

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
(повна назва)

Кафедра _____ Інформаційних управляючих систем _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти _____ другий (магістерський) _____

Дослідження методів управління життєвим циклом ІТ-проєкту зі створення
інформаційної системи для автоматизації складських операцій
(тема)

Виконав:

здобувач _____ 2 _____ року навчання,
групи _____ УПГІТМ-24-1 _____

Данило ІЩЕНКО

(власне ім'я, прізвище)

Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва спеціальності)

Тип програми _____ освітньо-наукова _____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____ Управління проєктами в
галузі інформаційних технологій _____

Керівник _____ проф каф ІУС Максим Євланов _____
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ ІУС _____



(підпис)

_____ Костянтин ПЕТРОВ _____

(власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
 Кафедра _____ Інформаційних управляючих систем _____
 Рівень вищої освіти _____ другий (магістерський) _____
 Спеціальність _____ 122 Комп'ютерні науки _____
 (код і повна назва)
 Тип програми _____ освітньо-наукова _____
 (освітньо-професійна або освітньо-наукова)
 Освітня програма _____ Управління проектами в галузі інформаційних технологій _____
 (повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

« 06 » квітня 20 26 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Іщенко Данилу Григорійовичу _____
 (прізвище, ім'я, по батькові)

1. Тема роботи _____ Дослідження методів управління життєвим циклом ІТ-проєкту зі створення інформаційної системи для автоматизації складських операцій _____

затверджена наказом університету від _____ 23 березня 2026 р. № 231Ст _____

2. Термін подання здобувачем роботи до екзаменаційної комісії _____ 15 травня 2026 р. _____

3. Вихідні дані до роботи _____ матеріали звіту з передатестаційної практики; науковотехнічні публікації та інтернет-джерела з тематики кваліфікаційної роботи _____

4. Перелік питань, що потрібно опрацювати в роботі _____ Аналіз предметної області складської логістики та проблем управління відповідними ІТ-проєктами. Дослідження та порівняльний аналіз каскадних і гнучких методів управління життєвим циклом програмного забезпечення. Обґрунтування розробки гібридної моделі управління (Water-Scrum-Fall) на базі концепції «Двомодальних ІТ». Моделювання бізнес-процесів (IDEF0, DFD), аналіз організаційної структури замовника та управління вимогами стейкхолдерів. Розробка ієрархічної структури робіт (WBS). Опис управлінських рішень щодо проєктування інфраструктури баз даних (MySQL) та вибору технологічного стека. Управління ітеративною розробкою програмного забезпечення та впровадженням практик DevOps. Ідентифікація проєктних ризиків, розробка стратегій реагування та управління
 якістю на етапі верифікації проєкту.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	02.03.2024-17.03.2026	Виконано
2	Дослідження технологій предметної області	18.03.2024-30.03.2026	Виконано
3	Постановка задачі дослідження	01.04.2024-06.04.2026	Виконано
4	Теоретичне дослідження обраних технологій	07.04.2024-12.04.2026	Виконано
5	Практичне дослідження обраних технологій	13.04.2024-17.04.2026	Виконано
6	Розробка тестового прототипу	18.04.2024-28.04.2026	Виконано
7	Аналіз недоліків та переваг тестового	14.04.2024-28.04.2026	Виконано
8	Моделювання висновків за виконаною роботою	29.04.2024-17.04.2026	Виконано
9	Написання пояснювальної записки	01.04.2024-29.04.2026	Виконано
10	Перевірка кваліфікаційної роботи нормоконтролем	10.05.2026	Виконано
11	Перевірка кваліфікаційної роботи на плагіат	14.05.2026	Виконано
12	Захист кваліфікаційної роботи в екзаменаційній комісії	20.05.2026	Виконано

Дата видачі завдання 06 квітня 2026 р.

Здобувач


(підпис)

Керівник роботи


(підпис)

проф каф ІУС Максим Євланов
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 79 с., 2 табл., 17 рис., 20 джерел.

АВТОМАТИЗАЦІЯ СКЛАДУ, БІЗНЕС-АНАЛІТИКА, ГІБРИДНА МОДЕЛЬ, ЖИТТЄВИЙ ЦИКЛ, МАТРИЦЯ РИЗИКІВ, УПРАВЛІННЯ ПРОЄКТАМИ, WATER-SCRUM-FALL, WBS, WMS.

Об'єктом дослідження в рамках магістерської роботи є процес управління життєвим циклом ІТ-проєкту зі створення корпоративних інформаційних систем.

Предметом дослідження є методи, моделі та інструментальні засоби управління розробкою та впровадженням системи автоматизації складських операцій.

Метою дослідницької роботи є дослідження та розробка методів управління життєвим циклом ІТ-проєкту для автоматизації складських операцій.

Методами дослідження виступають системний аналіз, предиктивне планування та гнучкі ітеративні підходи. В рамках виконання роботи був проведений аналіз предметної області, розглянуті функціональні особливості існуючих WMS-рішень та специфіка логістичного підприємства. Проведений аналіз організаційної структури, формалізовані вимоги до системи та сформульована постановка задачі управління проєктом. Також було обране та обґрунтоване технічне забезпечення для реалізації поставленої задачі.

Сфера застосування дослідження охоплює складську логістику та управління процесами розробки корпоративних ІТ-систем для автоматизації руху товарів.

ABSTRACT

Explanatory note to the qualification work: 79 pages, 2 tables, 17 figures, 20 sources.

BUSINESS ANALYTICS, HYBRID MODEL, LIFE CYCLE, PROJECT MANAGEMENT, RISK MATRIX, WAREHOUSE AUTOMATION, WATER-SCRUM-FALL, WBS, WMS.

The object of the study within the master's work is the IT project life cycle management process for creating corporate information systems.

The subject of the study includes the methods, models, and tools for managing the development and implementation of a warehouse management system (WMS).

The aim of the research work is to investigate and develop IT project life cycle management methods for automating warehouse operations.

The research methods involve systems analysis, predictive planning, and agile iterative approaches. As part of the work, an analysis of the subject area was carried out, the functional features of existing WMS solutions were considered, and the specifics of the logistics enterprise were examined. The organizational structure was analyzed, system requirements were formalized, and the project management problem statement was formulated. The technology stack for the implementation of the task was also selected and justified.

The scope of the study covers warehouse logistics and the management of corporate IT system development processes for automating the movement of goods.

ЗМІСТ

	С.
Скорочення та умовні позначки	8
Вступ.....	10
1 Формування та аналіз задачі дослідження методів управління іт-проєктом зі створення wms-системи.....	13
1.1 Аналіз актуальності задачі дослідження методів управління ІТ-проєктом зі створення WMS-системи.....	13
1.2 Аналіз існуючих рішень для автоматизації складських операцій	14
1.3 Аналіз існуючих методів та моделей управління життєвим циклом ІТ-проєктів	16
1.4 Обґрунтування мети створення гібридного методу та моделі ІТ-проєкту для автоматизації складу	18
1.5 Постановка задач магістерської кваліфікаційної роботи дослідження моделей і методів ІТ-проєкта.....	21
2 Теоретичне обґрунтування гібридної моделі управління життєвим циклом складських іт-проєктів.....	23
2.1 Аналіз сучасних теоретичних моделей та методів управління життєвим циклом складських ІТ-проєктів	23
2.2 Дослідження новітніх технологій та інноваційних підходів до управління ІТ-проєктами.....	25
2.2.1 Огляд інноваційних підходів та управлінських технологій	25
2.2.2 Детальний аналіз технологічних тенденцій та їх вплив на життєвий цикл	26
2.3 Оцінка інноваційного характеру та потенціалу технологічних рішень у сфері управління WMS	28
2.4 Вибір оптимальних методів та моделей для подальшого дослідження	30
2.4.1 Переваги обраної гібридної моделі управління.....	32
2.4.2 Обґрунтування вибору інтегрованого підходу	33

3 Практична реалізація етапів життєвого циклу іт-проєкту створення системи автоматизації складу	34
3.1 Практичне моделювання бізнес-процесів як інструмент фіксації обсягу проєкту	34
3.2 Моделювання потоків даних (DFD) та формування вимог до системи	37
3.3 Проєктування інформаційної інфраструктури та бази даних	39
3.4 Розробка ієрархічної структури робіт (WBS) та планування ітерацій	41
3.5 Управління якістю та методологія верифікації проєкту	44
4 Опис практичного застосування етапів життєвого циклу іт-проєкту створення системи автоматизації складу	47
4.1 Аналіз предметної галузі та організаційної структури об'єкта автоматизації	47
4.2 Практичне застосування методів управління інфраструктурними рішеннями та технологічним стеком	49
4.3 Управління якістю та ризиками цілісності даних на рівні бази даних.....	50
4.4 Управління ітеративною розробкою користувацьких інтерфейсів	52
4.5 Управління проєктними комунікаціями та формування управлінської звітності.....	56
4.6 Практичне управління інформаційною безпекою та процесами верифікації	57
4.7 Управління проєктними ризиками та стратегії реагування.....	58
4.8 Висновки до четвертого розділу.....	59
Висновки	60
Перелік джерел посилання	62
Додаток А. Графічний матеріал	64

