Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп’ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп’ютерні науки»
зі спеціальності
122 – Комп’ютерні науки

тема роботи:

« Розробка інформаційної системи управління торговим підприємством »

Виконав студент гр. КН-22-1 _____ Назаренко М. Д.

Керівник _____ Маринич І. А.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2026

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *26* » *січня* 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-22-1 *Назаренко Максиму Дмитровичу*

1. Тема кваліфікаційної роботи: *«Розробка інформаційної системи управління торговим підприємством»*

затверджено наказом по університету № *173с* від *30.06.2026 р.*

2. Термін здачі кваліфікаційної роботи: *10.06.2026 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом, додатки, презентація у Microsoft PowerPoint в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Маринич І. А.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.04.26*
2	*Розділ 1*	*05.04.26*
3	*Розділ 2*	*01.05.26*
4	*Висновки*	*25.05.26*
5	*Оформлення кваліфікаційної роботи*	*28.05.26*
6	*Підготовка презентації та графічного матеріалу*	*20.05.26*
7	*Підготовка доповіді до захисту*	*05.06.26*

6. Дата видачі завдання: *28.01.2026р.*

Керівник _____ / Маринич І. А./

7. Запевнення: *Я, Назаренко Максим Дмитрович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ /Назаренко М. Д./

АНОТАЦІЯ

Назаренко М. Д. Розробка інформаційної системи управляння торговим підприємством : кваліфікаційна робота бакалавра : 122 – Комп’ютерні науки. Кривий Ріг. Криворізький національний університет, 2026. 70 с.

Актуальність роботи полягає у тому, що сучасні торговельні підприємства потребують комплексної автоматизації управлінських процесів, цифровізації інформаційних потоків та оптимізації логістики для підвищення продуктивності праці, зменшення впливу людського фактора та покращення якості обслуговування клієнтів. Проблемою залишається недостатня структурованість процесів автоматизації та використання спрощених, вузькоспеціалізованих рішень.

Мета роботи полягає у розробці та практичній реалізації програмного комплексу для оптимізації управлінських процесів у межах торговельного підприємства

У першому розділі виконано аналіз об’єкта дослідження, надано детальну характеристику предметної області та сформовано її концептуальну модель. Проаналізовано існуючі рішення-аналоги, наведено аргументацію щодо вибору технологічного стеку для розробки та сформовано зміст технічного завдання.

У другому розділі описано процес побудови архітектури майбутнього програмного продукту, спроєктовано структури для зберігання даних та деталізовано реалізацію основних сценаріїв взаємодії. Виконано опис розроблених алгоритмів, структури класів інформаційної системи та дизайну графічного інтерфейсу, а також результати випробувань готового програмного комплексу на предмет відповідності технічним вимогам.

Ключові слова:

АРХИТЕКТУРА ПРОГРАМНОГО ПРОДУКТУ, БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, МОДЕЛЬ, ОПТИМІЗАЦІЯ УПРАВЛІНСЬКИХ ПРОЦЕСІВ

ЗМІСТ

ВСТУП

Ефективне ведення бізнесу можна порівняти з динамічним рухом, що потребує постійного балансу, безперервного розвитку та використання актуальних технологічних засобів. У контексті адміністрування життєвого циклу розробки програмного забезпечення та прикладних систем автоматизація стає фундаментом, що забезпечує необхідну швидкість, гнучкість і стабільність для адаптації до змінних запитів споживачів. Модернізація операційних процесів є складним завданням, проте правильно обрана стратегія дозволяє суттєво вдосконалити цикли розробки та підвищити якість продукції.

Спеціалізоване програмне забезпечення для корпоративної автоматизації є сукупністю інструментів, що призначені для усунення монотонних операцій, упорядкування робочих потоків та загального підвищення продуктивності. Окрему роль відіграють цифрові системи документообігу та автоматизованої звітності, які знімають значну частину адміністративного навантаження з фінансових підрозділів і менеджменту, дозволяючи формувати точні дані без тривалих ручних перевірок. Такі рішення можуть бути представлені як масштабними багатофункціональними платформами, так і окремими модулями для вирішення конкретних виробничих задач. Завдяки розвитку хмарних технологій підприємства мають можливість впроваджувати сучасні системи без значних капітальних витрат на власне серверне обладнання.

Іноді підприємства схильні до вибору бюджетних та спрощених варіантів автоматизації з метою мінімізації фінансових витрат. Зазвичай такі рішення мають лімітований набір можливостей, розширення яких потребує додаткових інвестицій, або ж мають вузьку спеціалізацію, що змушує купувати окремі продукти для кожного напряму діяльності та намагатися інтегрувати їх власноруч. Подібний підхід не дозволяє повною мірою задовольнити потреби персоналу та запити клієнтів, що суттєво гальмує досягнення цільових

показників бізнесу. Для ефективного масштабування, зростання продуктивності праці та покращення сервісу необхідно переходити від базових інструментів до комплексної автоматизації робочих процесів. Такий крок дозволяє мінімізувати безпосередню участь людини в операційному циклі, що своєю чергою зменшує кількість помилок, спричинених людським чинником.

Актуальність роботи полягає у тому, що сучасні торговельні підприємства потребують комплексної автоматизації управлінських процесів, цифровізації інформаційних потоків та оптимізації логістики для підвищення продуктивності праці, зменшення впливу людського фактора та покращення якості обслуговування клієнтів. Проблемою залишається недостатня структурованість процесів автоматизації, орієнтація керівництва на короткостроковий економічний ефект та використання спрощених, вузькоспеціалізованих рішень, що не забезпечують повної інтеграції та масштабування бізнес-процесів.

Об’єктом дослідження є система управління торговельного підприємства.

Предметом дослідження є процеси цифровізації та автоматизації інформаційних потоків в управлінській структурі підприємства.

Метою роботи є розробка інформаційної системи для управління логістикою та рухом товарних запасів на складі торговельного підприємства.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– Проаналізувати організаційну та функціональну структуру підприємства.

– Дослідити особливості використання внутрішньої інформації та документообігу.

– Розробити архітектуру бази даних для інформаційної системи.

– Створити користувацьку оболонку для взаємодії з даними та управління логістикою.

– Провести тестування та оцінку ефективності розробленої системи.

Практична цінність роботи полягає у створенні програмного засобу, який дозволяє оптимізувати управлінські процеси торговельного підприємства, підвищити продуктивність праці, зменшити кількість помилок, спричинених людським фактором, та забезпечити своєчасне формування точних управлінських даних для прийняття обґрунтованих рішень. Запропоноване рішення може застосовуватися як у масштабних організаціях, так і у підприємствах середнього рівня, забезпечуючи інтеграцію інформаційних потоків та підтримку стратегічного розвитку бізнесу.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОПИС ОСНОВНИХ ПОНЯТЬ

1.1 Характеристика предметної області

Проєктована база даних орієнтована на забезпечення діяльності торговельного закладу. Її основним завданням є інформаційне супроводження персоналу складських та торговельних підрозділів. Система має акумулювати відомості про товарну номенклатуру, операції з реалізації, складський облік та результативність роботи працівників, а також забезпечувати формування різноманітної аналітичної звітності. Підприємство спеціалізується на роздрібній торгівлі одягом, виступаючи ланкою між постачальниками та кінцевими покупцями.

Сфера інтересів магазину одягу охоплює такий комплекс послуг:

− закупівля продукції у постачальників;

− реалізація товарів покупцям;

− пошук нових контрагентів та клієнтської бази;

− генерація прозорих звітних документів.

Внутрішня структура організації включає три ключові відділи та посаду системного адміністратора, який підпорядковується безпосередньо керівнику. Старший продавець відповідає за документальний супровід товарів, здійснює прямі продажі, координує роботу торговельного персоналу, стежить за станом приміщень та займається маркетинговими заходами. Головний бухгалтер забезпечує фінансові розрахунки з контрагентами та здійснює нагляд за коректністю оформлення закупівельних операцій. Завідувач складу відповідає за моніторинг та управління товарними потоками в межах складського приміщення. Схематичне представлення ієрархії магазину наведено на відповідних рисунках 1.1 та 1.2.

Спираючись на проведений аналіз, можна визначити основні сутності досліджуваної предметної області:

– постачальники;

– операції з продажу;

– складський облік;

– персонал організації;

– товарний асортимент;

– інформаційні розсилки;

– замовлення допоміжних матеріалів.

Рисунок 1.1 – Організаційна схема

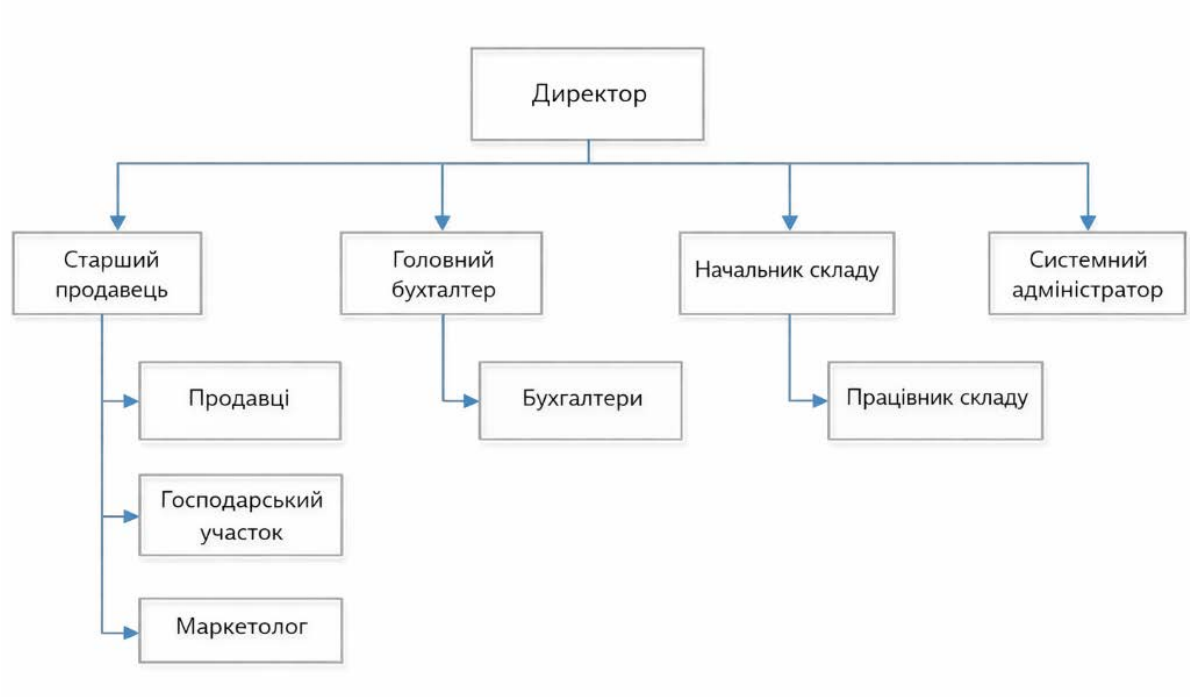

Рисунок 1.2 – Організаційна схема з посад

На основі зібраних відомостей можна зробити висновок, що організаційна структура даного підприємства має лінійно-функціональний характер. Результати цього аналізу будуть використані для подальшого проєктування автоматизованого комплексу.

1.2 Розробка концептуальної моделі предметної області

Створювана база даних призначена для взаємодії з такими групами користувачів:

− працівник складського підрозділу та торговельний персонал;

− керівник організації;

− фахівець фінансового відділу та адміністратор системи.

Функціональні можливості проєктуваної інформаційної системи включатимуть:

− адміністрування бази даних (створення записів, зчитування, коригування та переміщення даних до архіву);

− гарантування цілісності та логічної узгодженості збереженої інформації;

− реалізація механізмів захисту даних від стороннього втручання або випадкових помилок через розмежування рівнів доступу;

− виконання найбільш затребуваних запитів у готовому інтерфейсному вигляді.

Процес концептуального моделювання є обов'язковою стадією розробки інформаційного середовища[1]. Цей етап дає змогу чітко визначити архітектурну будову майбутньої бази даних.

1.3 Аналіз існуючих рішень

Окрім потреби в усуненні рутинних операцій шляхом автоматизації, постійна трансформація запитів клієнтів та персоналу стимулює активне

впровадження спеціалізованого програмного забезпечення. Деякі організації віддають перевагу єдиним багатофункціональним платформам для максимальної оптимізації внутрішніх процесів. Інші ж шукають рішення, здатні покращити якість взаємодії з покупцями або підвищити рівень задоволеності співробітників. Нижче наведено огляд та порівняльний аналіз популярних готових систем для ведення бізнесу, результати якого відображені на відповідній діаграмі порівняння аналогів.

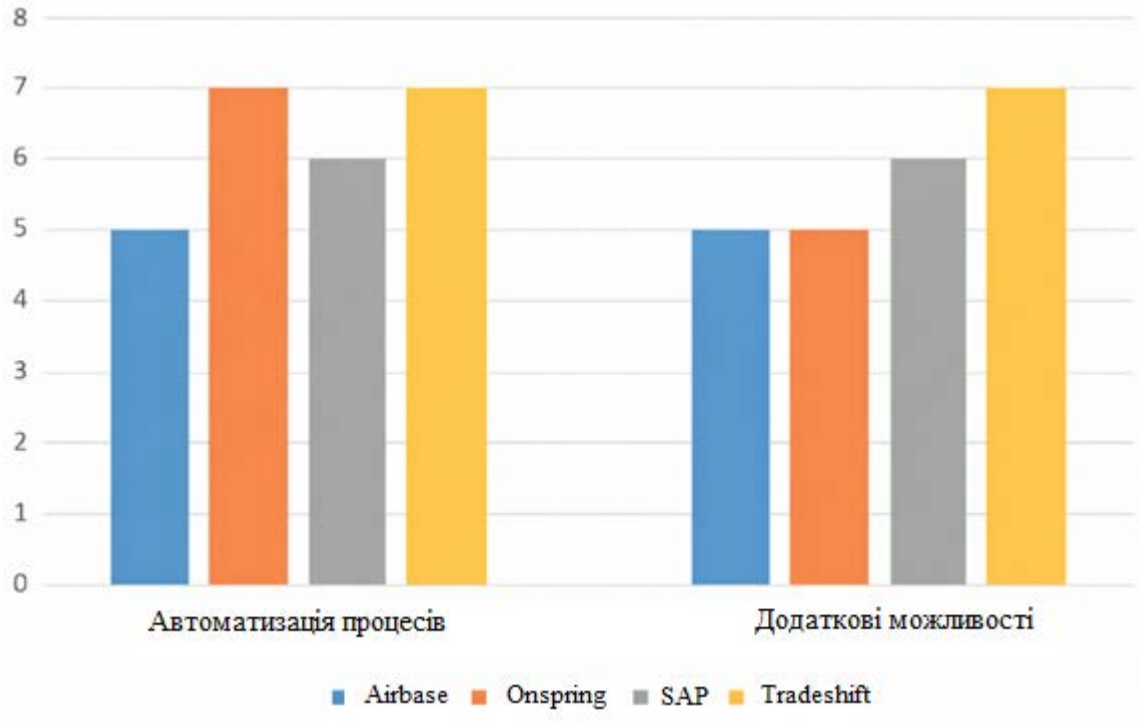

Рисунок 1.3 – Діаграма порівняння аналогів

1.3.1 Платформа Airbase

Airbase [5] є комплексним рішенням для адміністрування витрат, керування корпоративними картками та розрахунками за рахунками. Ця платформа забезпечує повну прозорість фінансових потоків компанії та дозволяє контролювати кожну грошову одиницю.

Система Airbase дозволяє ефективно вирішувати такі завдання:

− обробка запитів на витрати;

− багаторівневе схвалення операцій;

− проведення розрахунків за рахунками.

Також програма автоматизує процеси кодування та внесення даних до головної книги, забезпечуючи:

− моніторинг основних характеристик;

− генерацію детальних звітів про витрати.

Функціонал включає роботу з цифровими квитанціями, синхронізацію з банківськими рахунками та механізми відшкодування коштів. Платежі можуть виконуватися пакетами без необхідності ручного втручання. Платформа дозволяє створювати портали самообслуговування для оперативного отримання рахунків, виконувати звірку виписок та регулювати інвойси. Airbase підтримує роботу з іноземними мовами, аналіз витрат за проектами на основі контрактів та виявлення дубльованих транзакцій.

Серед додаткових можливостей системи варто виділити:

− адміністрування контрактів із моніторингом цін та каталогом додатків;

− можливість автономної інтеграції з іншими програмними продуктами;

− багаторівневий захист даних за допомогою шифрування SSL та відповідності стандартам PCI;

− підтримку різних методів оплати, включаючи банківські перекази та кредитні картки;

− систему визначення доходів з урахуванням відкладених надходжень.

1.3.2 Платформа Onspring

Onspring [6] представляє собою хмарне середовище для управління бізнесом, що забезпечує автоматизоване виявлення проблем та формування аналітики в режимі реального часу. Система вирізняється високою гнучкістю та масштабованістю, що робить її придатною для аудиту та управління ризиками. Розробники дотримуються концепції автоматизації, орієнтованої на людей, допомагаючи спростити робочі цикли.

Функціональні можливості системи охоплюють повний життєвий цикл взаємодії з постачальниками, включаючи закупівлі та стандартизовану адаптацію нових контрагентів. Також реалізовано індивідуальну оцінку

ризиків та відстеження стратегій їх мінімізації. Система дозволяє ідентифікувати дублікати постачальників у межах всього підприємства та гнучко налаштовувати права доступу для груп користувачів.

Інструментарій Onspring об'єднує стратегії управління, політики та комплаєнс, охоплюючи такі напрями:

− внутрішній аудит та надання гарантій;

− забезпечення безперервності діяльності та відновлення після збоїв;

− адміністрування інцидентів та корпоративних політик.

Додаткові переваги системи включають:

− аналітичний модуль із оперативною звітністю;

− автоматизацію динамічних процесів та інтеграційних потоків;

− персоналізовані інформаційні панелі;

− повну консолідацію даних із використанням складних розрахункових формул;

− технічну та освітню підтримку на етапі впровадження.

1.3.3 Платформа SAP

SAP [7], як один із глобальних лідерів IT-ринку, пропонує хмарну платформу для інтегрованого управління фінансами, продажами та витратами. На відміну від стандартних CRM або ERP систем, це рішення спрямоване на глибоке вдосконалення комунікації з постачальниками, що зазвичай реалізується через розрізнені додатки.

Використання SAP дозволяє централізовано керувати всіма стадіями співпраці - від реєстрації партнера до завершення контрактних відносин. Застосування технологій штучного інтелекту допомагає оптимізувати витрати на основі детального аналізу бази постачальників. Завдяки взаємодії в реальному часі та зворотному зв'язку, система суттєво економить трудові ресурси організації.

