*PriceP, 2))
               from DocDetails
               where DocId in (select DocId from Documents where
DocType = 2 and DocDate < @date) -- Реалізація товару
               group by ProductId

     insert into #temp (ProductId, Quantity, SumaP)
          select ProductId, Quantity = -1*sum(Quantity), SumaP =
-1*sum(round(Quantity*PriceP, 2))
               from DocDetails
               where DocId in (select DocId from Documents where
DocType = 3 and DocDate < @date) -- Повернення постачальнику
               group by ProductId


     select ProductId, Quantity = sum(Quantity), SumaP =
sum(SumaP)
          from #temp
          group by ProductId
END
GO
/****** Object:  StoredProcedure [dbo].[SelectZvitRealiz]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectZvitRealiz]
@date1 datetime, @date2 datetime
AS
BEGIN
     SET NOCOUNT ON;
     /*
     declare @date1 datetime, @date2 datetime
     select @date1 = convert(datetime, '28.07.2022', 104)
     select @date2 = convert(datetime, '26.10.2022', 104)
     exec SelectZvitRealiz @date1, @date2
     */
     -- залишок на початок періоду
     create table #t0 (ProductId int, Quantity numeric(18,3),
SumaP numeric(18,2))
     insert into #t0 (ProductId, Quantity, SumaP)
          exec dbo.SelectZalProductsOnDate @date1
```

```
     -- прихід товару
     select ProductId, Quantity = sum(Quantity), SumaP =
sum(round(Quantity*PriceP, 2))
          into #tp
          from DocDetails
          where DocId in (select DocId from Documents where
DocType = 1 and DocDate between @date1 and @date2)
          group by ProductId

     select ProductId = isnull(t1.ProductId, t2.ProductId),
               QuantityBeg = isnull(t1.Quantity, 0), SumaBeg =
isnull(t1.SumaP, 0),
               QuantityP = isnull(t2.Quantity, 0), SumaP =
isnull(t2.SumaP, 0)
          into #t1
          from #t0 t1 full join #tp t2 on t1.ProductId =
t2.ProductId
     drop table #t0
     drop table #tp

     -- розхід товару
     select ProductId, Quantity = sum(Quantity), SumaP =
sum(round(Quantity*PriceP, 2)), SumaR =
sum(round(Quantity*PriceR, 2))
          into #tr
          from DocDetails
          where DocId in (select DocId from Documents where
DocType = 2 and DocDate between @date1 and @date2)
          group by ProductId

     select ProductId = isnull(t1.ProductId, t2.ProductId),
               QuantityBeg = isnull(t1.QuantityBeg, 0), SumaBeg =
isnull(t1.SumaBeg, 0),
               QuantityP = isnull(t1.QuantityP, 0), SumaP =
cast(isnull(t1.SumaP, 0) as numeric(18,2)),
               QuantityR = isnull(t2.Quantity, 0), SumaR =
cast(isnull(t2.SumaP, 0) as numeric(18,2)), SumaR2 =
cast(isnull(t2.SumaR, 0) as numeric(18,2)),
               QuantityEnd = cast(0 as numeric(18,3)), SumaEnd =
cast(0 as numeric(18,2))
          into #t2
          from #t1 t1 full join #tr t2 on t1.ProductId =
t2.ProductId
     drop table #t1
     drop table #tr

     -- залишок на кінець періоду
     update #t2 set QuantityEnd = QuantityBeg + QuantityP -
QuantityR,
                    SumaEnd = SumaBeg + SumaP - SumaR
```

```
     select t1.*, SumaNac = SumaR2 - SumaR,
               t2.ProductName,    t2.Units,    t2.Trademark    --
додаткові характеристики з довідника товарів
          from #t2 t1 inner join Products t2 on t1.ProductId =
t2.ProductId
          order by t2.ProductName
     drop table #t2

END
GO
/****** Object:  StoredProcedure [dbo].[SelectZvitZamov]/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectZvitZamov]
@date1 datetime, @date2 datetime, @countDays int

AS

BEGIN
     SET NOCOUNT ON;
     /*
     declare @date1 datetime, @date2 datetime
     select @date1 = convert(datetime, '25.09.2022', 104)
     select @date2 = convert(datetime, '25.10.2022', 104)
     exec SelectZvitZamov @date1, @date2, 7
     */

     declare @DaysRealiz int
     select @DaysRealiz = datediff(dd, @date1, @date2) + 1