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІС – інформаційна система

ІТ – інформаційні технології

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ТЗД – термінал збору даних

ACID – Atomicity, Consistency, Isolation, Durability

API – Application Programming Interface

CI/CD – Continuous Integration / Continuous Deployment

DevOps – Development and Operations

DFD – Data Flow Diagram

EER – Enhanced Entity-Relationship

ERP – Enterprise Resource Planning

IaC – Infrastructure as Code

IDEF0 – Integration Definition for Function Modeling

IPMA – International Project Management Association

JWT – JSON Web Token

KPI – Key Performance Indicator

MVP – Minimum Viable Product

OBS – Organizational Breakdown Structure

PMBOK – Project Management Body of Knowledge

PMI – Project Management Institute

QA – Quality Assurance

RBAC – Role-Based Access Control

ROI – Return on Investment

SDLC – Software Development Life Cycle

SKU – Stock Keeping Unit

UI / UX – User Interface / User Experience

UML – Unified Modeling Language

UUID – Universally Unique Identifier

WBS – Work Breakdown Structure

WMS – Warehouse Management System

ВСТУП

У сучасному світі логістика та управління ланцюгами постачань є невід'ємною частиною глобальної економіки, а ефективна робота складських комплексів має вирішальне значення для конкурентоспроможності будь-якого підприємства. Складські процеси забезпечують безперебійний рух товарів, доступність продукції для споживачів та оптимізацію витрат підприємства. Проте, з огляду на зростаючі обсяги товарообігу та складність сучасних логістичних операцій, їхня надійність, швидкість і безпомилковість потребують постійного контролю та впровадження новітніх технологій[1].

Одним із ключових інструментів підвищення ефективності роботи складу є розробка та впровадження інформаційних систем автоматизації складських операцій (Warehouse Management System, WMS). З розвитком інформаційних технологій відкрилися нові можливості для удосконалення процесів управління такими системами. Сучасні світові тенденції в ІТ-галузі свідчать про перехід до мікросервісних архітектур, використання технологій контейнеризації та гнучких методологій розробки, що дозволяє створювати високонавантажені системи, здатні швидко адаптуватися до змін бізнес-вимог.

Однак, створення систем класу WMS є надзвичайно складним ІТ-проєктом, який характеризується високим рівнем ризиків через необхідність тісної взаємодії програмного забезпечення з апаратним обладнанням (сканерами, терміналами збору даних) та інтеграції з існуючими корпоративними системами. Помилки в управлінні на етапах ініціації, збору вимог або впровадження можуть призвести до серйозних фінансових втрат. Саме тому дослідження методів управління життєвим циклом таких проєктів є надзвичайно актуальним завданням, що дозволяє запобігти зривам термінів та перевищенню бюджетів.

Ця кваліфікаційна робота присвячена дослідженню методів управління життєвим циклом ІТ-проєкту зі створення інформаційної системи для автоматизації складських операцій. Метою роботи є аналіз сучасних підходів до

управління проєктами та обґрунтування вибору оптимальних моделей і технологічного стека для підвищення ефективності розробки та впровадження WMS-систем.

У ході дослідження вирішуються такі ключові задачі: розглядаються різноманітні підходи до управління (каскадні та гнучкі моделі), аналізується специфіка складських IT-проєктів, розробляється ієрархічна структура робіт (Work Breakdown Structure, WBS), а також досліджуються методи управління проєктними ризиками та інфраструктурними рішеннями на різних фазах життєвого циклу. Особлива увага у роботі приділяється формуванню гібридного підходу управління, який дозволяє поєднати предиктивне планування архітектури з ітеративною розробкою функціоналу.

У результаті проведеного дослідження обґрунтовано вибір технологічних та управлінських рішень (зокрема, застосування Continuous Integration / Continuous Deployment, Docker та транзакційних баз даних), що демонструють високу ефективність у процесі реалізації складських проєктів. Виявлено, що комплексне застосування цих методів дозволяє значно покращити процес створення системи порівняно з традиційними підходами управління.

Отримані результати можуть бути використані як основа для створення сучасних та надійних систем автоматизації, які сприятимуть економічній ефективності експлуатації складів. Таким чином, ця кваліфікаційна робота є внеском у розвиток методологій управління IT-проєктами, демонструючи потенціал використання сучасних підходів для підвищення надійності та якості розробки корпоративного програмного забезпечення.

Об'єкт дослідження – процес управління життєвим циклом IT-проєкту зі створення корпоративних інформаційних систем.

Предмет дослідження – методи, моделі та інструментальні засоби управління розробкою та впровадженням WMS.

Методи дослідження. Для розв'язання поставлених у кваліфікаційній роботі завдань використано комплексний підхід, що включає: методи системного аналізу та функціонального моделювання (Integration Definition for

Function Modeling, IDEF0, Data Flow Diagram, DFD) – для формалізації вимог та опису бізнес-процесів складу; методи емпіричного управління процесами (Agile/Scrum) – для організації ітеративної розробки користувацьких інтерфейсів; методи предиктивного планування (Waterfall) – для проєктування архітектури баз даних; об'єктно-орієнтований підхід та інструментарій Unified Modeling Language (UML) – для проєктування програмної логіки; методи управління ризиками – для ідентифікації та мінімізації загроз на етапі впровадження [2].

Наукова новизна одержаних результатів полягає у подальшому розвитку методів управління логістичними ІТ-проєктами. Зокрема, набула подальшого розвитку концепція «Двомодальних ІТ» (Bimodal IT) через її адаптацію до специфіки розробки WMS-систем. Вперше запропоновано та обґрунтовано гібридну модель управління «Water-Scrum-Fall» у поєднанні з практиками Development and Operations (DevOps), що дозволяє усунути системний конфлікт між необхідністю забезпечення жорсткої транзакційної цілісності бази даних складу та потребою у гнучкій адаптації клієнтських інтерфейсів[3-4].

Практичне значення одержаних результатів. Запропоновані моделі та управлінські рішення доведено до стадії практичного застосування. Розроблена ієрархічна структура робіт (WBS), матриця управління ризиками та обраний технологічний стек (Java, Spring Boot, React, MySQL) можуть бути використані ІТ-компаніями як типовий шаблон (Baseline) для планування та реалізації реальних проєктів автоматизації складських комплексів [3].

1 ФОРМУВАННЯ ТА АНАЛІЗ ЗАДАЧІ ДОСЛІДЖЕННЯ МЕТОДІВ УПРАВЛІННЯ ІТ-ПРОЄКТОМ ЗІ СТВОРЕННЯ WMS-СИСТЕМИ

1.1 Аналіз актуальності задачі дослідження методів управління ІТ-проектом зі створення WMS-системи

У сучасному глобалізованому світі ефективність логістичних ланцюгів постачання є одним із найважливіших факторів, що визначають конкурентоспроможність як окремих підприємств, так і національних економік загалом. Зі стрімким розвитком електронної комерції (e-commerce), переходом до концепції багатоканального продажу та підвищенням очікувань споживачів щодо швидкості доставки, складські комплекси перетворилися з місць простого зберігання товарів на високодинамічні центри обробки та розподілу інформації і матеріальних потоків.

Актуальність дослідження методів та моделей управління життєвим циклом ІТ-проекту зі створення WMS визначається низкою ключових факторів, що формують сучасний ландшафт галузі.

Зростання складності логістичних бізнес-процесів – сучасні склади оперують десятками тисяч активних позицій номенклатури. Процеси приймання, комплектації за стратегіями хвильового або партійного відбору вимагають бездоганної алгоритмізації. Створення ІТ-системи, здатної керувати такими процесами, є складним проектом, що вимагає застосування передових методів управління життєвим циклом для уникнення зриву строків розробки та перевищення бюджету.

Глибока інтеграція з апаратним забезпеченням (Hardware Integration) – на відміну від класичних веб-додатків, системи WMS тісно взаємодіють із фізичним світом. Це вимагає інтеграції з терміналами збору даних (ТЗД), принтерами етикеток, сканерами штрих-кодів, RFID-зчитувачами, а в перспективі – з автоматизованими системами керування конвеєрами та роботами. Управління таким ІТ-проектом ускладнюється необхідністю синхронізації циклів розробки

програмного забезпечення з циклами постачання та налаштування обладнання.

Критичність систем для безперервності бізнесу – інформаційні системи управління складом належать до класу Mission Critical. Будь-який простий WMS або помилка під час її впровадження може призвести до повної паралізації відвантажень, накопичення черг транспорту та прямих фінансових і репутаційних збитків. Це підвищує вимоги до управління якістю (Quality Management) та управління ризиками на всіх етапах життєвого циклу IT-проєкту.