Додатковий функціонал SAP включає:

− вибір контрагентів на основі актуальних профілів ризиків, географії та категорій;

− проведення переговорів та фіксацію умов KPI;

− матричне представлення даних про постачальників для контролю ефективності;

− інтегроване управління матеріальними потоками на єдиній платформі.

1.3.4 Платформа Tradeshift

Tradeshift [8] є універсальною платформою для цифровізації торгівлі та автоматизації взаємодії з постачальниками. Система орієнтована на перехід до безпаперового документообігу та безконтактне виконання циклу від замовлення до оплати. Програма сприяє зростанню бізнесу через створення сучасних мереж постачання B2B.

Користувачі отримують безкоштовний доступ до інструментів аналітики та засобів співпраці. Платформа забезпечує повну видимість транзакцій та збір даних безпосередньо з ERP-рішень компанії. Уся комунікація між сторонами документується, що позбавляє необхідності пошуку інформації в електронній пошті та обробки паперових квитанцій.

Серед специфічних можливостей Tradeshift:

− автоматизація платіжних циклів через модуль Tradeshift Pay;

− гарантована адаптація постачальників із реальними вигодами для обох сторін;

− інтелектуальна панель управління з підтримкою ШІ;

− інструменти для наповнення ланцюга постачання пропозиціями від перевірених партнерів.

Для більшої наочності представимо порівняння розглянутих систем у вигляді таблиці 1.1.

Система	Переваги	Недоліки	Висновок
Airbase	Повна прозорість фінансових потоків; обробка запитів на витрати; багаторівневе схвалення; автоматизація кодування та внесення даних; робота з цифровими квитанціями; синхронізація з банками; багатомовність; захист даних за SSL та PCI; підтримка різних методів оплати	Орієнтована головним чином на фінансові процеси; обмежена інтеграція з іншими бізнес-процесами; можливий надлишковий функціонал для малого чи середнього підприємства	Добре підходить для управління витратами та фінансами, але не забезпечує комплексної автоматизації всіх бізнес-процесів
Onspring	Гнучкість та масштабованість; автоматизоване виявлення проблем; управління ризиками; повний цикл роботи з постачальниками; індивідуальна оцінка ризиків; персоналізовані інформаційні панелі; аналітика в реальному часі	Основний фокус на аудит та комплаєнс; можлива складність налаштування для користувачів без досвіду; не повністю охоплює логістичні та складські процеси	Підходить для аудиту та управління ризиками, але не для комплексного контролю логістики та руху товарів
SAP	Централізоване управління співпрацею з постачальниками; інтегроване управління фінансами та матеріальними потоками; підтримка AI для оптимізації витрат; реальний час та зворотний зв’язок	Висока вартість; надлишковий функціонал для середніх підприємств; складність впровадження	Ідеально для великих корпорацій, але перевантажена для малого/середнього бізнесу і неефективна з точки зору вартості та ресурсів
Tradeshift	Універсальна платформа для безпаперового документообігу; автоматизація циклу від замовлення до оплати; аналітика та співпраця B2B; інтелектуальні панелі з підтримкою ШІ	Основний акцент на B2B та постачальників; не повністю покриває внутрішні управлінські процеси; може містити функції, що не потрібні конкретному підприємству	Підходить для цифровізації торгівлі та B2B взаємодії, але не забезпечує повного управління внутрішніми процесами підприємства

Проведене порівняння функціональних та додаткових можливостей показало, що жоден із розглянутих варіантів не відповідає вимогам підприємства повною мірою. У випадках, коли програмне забезпечення було близьким до необхідних параметрів, воно містило надлишковий функціонал, що суттєво підвищував вартість системи. З огляду на це, було прийнято рішення про розробку власного інформаційного продукту з точно визначеним набором функцій.

1.4 Огляд та обґрунтування вибору інструментарію розробки

Для ефективної розробки інформаційної системи управління торговим підприємством розглянуто три основні варіанти реалізації системи, які повинні забезпечити швидку роботу, надійне зберігання даних та зручний інтерфейс користувача:

- C# + SQL Server
- Java + MySQL
- Python + PostgreSQL

1.4.1 Мова програмування

C#[4]:

– Переваги: об’єктно-орієнтований підхід, інтеграція з .NET Framework, підтримка багатопотоковості, зручний GUI (WinForms/WPF).

– Недоліки: обмежена кросплатформеність.

– Висновок: найкращий варіант для Windows-додатку з багатокористувацьким доступом та інтерактивним інтерфейсом.

Java:

– Переваги: кросплатформеність, Open Source, велика кількість бібліотек для бізнес-логіки.

– Недоліки: складніше реалізувати настільний GUI, потребує додаткових фреймворків.

– Висновок: підходить для веб-додатків або корпоративних систем, але не оптимальний для швидкого Windows GUI.

Python:

– Переваги: швидке прототипування, аналітичні можливості, Open Source.

– Недоліки: менш ефективний для інтерактивних GUI і великих бізнес-процесів без додаткових бібліотек.

– Висновок: добре для аналітичних модулів і прототипів, але не для повноцінної автоматизованої платформи.

Висновок: Обрано C# як основну мову програмування.

Переваги використання C#:

– Об’єктно-орієнтований підхід, що дозволяє моделювати бізнес-логіку підприємства через класи та об’єкти.

– Підтримка сучасних бібліотек .NET Framework та .NET Core, що забезпечує гнучкість у розробці.

– Інтеграція з різними типами баз даних через ORM (Entity Framework).

– Зручність створення графічного інтерфейсу користувача (WinForms, WPF), що робить систему зрозумілою для співробітників підприємства.

– Можливість легко реалізувати багатопоточність, асинхронні операції та обробку подій, що важливо для швидкого реагування системи на зміни даних.

Застосування в системі:

– Реалізація модулів управління товарами, замовленнями, клієнтами та фінансами.

– Створення форм для введення, редагування та перегляду даних.

– Вбудована логіка для розрахунку фінансових показників, формування звітів і контролю бізнес-процесів.

1.4.2 Система управління базами даних

SQL Server[3,15]:

– Переваги: надійність, підтримка транзакцій, інтеграція з C#, масштабованість.

– Недоліки: ліцензійні витрати для великих компаній.

MySQL:

– Переваги: Open Source, кросплатформова підтримка, підходить для веб-систем.

– Недоліки: обмежена підтримка складних транзакцій у порівнянні з SQL Server.

PostgreSQL:

– Переваги: потужна аналітика, Open Source, гнучкість у типах даних.

– Недоліки: потребує додаткового налаштування для інтеграції з настільними GUI.

Висновок: Для зберігання інформації обрана реляційна база даних SQL Server обрано як оптимальний для інтеграції з C# і забезпечення повної функціональності настільної системи.

Переваги:

– Надійне зберігання структурованих даних про товари, клієнтів, замовлення, постачальників та фінансові транзакції.

– Підтримка складних SQL-запитів для отримання аналітики та звітів.

– Можливість виконання транзакцій, що гарантує цілісність даних при одночасній роботі кількох користувачів.

– Масштабованість, що дозволяє розширювати систему у разі росту підприємства.

Структура бази даних:

– Таблиця Товари: код товару, назва, опис, ціна, кількість на складі.

– Таблиця Клієнти: код клієнта, ПІБ, контактна інформація, історія покупок.

– Таблиця Замовлення: код замовлення, дата, клієнт, список товарів, сума.

– Таблиця Рахунки: номер рахунку, замовлення, дата оплати, статус.

– Таблиця Склад: залишки товарів, місце зберігання, рух товарів.

1.4.3 Інструменти розробки [2,7,19]

– Visual Studio: основне середовище розробки, що забезпечує інтеграцію з C#, зручне відлагодження, генерацію коду та інтерфейсів.

– Entity Framework (EF): ORM-бібліотека, яка дозволяє працювати з даними бази через об’єкти C#, зменшуючи складність SQL-запитів та прискорюючи розробку.

– SQL Server Management Studio (SSMS): інструмент адміністрування бази даних, що дозволяє створювати таблиці, зв’язки та проводити тестування запитів.

– Бібліотеки для звітності: можливість формувати звіти для управлінців підприємства, включаючи графіки продажів, залишки товарів та фінансові підсумки.

Альтернативні варіанти:

– Java + NetBeans/Eclipse + MySQL Workbench

– Python + PyCharm + pgAdmin

Висновок: Комбінація C# + SQL Server обрана як оптимальна для реалізації повнофункціональної системи управління торговим підприємством. Причини вибору:

– Забезпечує надійність та швидкість роботи системи.

– Дозволяє реалізувати зручний інтерфейс користувача.

– Підтримує масштабування та інтеграцію з додатковими модулями (API, електронний документообіг).

– Полегшує обслуговування та розвиток системи у майбутньому.

1.4.4 Звітність та аналітика

– C# дозволяє реалізувати модулі аналітики та звітності: продажі, залишки на складі, фінансові показники.

– Java + MySQL підійде для веб-звітності через додаткові бібліотеки (JasperReports).

– Python + PostgreSQL дозволяє розробляти аналітичні модулі та графіки через Pandas/Matplotlib.

Висновок: Для інтерактивних звітів обрано C# з бібліотеками для побудови графіків і таблиць у GUI.

1.4.5 Можливості розширення системи

– C# + SQL Server: інтеграція з API постачальників, модулі прогнозування продажів, мобільний доступ.

– Java + MySQL: зручне масштабування для веб-клієнтів, інтеграція з онлайн-магазинами.

– Python + PostgreSQL: аналітичні модулі, прогнозування, машинне навчання.

Висновок: C# + SQL Server забезпечує найповніший баланс між функціональністю, інтерактивністю та надійністю.

– Інтеграція з онлайн-магазинами та платформами постачальників.

– Впровадження модулів прогнозування продажів та потреб складу на основі історичних даних.

– Використання аналітичних дашбордів для керівництва.

– Підключення мобільних додатків для віддаленого доступу до системи.

1.4.6 Порівняльна таблиця технологій

Для більшої наочності та зручності порівняння усіх розглянутих варіантів реалізації інформаційної системи зведемо їхні переваги, недоліки та обґрунтування вибору у вигляді порівняльної таблиці 1.2

Це дозволить наочно оцінити сильні та слабкі сторони кожного технологічного рішення та підтвердити обґрунтованість обраного варіанту.

Висновок: Обрана комбінація C# + SQL Server забезпечує оптимальне поєднання надійності, інтерактивності та масштабованості, тоді як Java + MySQL і Python + PostgreSQL більш підходять для веб-рішень або аналітичних модулів. Таблиця 1.2 наочно демонструє переваги і обмеження кожного варіанту та підтверджує обґрунтованість вибору технологій для конкретної інформаційної системи.

Таблиця 1.2 – Порівняльна таблиця можливих варіантів реалізації системи

Варіант	Переваги	Недоліки	Обґрунтування вибору
C# + SQL Server	- Надійність та стабільність - Повна інтеграція з Windows - Зручний графічний інтерфейс - Потужна підтримка .NET	- Обмежена кросплатформеність - Вартість ліцензії SQL Server	Оптимальний для малого та середнього торгового підприємства з Windows GUI, багатокористувацьким доступом та інтеграцією з базою даних.
Java + MySQL	- Кросплатформеність - Open Source - Велика спільнота та бібліотеки	- Складніший настільний GUI - Потребує додаткових фреймворків	Підходить для веб-версій або кросплатформених корпоративних систем.
Python + PostgreSQL	- Швидке прототипування - Підходить для аналітики - Open Source	- Менш ефективний для інтерактивного GUI - Потребує додаткових бібліотек для складної бізнес-логіки	Добрий для аналітичних модулів та прототипів, але не для повноцінної настільної автоматизованої системи.

Висновки до розділу:

У першому розділі було проведено детальний аналіз предметної області та існуючих рішень для управління торговим підприємством, а також обґрунтовано підхід до розробки власної інформаційної системи. Було визначено ключові бізнес-процеси підприємства, такі як закупівля, зберігання, продаж, облік та взаємодія з клієнтами, і виявлено основні потреби: автоматизація документообігу, контроль фінансових потоків та оптимізація управлінських операцій.

Побудовано концептуальну модель предметної області, яка відображає основні сутності та їх взаємозв’язки, що дозволяє формалізувати процеси підприємства та слугує основою для подальшої автоматизації. Проведено огляд існуючих рішень, таких як Airbase, Onspring, SAP та Tradeshift, з визначенням їхніх переваг та обмежень. Аналіз показав, що готові системи часто мають високу вартість, складність впровадження та не завжди відповідають потребам малого та середнього бізнесу.

Розглянуто технології та інструментарій для розробки власної системи, включаючи мови програмування (C#, Java, Python), системи управління базами даних (SQL Server, MySQL, PostgreSQL), інструменти розробки та можливості звітності й аналітики. Порівняльний аналіз продемонстрував, що комбінація C# та SQL Server є оптимальною, забезпечуючи надійну роботу, інтерактивний інтерфейс, швидку розробку та можливість подальшого розвитку системи.

Таким чином, проведений аналіз дозволяє зробити висновок, що для реалізації інформаційної системи управління торговим підприємством доцільно обирати C# разом із SQL Server, що забезпечить реалізацію необхідних функцій та ефективне управління бізнес-процесами.

РОЗДІЛ 2
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТОРГОВОГО ПІДПРИЄМСТВА

2.1 Розробка архітектури програмного продукту

Сучасні організації активно використовують методологію моделювання бізнес-процесів для візуальної фіксації, осмислення та вдосконалення внутрішніх операцій. Будучи невід'ємним елементом менеджменту, таке моделювання виступає інструментом для опису поточної ситуації (стану «як є») та формування бачення майбутньої структури («як має бути») з урахуванням запланованих інновацій. Схематичне представлення моделі є лише засобом досягнення мети, а не кінцевим показником ефективності. Саме діаграма дозволяє чітко визначити подальшу послідовність дій, встановити логічний порядок операцій, усунути зайві чи проміжні ланки та наочно продемонструвати ключові етапи. Це один із найбільш дієвих методів візуалізації бізнес-середовища[9]. Підсумком такого підходу є якісне вдосконалення робочих циклів та раціональний розподіл навантаження між фахівцями, що мінімізує непродуктивні витрати. Основний акцент при цьому робиться на діях, що створюють додаткову вартість, покращують сервіс для споживачів та скорочують втрати часових і трудових ресурсів.

Виділяють два базові типи моделей бізнес-процесів[11]:

− модель «AS-IS» або базова модель (відображення поточного стану);

− модель «TO-BE» (проєктована оновлена ситуація).

Вони застосовуються для детального аналізу, апробації, впровадження та модернізації процесів. Наведемо перелік найбільш поширених технік, що можуть застосовуватися як самостійно, так і в комбінації з іншими формалізованими підходами:

− методика блок-схем;

– діаграми потоків даних (метод Юрдона);

– діаграми ролей та дій (RAD);

– діаграми взаємодії ролей (RID);

– діаграма Ганта;

– інтегроване визначення для моделювання функцій (IDEF);

– кольорові мережі Петрі (CPN);

– діаграма активності UML;

– моделі трансформаційних процесів;

– ієрархічні моделі процесів;

– методи візуалізації.

Для виконання завдань даної роботи використано підхід інтегрованого визначення для моделювання функцій IDEF[12]. У межах дослідження було побудовано контекстну діаграму, яка демонструє вхідні та вихідні ресурси, нормативні правила та механізми управління складським господарством до моменту впровадження інформаційного продукту, що відповідає моделі «AS-IS» (рисунок 2.1).

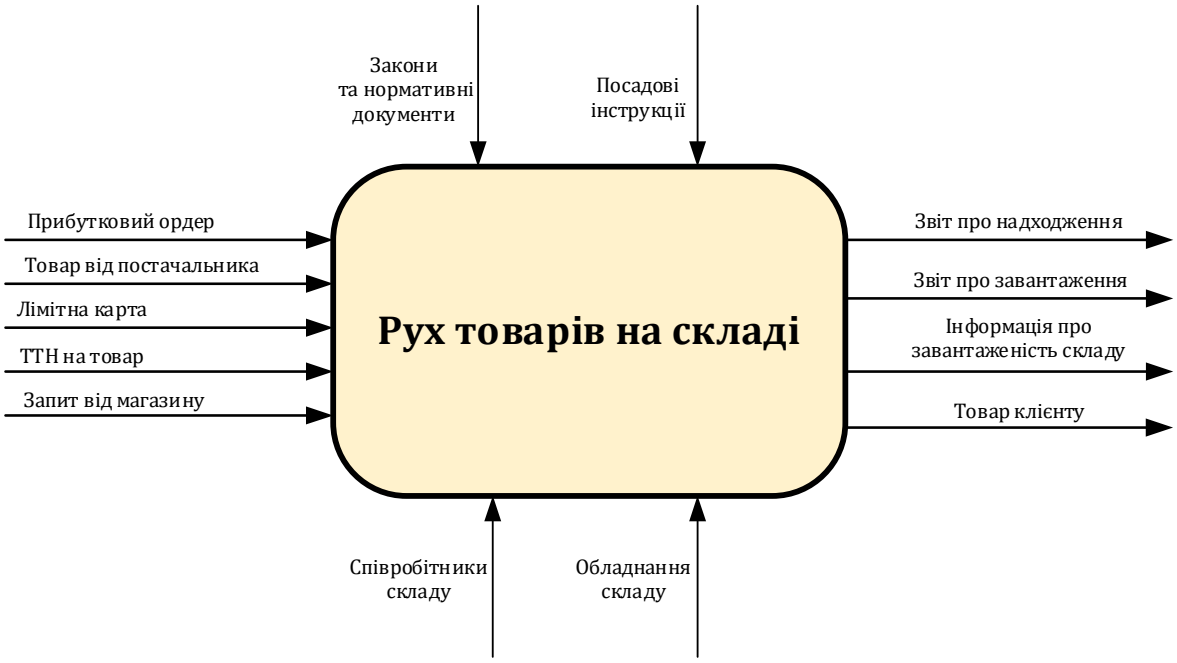

Рисунок 2.1 – Контекстна діаграма функціональної моделі

Контекстна схема верхнього рівня моделі «AS-IS» деталізована на п’ять функціональних блоків (Приймання товарів, Зберігання продукції, Комплектування замовлень за запитом, Відвантаження товарів та Проведення інвентаризації), як показано на рисунку 2.1, що формують перший рівень ієрархії поточної моделі:

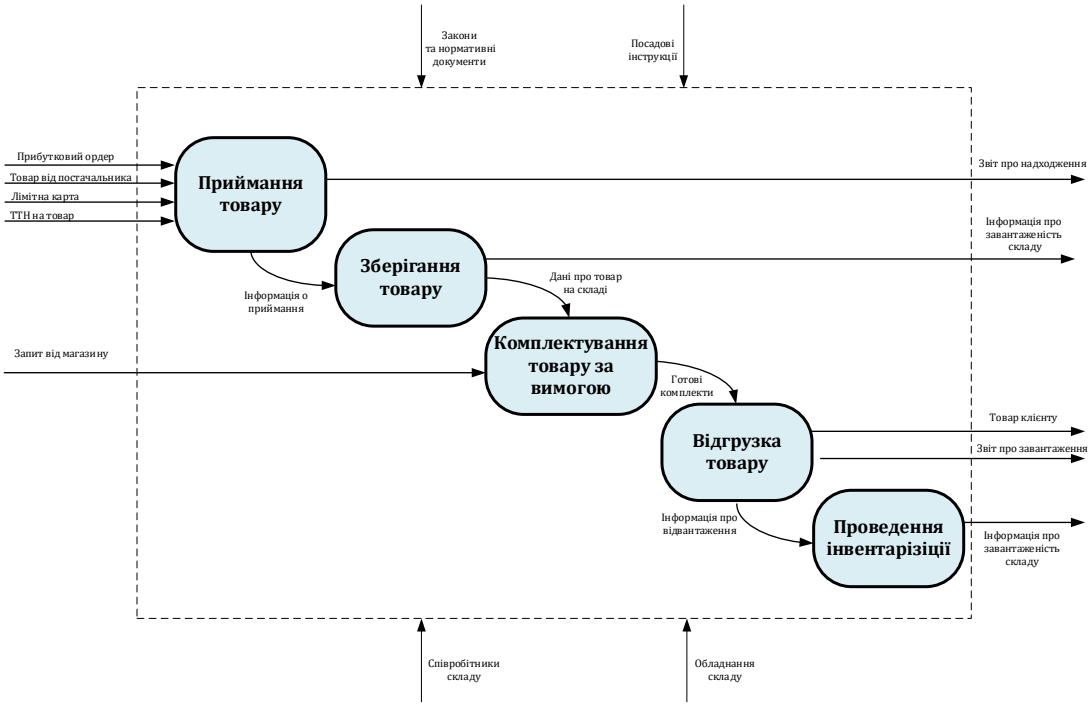

Рисунок 2.2 – Деталізований перший рівень функціональної моделі системи

Спираючись на таку структуру, з’являється можливість детальніше проаналізувати кожен етап та сформувати модель «як має бути». Перелік бізнес-процесів, що входять до складу цієї діаграми, наведено далі.