     -- розхід товару
     select ProductId, SoldAll = sum(Quantity)
          into #t0
          from DocDetails
          where  DocId  in  (select  DocId  from  Documents  where
DocType = 2 and DocDate between @date1 and @date2)
          group by ProductId

     select t1.*, Zal = isnull(t2.Quantity, 0)
          into #t1
          from  #t0  t1  left  join  dbo.ZalProducts()  t2  on
t1.ProductId = t2.ProductId

     alter table #t1 add
          SoldDay numeric(18,3) not null default 0,
          Need numeric(18,0) not null default 0,
          Ord numeric(18,0) not null default 0
     update #t1 set SoldDay = round(SoldAll / @DaysRealiz, 3)
     update #t1 set Need = round(@countDays * SoldDay, 0)
```

```
     update #t1 set Ord = case when Zal >= Need then 0 else
round(Need - Zal, 0) end
     select t1.*,
               t2.ProductName,    t2.Units,    t2.Trademark    --
додаткові характеристики з довідника товарів
          from #t1 t1 inner join Products t2 on t1.ProductId =
t2.ProductId
          order by t2.ProductName

     drop table #t0
     drop table #t1
END
GO
```

**Додаток Г**

## Скрипт основних функцій клієнтської частини ПЗ

Лістинг Г.1 – Клас прихідних документів

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace Torg
{
    public partial class DocumentsP : Form
    {
        SqlDataAdapter _dataAdapter;
        private DataSet _dataSet;

        public DataRow Datarow;

        private int _docType = 1;
        public DocumentsP()
        {
            InitializeComponent();
        }

        public virtual void ShowForm(bool modal)
        {
            if (_dataSet == null) _dataSet = new DataSet();
            _dataSet.Clear();

            _dataAdapter = GetDataAdapter();
            _dataAdapter.Fill(_dataSet);
            gridControl1.DataSource = _dataSet.Tables[0];

            if (modal)
            {
                simpleButtonChoose.Enabled = true;
                ShowDialog();
            }
            else
            {
                MdiParent = ServLib.MdiParent;
                simpleButtonChoose.Enabled = false;
                Show();
            }

        }
        private SqlDataAdapter GetDataAdapter()
        {
            SqlDataAdapter da = new
SqlDataAdapter("torg.dbo.DocumentsSelectAll",
ServLib.Connection)
```

```
            {
                SelectCommand     =     {     CommandType     =
CommandType.StoredProcedure}
            };
            da.SelectCommand.Parameters.AddWithValue("@DocType",
_docType);

            SqlCommand        updateCommand        =        new
SqlCommand("torg.dbo.DocumentsUpdate", ServLib.Connection)
            {
                CommandType = CommandType.StoredProcedure
            };
            SqlParameter                    p                    =
updateCommand.Parameters.Add("@DocId",     SqlDbType.Int,     4,
"DocId");
            p.Direction = ParameterDirection.InputOutput;
            updateCommand.Parameters.Add("@ClientId",
SqlDbType.Int, 4, "ClientId");
            updateCommand.Parameters.Add("@Suma",
SqlDbType.Decimal, 18, "Suma");
            updateCommand.Parameters.Add("@DocDate",
SqlDbType.DateTime, 8, "DocDate");
            updateCommand.Parameters.Add("@DocType",
SqlDbType.SmallInt, 2, "DocType");

            da.UpdateCommand = updateCommand;
            da.InsertCommand = updateCommand;

            SqlCommand        deleteCommand        =        new
SqlCommand("torg.dbo.DocumentsDelete", ServLib.Connection)
            {
                CommandType = CommandType.StoredProcedure
            };
            deleteCommand.Parameters.Add("@DocId",
SqlDbType.Int, 4, "DocId");
            da.DeleteCommand = deleteCommand;

            return da;
        }
        private  void  simpleButtonCreate_Click(object  sender,
EventArgs e)
        {
            gridControl1.Focus();
            DataRow dr = _dataSet.Tables[0].NewRow();
            _dataSet.Tables[0].Rows.Add(dr);

            ShowEditForm(dr, true);

        }
        private void ShowEditForm(DataRow dr, bool newrec)
        {
            if (newrec)
```

```
            {
                ServLib.SetDefaultValue(dr);
                dr["DocType"] = _docType;
                dr["DocDate"] = DateTime.Now;
                _dataAdapter.Update(_dataSet);
            }
            using (DocDetailsP editForm = new DocDetailsP(dr))
            {
                editForm.ShowForm();
            }

            _dataAdapter.Update(_dataSet);
        }

        private void simpleButtonEdit_Click(object sender,
EventArgs e)
        {
            gridControl1.Focus();
            DataRow dr = ServLib.GetCurrentRow(gridView1);
            if (dr == null) return;

            ShowEditForm(dr, false);
        }

        private void simpleButtonDelete_Click(object sender,
EventArgs e)
        {
            gridControl1.Focus();
            DataRow dr = ServLib.GetCurrentRow(gridView1);
            if (dr == null) return;

            if (MessageBox.Show( "Знищити вибраний запис?",
"Увага!           Знищення!",           MessageBoxButtons.YesNo,
MessageBoxIcon.Question,   MessageBoxDefaultButton.Button2)   !=
DialogResult.Yes) return;

            dr.Delete();
            _dataAdapter.Update(_dataSet);
        }

        private void simpleButtonChoose_Click(object sender,
EventArgs e)
        {
            gridControl1.Focus();
            DataRow dr = ServLib.GetCurrentRow(gridView1);
            if (dr == null) return;

            Datarow = dr;

            Close();
        }
    }
```

```
}
```