Технологічна еволюція та необхідність модернізації – багато підприємств досі використовують застарілі, монолітні або «самописні» облікові системи, які не здатні масштабуватися під зростаючі навантаження. Проєкти з міграції на сучасні платформи (з використанням мікросервісної архітектури, транзакційних баз даних та контейнеризації) є надзвичайно ризикованими. Вони потребують чіткої методології управління переходом, міграцією даних та адаптацією персоналу.

Підсумовуючи, враховуючи критичну важливість складської логістики, високі вимоги до обробки транзакцій та глибоку апаратно-програмну інтеграцію, заглиблення у вивчення методів і моделей управління IT-проєктами класу WMS є обов'язковим та вкрай актуальним завданням для забезпечення успішної цифрової трансформації підприємств.

1.2 Аналіз існуючих рішень для автоматизації складських операцій

Для обґрунтування необхідності вдосконалення методів управління IT-проєктами в галузі складської логістики необхідно проаналізувати існуючий ринок рішень WMS. Зважаючи на різноманітність існуючих систем, їх можна класифікувати за архітектурним підходом та стратегією впровадження, що безпосередньо впливає на життєвий цикл проєкту.

Модулі WMS у складі глобальних систем Enterprise Resource Planning

(ERP) (наприклад, SAP EWM, Oracle WMS). Ці рішення є частиною великих корпоративних екосистем. Вони забезпечують безшовну інтеграцію з фінансовим та кадровим обліком підприємства. Перевагами даного рішення є висока надійність, стандартизовані процеси, потужна аналітика. Недоліками ж з точки зору управління проектом є те, що життєвий цикл впровадження таких систем базується на суворій каскадній моделі (Waterfall) і може тривати від 1 до 3 років. Проекти є надзвичайно дорогими, неповороткими та супроводжуються величезним обсягом документації, що робить їх недоступними для середнього бізнесу. Зміни вимог на пізніх етапах майже неможливі без значних фінансових втрат.

Вузькоспеціалізовані "Best-of-Breed" системи (наприклад, Manhattan Associates, Blue Yonder). Це потужні, автономні системи, створені виключно для управління складом, які мають найсучасніші алгоритми оптимізації маршрутів та розміщення. Їх перевагами є найкращий у своєму класі функціонал, гнучкість налаштувань під специфічні складські потреби (наприклад, холодильні склади або склади небезпечних речовин). Недоліками з точки зору управління проектом є те, що життєвий цикл ускладнюється етапом інтеграції (Integration Phase). Оскільки система є зовнішньою, ІТ-команді доводиться витратити левову частку часу проекту на розробку шлюзів (Application programming interface, API), проміжних баз даних та шин повідомлень для синхронізації з існуючою корпоративною ERP-системою замовника. Це створює високі інтеграційні ризики.

Хмарні Software as a Service (SaaS) WMS-рішення. Системи, що надаються за моделлю підписки (SaaS) і розгортаються на серверах провайдера. Перевагами є швидкий старт проекту (Time-to-Market), відсутність необхідності розгортати власну серверну інфраструктуру, регулярні оновлення. Недоліками ж обмежені можливості кастомізації (зміни бізнес-логіки під унікальні процеси замовника). Проектний менеджер стикається з ризиками інформаційної безпеки та залежності від якості інтернет-з'єднання на складі, що критично впливає на швидкодію терміналів збору даних.

Індивідуальна (Custom) розробка WMS-системи. Розробка IT-рішення "з нуля" спеціально під потреби конкретного підприємства з використанням сучасного технологічного стека (наприклад, Java, Spring Boot, React, MySQL). Перевагами є повна відповідність бізнес-процесам замовника, відсутність ліцензійних платежів за кожного користувача, незалежність від вендора, можливість побудови сучасної мікросервісної архітектури. Недоліками з точки зору управління проектом є ті факти, що дане рішення вимагає найвищого рівня компетенції у сфері управління життєвим циклом (Project Life Cycle Management). Відсутність готового «коробкового» функціоналу означає, що команда повинна пройти всі фази: від збору первинних вимог і проєктування архітектури баз даних до програмування, навантажувального тестування та інтеграції з обладнанням.

Аналіз існуючих рішень показує, що незалежно від обраного типу системи, управління IT-проєктом її створення або впровадження стикається з низкою системних проблем: перевищення бюджетів, зрив календарних графіків через непередбачені проблеми з обладнанням та високий рівень опору персоналу складу новим технологіям. Це свідчить про недостатню ефективність класичних методів управління для проєктів такого класу.

1.3 Аналіз існуючих методів та моделей управління життєвим циклом IT-проєктів

Управління життєвим циклом проєкту визначає організаційну структуру, логічну послідовність виконання робіт, точки контролю та механізми адаптації команди до змін. В сучасній IT-індустрії існує кілька фундаментальних підходів та стандартів, які необхідно проаналізувати в контексті розробки складських систем.

Фундаментальним стандартом у галузі є керівництво Project Management

Body of Knowledge (PMBOK) від Інституту управління проєктами (Project Management Institute, PMI).[5,6] Згідно з PMBOK, проєктний менеджмент реалізується через 49 процесів, які згруповані у п'ять основних груп:

- група процесів ініціації – формальне визнання необхідності створення WMS-системи, розробка Статуту проєкту та первинна ідентифікація стейкхолдерів (директор з логістики, IT-директор, лінійний персонал складу);
- група процесів планування – найбільш об'ємна частина, що включає збір вимог, створення WBS, розробку розкладу (діаграма Ганта), планування бюджету, ресурсів та ідентифікацію ризиків;
- група процесів виконання – безпосередня розробка програмного забезпечення, написання коду, закупівля та налаштування складських сканерів і серверів;
- група процесів моніторингу та контролю – відстеження виконання робіт відносно базового плану, управління змінами та контроль якості розробки;
- група процесів закриття – формальна передача розробленої системи в промислову експлуатацію, підписання актів приймання-передачі та вивільнення команди.

Окрім процесного підходу, критичним є вибір моделі життєвого циклу розробки програмного забезпечення (Software Development Life Cycle, SDLC).

Предиктивні моделі (Waterfall / Каскадна модель). Ця модель передбачає жорстку послідовність фаз, де кожна наступна фаза починається лише після повного завершення попередньої (Аналіз → Дизайн → Розробка → Тестування → Впровадження). Переваги: Забезпечує високий рівень контролю, прозорість бюджету та деталізовану документацію. У проєктах WMS цей підхід чудово працює для апаратної частини (закупівля серверів, прокладка мережі на складі), де неможливо діяти ітеративно. Недоліки: фатальна негнучкість до змін. Якщо на етапі тестування виявляється, що логіка маршрутизації комірника, затверджена місяці тому на етапі дизайну, є незручною на практиці, вартість внесення змін у вже написаний код та спроектовану базу даних буде катастрофічною.

Адаптивні (Гнучкі) методології (Agile, Scrum, Kanban). Гнучкі методології змістили фокус із жорсткої документації на створення робочого продукту та тісну співпрацю із замовником. Життєвий цикл розбивається на короткі ітерації (спринти тривалістю 2-4 тижні).[7,8] Управління вимогами здійснюється через динамічний беклог продукту. Переваги: швидка адаптація до змін бізнес-вимог логістики. Замовник може тестувати окремі модулі (наприклад, тільки модуль приймання товару) вже через місяць після старту проєкту. Недоліки у контексті WMS: суто гнучкий підхід є небезпечним для проєктів автоматизації складів. Неможливо "ітеративно" проєктувати реляційну базу даних з високими вимогами Atomicity, Consistency, Isolation, Durability (ACID) або архітектуру серверної безпеки – помилки фундаменту на ранніх етапах спричинять колапс усієї системи згодом. Крім того, складно синхронізувати 2-тижневі спринти розробки із 3-місячним циклом закупівлі та постачання специфічного складського обладнання з-за кордону.

1.4 Обґрунтування мети створення гібридного методу та моделі ІТ-проєкту для автоматизації складу

Аналіз існуючих методологій доводить, що застосування виключно детермінованої (каскадної) або виключно емпіричної (гнучкої) моделі для розробки комплексної системи автоматизації складських операцій є неефективним і несе значні ризики. Каскадна модель не здатна впоратися з динамікою сучасних бізнес-вимог логістики, тоді як чистий Agile створює загрози для критичної інфраструктури, де помилки проєктування баз даних або апаратних інтеграцій є фатальними. Отже, виникає об'єктивна необхідність у створенні комбінованого (гібридного) методу управління життєвим циклом.

Для подолання виявленого системного конфлікту між потребою у жорсткій стабільності складської інфраструктури та динамічністю бізнес-процесів

логістики, пропонується синтезувати модель управління на засадах концепції Bimodal IT (Gartner) [4]. Метою застосування цієї стратегії є створення дворівневого механізму управління ризиками:

- стратегічна стабілізація (Режим 1) – спрямована на виключення критичних помилок у фундаментальних вузлах (архітектура бази даних (БД), ERP-інтеграції, апаратна частина), мета – мінімізація вартості змін на пізніх етапах через детерміноване планування за Waterfall;

- операційна адаптивність (Режим 2) – орієнтована на максимізацію користувацької цінності та UX-інновації, мета – забезпечити гнучкість розробки клієнтських інтерфейсів (React) та логіки (Spring Boot) через ітерації Scrum, що дозволяє системі еволюціонувати разом із потребами складу без шкоди для її цілісності.