Бізнес-процес «Приймання товару» (рисунок 2.3) деталізує процедуру надходження продукції на склад. Даний операційний цикл можна розділити на три основні складові:

− Приймання та перевірка супровідної документації;

− Розвантаження продукції;

− Верифікація кількісних та якісних показників товару.

На цьому рівні вибудувано модель первинного руху товарних потоків, коли продукція надходить до організації та проходить початкові стадії - від моменту доставки до звірки відповідності накладних. Критично важливим аспектом цього етапу є коректність оформлення документів про приймання. Навіть незначні похибки можуть призвести до втрати актуальності інформаційної бази та спричинити матеріальні збитки. Чітка послідовність дій у кожному підпроцесі дозволяє звести ймовірність виникнення помилок до мінімуму. Даний бізнес-процес представлено нижче:

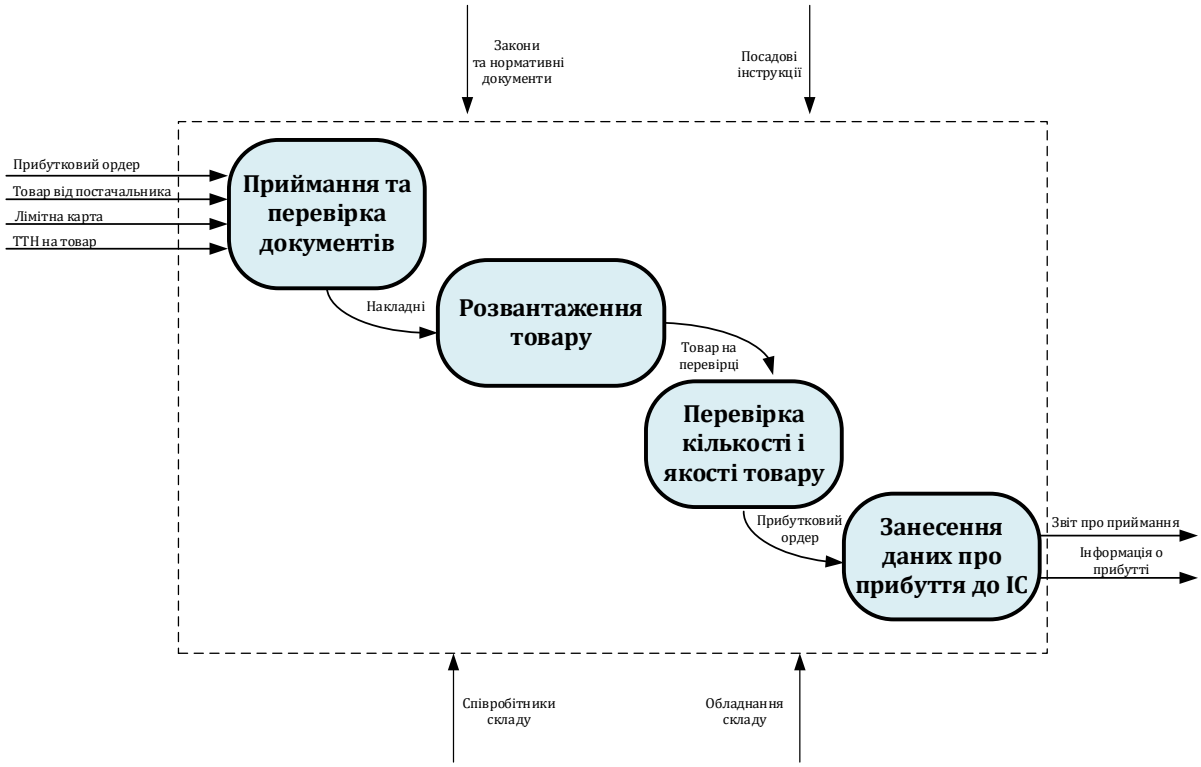

Рисунок 2.3 –Бізнес-процес «Приймання товару»

Бізнес-процес «Зберігання товарів» (рисунок 2.4) описує сукупність дій, пов’язаних із перебуванням продукції на території складу. Цей функціональний блок складається з чотирьох послідовних частин:

− визначення місця для розміщення товарних одиниць;

− переміщення вантажів до зони зберігання;

− безпосереднє розміщення продукції на визначених позиціях;

− систематичний контроль наявності та стану запасів.

Цей рівень є наступним логічним кроком у загальному ланцюгу обробки товару. Одразу після завершення етапу надходження стає необхідним організоване розкладання продукції. Дана фаза є критично важливою для компаній, що експлуатують великі складські площі, багатоярусні стелажні системи або мережу територіально віддалених сховищ. Володіння точними відомостями про конкретну локацію товару дає змогу максимально швидко розраховувати подальші комерційні дії щодо його реалізації. Функціональна модель процесу зберігання товарів представлена далі:

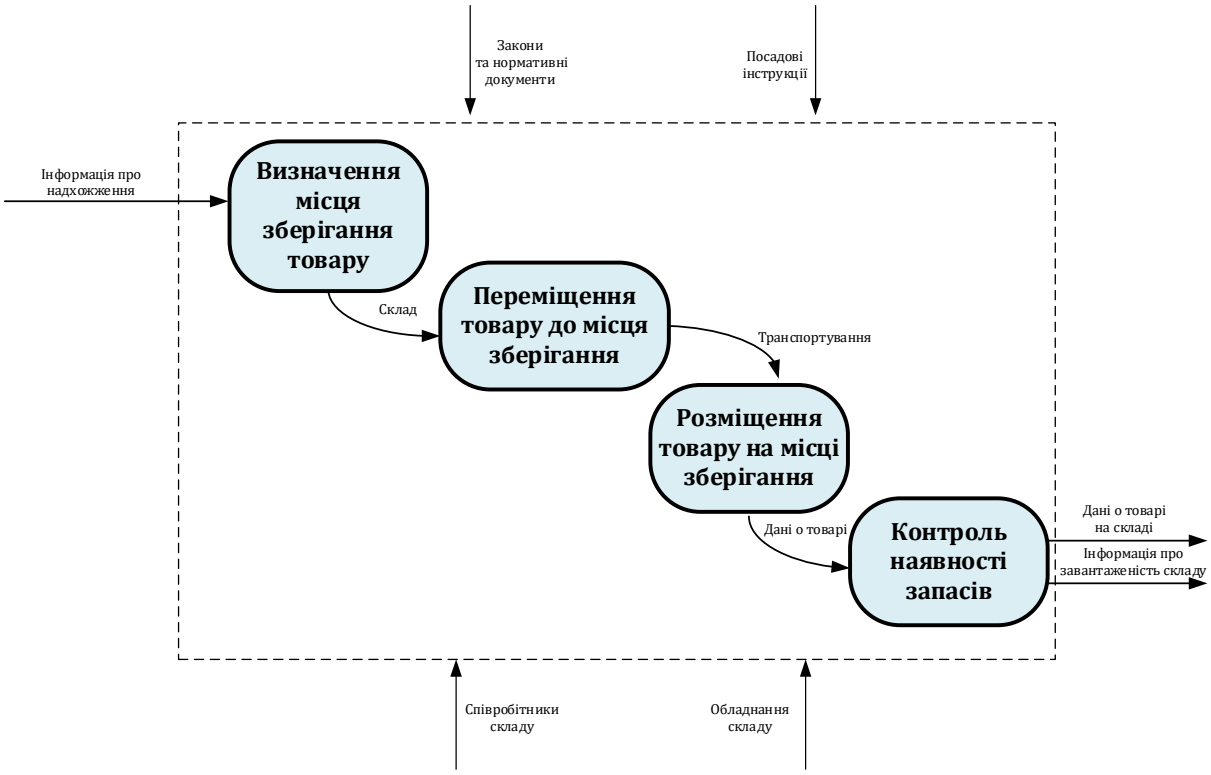

Рисунок 2.4 – Бізнес-процес «Зберігання товарів»

Бізнес-процес (рисунок 2.5) «Комплектування товару за запитом» передбачає процедуру збирання продукції згідно з отриманими замовленнями. Він розподілений на 5 основних стадій:

− отримання документації за замовленнями;

− вибірка товарів відповідно до специфікації;

− безпосереднє комплектування замовлення;

− підготовка супровідного пакету документів;

– пакування та опломбування продукції.

Це найбільш масштабний бізнес-процес серед усіх наведених функціональних моделей торговельної організації. На даному етапі одночасно залучено персонал, що відповідає за формування документальної бази, та працівників, які здійснюють фізичну збірку. Високий рівень концентрації та уважність фахівців є вирішальними факторами успішної реалізації товару кінцевому покупцеві. Процес комплектування товарів деталізовано далі:

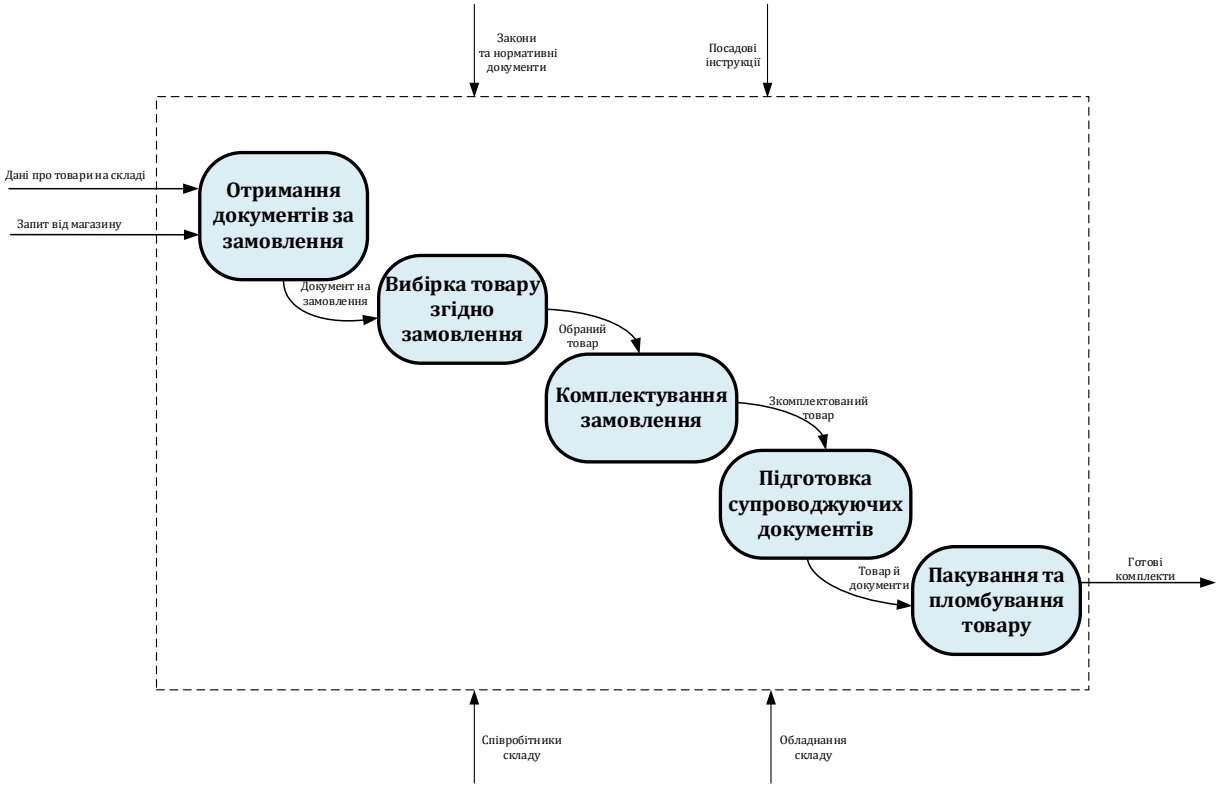

Рисунок 2.5 – Бізнес-процес «Комплектування товару за запитом»

Бізнес-процес «Відвантаження товарів» (рисунок 2.6) передбачає організацію доставки продукції кінцевим споживачам. Даний операційний цикл можна структурувати за трьома основними напрямами:

– оптимальне групування сформованих замовлень;

– підготовка та оформлення транспортних накладних;

– завантаження товарних одиниць у транспортний засіб.

Завдяки суворому дотриманню регламенту дій у межах цього процесу вдається мінімізувати ризики виникнення пересортиці або порушення

комплектації замовлень. Це, у свою чергу, дозволяє суттєво зменшити кількість претензій та випадків повернення продукції від клієнтів. Детальніше з архітектурою побудованого бізнес-процесу функціональної моделі можна ознайомитися нижче:

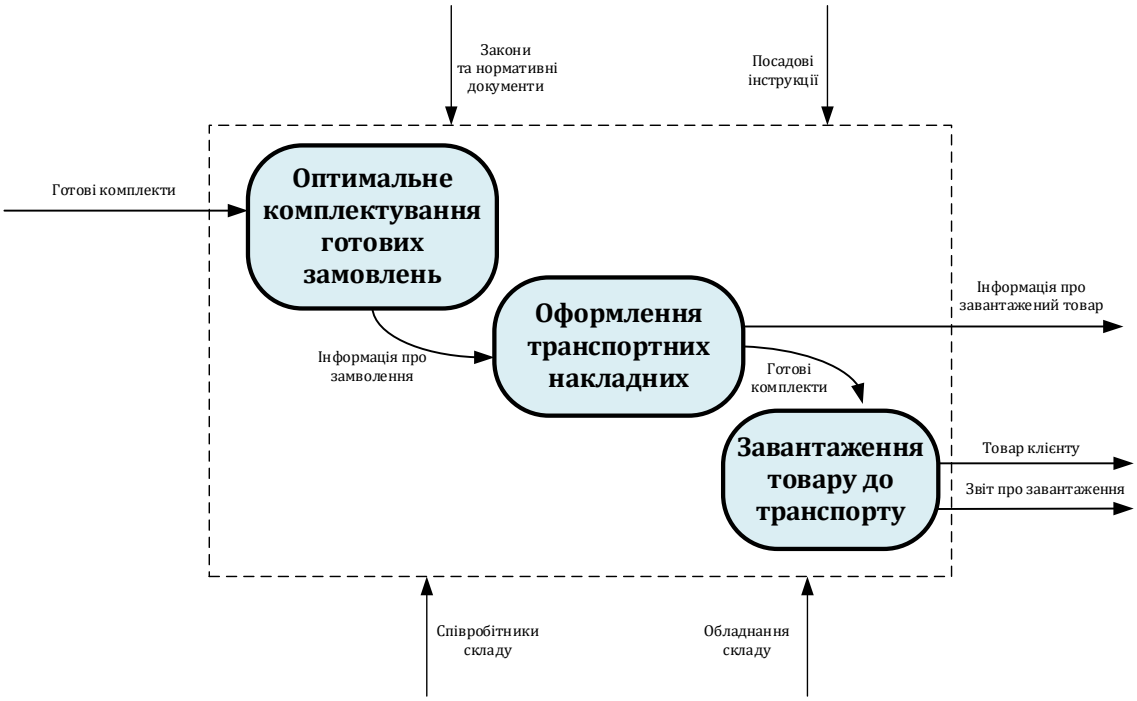

Рисунок 2.6 – Бізнес-процес «Відвантаження товарів»

Бізнес-процес (рисунок 2.7) «Проведення інвентаризації» охоплює сукупність заходів, спрямованих на перевірку фактичної наявності продукції в межах складського господарства. Цей операційний цикл можна розділити на три ключові складові:

− формування документальної бази про залишки товарів на складі;

− порівняння реальної кількості продукції із обліковими даними;

− підготовка фінального звіту про підсумки проведеної ревізії.

Цей етап дозволяє актуалізувати відомості щодо відповідності наявних товарних одиниць документації з надходження та реалізації активів, формуючи таким чином вичерпну аналітику щодо переміщення товарів. Необхідність систематичного здійснення інвентаризації підтверджується потребою у

безперервному контролі над бізнес-процесами, що дає змогу вчасно виявляти недостачі та уникати фінансових втрат підприємства.