Лістинг Г.2 – Клас звіту руху коштів

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace Torg
{
    public partial class ZvitCash : Form
    {
        SqlDataAdapter _dataAdapter;
        private DataSet _dataSet;

        public ZvitCash()
        {
            InitializeComponent();
            dateEdit1.DateTime = DateTime.Today.AddDays(-30);
            dateEdit2.DateTime = DateTime.Today;
        }

        public void ShowForm(bool modal)
        {
            if (modal)
            {
                ShowDialog();
            }
            else
            {
                MdiParent = ServLib.MdiParent;
                Show();
            }
        }
        private SqlDataAdapter GetDataAdapter()
        {
            SqlDataAdapter da = new
SqlDataAdapter("torg.dbo.SelectZalCash", ServLib.Connection)
            {
                SelectCommand = { CommandType =
CommandType.StoredProcedure}
            };
            da.SelectCommand.Parameters.AddWithValue("@date1",
dateEdit1.EditValue);
            da.SelectCommand.Parameters.AddWithValue("@date2",
dateEdit2.EditValue);

            return da;
        }
        private void simpleButtonChoose_Click(object sender,
EventArgs e)
```

```
        {
            gridControl1.Focus();
            GetData();
        }
        private void GetData()
        {
            if (_dataSet == null) _dataSet = new DataSet();
            _dataSet.Clear();

            _dataAdapter = GetDataAdapter();
            _dataAdapter.Fill(_dataSet);
            gridControl1.DataSource = _dataSet.Tables[0];
        }
    }
}
```

Лістинг Г.3 – Редагування однієї позиції товару

```
using System;
using System.Data;
using System.Windows.Forms;

namespace Torg
{
    public partial class DocDetailsPEditForm : Form
    {
        private DataRow _dr;
        public DocDetailsPEditForm(DataRow dr)
        {
            InitializeComponent();
            _dr = dr;
        }
        public virtual DialogResult ShowForm()
        {
            textEditName.EditValue = _dr["ProductName"];
            textEditUnits.EditValue = _dr["Units"];
            textEditPriceP.EditValue = _dr["PriceP"];
            textEditQuantity.EditValue = _dr["Quantity"];