Така дво-modalна архітектура управління дозволить не просто розробити програмне забезпечення (ПЗ), а створити живу систему, де надійне "ядро" захищене від хаотичних змін, а "оболонка" залишається максимально гнучкою.

Метою створення гібридної моделі управління IT-проектом є забезпечення оптимального балансу між надійністю базової інфраструктури (Режим 1) та гнучкістю бізнес-логіки (Режим 2). Такий підхід необхідний для досягнення низки ключових завдань.

Оптимізація "Кривої вартості змін" (Cost of Change Curve) – у класичному проектному менеджменті вартість виправлення помилки зростає експоненціально з кожною наступною фазою життєвого циклу. Гібридний підхід дозволяє "заморозити" фундаментальні вимоги до обладнання та баз даних на ранніх етапах (фіксує високу вартість змін), але залишає програмні інтерфейси та алгоритми маршрутизації гнучкими до самого кінця розробки (мінімізуючи вартість їх рефакторингу);

Зниження архітектурних та інтеграційних ризиків – використання предиктивного (Waterfall) підходу на стадії ініціації та планування дозволяє провести глибокий системний аналіз. Це включає створення детальних специфікацій для API-шлюзів та проведення аудиту топології складу. Такий

"фундамент" гарантує, що розроблені згодом гнучкі мікросервіси гарантовано зможуть обмінюватися даними з існуючими обліковими системами підприємства;

Підвищення швидкості розробки та адаптивності – реалізація серверної логіки та користувацьких мобільних додатків для терміналів збору даних здійснюється через короткі ітерації. Регулярна демонстрація (Sprint Review) проміжних результатів лінійному персоналу унеможливорює ситуацію, коли кінцевий продукт, розроблений за кілька місяців, виявляється ергономічно незручним для використання в реальних складських умовах (наприклад, кнопки інтерфейсу занадто малі для роботи в рукавичках);

Управління технологічним стеком як інтегруючим фактором – гібридний метод управління має не лише поєднувати адміністративні фреймворки, але й спиратися на відповідний технологічний менеджмент. Сполучною ланкою між жорсткою інфраструктурою та гнучкою розробкою виступають практики DevOps. Впровадження контейнеризації (Docker) та Continuous Integration / Continuous Deployment (CI/CD) дозволяє Agile-командам щодня розгортати новий код у тестових середовищах, не порушуючи стабільності базової архітектури, спроектованої за Waterfall;

Ефективне управління змінами (Organizational Change Management) – впровадження нової WMS завжди супроводжується стресом для персоналу та організаційним опором. Гібридний метод управління життєвим циклом передбачає планомірне, поетапне розгортання розробленого програмного забезпечення (Phased Rollout). Це дозволяє керувати соціально-психологічними ризиками, забезпечуючи плавне навчання співробітників без критичної зупинки логістичних бізнес-процесів компанії.

Отже, обґрунтованою метою даного дослідження є створення такої комбінованої моделі управління, яка дозволить стандартизувати процес розробки та впровадження WMS, критично знизити ризики інтеграційних конфліктів та забезпечити оперативну реакцію ІТ-команди на зміни вимог бізнесу без втрати контролю над бюджетом, якістю та строками проєкту.

1.5 Постановка задач магістерської кваліфікаційної роботи дослідження моделей і методів IT-проєкта

Враховуючи актуальність проблеми, проаналізовані недоліки існуючих підходів та обґрунтовану потребу в гібридній методології, постановка задачі даної кваліфікаційної роботи полягає у комплексному дослідженні та розробці моделей і методів управління життєвим циклом для створення IT-проєкту автоматизації складу.

Для досягнення цієї мети передбачається виконання наступних науково-практичних завдань:

- деталізація фаз життєвого циклу – розробити спеціалізовану модель життєвого циклу, яка враховує специфіку розробки WMS, включаючи фази збору топологічних вимог складу, проєктування мікросервісної архітектури, апаратного тестування та інтеграції з ERP-системами;
- структурна декомпозиція проєкту – розробити концептуальну WBS, декомпозувавши складний логістичний IT-проєкт на керовані етапи та пакети робіт;
- технологічний менеджмент – дослідити вплив обраного сучасного технологічного стека (Java, Spring Boot, React, MySQL) та інфраструктурних рішень (Docker, пайплайни CI/CD) на управління розкладом та зниження технічних ризиків життєвого циклу проєкту:
- управління проєктними ризиками – провести ідентифікацію потенційних технічних, організаційних та інтеграційних ризиків (зокрема, ризиків міграції даних та опору персоналу) та сформулювати матрицю стратегій реагування на них;
- валідація запропонованих підходів – провести аналіз очікуваних результатів від застосування розробленого гібридного методу управління та сформулювати висновки щодо ефективності та перспективності його використання для успішної реалізації IT-проєктів складської логістики.

Вирішення цих завдань дозволить сформувати цілісну методологію управління складськими ІТ-проектами, що базується на поєднанні найкращих управлінських практик та передових технологічних рішень.

2 ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ГІБРИДНОЇ МОДЕЛІ УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ СКЛАДСЬКИХ ІТ-ПРОЄКТІВ

2.1 Аналіз сучасних теоретичних моделей та методів управління життєвим циклом складських ІТ-проєктів

Теоретичне вирішення задачі управління життєвим циклом ІТ-проєкту зі створення WMS базується на комплексному застосуванні теорії управління проєктами, системного аналізу, теорії масового обслуговування та інженерії програмного забезпечення. Сучасний стан розвитку ІТ-менеджменту пропонує низку теоретичних моделей, кожна з яких має свої парадигми щодо управління часом, вартістю, обсягом робіт та якістю, що формують класичний «проєктний трикутник» (Project Iron Triangle).

Розглядаючи теоретичні основи управління ІТ-проєктами, необхідно виділити кілька фундаментальних підходів, які історично склалися в індустрії розробки програмного забезпечення. Перш за все, це предиктивні (каскадні) моделі управління. В основі каскадної моделі (Waterfall), вперше описаної Вінстоном Ройсом, лежить детермінований підхід, який передбачає сувору лінійну послідовність проходження фаз життєвого циклу: збір вимог, системне проєктування, безпосереднє кодування, модульне та інтеграційне тестування, розгортання та підтримка. З теоретичної точки зору, цей метод ідеально підходить для управління апаратною складовою WMS-проєктів (наприклад, закупівля серверних стійок, монтаж конвеєрів та точок доступу Wi-Fi на складі), оскільки ці процеси підкоряються законам класичної логістики і не можуть бути виконані ітеративно. Відповідно до стандартів International Project Management Association (IPMA) та традиційних редакцій PMBOK, предиктивна модель гарантує високий рівень контролю над бюджетом та розкладом завдяки формуванню детальної WBS на етапі планування. Проте головним теоретичним недоліком цієї моделі є надзвичайно висока вартість внесення змін на пізніх фазах життєвого циклу.

З іншого боку, еволюція теорії управління програмним забезпеченням призвела до появи емпіричних моделей, заснованих на маніфесті Agile. Теоретичною базою гнучких методів (зокрема Scrum, Kanban) є теорія складних адаптивних систем та принцип емпіричного контролю процесів. Згідно з цією теорією, розробка багаторівневого ПЗ є процесом з високим рівнем невизначеності, який неможливо на 100% спланувати наперед. Тому управління здійснюється через три стовпи емпіризму: прозорість, регулярну інспекцію та адаптацію. Життєвий цикл у таких моделях декомпонується на серію коротких ітерацій (спринтів), кожна з яких завершується постачанням робочого інкремента продукту.

Для систематизації теоретичних відомостей доцільно провести порівняльний аналіз основних моделей за ключовими управлінськими критеріями, що наведений нижче в таблиці 2.1.

Таблиця 2.1 – Теоретичне порівняння моделей управління життєвим циклом

Характеристика	Предиктивна теорія (Waterfall)	Емпірична теорія (Agile/Scrum)
Базова парадигма	Планування керує виконанням (Plan-driven)	Цінність та бізнес-пріоритети керують виконанням (Value-driven)
Управління змінами	Зміни є деструктивними, їх потрібно уникати за допомогою суворих процедур управління змінами	Зміни є природними та очікуваними; вони вітаються навіть на пізніх етапах розробки
Модель життєвого циклу	Лінійно-послідовна (один масштабний макро-цикл розробки)	Ітеративно-інкрементальна (безліч коротких мікро-циклів)

Кінець таблиці 2.1

Характеристика	Предиктивна теорія (Waterfall)	Емпірична теорія (Agile/Scrum)
Оцінка прогресу	Відсоток виконання базового плану (Baseline) та освоєння бюджету	Постачання робочого програмного забезпечення, що задовольняє критерії Definition of Done (DoD)
Ступінь ризику інтеграції	Високий (інтеграція відбувається наприкінці життєвого циклу)	Низький (безперервна інтеграція є частиною кожного спринту)

Аналіз сучасних теоретичних підходів доводить, що для розробки складних логістичних ІТ-проектів використання лише однієї парадигми створює системний конфлікт. Бізнес-вимоги потребують адаптивності та гнучкості, тоді як інфраструктурні обмеження складу та вимоги до обробки даних у реальному часі потребують жорсткого та детермінованого планування. Це зумовлює необхідність пошуку та обґрунтування новітніх комбінованих теоретичних рішень.