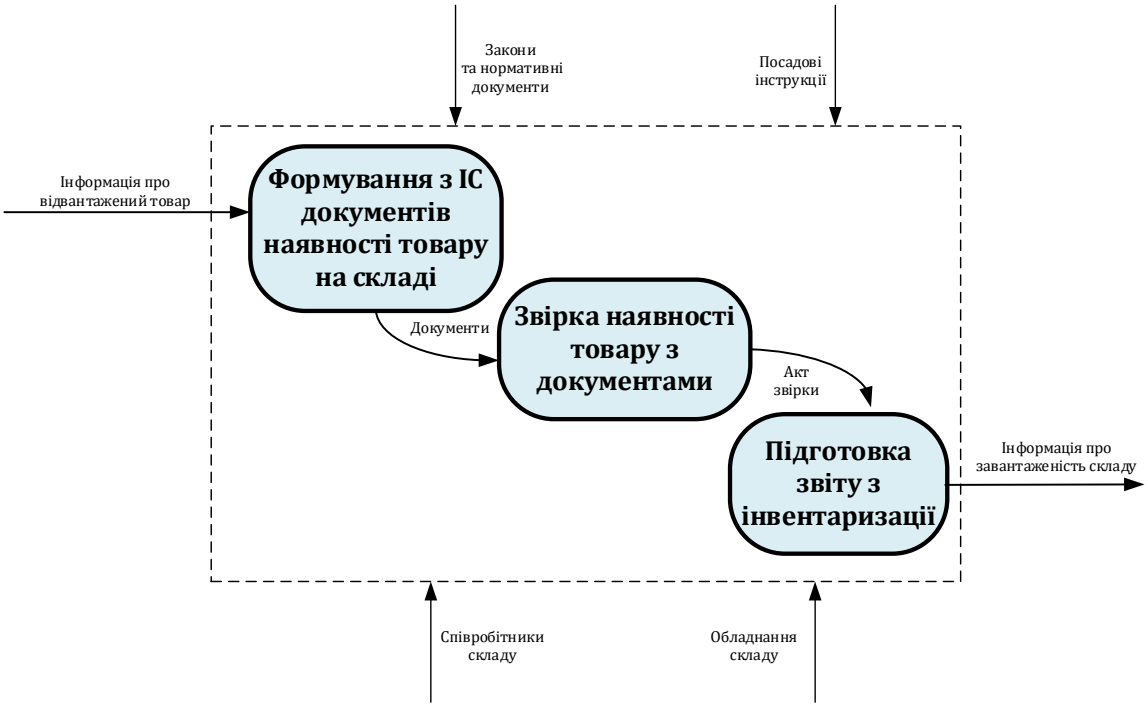

Рисунок 2.7 – Бізнес-процес «Проведення інвентаризації»

Адміністрування складського господарства виступає фундаментальним та водночас складним завданням, особливо для тих суб'єктів господарювання, чий успіх у ланцюгу постачання безпосередньо залежить від продуктивності та точності логістичних операцій. Неефективність у цій сфері здатна суттєво погіршити фінансові показники через зростання операційних витрат, падіння загальної результативності та репутаційні втрати серед клієнтів. Саме тому пріоритетним напрямом розвитку бізнесу має стати впровадження структурованих методик менеджменту, що гарантують безперебійне функціонування об’єкта. Шляхом ідентифікації та усунення профільних ризиків підприємства можуть не лише оптимізувати свою діяльність, а й якісніше задовольняти потреби споживачів. Нижче розглянуто типові деструктивні чинники, що виникають у процесі управління сучасним складом.

Трудові ресурси є ключовою ланкою складської екосистеми, тому грамотне адміністрування персоналу безпосередньо впливає на динаміку

виконання всіх технологічних процедур. Координація великої кількості працівників - від допоміжного персоналу до керівного складу - нерідко супроводжується складнощами, особливо за умов використання дороговартісного обладнання. Частим явищем є нераціональна організація внутрішнього простору, що спричиняє неефективне використання площ та потребу у залученні надлишкової робочої сили для компенсації логістичних прорахунків. Ще одним вагомим деструктивним фактором виступає дефіцит сучасних технологій. Це призводить до архаїзації процесів та необхідності ручного виконання тих операцій, які доцільно було б автоматизувати.

Керівництво складських комплексів також часто стикається з викликами у сфері контролю пошкоджень продукції. Основною причиною таких збитків є недостатній рівень професійної підготовки кадрів та обмеженість ресурсів для коректного маніпулювання вантажами. Відсутність належного навчання призводить до випадкових механічних пошкоджень під час зберігання. Крім того, дефіцит спеціалізованого оснащення (наприклад, гідравлічних візків або захисних пакувальних засобів) також провокує псування майна. Не варто ігнорувати і вплив зовнішніх чинників, таких як несприятливі погодні умови чи природні катаклізми, які здатні завдати шкоди товарам за умови недостатньої захищеності самого об’єкта.

Однією з найбільш гострих проблем залишається неточність інвентаризаційних даних, що має деструктивний вплив на рентабельність бізнесу. Причинами таких розбіжностей часто стають помилки персоналу при підрахунку, фізичне псування або втрата одиниць товару, а також десинхронізація між реальними залишками та електронними записами. Для нівелювання цих ризиків доцільно впроваджувати систематичні перевірки, застосовувати технології штрих-кодування, інтегрувати спеціалізоване програмне забезпечення для контролю запасів та забезпечувати безперервне навчання працівників правилам ведення обліку. Систематичний моніторинг та своєчасне оновлення бази даних дозволяють звести до мінімуму ймовірність

виникнення дефіциту чи надлишку продукції, забезпечуючи стабільність логістичних операцій.

Складські комплекси нерідко потерпають від проблеми надлишкових операцій, до яких належать завдання, що не мають практичної цінності та не приносять користі організації. До таких дій відносяться монотонні ручні процедури, паперовий документообіг та механічне введення інформації, які можна було б автоматизувати або повністю виключити. Надмірна активність персоналу призводить до зниження загальної ефективності, виникнення помилок та зростання витрат на утримання складу. Це зумовлено тим, що на більшості підприємств досі функціонують морально застарілі системи, які вимагають виконання зайвих кроків для реалізації звичних завдань. Нераціонально спроєктовані алгоритми роботи та відсутність засобів автоматизації також провокують дублювання функцій.

Неоптимальне комплектування на складах описує ситуацію, за якої працівники не спроможні здійснювати підбір необхідних товарних позицій найбільш продуктивним способом. Подібний стан справ спричиняє великі втрати часу та ресурсів, суттєво знижуючи загальну виробітку. Однією з фундаментальних причин низької ефективності відбору є дефіцит спеціалізованих систем та технологій, призначених для вдосконалення цього процесу. Також негативно впливає відсутність належної підготовки кадрів та прогалини в комунікації між співробітниками, що породжує плутанину та помилки. Невдале планування складських приміщень та помилки в дизайні простору також заважають швидкому переміщенню та комплектуванню замовлень.

У цілому, усунення недоліків та модернізація процесів відбору товарів має вирішальне значення для успішного функціонування бізнесу. Після впровадження розробленої інформаційної системи (що відповідає моделі «TO-BE») [14]контекстна діаграма верхнього рівня набуває вигляду, представленого на рисунку 2.8.

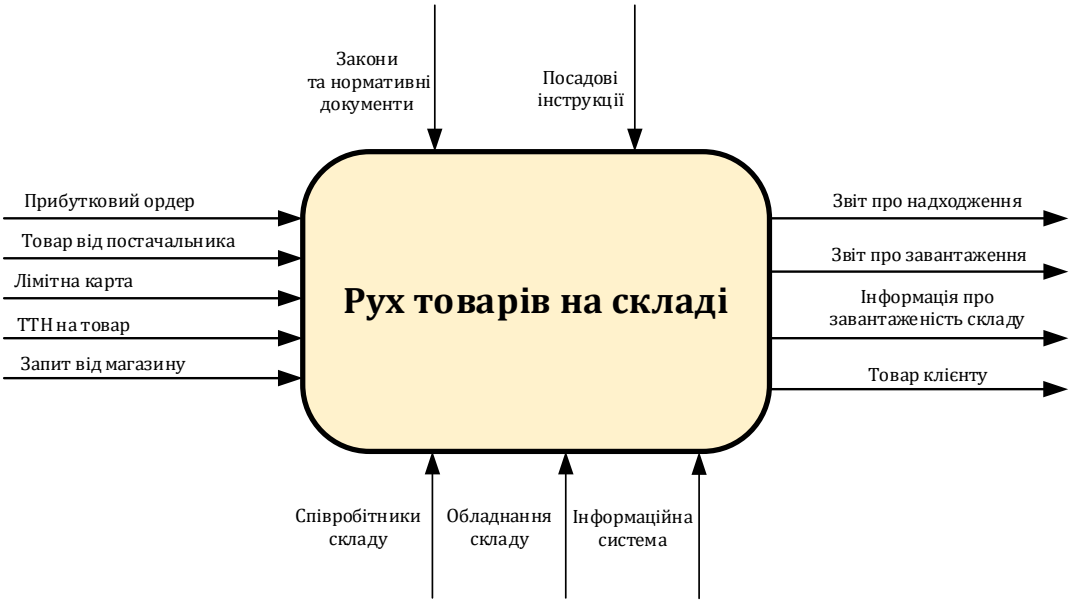

Рисунок 2.8 – Контекстна діаграма функціональної моделі системи

Контекстна схема найвищого рівня в межах проєктованої моделі «TO-BE» структурована за чотирма ключовими функціональними напрямами, як продемонстровано на рисунку 2.9, що фактично формують перший ієрархічний рівень оновленої системи.

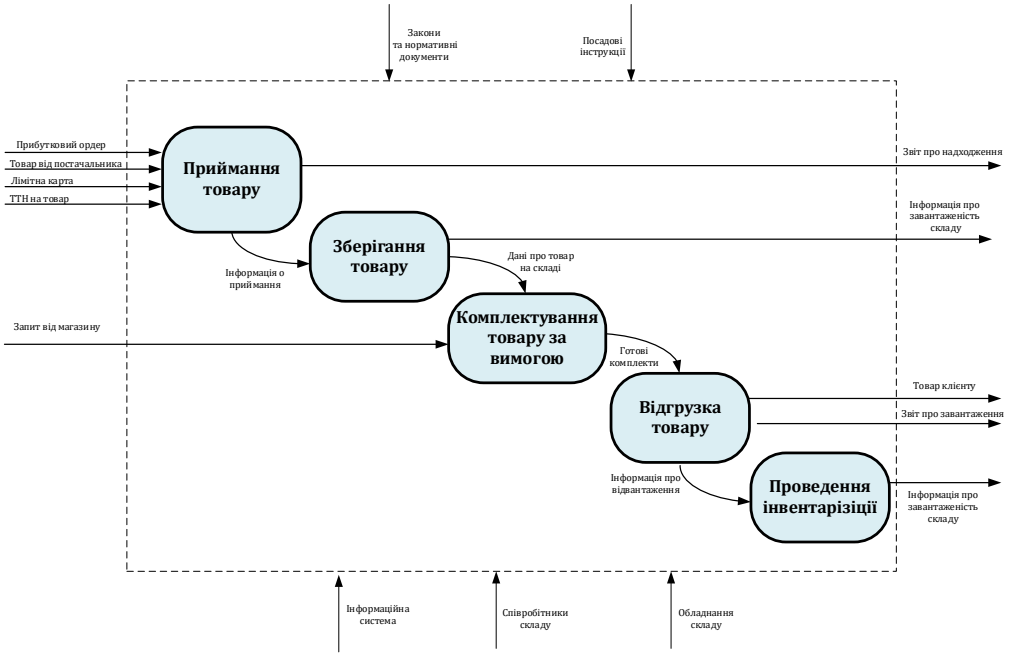

Рисунок 2.9 – Деталізований перший рівень функціональної моделі системи

До переліку бізнес-процесів, які деталізовані в межах цієї діаграми, належать:

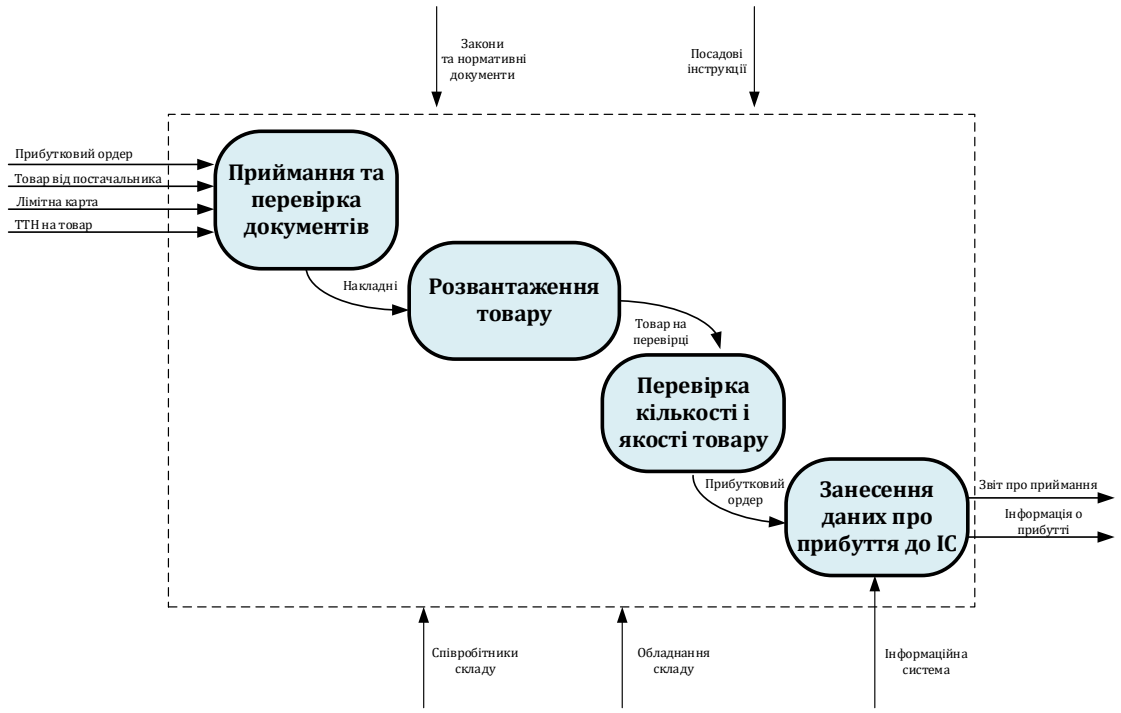

Рисунок 2.10 – Бізнес-процес «Приймання товарної продукції»

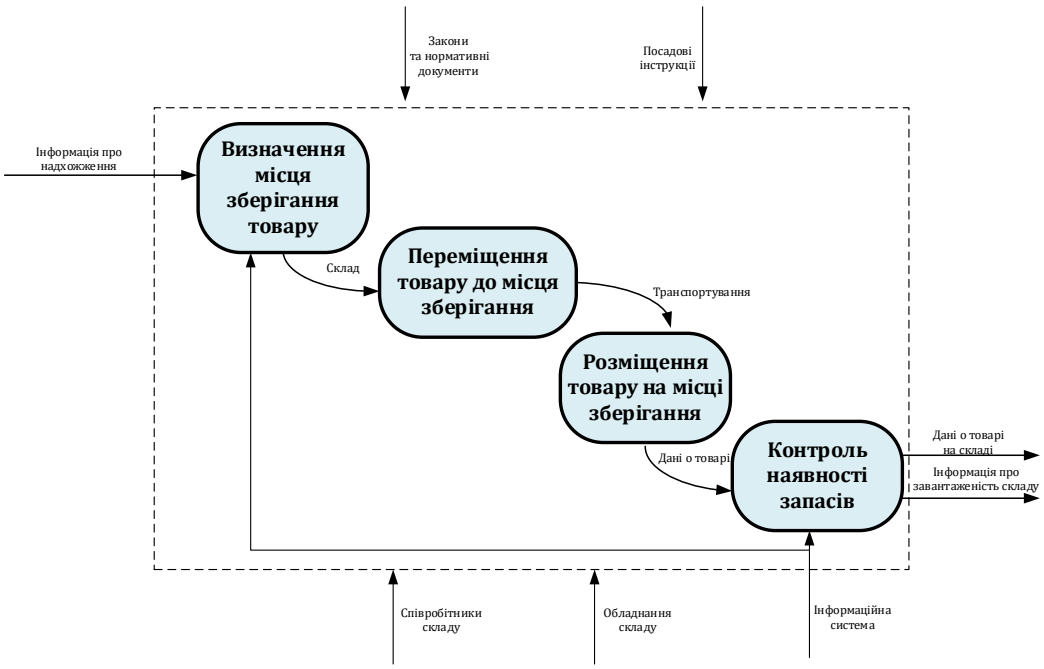

Рисунок 2.11– Бізнес-процес «Організація зберігання товарів»

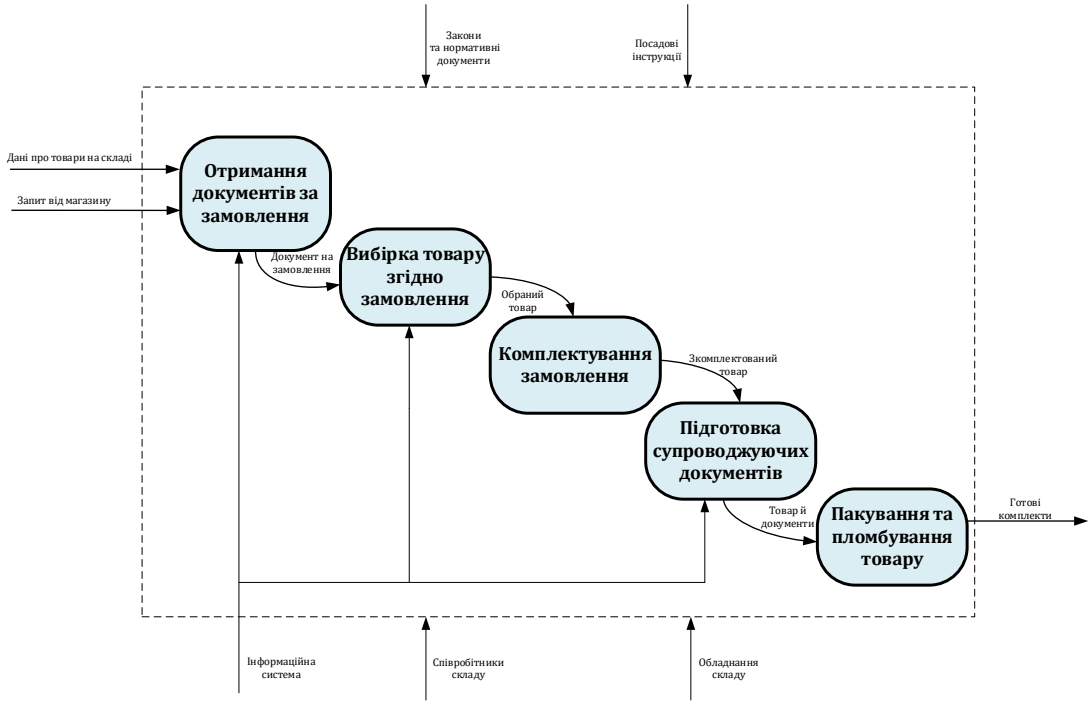

Рисунок 2.12 – Бізнес-процес «Комплектування замовлень згідно із запитами»