            return ShowDialog();
        }
        private    void    simpleButtonOk_Click(object    sender,
EventArgs e)
        {
            _dr["ProductName"] = textEditName.EditValue;
            _dr["Units"] = textEditUnits.EditValue;
            _dr["PriceP"] = textEditPriceP.EditValue;
            _dr["Quantity"] = textEditQuantity.EditValue;
        }
    }
}
```

Лістинг Г.4 – Лістинг редагування прихідної накладної

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace Torg
{
    public partial class DocDetailsP : Form
    {
        SqlDataAdapter _dataAdapter;
        private DataSet _dataSet;

        public DataRow DocumentsDatarow;

        public DocDetailsP(DataRow dr)
        {
            InitializeComponent();
            DocumentsDatarow = dr;
        }

        public DialogResult ShowForm()
        {
            textEditSuma.EditValue = DocumentsDatarow["Suma"];
            dateEdit1.EditValue = DocumentsDatarow["DocDate"];
            textEditClient.EditValue                                    =
DocumentsDatarow["ClientName"];
            textEditClient.Tag = DocumentsDatarow["ClientId"];

            if (_dataSet == null) _dataSet = new DataSet();
            _dataSet.Clear();

            _dataAdapter = GetDataAdapter();
            _dataAdapter.Fill(_dataSet);
            gridControl1.DataSource = _dataSet.Tables[0];

           return ShowDialog();

        }
        private SqlDataAdapter GetDataAdapter()
        {
            SqlDataAdapter           da           =           new
SqlDataAdapter("torg.dbo.DocDetailsSelectByDocId",
ServLib.Connection)
            {
                SelectCommand     =     {     CommandType     =
CommandType.StoredProcedure}
            };
            da.SelectCommand.Parameters.AddWithValue("@DocId",
DocumentsDatarow["DocId"]);
```

```
            SqlCommand        updateCommand        =        new
SqlCommand("torg.dbo.DocDetailsUpdate", ServLib.Connection)
            {
                CommandType = CommandType.StoredProcedure
            };
            SqlParameter p = updateCommand.Parameters.Add("@Id",
SqlDbType.Int, 4, "Id");
            p.Direction = ParameterDirection.InputOutput;
            updateCommand.Parameters.Add("@DocId",
SqlDbType.Int, 4, "DocId");
            updateCommand.Parameters.Add("@ProductId",
SqlDbType.Int, 4, "ProductId");
            updateCommand.Parameters.Add("@Quantity",
SqlDbType.Decimal, 18, "Quantity");
            updateCommand.Parameters.Add("@PriceR",
SqlDbType.Decimal, 9, "PriceR");
            updateCommand.Parameters.Add("@PriceP",
SqlDbType.Decimal, 9, "PriceP");

            da.UpdateCommand = updateCommand;
            da.InsertCommand = updateCommand;

            SqlCommand        deleteCommand        =        new
SqlCommand("torg.dbo.DocDetailsDelete", ServLib.Connection)
            {
                CommandType = CommandType.StoredProcedure
            };
            deleteCommand.Parameters.Add("@Id",    SqlDbType.Int,
4, "Id");
            da.DeleteCommand = deleteCommand;

            return da;
        }
        private   void   simpleButtonCreate_Click(object   sender,
EventArgs e)
        {
            gridControl1.Focus();

            Products product = new Products();
            product.ShowForm(true);
            if (product.Datarow == null) return;

            DataRow dr = _dataSet.Tables[0].NewRow();
            _dataSet.Tables[0].Rows.Add(dr);
            ServLib.SetDefaultValue(dr);
            dr["DocId"] = DocumentsDatarow["DocId"];
            dr["ProductId"] = product.Datarow["ProductId"];
            dr["ProductName"] = product.Datarow["ProductName"];
            dr["Units"] = product.Datarow["Units"];
            dr["PriceP"] = product.Datarow["PriceP"];

            if (ShowEditForm(dr))
```

```
            {
                _dataAdapter.Update(_dataSet);
            }
            else
            {
                _dataSet.Tables[0].Rows.Remove(dr);
            }

            CalcSum();
        }
        private bool ShowEditForm(DataRow dr)
        {

            using (DocDetailsPEditForm docDetailsEditForm = new
DocDetailsPEditForm(dr))
            {
                if       (docDetailsEditForm.ShowForm()       ==
DialogResult.OK) return true;
            }

            return false;
        }

        private   void   simpleButtonEdit_Click(object   sender,
EventArgs e)
        {
            gridControl1.Focus();
            DataRow dr = ServLib.GetCurrentRow(gridView1);
            if (dr == null) return;

            if (ShowEditForm(dr))
            {
                _dataAdapter.Update(_dataSet);
            }
            CalcSum();

        }

        private   void   simpleButtonDelete_Click(object   sender,
EventArgs e)
        {
            gridControl1.Focus();
            DataRow dr = ServLib.GetCurrentRow(gridView1);
            if (dr == null) return;

            if   (MessageBox.Show(   "Знищити   вибраний   запис?",
"Увага!           Знищення!",           MessageBoxButtons.YesNo,
MessageBoxIcon.Question,   MessageBoxDefaultButton.Button2)   !=
DialogResult.Yes) return;

            dr.Delete();
            _dataAdapter.Update(_dataSet);
```

```
            CalcSum();

        }

        private  void  simpleButtonClient_Click(object  sender,
EventArgs e)
        {
            Clients clients = new Clients();
            clients.ShowForm(true);
            if (clients.Datarow == null) return;

            textEditClient.EditValue                              =
clients.Datarow["ClientName"];
            textEditClient.Tag = clients.Datarow["ClientId"];
        }

        private void CalcSum()
        {
            decimal suma = 0;
            foreach (DataRow dr in _dataSet.Tables[0].Rows)
            {
                decimal price = Convert.ToDecimal(dr["PriceP"]);
                decimal                quantity                =
Convert.ToDecimal(dr["Quantity"]);
                suma += ServLib.Round(price * quantity, 2);
            }

            textEditSuma.EditValue = suma;
        }

        private  void  DocDetailsP_FormClosing(object  sender,
FormClosingEventArgs e)
        {
            DocumentsDatarow["Suma"] = textEditSuma.EditValue;
            DocumentsDatarow["DocDate"] = dateEdit1.EditValue;
            DocumentsDatarow["ClientName"]                        =
textEditClient.EditValue;
            DocumentsDatarow["ClientId"] = textEditClient.Tag;
        }
    }
}
```