2.2 Дослідження новітніх технологій та інноваційних підходів до управління ІТ-проектами

2.2.1 Огляд інноваційних підходів та управлінських технологій

Огляд інноваційних підходів у сфері ІТ-менеджменту передбачає концептуальний перехід від суто адміністративного управління ресурсами до глибокого технологічного менеджменту. Сучасною інновацією в управлінні життєвим циклом є парадигма DevOps, яка стирає теоретичну та організаційну межу між фазою розробки програмного забезпечення та фазою його

промислової експлуатації і супроводу [9].

Теоретично DevOps розглядається як набір інженерних та управлінських практик, що спрямовані на автоматизацію процесів інтеграції та доставки програмного забезпечення. Для масштабних WMS-систем це означає фундаментальну зміну життєвого циклу проєкту: етап тестування та інтеграції перестає бути окремою, розтягнутою в часі фазою. Натомість він перетворюється на безперервний фоновий процес. Кожна зміна в коді, написана розробником, автоматично компілюється, тестується (Unit-тести, інтеграційні тести) і розгортається у тестовому середовищі за допомогою конвеєрів безперервної інтеграції та постачання. З точки зору управління проєктом, це радикально знижує ризики зриву дедлайнів на етапі релізу, оскільки помилки виявляються за лічені хвилини після їх створення.

Ще одним інноваційним підходом, що впливає на управління проєктом, є концепція «Інфраструктура як код» (Infrastructure as Code, IaC) та широке використання технологій контейнеризації. З управлінської точки зору, застосування контейнерів повністю ізолює програмне забезпечення від середовища, в якому воно виконується [10]. Проєктний менеджер більше не стикається з класичною проблемою інтеграційного етапу, коли модуль маршрутизації комірників ідеально працює на локальному комп'ютері розробника, але дає непередбачуваний збій на бойовому сервері складу через різні версії бібліотек або операційної системи. Контейнеризація забезпечує стовідсоткову ідентичність середовищ, що робить життєвий цикл максимально передбачуваним та керованим.

2.2.2 Детальний аналіз технологічних тенденцій та їх вплив на життєвий цикл

Аналіз сучасних технологічних тенденцій вказує на те, що архітектура

програмного забезпечення безпосередньо диктує методи управління ним (згідно із законом Конвея). Сучасний ринок корпоративного програмного забезпечення переживає масовий перехід від монолітних архітектур до розподілених систем та мікросервісів [11-12].

Мікросервісна архітектура передбачає розбиття комплексної WMS-системи на десятки незалежних модулів, що спілкуються між собою за допомогою легковагових протоколів (найчастіше HTTP/REST або шин повідомлень). Ця технологічна тенденція має потужний та багатогранний вплив на методи управління IT-проєктом.

Розпаралелювання робіт (Parallel Execution): проєктний менеджер має змогу розділити велику команду розробників на кілька автономних крос-функціональних груп (Squads). Кожна група відповідає за свій конкретний мікросервіс (наприклад, управління запасами, авторизація, інтеграція зі сканерами). Вони можуть проєктувати, розробляти, тестувати та впроваджувати свої модулі незалежно одна від одної, що суттєво скорочує загальний час реалізації проєкту (Time-to-Market).

Управління технологічним стеком: мікросервісний підхід дозволяє використовувати різні технології та мови програмування в межах одного проєкту, оптимізуючи інструменти під конкретну задачу. Наприклад, для створення високоінтерактивних клієнтських інтерфейсів (як для десктопних браузерів, так і для терміналів збору даних) доцільно використовувати JavaScript та бібліотеку React, що забезпечує швидкий рендеринг та реактивний UX. Водночас ядро бізнес-логіки та бекенд-інтеграції може розроблятися на базі надійних технологій корпоративного рівня.

Управління базами даних: однією з найкритичніших тенденцій є міграція на сучасні реляційні системи управління базами даних (СУБД). В межах даного дослідження розглядається управління інфраструктурою на базі MySQL. Використання рушія InnoDB у MySQL забезпечує необхідний рівень ACID за рахунок механізмів блокування рядків (row-level locking), що є безальтернативною вимогою для складських систем. Управління етапом

розгортання такої БД вимагає глибокого планування процедур та написання DDL-скриптів [13].

Управління ідентифікацією та консистенцією даних: у розподілених системах складської логістики критичною проблемою є генерація унікальних ідентифікаторів для мільйонів операцій (переміщень, замовлень, комірок). Використання традиційних імовірнісних методів генерації випадкових чисел створює ризик колізій, що призводить до збоїв на етапі експлуатації. В рамках управління архітектурними стандартами проєкту впроваджується практика використання детермінованих рядкових префіксів для генерації проєктних ID. Застосування чіткої системи префіксів (наприклад, `whOrder_`, `binLocation_`, `stockMove_`) замість абстрактних Universally Unique Identifier (UUID) усуває колізії, робить логи системи зрозумілими для аналізу та суттєво скорочує тривалість фази тестування та відлагодження.

Уніфікація інфраструктури розробки: Налаштування та управління середовищами розробки здійснюється за допомогою конфігурацій Docker. Це дозволяє команді тестувальників миттєво розгортати копії бази даних MySQL та мікросервісів на своїх машинах однією командою, оптимізуючи життєвий цикл гарантування якості (QA Lifecycle).

2.3 Оцінка інноваційного характеру та потенціалу технологічних рішень у сфері управління WMS

Оцінка інноваційного характеру запропонованих теоретичних та управлінських рішень базується на критеріях адаптивності системи, надійності обробки даних та швидкості постачання бізнес-цінності. Фундаментальним інноваційним характером вирішення поставленої задачі є застосування концепції Двомодальних IT (Bimodal IT), запровадженої аналітичною компанією Gartner, яка в даній роботі спеціально адаптується для управління життєвим циклом

складських інформаційних систем.

Двомодальне управління – це інноваційна організаційна практика управління двома окремими, але тісно узгодженими режимами постачання ІТ-проєкту.

Режим 1 (Орієнтація на надійність та безпеку): цей режим управляє проєктуванням базової архітектури, серверною інфраструктурою, процесами міграції баз даних на MySQL та розробкою критичних API-шлюзів для інтеграції з ERP-системами. Управління тут здійснюється за допомогою класичних предиктивних методів. Потенціал цього рішення полягає у гарантуванні абсолютної стабільності фундаментальних вузлів системи. У складській логістиці, де вартість хвилини простою вимірюється тисячами доларів, ретельне предиктивне планування ядра системи є безальтернативним базисом безпеки.

Режим 2 (Орієнтація на гнучкість та швидкість): цей режим керує розробкою та постійним вдосконаленням користувацьких інтерфейсів, клієнтських React-додатків для планшетів менеджерів та мобільних сканерів. Управління тут відбувається виключно за методологіями Agile (Scrum/Kanban). Потенціал цього підходу полягає у здатності ІТ-команди миттєво реагувати на зворотний зв'язок від кінцевих користувачів (комірників) та адаптувати ергономіку програмного забезпечення без втручання в серверну логіку.

Інноваційність запропонованого вирішення задачі полягає у синергетичному поєднанні цих двох, здавалося б, антагоністичних режимів за допомогою практик DevOps. Тобто, надійний фундамент (Режим 1), створений на основі реляційних транзакційних баз, розгортається і тестується автоматично завдяки CI/CD конвеєрам та Docker-образам. Це створює безпечний "пісочний" простір для команди Режиму 2, яка може щотижня випускати оновлення інтерфейсів і бізнес-логіки, не ризикуючи порушити цілісність даних складу.

Такий гібридний підхід є значним інноваційним кроком порівняно з традиційним управлінням. Він ефективно усуває історичний компроміс між швидкістю розробки ПЗ та надійністю критичної логістичної інфраструктури.

2.4 Вибір оптимальних методів та моделей для подальшого дослідження

На основі проведеного глибокого теоретичного та технологічного аналізу, для подальшого практичного моделювання у цій кваліфікаційній роботі обирається гібридна модель управління життєвим циклом (Water-Scrum-Fall) у тісній комбінації з мікросервісним технологічним стеком, детермінованою ідентифікацією даних та інфраструктурними практиками DevOps.

Для візуалізації обраного управлінського підходу розроблено концептуальну схему життєвого циклу (рисунок 2.1).