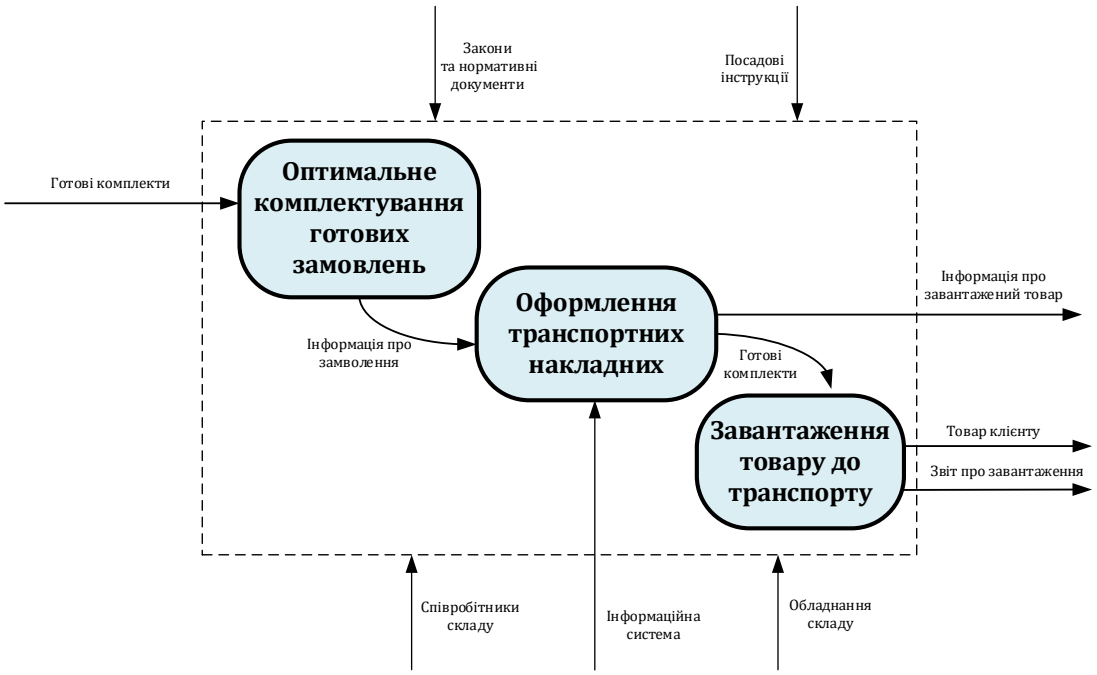

Рисунок 2.13 – Бізнес-процес «Відвантаження продукції споживачам»

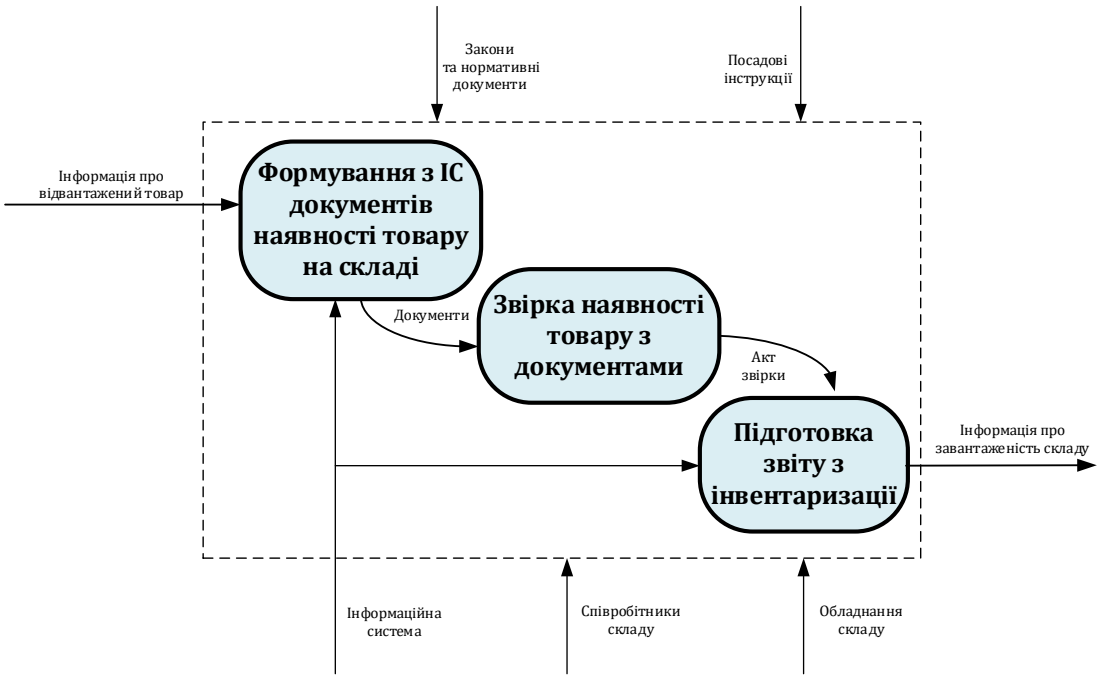

Рисунок 2.14 – Бізнес-процес «Здійснення контрольних інвентаризаційних заходів»

Розроблені функціональні моделі забезпечили наочне представлення бізнес-циклів, що стало підґрунтям для початку робіт із технічного проєктування автоматизованого комплексу.

2.2 Проєктування структур зберігання даних

Реляційна модель сутність-зв’язок виступає інструментом для визначення ключових об’єктів, що будуть інтегровані в інформаційне середовище, а також для опису характеру взаємодії між ними. Графічне відображення архітектури бази даних у вигляді ER-діаграми наведено на рисунку 2.15.

Модель даних ER (сутність-зв’язок) [19]визначає загальну архітектурну схему, що графічно відображає цілісну логічну структуру бази даних. Діаграма відношень між сутностями деталізує характер взаємодії між об’єктами, які інтегровані в інформаційну систему.

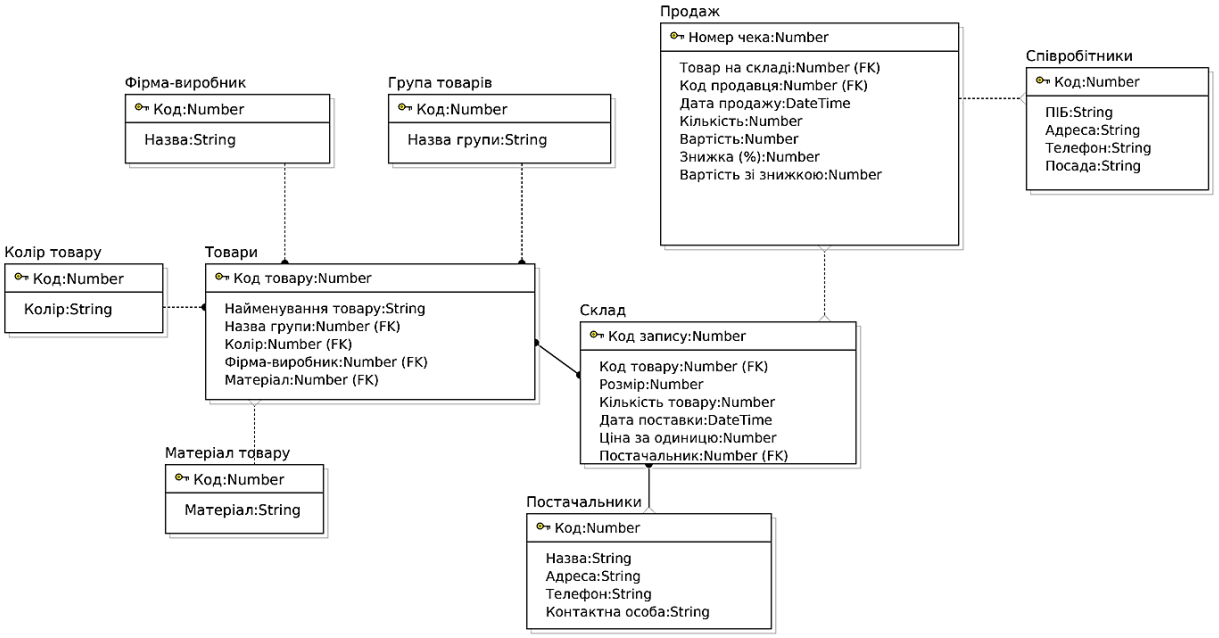

Рисунок 2.15 - ER-діаграма БД, що розробляється

ER-моделі застосовуються для структурування об’єктів реального світу, таких як персонал, матеріальні активи або організації, а також для встановлення логічних зв’язків між ними. Простими словами, ER-діаграма виступає структурним фундаментом для організації бази даних[20]. У наведених нижче таблицях 2.1– 2.9 описано атрибутивний склад кожної таблиці системи.

Таблиця 2.1 - Схема відношення «Співробітники»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код співробітника	Код	INT(4)	Первинний ключ, унікальний
2	ПІБ співробітника	ПІБ	CHAR(50)	Обов'язкове поле
3	Адреса проживання співробітника	Адреса	CHAR(100)	
4	Контактний телефон співробітника	Телефон	INT(40)	Обов'язкове поле
5	Посада співробітника	Посада	CHAR(100)	Обов'язкове поле

Таблиця 2.2 - Схема відношення «Матеріал товару»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код матеріалу товару	Код	INT(4)	Первинний ключ, унікальний
2	Назва матеріалу товару	Матеріал	CHAR(50)	Обов'язкове поле

Таблиця 2.3 - Схема відношення «Постачальники»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код фірми-постачальника	Код	INT(4)	Первинний ключ, унікальний
2	Назва фірми-постачальника	Назва	CHAR(50)	Обов'язкове поле
3	Адреса фірми-постачальника	Адреса	CHAR(100)	
4	Телефон фірми-постачальника	Телефон	INT(40)	Обов'язкове поле
5	ПІБ контактної особи постачальника	Контактна особа	CHAR(100)	

Таблиця 2.4 - Схема відношення «Група товарів»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код групи товарів	Код	INT(4)	Первинний ключ, унікальний
2	Назва групи товарів	Назва групи	CHAR(50)	Обов'язкове поле

Таблиця 2.5 - Схема відношення «Продаж»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код чека при продажу	Номер чека	INT(4)	Первинний ключ, унікальний
2	Наявність товару на складі	Товар на складі	INT(4)	Зовнішній ключ до «Склад»
3	Код продавця, що здійснив продаж	Код продавця	INT(4)	Зовнішній ключ до «Співробітники»
4	Дата здійснення операції	Дата продажу	DATETIME	Обов'язкове поле
5	Кількість реалізованих товарів	Кількість	INT(4)	Обов'язкове поле
6	Вартість товару, що продається	Вартість	INT(20)	Обов'язкове поле
7	Відсоток знижки	Знижка (%)	INT(20)	Обов'язкове поле
8	Підсумкова сума до сплати	Вартість зі знижкою	INT(20)	Обов'язкове поле

Таблиця 2.6 - Схема відношення «Склад»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код запису на складі	Код запису	INT(4)	Первинний ключ, унікальний
2	Код товару	Код товару	INT(1)	Зовнішній ключ до «Товари»
3	Розмір товару, що завозиться	Розмір	INT(4)	Обов'язкове поле
4	Кількість товару, що завозиться	Кількість товару	INT(4)	Обов'язкове поле
5	Вартість однієї одиниці товару	Ціна за одиницю	INT(4)	Обов'язкове поле
6	Код фірми-постачальника	Постачальник	INT(1)	Зовнішній ключ до «Постачальники»

Таблиця 2.7 - Схема відношення «Товари»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код товару	Код товару	INT(4)	Первинний ключ, унікальний
2	Назва товару від постачальника	Найменування товару	CHAR(50)	Обов'язкове поле
3	Код відповідної групи товарів	Назва групи	INT(1)	Зовнішній ключ до «Група товарів»
4	Код кольору товару	Колір	INT(1)	Зовнішній ключ до «Колір товару»
5	Код фірми-виробника	Фірма-виробник	INT(1)	Зовнішній ключ до «Виробник»
6	Код матеріалу товару	Матеріал	INT(1)	Зовнішній ключ до «Матеріал товару»

Таблиця 2.8 - Схема відношення «Фірма-виробник»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код фірми-виробника	Код	INT(4)	Первинний ключ, унікальний
2	Назва фірми-виробника	Назва	CHAR(50)	Обов'язкове поле

Таблиця 2.9 - Схема відношення «Колір товару»

№ рядка	Зміст поля	Назва поля	Тип, довжина	Примітки
1	Код кольору товару	Код	INT(4)	Первинний ключ, унікальний
2	Назва кольору товару	Колір	CHAR(50)	Обов'язкове поле

В описаних вище таблицях набір атрибутів дає змогу деталізувати властивості та характеристики відомостей, що використовуватимуться в інформаційному середовищі. Це забезпечує можливість раціонального перенесення бізнес-алгоритмів у площину конкретних правил зберігання інформації, їхнього коректного завантаження до бази, побудови функціональних зв’язків між таблицями та гарантування цілісності даних.

2.3 Опис реалізації варіантів використання

Сценарій використання демонструє алгоритм поведінки системи або додатка за певних умов. Чим детальніше буде пропрацьовано опис кроків, тим точніше вдасться вибудувати моделі поведінки. Ключовою метою визначення варіантів використання є досягнення консенсусу між розробниками та замовниками щодо функціональних меж та особливостей роботи програмного продукту. Такі сценарії описують системну активність, зокрема процеси взаємодії між комплексом та його користувачами. Під користувачем розуміється людина, конкретна роль, організаційна структура або інша зовнішня система[21]. Виступаючи в ролі актора, він взаємодіє із середовищем для досягнення встановленої мети через певну послідовність дій. Назва сценарію зазвичай відображає кінцеву ціль (наприклад, «моніторинг швидкості об’єкта» або «виплата коштів»), щоб з першого погляду було зрозуміло, який функціонал розглядається.

Поняття варіантів використання активно застосовується у сфері розробки програмного забезпечення та проектування складних систем ще з початку 1990-х років. Сьогодні цей метод залишається популярним, оскільки дозволяє задокументувати можливості планованої системи за допомогою зрозумілих моделей. Серед переваг підходу варто виділити наступні[17]:

− суттєве покращення загального розуміння процесів взаємодії між актором та програмним середовищем;

− можливість чіткої ідентифікації всіх можливих сценаріїв розвитку подій;

− спрощення процесу формування функціональних вимог та створення відповідних тестових прикладів на основі завдань учасників.

Діаграми варіантів використання дозволяють візуалізувати функціональні випадки та дійових осіб із урахуванням наявних зв’язків між ними. Вони забезпечують цілісний огляд структури продукту, проте, на відміну від технічних специфікацій, акцентують увагу не на алгоритмах, а на взаємовідносинах між набором функцій та залученими сторонами. В межах уніфікованої мови моделювання UML [24] такі схеми класифікуються як діаграми поведінки. Основними компонентами моделі виступають: актори, прецеденти (варіанти використання), типи відношень та системні межі. Спільне використання детальних описів та графічних діаграм дозволяє отримати повне уявлення про функціонування системи та виявити її потенційні переваги чи слабкі місця.

Варіанти використання є дієвим інструментом для осмислення системи з позиції кінцевого споживача. Проте на початкових етапах компанії можуть стикатися із труднощами при виборі контенту та формулюванні тем. Зі збільшенням кількості залучених фахівців процес деталізації вимог стає складнішим, але водночас і якіснішим. При підготовці описів слід уникати надмірної перевантаженості деталями, що потребує певного досвіду. Також не рекомендується використовувати шаблонні формулювання для описів ініціюючих подій (тригерів) або умов, оскільки вони не дозволяють точно передати специфіку завдань. Логіку впровадження функцій часто можна визначити через аналіз передумов та результатів: якщо певна дія має результат X, який є необхідною умовою для іншої дії, то вона повинна бути реалізована раніше.

Спираючись на аналіз вимог технічного завдання, у проєктованій системі визначено таких дійових осіб:

– бухгалтер - контролює операційну діяльність складу та відповідає за коректність звітних даних;

– працівник складу - координує безпосередні робочі процеси складського приміщення;

– фахівець магазину - виконує процедури з оформлення замовлень.

Враховуючи потреби цих груп користувачів, можна виділити ключові варіанти використання, що охоплюють основні операції торговельної організації: логістичне переміщення товарів всередині складу, формування замовлення продукції, проведення перевірки залишків (інвентаризація), підготовка замовлень до відправлення, аудит діяльності персоналу, резервування місць для зберігання, підготовка супровідної транспортної документації, процедура отримання товарів зі складу, реєстрація вхідної продукції, верифікація документів та генерація аналітичних звітів. Візуалізація виявлених сценаріїв представлена на діаграмі варіантів [18] використання на рисунку 2.16.

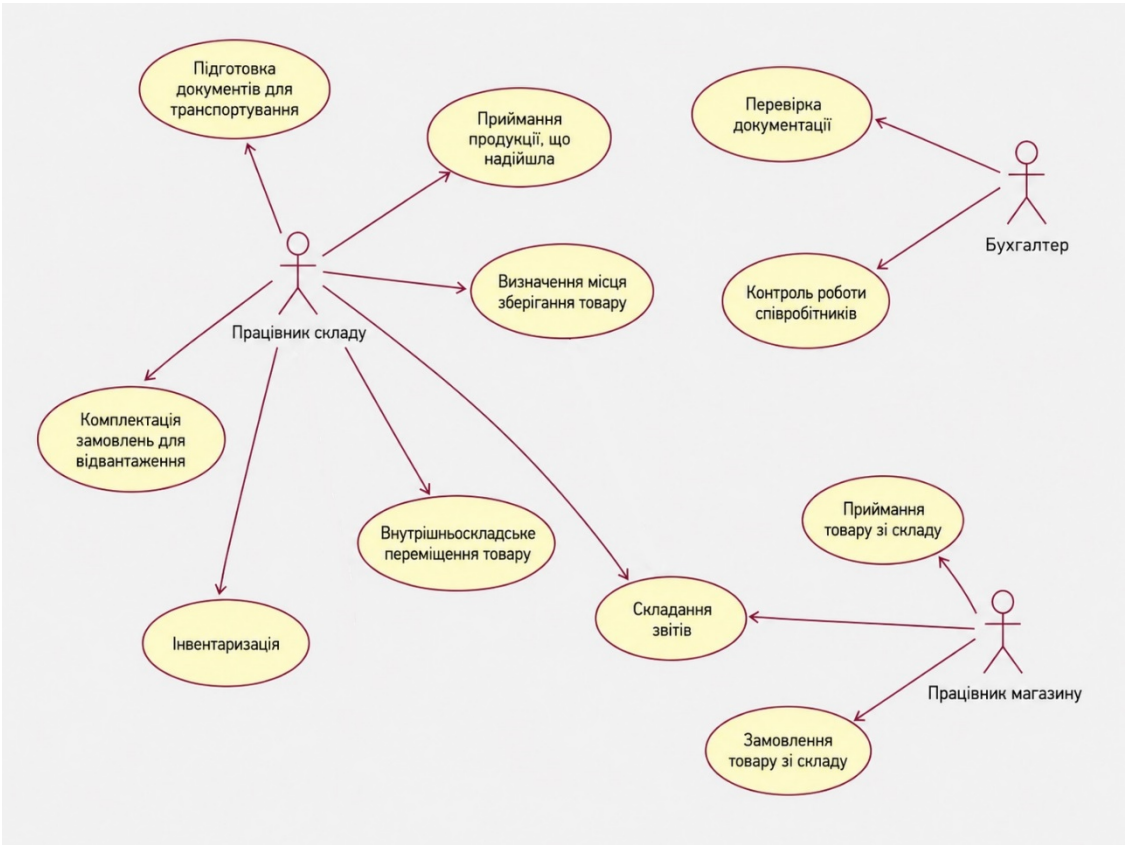

Рисунок 2.16 – Діаграма варіантів використання системи

Під час практичної реалізації розбиття сценаріїв використання на окремі фрагменти, що впроваджуються в межах одного спринту, виявляється надзвичайно корисним. На практиці аналіз випадків некоректного використання також активно сприяє задоволенню запитів клієнтів, тим більше, що їх так само можна представити графічно за допомогою діаграм. Завдяки такій візуалізації розробники мають можливість краще зрозуміти позицію кінцевих споживачів та сформулювати точні вимоги.