Ця модель виступає не просто як набір адміністративних регламентів, а як комплексне управлінсько-технологічне рішення, що дозволяє раціонально декомпонувати складний проєкт створення WMS на керовані етапи та звести до мінімуму операційні ризики підприємства-замовника.

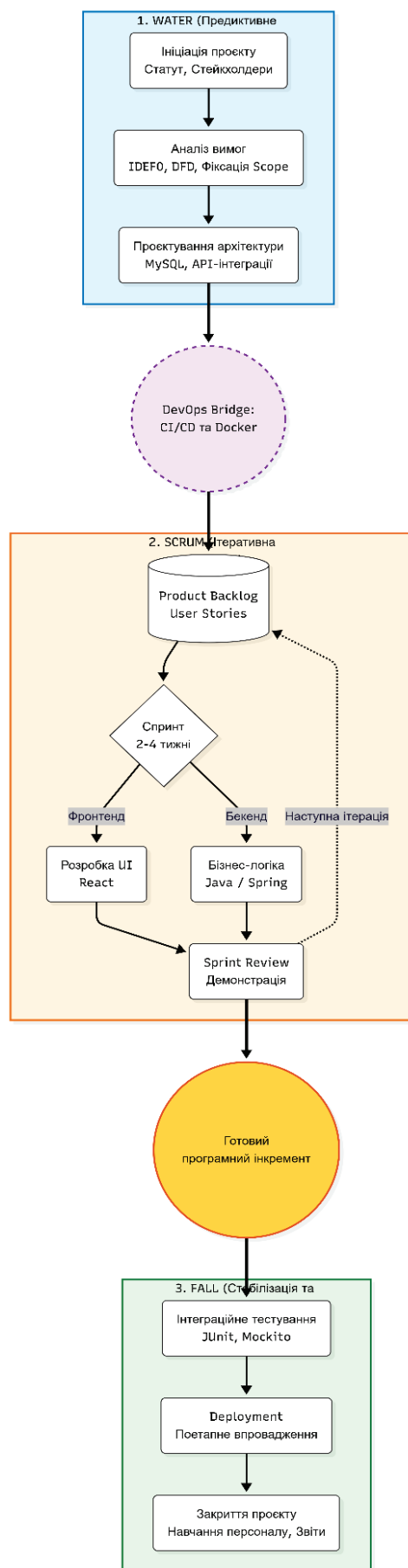


Рисунок 2.1 – Концептуальна схема гібридної моделі життєвого циклу Water-Scrum-Fall

2.4.1 Переваги обраної гібридної моделі управління

Першою і стратегічно найважливішою перевагою обраного методу є оптимізація кривої вартості змін (Boehm's Curve of Cost of Change). У класичному проєктному менеджменті вартість виправлення помилки експоненціально зростає на кожній наступній фазі життєвого циклу. Завдяки предиктивному плануванню на стадії ініціації, фундаментальні архітектурні рішення (такі як логічна структура бази даних, застосування детермінованих префіксів для генерації ключів та протоколи апаратної інтеграції) фіксуються заздалегідь. Це запобігає катастрофічно дорогим переробкам ядра системи. Водночас ітеративний підхід до клієнтської розробки дозволяє змінювати вимоги до інтерфейсів на пізніх етапах майже безкоштовно.

Другою суттєвою перевагою є максимальна прозорість та керованість очікувань стейкхолдерів. Замовник (наприклад, топ-менеджмент логістичної компанії) отримує чіткий календарний графік і бюджетну оцінку на ранній стадії проєкту, що необхідно для фінансового планування. Водночас, під час фази розробки, замовник регулярно бере участь у демонстраціях функціоналу (Sprint Review), постійно бачачи реальний прогрес і контролюючи напрямок розвитку продукту.

Третьою перевагою є високоефективне управління ризиками впровадження. Запропонований гібридний метод органічно підтримує стратегію поетапного розгортання (Phased Rollout). Завдяки контейнеризації мікросервісів, нова ІТ-система може впроваджуватися на діючому складі не одномоментно (що часто призводить до паралічу роботи), а локалізовано по зонах. Спочатку автоматизується зона приймання товару, після її стабілізації — зона зберігання, і так далі. Це знижує ризик операційної зупинки бізнесу до прийнятного мінімуму.

2.4.2 Обґрунтування вибору інтегрованого підходу

Детальне обґрунтування вибору інтегрованого гібридного методу управління спирається на його доведену ефективність у складних корпоративних проєктах. Сфера автоматизації складських операцій є безкомпромісною: ціна програмної помилки або системного збою тут вимірюється не просто незадоволеністю користувача, а зупинкою ланцюгів постачання та значними фінансовими збитками. У таких умовах використання суто емпіричних гнучких методів (Agile) без твердого архітектурного фундаменту є неприпустимо ризикованим кроком. З іншого боку, застосування суто каскадних методів (Waterfall) є економічно невиправданим через довготривалий цикл розробки та ризик створення морально застарілого продукту, інтерфейс якого не відповідає реальним потребам комерсійців.

Інтегрований вибір гібридного методу управління (Bimodal IT) у поєднанні із застосуванням надійних технологій для серверної логіки, бази даних MySQL для абсолютної транзакційної цілісності, React для динамічного та ергономічного фронтенду, а також Docker-конфігурацій для стабільності середовищ розробки, створює потужний синергетичний ефект. Це управлінське рішення надає керівнику проєкту вичерпний інструментарій для забезпечення високого рівня контролю над технічними ресурсами та розкладом.

Водночас такий підхід гарантує постачання інноваційного, високоякісного програмного продукту, який максимально швидко адаптується до мінливих і динамічних потреб сучасної складської логістики. Саме тому ця теоретична та технологічна модель визнається оптимальною, науково обґрунтованою і буде використана як надійний базис для розробки структурних компонентів системи та практичного вирішення проєктної задачі у наступних розділах кваліфікаційної роботи.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ ІТ-ПРОЄКТУ СТВОРЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДУ

3.1 Практичне моделювання бізнес-процесів як інструмент фіксації обсягу проєкту

У межах реалізації практичної фази проєкту зі створення WMS, згідно з гібридною моделлю управління, першочерговим етапом виступив детальний аналіз та формалізація бізнес-процесів підприємства. На відміну від хаотичного збору вимог, було застосовано методологію функціонального моделювання IDEF0, що дозволило декомпонувати складну складську діяльність на керовані підзадачі та зафіксувати обсяг проєкту на предиктивній фазі життєвого циклу [14].

Головною функцією системи визначено «Облік складської діяльності».

На рисунку 3.1 представлена контекстна діаграма, на якій зображена основна функція системи. Вхідними даними для даної функції виступають:

- логін та пароль (для ідентифікації);
- дані про товари, замовлення, склади та сектори складу;
- дані користувача.

Вихідними даними для даної функції виступають: дані авторизації, інформація про виконану складську операцію та замовлення, статус замовлення, інформація про склад та сектор, а також зареєстровані користувачі та звіти про їхню роботу. В якості управління виступають керівництво користувача та нормативна документація, а механізмами — безпосередньо ІС, адміністратори та менеджери складу.

Для розподілу зон відповідальності та виявлення інтеграційних ризиків була проведена декомпозиція процесів, що зображена на рис 3.2.

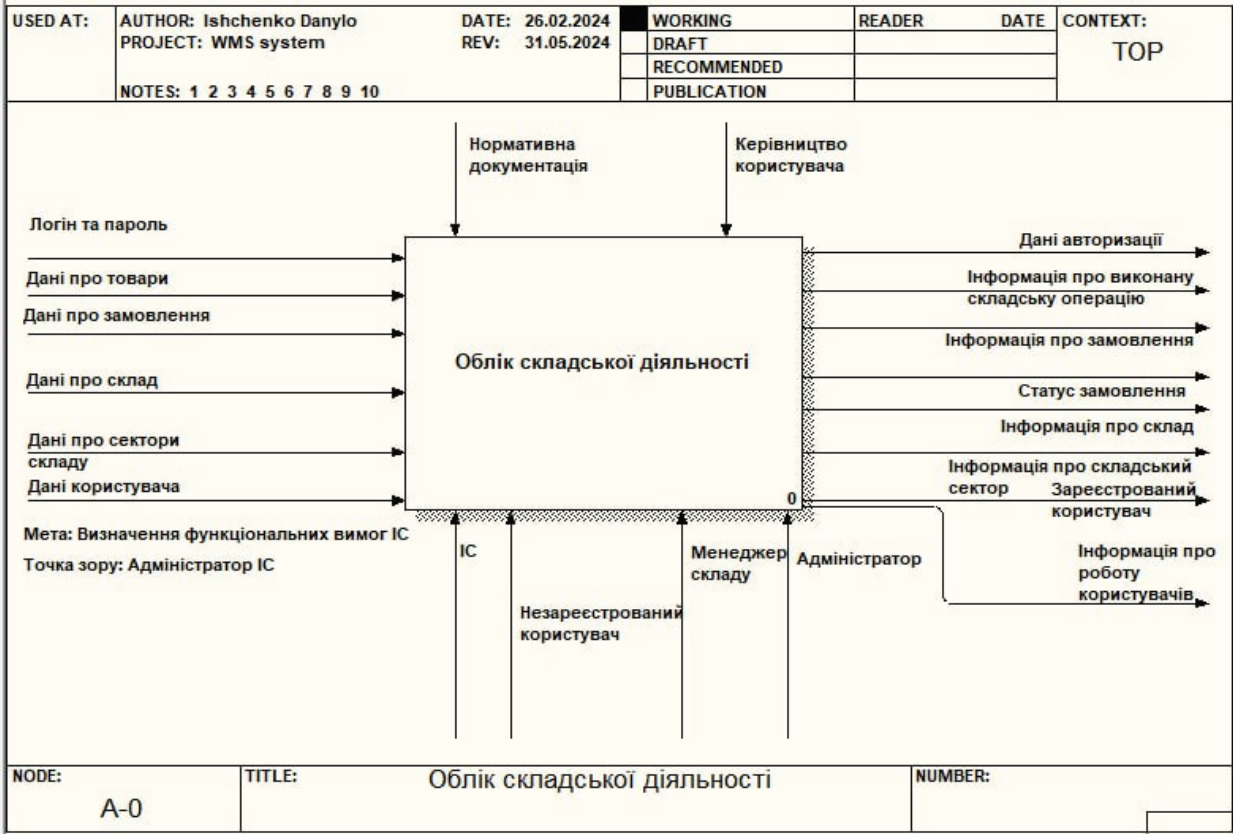


Рисунок 3.1 – Схема бізнес процесу в нотації IDEF0

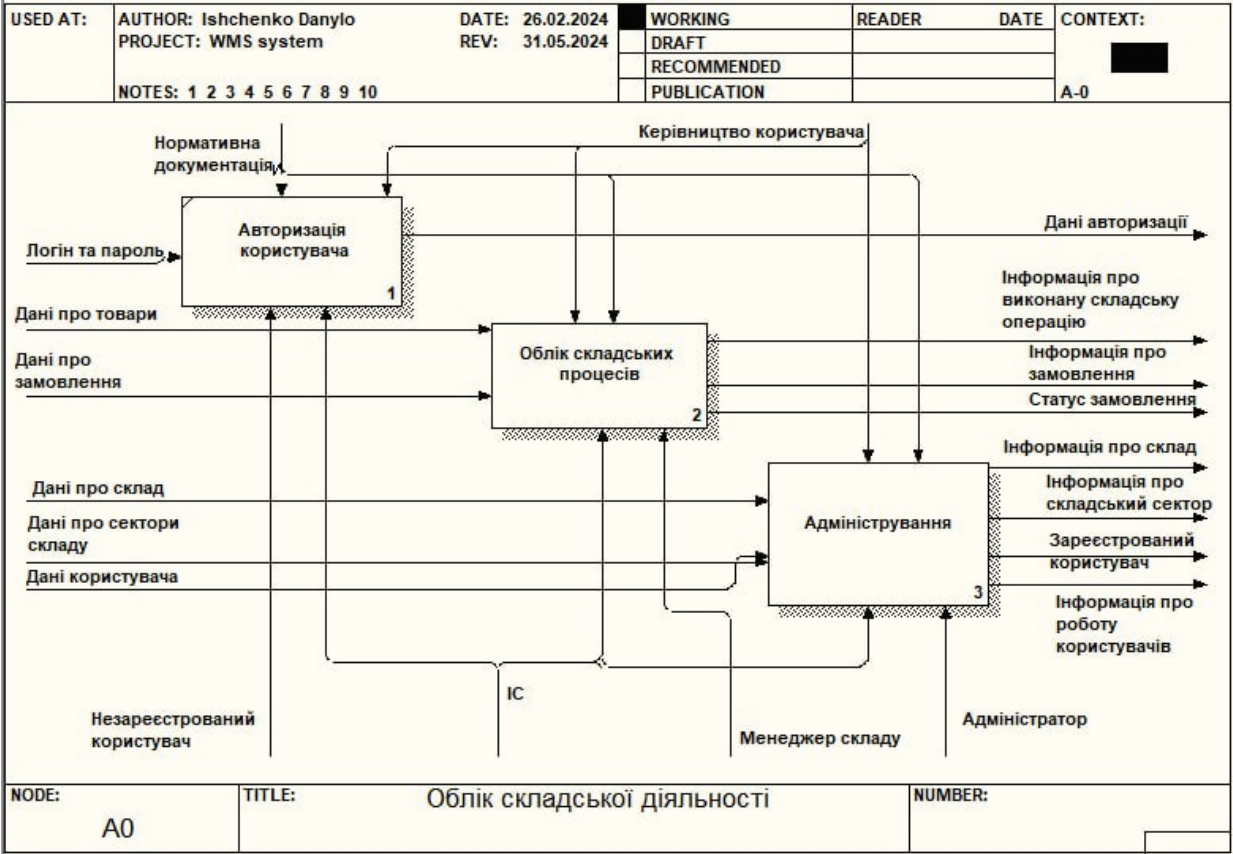


Рисунок 3.2 – Діаграма декомпозиції функції «Облік складської діяльності»

Процес управління вимогами на практиці включав розділення функціоналу за ролями стейкхолдерів. Це дозволило мінімізувати конфлікти інтересів між адміністративним та лінійним персоналом.

Було визначено, що менеджер складу фокусується на оперативній діяльності (прийом, відвантаження), тоді як адміністратор керує системними налаштуваннями. Для глибшого розуміння операційних ризиків була проведена подальша декомпозиція найбільш динамічного процесу — виконання замовлень.

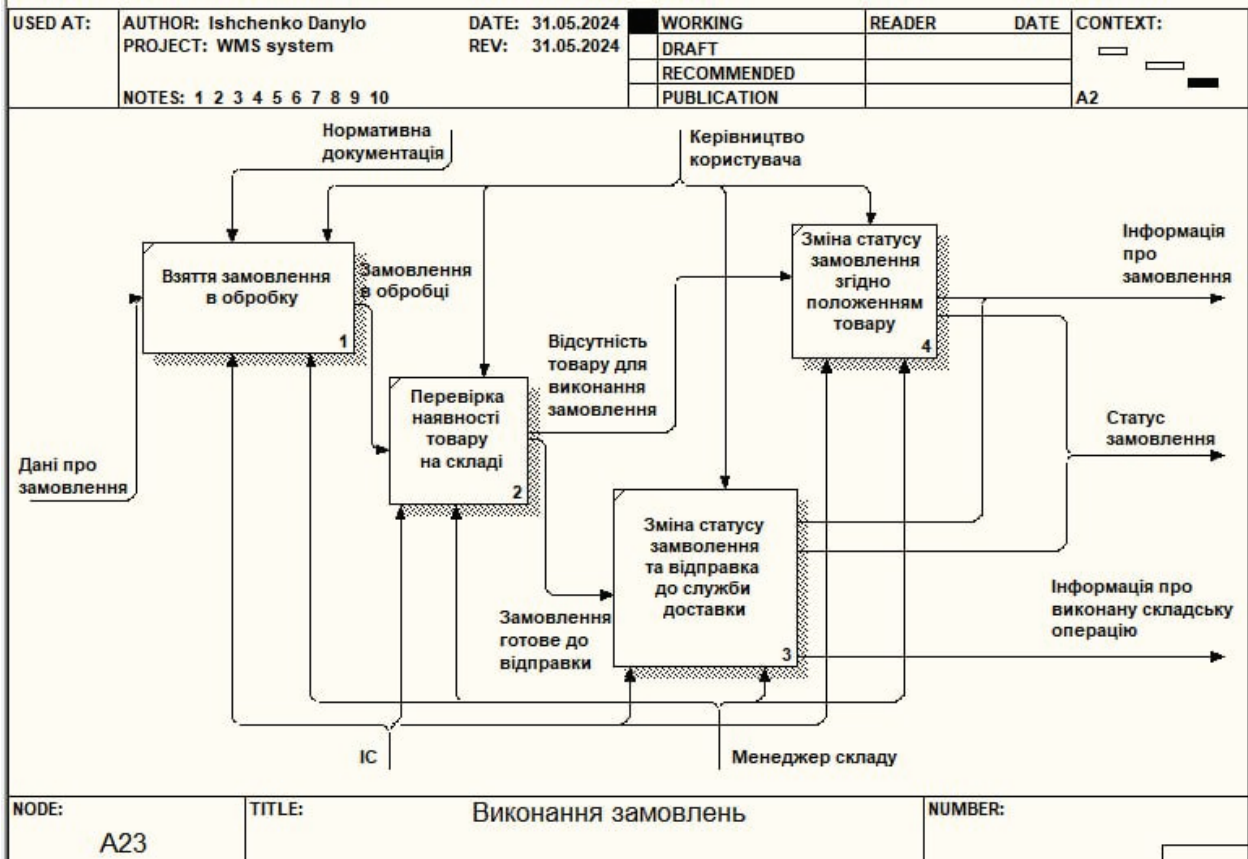


Рисунок 3.3 – Діаграма декомпозиції функції «Виконання замовлень»

Як видно з рис. 3.3, деталізація цього блоку дозволила ідентифікувати критичний етап – перевірку наявності товару на складі. Виявлення цього кроку на стадії предиктивного моделювання дозволило заздалегідь спроектувати механізми обробки нестачі товарів, уникнувши простоїв на етапі експлуатації.