2.4 Розробка алгоритмів реалізації варіантів використання

У методології UML діаграма активності [23] дає уявлення про поведінку системи через опис хронологічної послідовності дій у межах певного процесу. Такі схеми мають певну схожість із блок-схемами, оскільки вони відображають логічні переходи між окремими операціями.

Проте на діаграмах активності також можуть бути представлені паралельні, одночасні або альтернативні інформаційні потоки. Завдяки застосуванню цього інструмента моделювання фахівці краще усвідомлюють завдання та технічні вимоги до продукту. Діаграма UML відкриває широкі можливості для професійної комунікації між розробниками, взаємодії із замовником та верифікації реалізованого функціоналу на відповідність вимогам.

Діаграми активності UML входять до категорії діаграм поведінки. У той час як структурна схема фіксує статичний стан системи - тобто наявні об’єкти, їхні ієрархії та зв’язки у конкретний момент часу - діаграми поведінки описують динамічний рух даних. Окрім діаграм активності, до цієї групи належать діаграми варіантів використання та діаграми станів. За своїм призначенням та нотацією ці схеми нагадують алгоритмічні блок-схеми програм, проте вони адаптовані до принципів об’єктно-орієнтованого проектування.

Нижче, на рисунку 2.17, наведено діаграму активності для складського підрозділу. Побудова чіткої взаємодії між ключовими фахівцями дозволяє оптимізувати операційні цикли та усунути зайві етапи роботи.

Діаграма станів[16], яку також називають діаграмою кінцевого автомата або схемою переходів, візуалізує цикл змін, які об’єкт проходить протягом свого існування.

Її використовують для опису динаміки системи, окремих підсистем, модулів або програмних класів. Особливості застосування інтерфейсів у системі також можуть бути відображені за допомогою таких діаграм. Особлива увага приділяється переходам між різними станами об’єкта, подіям-тригерам та властивостям, якими об’єкт володіє або має володіти до моменту зміни свого стану.

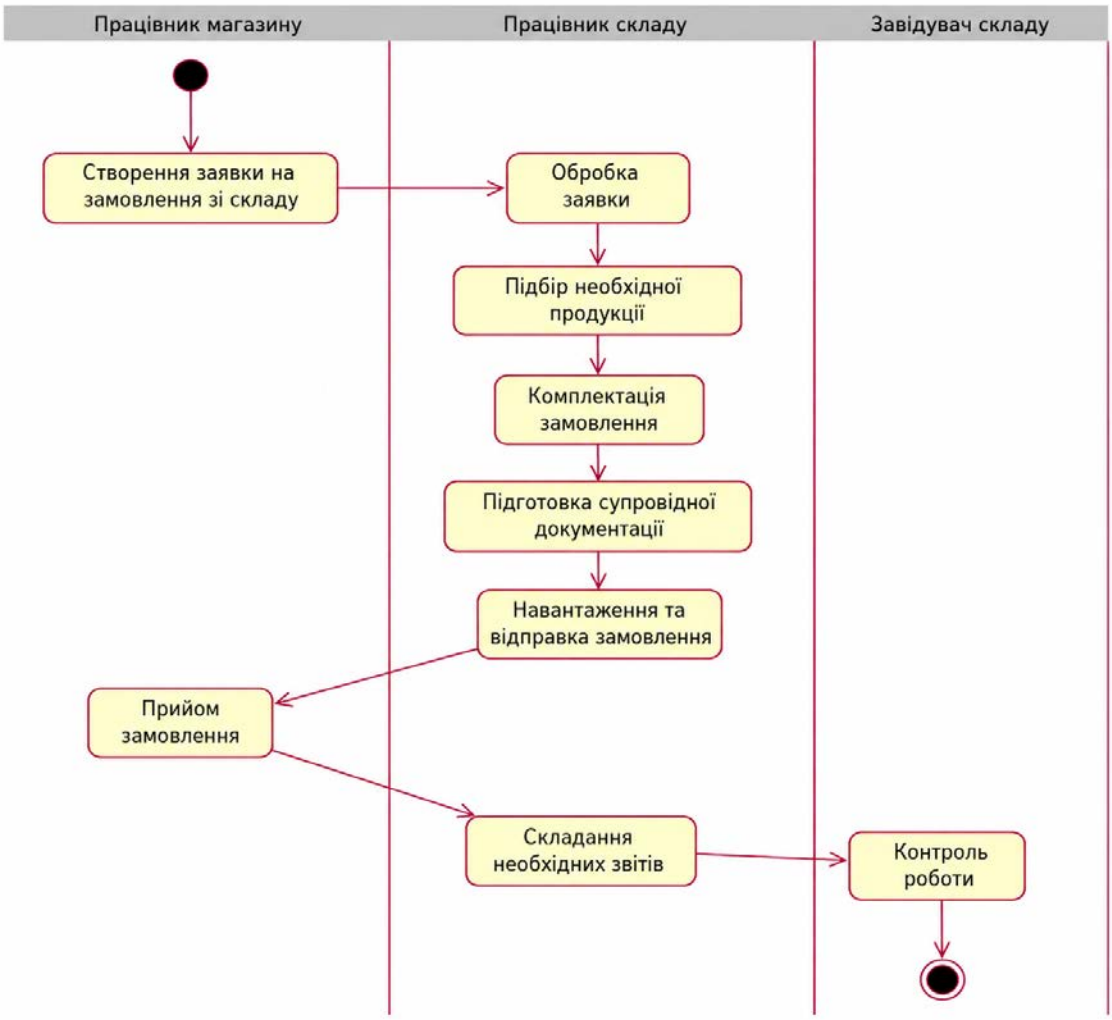

Рисунок 2.17 – Діаграма активності

Фундаментом для побудови діаграм станів стала концепція, розроблена ізраїльським професором інформатики Девідом Харелем, який у 1987 році опублікував працю «Діаграми станів: візуальний формалізм для складних систем». Лише згодом цей тип схем було включено до переліку 14 основних типів діаграм уніфікованої мови моделювання (UML) та, відповідно, до мови моделювання систем (SysML).

Початкова теорія була значно розширена і наразі охоплює такі складні елементи, як ієрархічно вкладені стани, ортогональні зони та операції, виконання яких може одночасно залежати від поточного статусу системи та конкретної вхідної події. Також було реалізовано підтримку вхідних та вихідних дій, які мають більшу прив'язку до самого стану об'єкта, ніж до процесу переходу. На рисунку 2.18 наведено діаграму станів для програмних класів, що відповідають за функціонування складських операцій.

Рисунок 2.18 - Діаграма станів

Діаграми станів часто знаходять своє застосування у сфері вбудованих систем для здійснення моніторингу, керування та точного регулювання функціональних вузлів або процесів обробки сигналів. Вбудовані рішення, які є синергією програмного забезпечення та апаратної складової, виконують безліч сервісних операцій, невидимих для пересічного користувача, - від роботи автомобільних систем та авіоніки до функціонування звичайних холодильників чи ліфтів. Саме вони відповідають за логіку видачі банкнот у банкоматах або спрацювання систем захисту від протікання AquaStop у побутовій техніці. Суттєві сегменти світової економіки, як-от медичне приладобудування, сфера розваг, телекомунікації та транспортна галузь, критично залежать від стабільності вбудованих систем, а отже, і від детального опрацювання діаграм станів. Ба більше, використання таких діаграм є загальноприйнятим стандартом у менеджменті продуктів, управлінні проєктами та на етапі аналізу вимог. Таким чином, розробка алгоритмів стає ключовим етапом побудови будь-якого автоматизованого комплексу, оскільки дозволяє візуалізувати послідовність дій для реалізації головної мети — переведення процесів у цифровий формат. Завдяки алгоритмізації сценаріїв використання стає зрозумілим, як саме програмний продукт має реагувати на ті чи інші зовнішні запити.

2.5 Проєктування класів інформаційної системи

Класична структура діаграми класів передбачає поділ кожного блоку на три функціональні зони[20]. У верхній частині зазначається безпосередньо найменування класу, що ідентифікує його в системі. Середня частина призначена для опису атрибутів класу, що фактично є сукупністю внутрішніх змінних та параметрів. Нижня ж частина відведена для операцій та методів - тобто конкретних дій, які можуть бути ініційовані або виконані в межах цього класу. На рисунку 2.19 продемонстровано архітектуру класів Software

Architecture Class Diagram, розроблену для автоматизації складської діяльності.

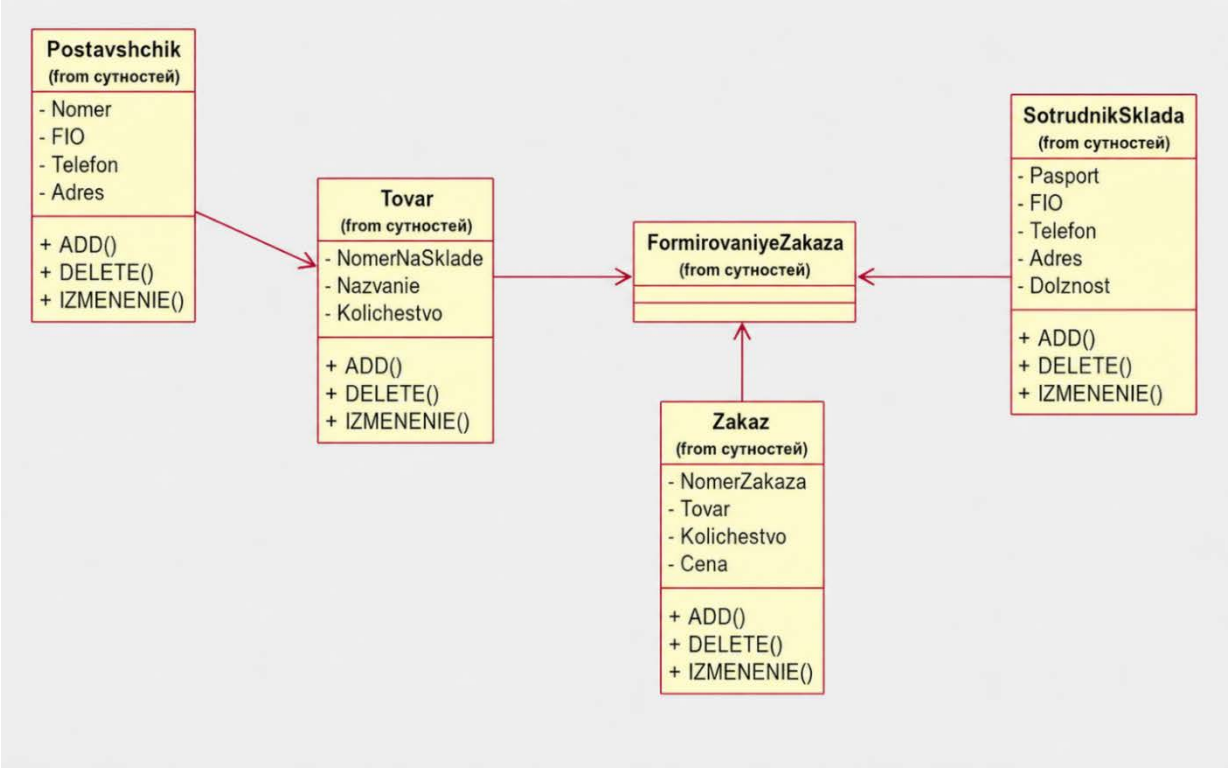

Рисунок 2.19 – Діаграма класів

На рисунку 2.20 представлено діаграму «сутність-зв'язок» для складу.

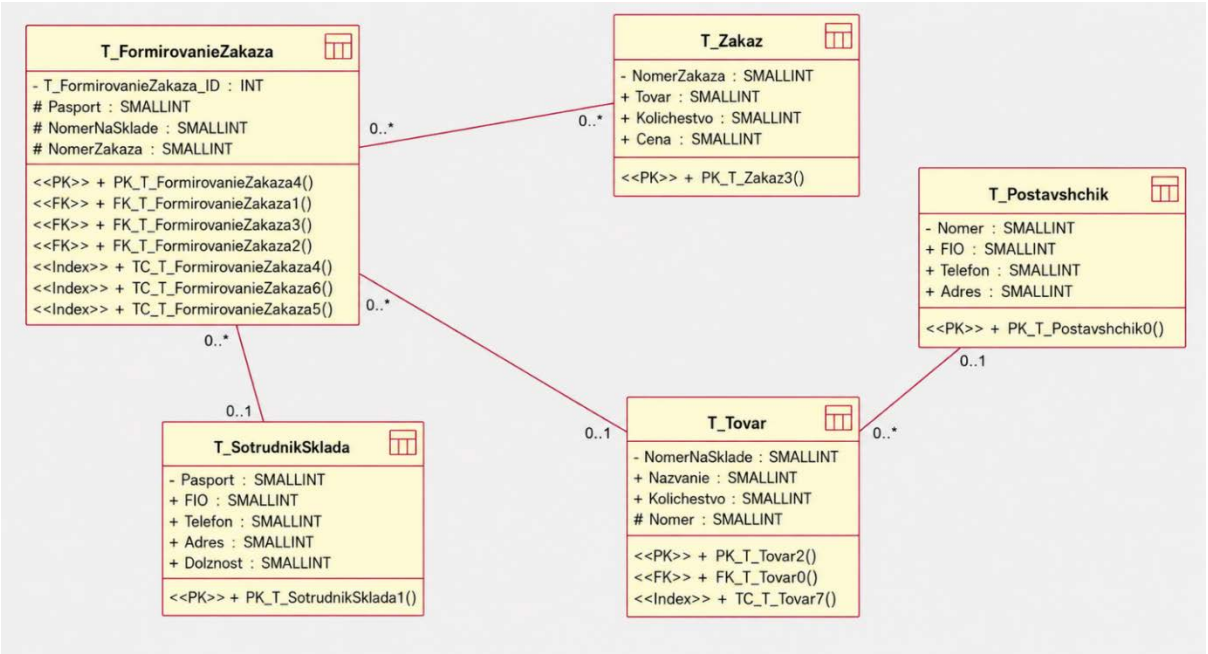

Рисунок 2.20 – Діаграма «сутність-зв'язок»

Застосування такої структури дозволяє розробникам чітко розмежувати відповідальність між окремими програмними компонентами та забезпечити гнучкість при подальшому масштабуванні системи.

Діаграми класів - інструмент, який часто використовується для представлення структури класів та їх відносин. Вони можуть допомогти зрозуміти, як працює програма, і можуть бути корисні для усунення несправностей або створення коду [11].

2.6 Розробка графічного інтерфейсу

Windows Forms (WinForms)[25], як складова частина Microsoft .NET Framework, дозволяє оперативно та зручно проєктувати графічні інтерфейси користувача (GUI) для десктопних застосунків під управлінням ОС Windows. Зовнішній вигляд головної сторінки розробленого інформаційного середовища представлено на рисунку 2.21.

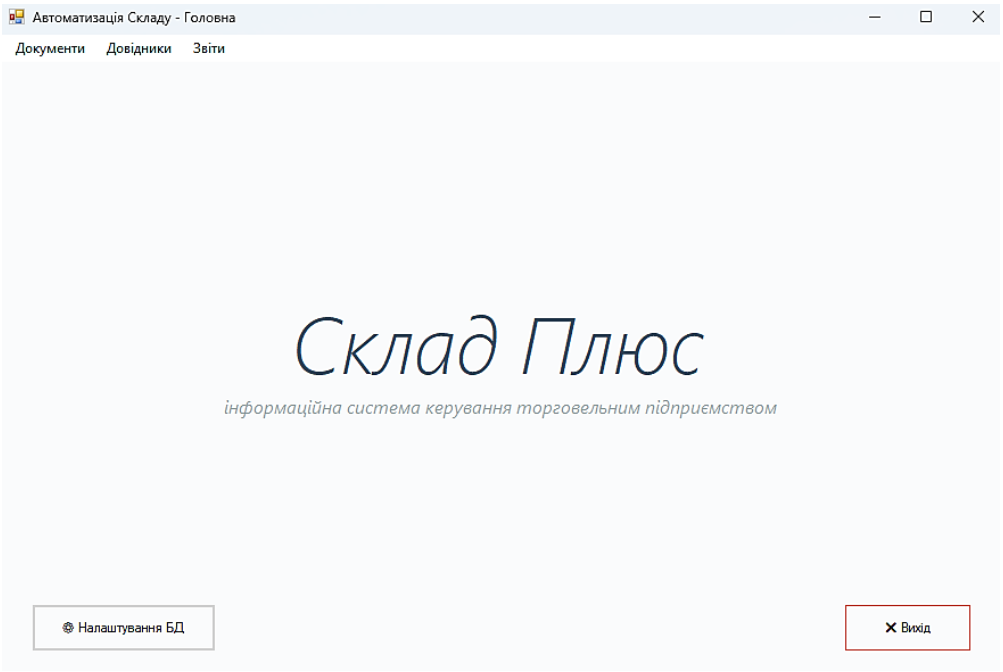

Рисунок 2.21 – Вигляд головної сторінки інформаційної системи

Нижче буде проілюстровано зовнішній вигляд основник вкладок та сторінок.

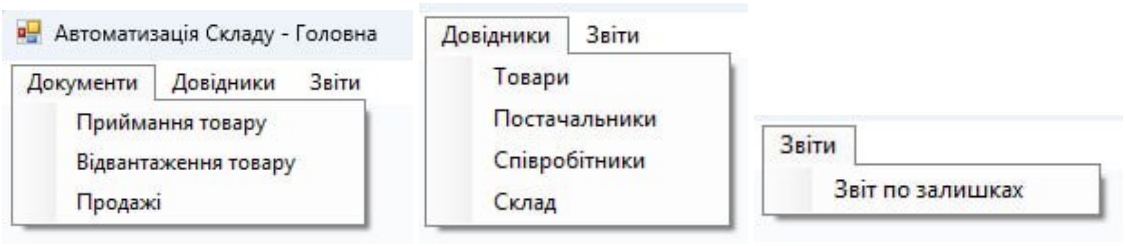

Рисунок 2.22 – Основні вкладки головного меню

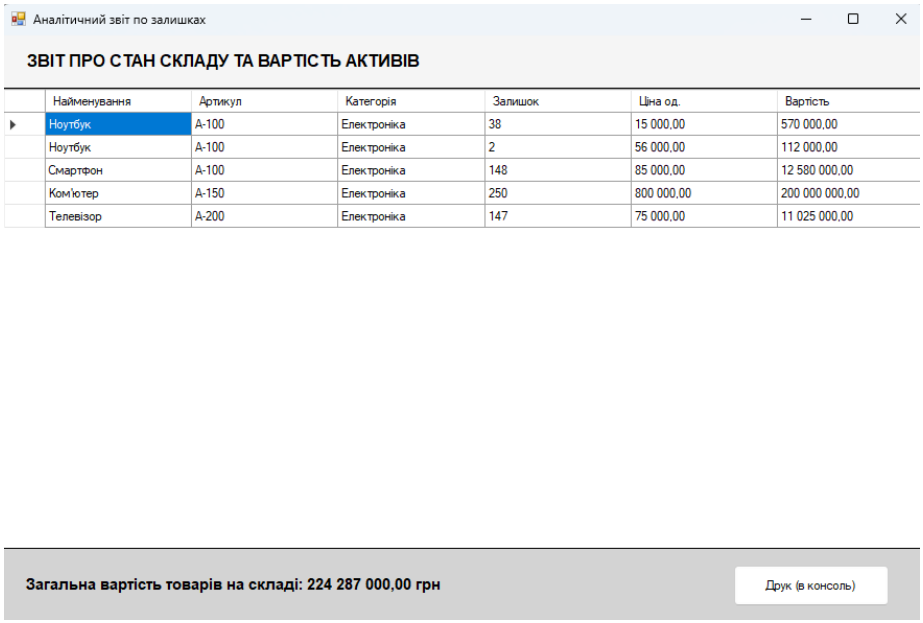

Рисунок 2.23 – Вигляд сторінки інформаційної системи «Аналітичний звіт по залишкам»

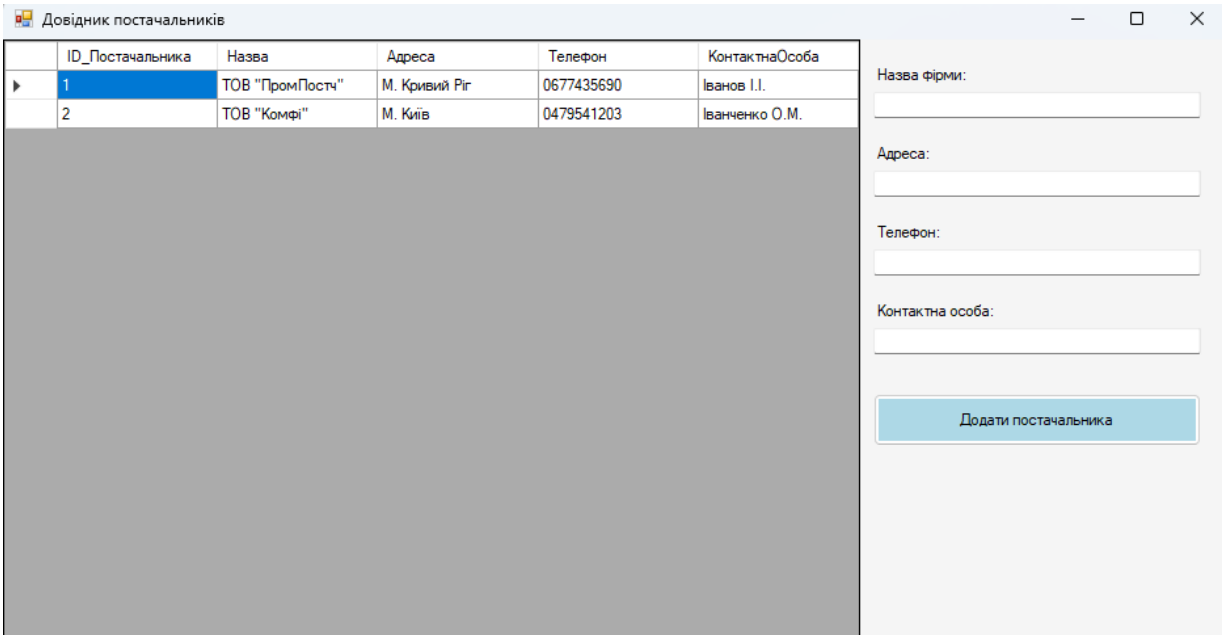

Рисунок 2.24 – Вигляд сторінки інформаційної системи «Довідник постачальників»

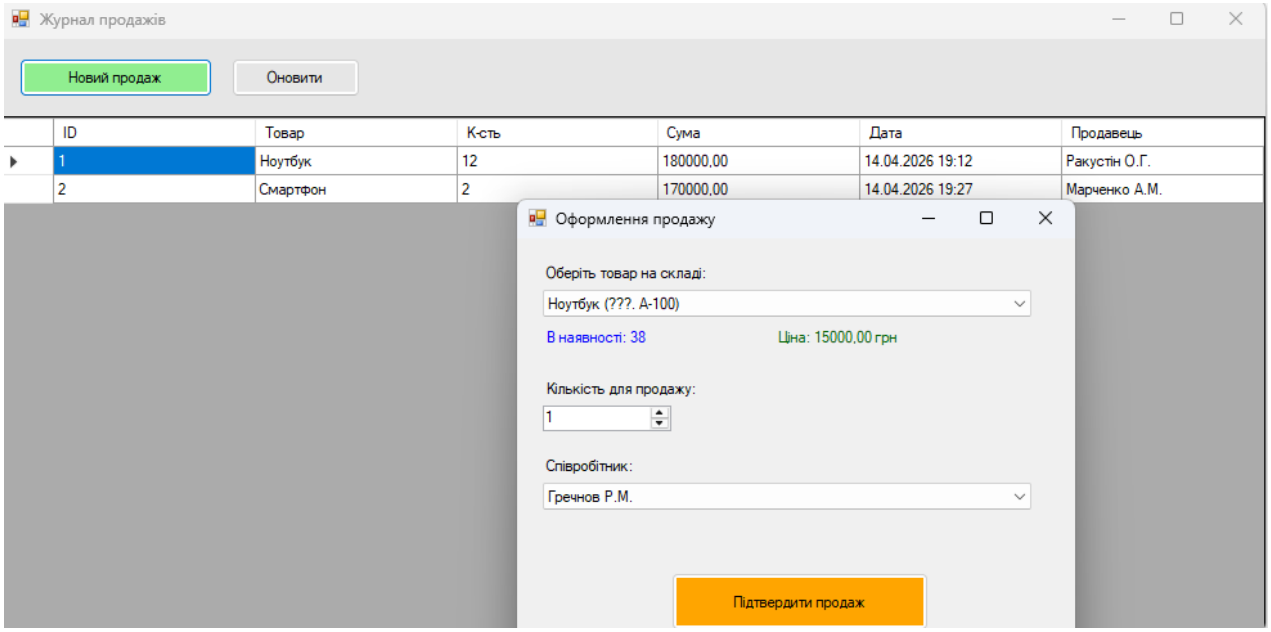

Рисунок 2.25 – Вигляд сторінки інформаційної системи «Журнал продажів»

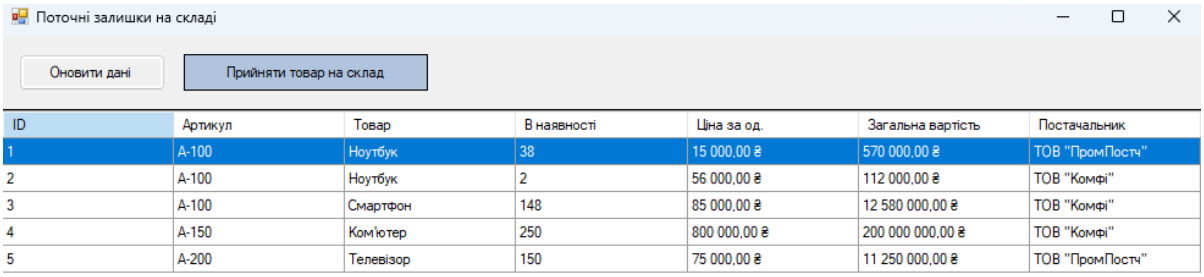

Рисунок 2.26 – Вигляд сторінки інформаційної системи «Поточні залишки на складі»

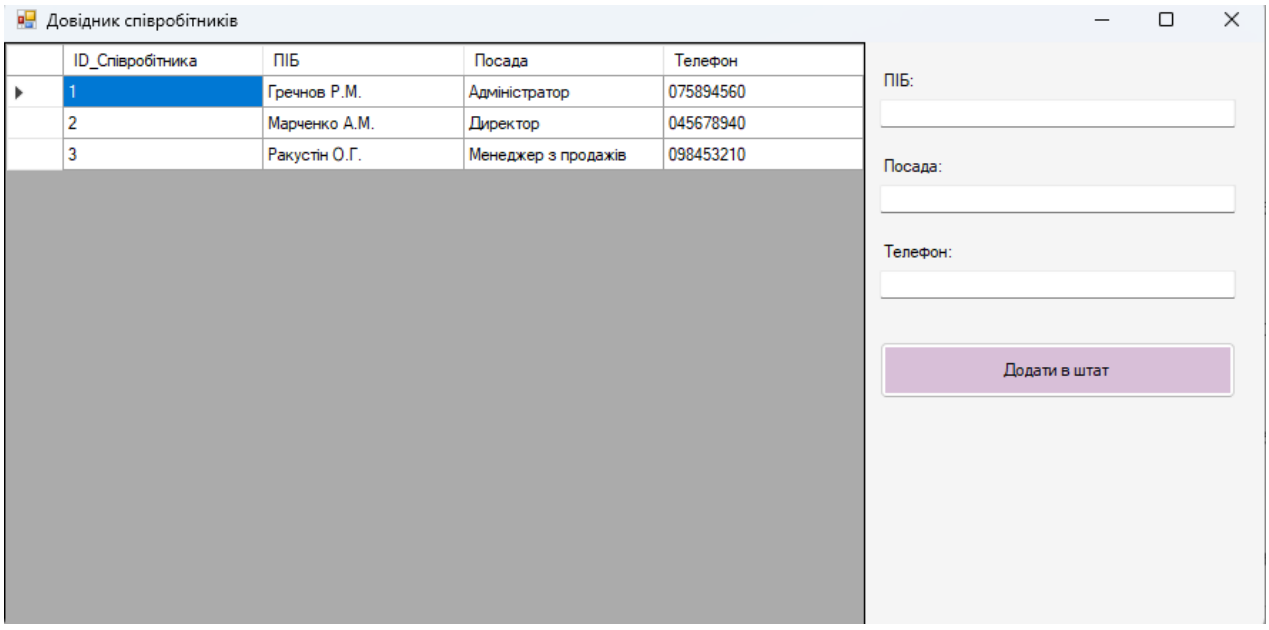

Рисунок 2.27 – Вигляд сторінки інформаційної системи «Довідник співробітників»

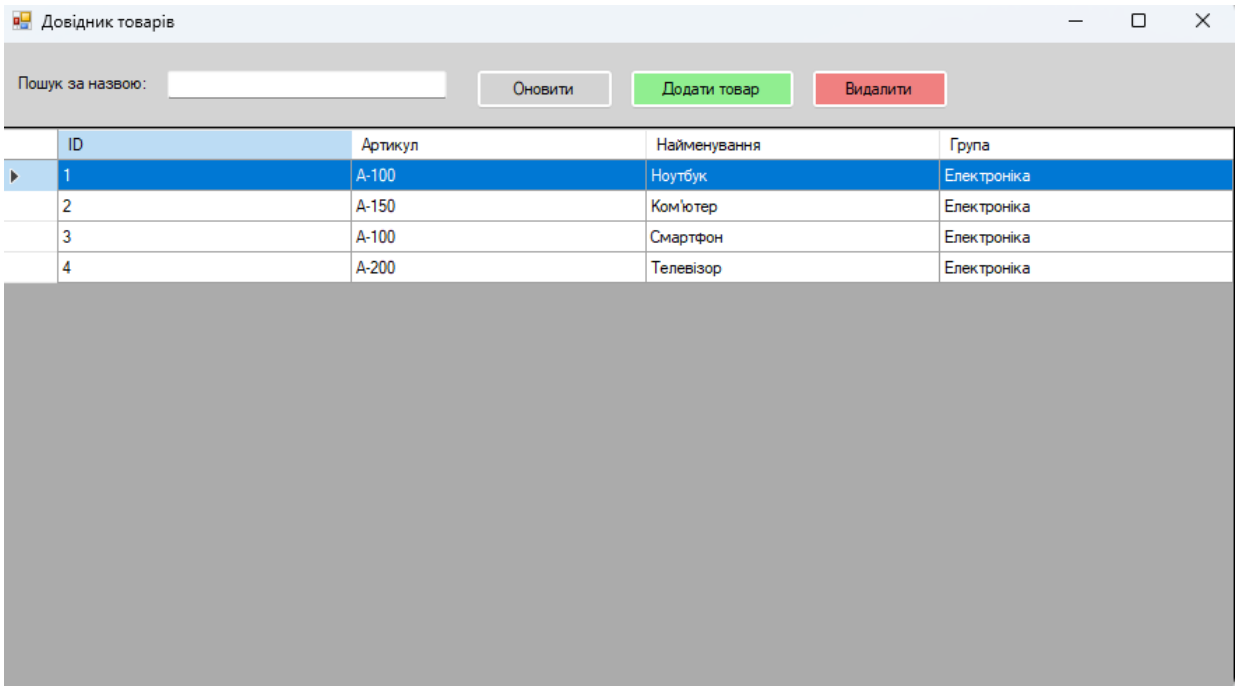

Рисунок 2.28 – Вигляд сторінки інформаційної системи «Довідник товарів»

На представлених сторінках можна переглядати дані, видаляти рядки таблиці та додавати інформацію. Щоб переглянути зміни, необхідно натиснути кнопку «Оновити».

2.7 Тестування програмної системи

Методика перевірки за принципом «чорної скриньки» являє собою підхід до аналізу програмного забезпечення, який також відомий як поведінкове або функціональне тестування. Цей інструментарій орієнтований на дослідження функціональних та нефункціональних характеристик продукту без заглиблення у його внутрішню архітектуру чи специфіку програмної реалізації. Для виконання такого циклу випробувань фахівцеві не обов’язково володіти навичками розробки або мати безпосередній доступ до вихідного коду. Дана концепція була створена для верифікації відповідності системи запитам користувачів та технічним специфікаціям. Під час роботи тестувальник формує вибірку коректних та помилкових вхідних параметрів, після чого аналізує валідність отриманої відповіді. Оскільки внутрішні механізми функціонування залишаються прихованими, оцінюється виключно зовнішня реакція середовища та коректність генерованих вихідних даних.

На противагу вищеописаному підходу, тестування «білої скриньки» передбачає детальний аудит програмного коду та внутрішньої логіки продукту. Цей метод найчастіше застосовується на етапі модульної перевірки окремих компонентів, хоча його використання можливе і під час інтеграційних випробувань. Для успішної реалізації такої стратегії виконавець повинен мати ґрунтовні знання у технологічному стеку, що використовувався під час написання програми.

Гібридом цих двох концепцій є методика «сірої скриньки». Вона базується на поєднанні підходів і потребує від фахівця часткового розуміння архітектурних рішень та внутрішніх алгоритмів системи. Застосування цього методу сприяє кращому покриттю коду тестами, оскільки дозволяє розробляти сценарії, що охоплюють різні логічні шляхи виконання програми. Проте ключовою особливістю залишається обмежений доступ до вихідних файлів, через що підготовка ефективних тестових ситуацій може потребувати значних часових витрат.

Для практичного випробування функціональності системи додаємо кілька нових записів до бази даних через призначений для користувача інтерфейс у таблицю «Довідник товарів». Відкриємо відповідну екранну форму в додатку, яку було проілюстровано на рисунку 2.28.

При натисканні кнопки «Додати» інформація повинна додатися до таблиці. Щоб переглянути зміну, натисніть кнопку «Оновити».

Виділимо доданий рядок і при натисканні кнопки «Видалити» цей запис буде видалено. Після натискання кнопки «Оновити» рядок буде видалено.

На рисунку 2.29 наведено приклад проведення продажу товарів зі складу, а на рисунку 2.30 наведено сторінку на якій вже відображається продаж з попереднього рисунку.

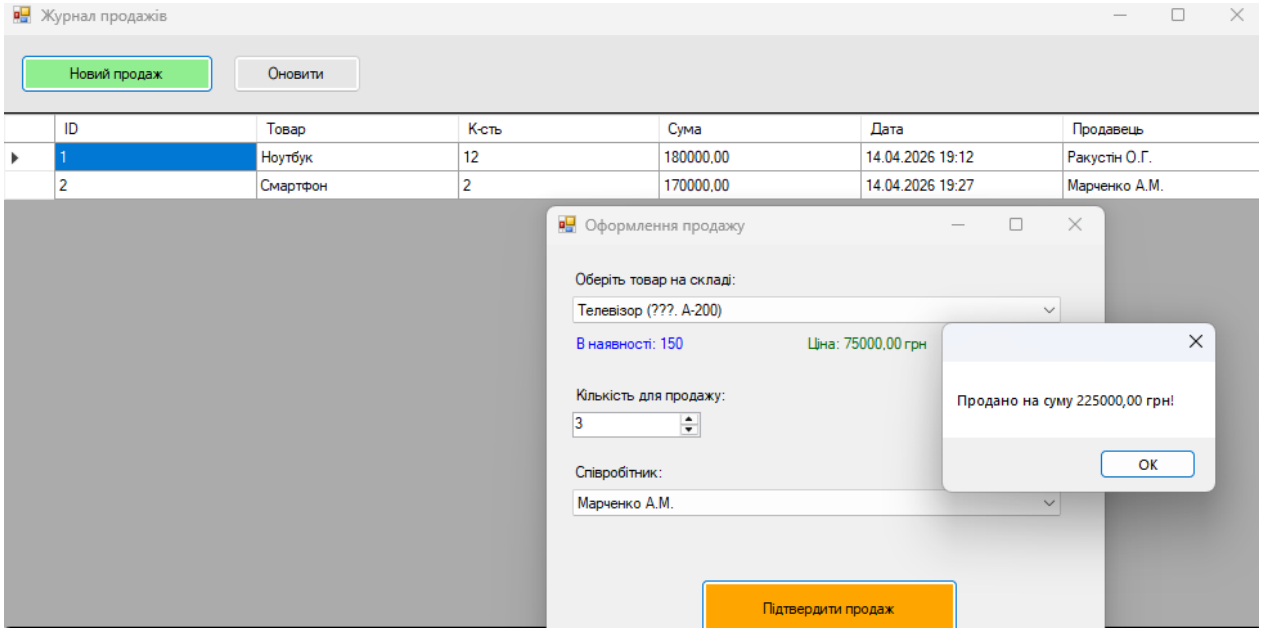

Рисунок 2.29– Вид тестової реалізації продажу товару зі складу

Після натискання кнопки «Підтвердження продажу» з'явиться оновлена таблиця на рисунку 2.30.

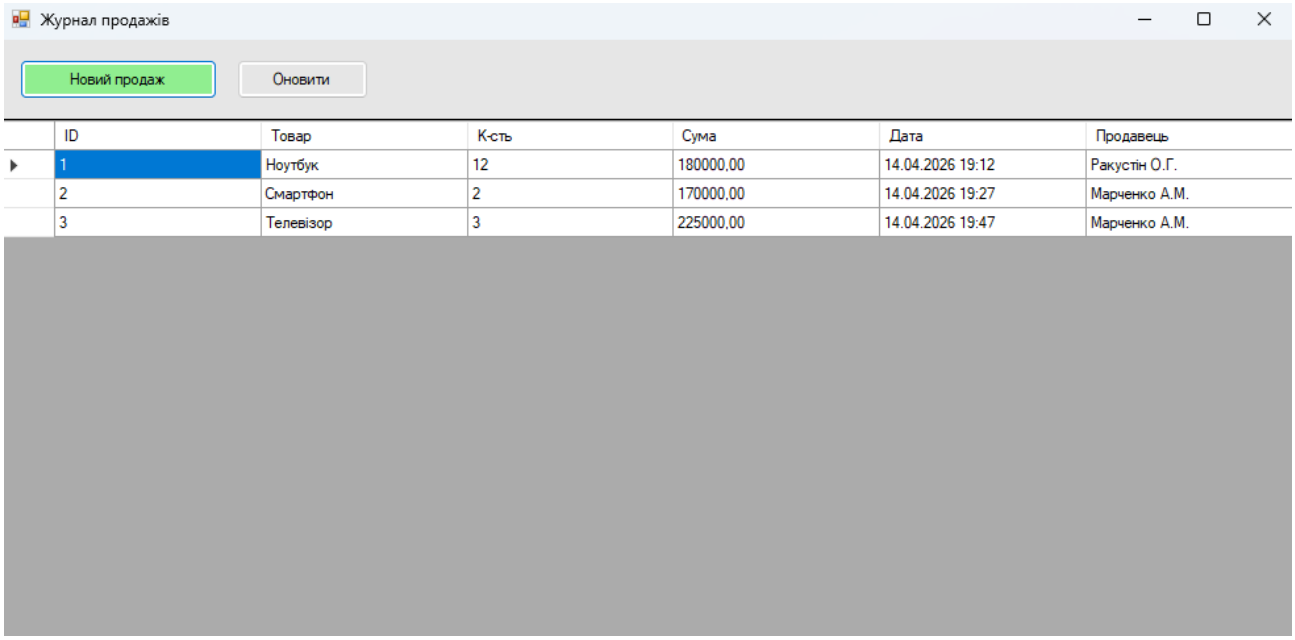

Рисунок 2.30 – Вид оновленої сторінки після проведеного продажу товару

Реалізований комплекс випробувань програмного забезпечення не виявив критичних помилок під час його експлуатації. Тестування виступає фундаментальною стадією життєвого циклу створення продукту, оскільки воно сприяє своєчасному виявленню дефектів, підвищує загальну якість розробки, прискорює етап практичного впровадження та мінімізує загрози втрати інформації користувачами. Окрім цього, верифікація допомагає налагодити ефективний зворотний зв'язок із персоналом та підвищити рівень задоволеності від роботи з інструментом. У цілому, процедура тестування забезпечує позитивні результати для всіх учасників процесу проектування.

Висновок до розділу:

У другому розділі було здійснено розробку та практичну реалізацію інформаційної системи для торгового підприємства. Розроблена архітектура забезпечує модульність та масштабованість програмного продукту, а проєктування структур зберігання даних у SQL Server гарантує ефективне та безпечне управління інформацією. Реалізація варіантів використання та алгоритмів їх роботи дозволяє автоматизувати ключові бізнес-процеси, а

проєктування класів і графічного інтерфейсу на C# забезпечує зручну взаємодію користувача з системою. Проведене тестування підтвердило стабільність, коректність функцій та відповідність системи поставленим вимогам, що робить її готовою до практичного впровадження

ВИСНОВКИ

Кінцевим продуктом даної роботи стало створення функціонального програмного засобу для цифровізації діяльності торговельного об'єкта. Сформована база даних орієнтована на максимальне спрощення процесів обробки, внесення та оперативного пошуку відомостей про товарний асортимент. Впровадження такої системи є необхідним для переходу до автоматизованого обліку, надійного зберігання та швидкого вилучення даних про продукцію, штатних працівників та контрагентів. Більш того, розробка власного програмного рішення дозволяє організації уникнути залежності від зовнішніх факторів, таких як раптове припинення технічної підтримки або анулювання ліцензій на готові продукти через регіональні обмеження.

У роботі проведено всебічний аналіз предметної області управління торговим підприємством та існуючих інформаційних систем, що дозволило визначити ключові бізнес-процеси - закупівлю, зберігання, продаж, облік та взаємодію з клієнтами - та потреби підприємства в автоматизації документообігу, контролі фінансових потоків і оптимізації управлінських операцій. Було проаналізовано існуючі рішення, такі як Airbase, Onspring, SAP та Tradeshift, та обґрунтовано доцільність розробки власної системи на основі комбінації C# і SQL Server, що забезпечує надійність, швидкість роботи та можливість подальшого розвитку.

У другому розділі здійснено практичну реалізацію інформаційної системи, включно з проєктуванням архітектури, структур бази даних, алгоритмів та класів, розробкою графічного інтерфейсу й тестуванням системи. Реалізовані функції автоматизують ключові бізнес-процеси, забезпечують ефективну взаємодію користувача з системою та гарантують стабільність і коректність роботи програмного комплексу.

Практична цінність роботи полягає у створенні програмного засобу, який дозволяє підвищити продуктивність праці, зменшити вплив людського фактору, забезпечити своєчасне формування точних управлінських даних і

підтримку стратегічного розвитку підприємства. Запропоноване рішення може ефективно застосовуватися як у масштабних організаціях, так і в підприємствах середнього рівня, забезпечуючи інтеграцію інформаційних потоків та оптимізацію управлінських процесів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1 A. Sharma, M. Tiwari. Optimization of warehouse layout using genetic algorithms / *International Journal of Production Research* / 2015. Vol. 53, No. 13. P. 4020–4032.

2 Лук’яненко С.О. Числові методи в інформатиці: Навч. посіб. Київ: “Видавництво “Політехніка”, 2007. 140с.

3 Харрісон Алан, Ван Хоук Ремко. Управління логістикою: Розробка стратегій логістичних операцій. Дніпропетровськ: Баланс Бізнес Букс, 2007.

4 Шилдт, Г. Програмування на C#, 6-е вид. Київ : Видавництво "Основа", 2018. 784 с.

5 Платформа Airbase URL: https://www.airbase.com/(дата звернення 15.03.2026)

6 Платформа Onspring URL: https://onspring.com/platform/overview/ (дата звернення 15.03.2026)

7 Платформа SAP URL: https://www.sap.com/ukraine/index.html (дата звернення 15.03.2026

8 Платформа Tradeshift URL: https://go.tradeshift.com/?Current Screen=0 (дата звернення 15.03.2026)

9 Албахарі, Дж. C# 9.0 in a Nutshell: The Definitive Reference, 1-е вид. Нью-Йорк : O'Reilly Media, 2020. 1184 с.

10 Тротман, Е. C# для професіоналів. Київ : Видавництво "Основа", 2017. 512 с.

11 Microsoft SQL Server - URL:https://www.techtarget.com/searchdata management/definition/SQL-Server (дата звернення 15.04.2026).

12 SQL Server – is a relational database management system (RDBMS) URL: https://www.sqlsplus.com/sql-server-is-a-relational-database-managemen tsystem-rdbms/ (дата звернення 15.04.2026).

13 What is a relational database? - URL: https://azure.microsoft.com/enus/resources/cloud-computing-dictionary/what-is-a-relational-database (дата звернення 15.04.2026).

14 What Is SQL Server – Definition And Architecture - URL: https://analyticsplanets.com/what-is-sql-server-definition-and-architecture-details/ (дата звернення 10.05.2026).

15 What is SQL Server- URL: https://www.sqlservertutorial.net/gettingstarted/what-is-sql-server/ (дата звернення 10.05.2026).

16 Ambler, S. The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge: Cambridge University Press, 2004.

17 Fowler, M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley Professional, 2002.

18 Pressman, R. S., Maxim, B. R. Software Engineering: A Practitioner’s Approach. New York: McGraw-Hill Education, 2014.

19 Sommerville, I. Software Engineering. 10th ed. Pearson, 2015.

20 Silberschatz, A., Korth, H. F., Sudarshan, S. Database System Concepts. 7th ed. McGraw-Hill, 2020.

21 Elmasri, R., & Navathe, S. B. Fundamentals of Database Systems. 7th ed. Pearson, 2016.

22 Connolly, T., & Begg, C. Database Systems: A Practical Approach to Design, Implementation and Management. 6th ed. Pearson, 2015.

23 Laudon, K. C., & Laudon, J. P. Management Information Systems: Managing the Digital Firm. 16th ed. Pearson, 2020.

24 Hoffer, J. A., Ramesh, V., & Topi, H. Modern Database Management. 13th ed. Pearson, 2016.

25 Larman, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed. Prentice Hall, 2004.

26 Моркун Н. В., Завсегдашня І. В. Методичні вказівки до виконання кваліфікаційної роботи бакалавру для студентів спеціальності 122 “Комп’ютерні науки”. Кривий Ріг : Видавничий центр КНУ, 2021. 50 с.

27 ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

28 ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

29 ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація).

ДОДАТОК А

Лістинг коду головної форми IC Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApp1
{
   public partial class Form1 : Form
   {
      // Елементи меню
      private MenuStrip mainMenuStrip;
      private ToolStripMenuItem menuDocuments;
      private ToolStripMenuItem menuDirectories;
      private ToolStripMenuItem menuReports;
      private Label lblWelcome;
      public Form1()
      {
         // Налаштування вікна
         this.Text = "Автоматизація Складу - Головна";
         this.Size = new Size(800, 600);
         this.StartPosition = FormStartPosition.CenterScreen;
         InitCustomControls();
      }
      private void InitCustomControls()
      {
         // 1. Налаштування вікна
         this.Text = "Автоматизація Складу - Головна";
         this.Size = new Size(900, 600);
         this.MinimumSize = new Size(800, 500);
         this.StartPosition = FormStartPosition.CenterScreen;
         this.BackColor = Color.FromArgb(250, 251, 252);
         // 2. Головне меню
         mainMenuStrip = new MenuStrip();
         mainMenuStrip.BackColor = Color.White;
         menuDocuments = new ToolStripMenuItem("Документи");
         menuDocuments.DropDownItems.Add("Приймання товару", null, (s, e)
=> { new InboundForm().ShowDialog(); });
         menuDocuments.DropDownItems.Add("Відвантаження товару", null, (s,
e) => { new SalesForm().ShowDialog(); });
```

```
            menuDocuments.DropDownItems.Add("Продажі", null, (s, e) => { new
SalesForm().ShowDialog(); });
            menuDirectories = new ToolStripMenuItem("Довідники");
            menuDirectories.DropDownItems.Add("Товари", null, (s, e) => { new
ProductsForm().ShowDialog(); });
            menuDirectories.DropDownItems.Add("Постачальники", null, (s, e) => {
new SuppliersForm().ShowDialog(); });
            menuDirectories.DropDownItems.Add("Співробітники", null, (s, e) => {
new EmployeesForm().ShowDialog(); });
            menuDirectories.DropDownItems.Add("Склад", null, (s, e) => { new
WarehouseForm().ShowDialog(); });
            menuReports = new ToolStripMenuItem("Звіти");
            menuReports.DropDownItems.Add("Звіт по залишках", null, (s, e) => {
new ReportForm().ShowDialog(); });
            mainMenuStrip.Items.AddRange(new ToolStripItem[] { menuDocuments,
menuDirectories, menuReports });
            this.MainMenuStrip = mainMenuStrip;
            this.Controls.Add(mainMenuStrip);
            // 3. Панель для тексту
            TableLayoutPanel centerLayout = new TableLayoutPanel();
            centerLayout.Dock = DockStyle.Fill;
            centerLayout.ColumnCount = 1;
            centerLayout.RowCount = 4; // 4 рядки для гнучкості
            centerLayout.ColumnStyles.Add(new ColumnStyle(SizeType.Percent,
100));
            // Оформлення
            centerLayout.RowStyles.Add(new RowStyle(SizeType.Percent, 50)); //
Верхній відступ
            centerLayout.RowStyles.Add(new RowStyle(SizeType.AutoSize)); //
Рядок для заголовка
            centerLayout.RowStyles.Add(new RowStyle(SizeType.AutoSize)); //
Рядок для підзаголовка
            centerLayout.RowStyles.Add(new RowStyle(SizeType.Percent, 50)); //
Нижній відступ (рівний верхньому)
            this.Controls.Add(centerLayout);

            Label lblTitle = new Label();
            lblTitle.Text = "Склад Плюс";
            lblTitle.Font = new Font("Segoe UI Light", 52, FontStyle.Italic);
            lblTitle.ForeColor = Color.FromArgb(52, 73, 94);
            lblTitle.AutoSize = true;
```

```
            lblTitle.Anchor = AnchorStyles.None;

            Label lblSubtitle = new Label();
            lblSubtitle.Text = "інформаційна система керування торговельним підприємством";
            lblSubtitle.Font = new Font("Segoe UI", 12, FontStyle.Italic);
            lblSubtitle.ForeColor = Color.FromArgb(149, 165, 166);
            lblSubtitle.AutoSize = true;
            lblSubtitle.Anchor = AnchorStyles.None;

            centerLayout.Controls.Add(lblTitle, 0, 1);
            centerLayout.Controls.Add(lblSubtitle, 0, 2);
            // 4. Кнопки
            Button btnSettings = new Button();
            btnSettings.Text = "⚙ Налаштування БД";
            btnSettings.Size = new Size(160, 40);
            btnSettings.Location = new Point(30, this.ClientSize.Height - 60);
            btnSettings.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;
            btnSettings.FlatStyle = FlatStyle.Flat;
            btnSettings.FlatAppearance.BorderColor = Color.LightGray;
            btnSettings.Click += (s, e) => { new ConnectForm().ShowDialog(); };
            this.Controls.Add(btnSettings);
            btnSettings.BringToFront(); // Поверх панелі
            Button btnExit = new Button();
            btnExit.Text = " ✕ Вихід";
            btnExit.Size = new Size(110, 40);
            btnExit.Location = new Point(this.ClientSize.Width - 140, this.ClientSize.Height - 60);
            btnExit.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;
            btnExit.FlatStyle = FlatStyle.Flat;
            btnExit.FlatAppearance.BorderColor = Color.FromArgb(231, 76, 60);
            btnExit.Click += (s, e) => Application.Exit();
            this.Controls.Add(btnExit);
            btnExit.BringToFront(); // Поверх панелі
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

ДОДАТОК Б

Лістинг коду модуля AddProductForm.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApp1
{
   public partial class AddProductForm : Form
   {
      private TextBox txtArticul, txtName;
      private ComboBox cbGroup;
      private Button btnSave;
      public AddProductForm()
      {
         this.Text = "Додати новий товар";
         this.Size = new Size(400, 300);
         this.StartPosition = FormStartPosition.CenterParent;
         this.FormBorderStyle = FormBorderStyle.FixedDialog;
         InitUI();
         LoadGroups(); // Завантаження категорії при створенні форми
      }
      private void InitUI()
      {
         Label lbl1 = new Label { Text = "Артикул:", Location = new Point(20, 20), AutoSize
= true };
         txtArticul = new TextBox { Location = new Point(20, 40), Width = 340 };
         Label lbl2 = new Label { Text = "Назва товару:", Location = new Point(20, 80),
AutoSize = true };
         txtName = new TextBox { Location = new Point(20, 100), Width = 340 };
         Label lbl3 = new Label { Text = "Категорія (Група):", Location = new Point(20, 140),
AutoSize = true };
         cbGroup = new ComboBox { Location = new Point(20, 160), Width = 340,
DropDownStyle = ComboBoxStyle.DropDownList };
         btnSave = new Button { Text = "Зберегти", Location = new Point(130, 210), Size =
new Size(120, 35), BackColor = Color.LightBlue };
         btnSave.Click += BtnSave_Click;
```

```
            this.Controls.AddRange(new Control[] { lbl1, txtArticul, lbl2, txtName, lbl3, cbGroup, btnSave });
        }
        // Метод витягування списку груп з бази
        private void LoadGroups()
        {
            try
            {
                using (SqlConnection conn = DatabaseHelper.GetConnection())
                {
                    string query = "SELECT ID_Группи, Назва FROM ГруппиТоварів";
                    SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
                    DataTable dt = new DataTable();
                    adapter.Fill(dt);
                    cbGroup.DataSource = dt;
                    cbGroup.DisplayMember = "Назва";    // Що бачить користувач
                    cbGroup.ValueMember = "ID_Группи"; // Що ми реально відправляємо в SQL
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка завантаження груп: " + ex.Message);
            }
        }
        private void BtnSave_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtName.Text) || cbGroup.SelectedValue == null)
            {
                MessageBox.Show("Заповніть назву та оберіть групу!");
                return;
            }
            try
            {
                using (SqlConnection conn = DatabaseHelper.GetConnection())
                {
                    string query = "INSERT INTO Товари (Артикул, Назва, ID_Группи) VALUES (@art, @name, @groupId)";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@art", txtArticul.Text);
                    cmd.Parameters.AddWithValue("@name", txtName.Text);
```

```
                    cmd.Parameters.AddWithValue("@groupId", cbGroup.SelectedValue);
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Товар успішно додано!");
                    this.DialogResult = DialogResult.OK; // Сигнал для головної форми, що треба
оновити таблицю
                    this.Close();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при збереженні: " + ex.Message);
            }
        }
        private void AddProductForm_Load(object sender, EventArgs e)
        {
        }
    }
}
```