3.2 Моделювання потоків даних та формування вимог до системи

Для забезпечення прозорості руху інформації на всіх етапах життєвого циклу розроблено модель потоків даних (Data Flow Diagram, DFD). З управлінської точки зору, DFD описує процеси передачі даних із вхідних форм у сховище файлів (базу даних) та формування вихідних звітів, що є критично важливим для проєктування інформаційної безпеки проєкту.

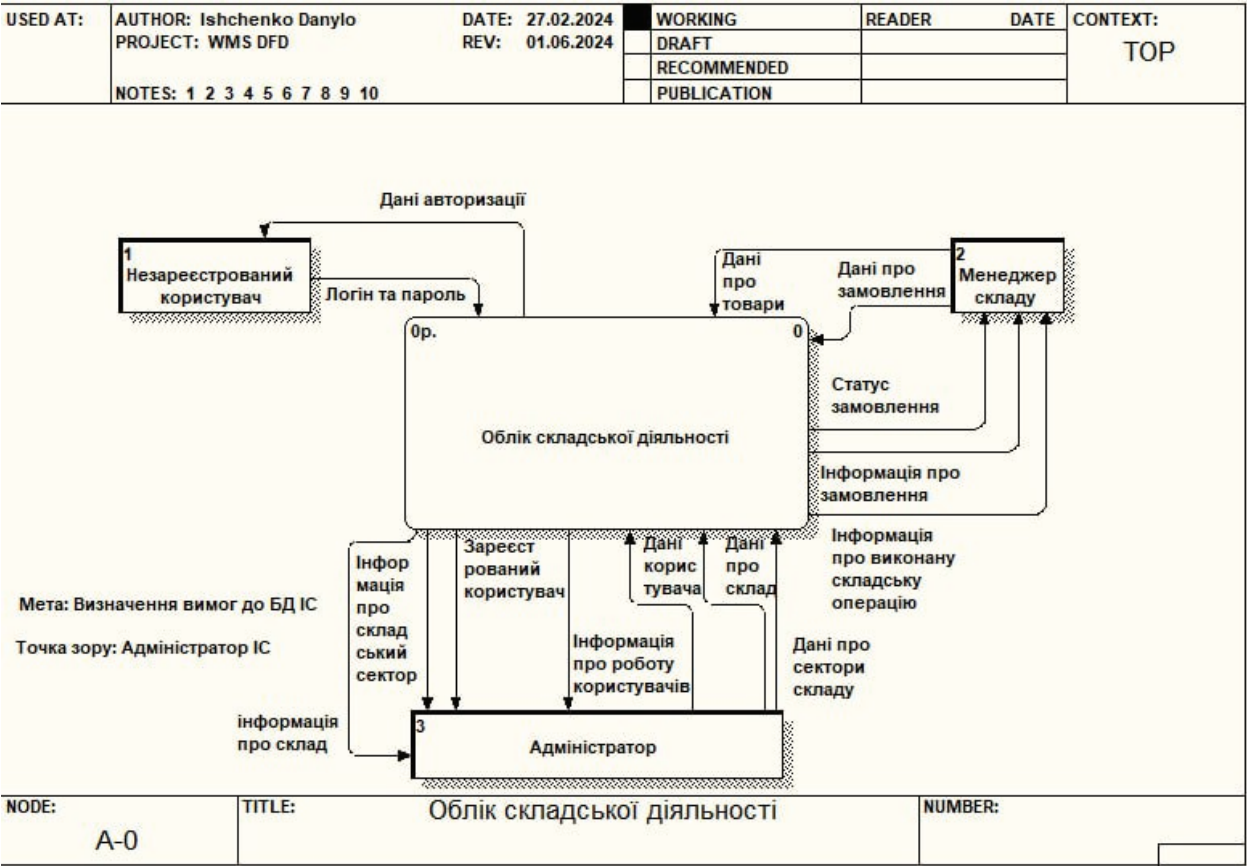


Рисунок 3.4 – Контекстна діаграма потоків даних DFD

Зовнішніми сутностями для головної функції на цій діаграмі виступають: незареєстрований користувач, менеджер складу та адміністратор. Для переходу до гнучкої фази розробки (Scrum) і формування беклогу продукту, статичні вимоги DFD були трансформовані у діаграму прецедентів (Use Case).

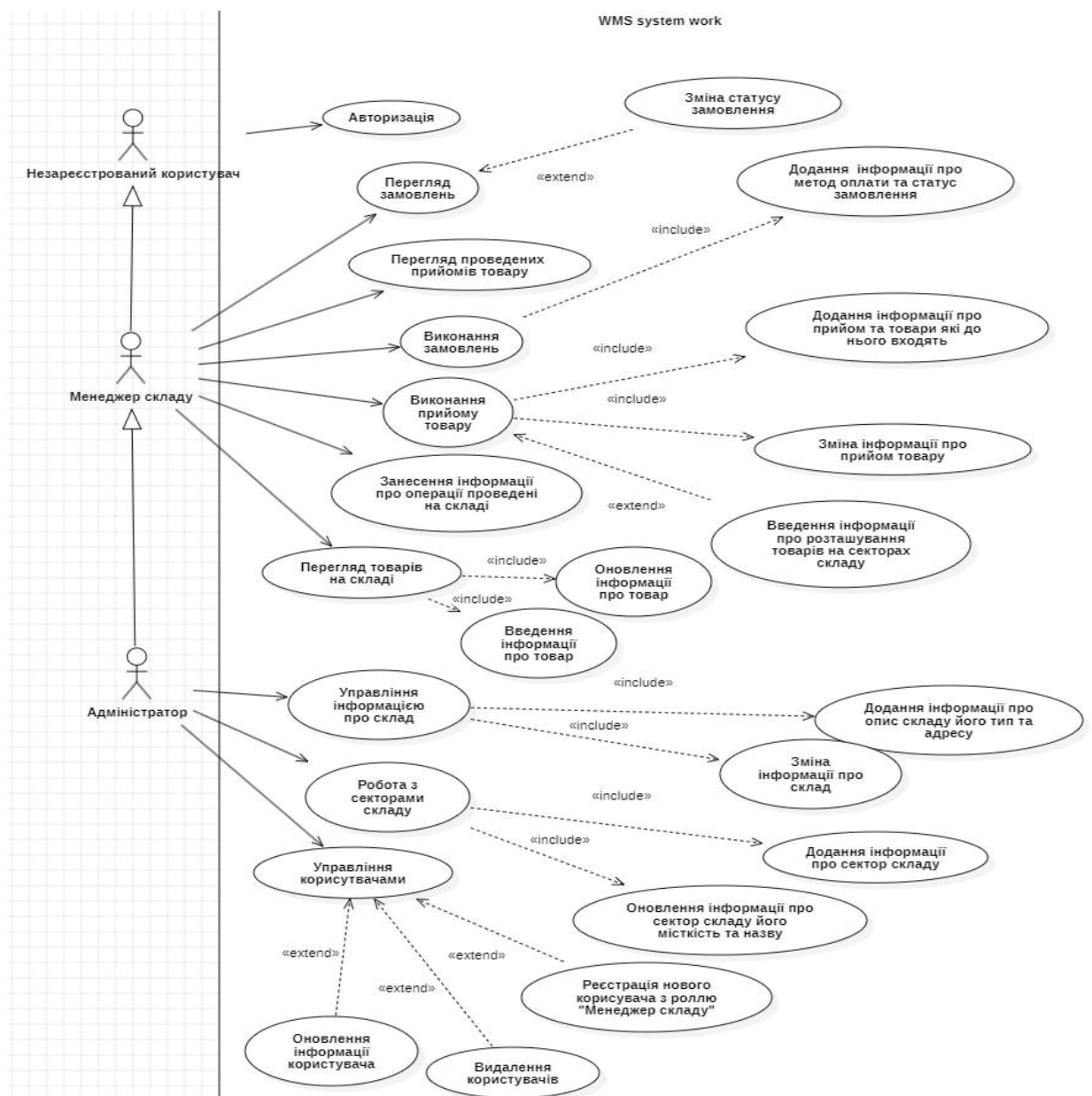


Рисунок 3.5 – Діаграма прецедентів (Use Case Diagram)

На рис. 3.5 відображені всі основні сценарії взаємодії користувачів із системою. Кожен прецедент (наприклад, "Перегляд товарів на складі" або "Управління користувачами") фактично виступає окремим Епіком або User Story, які розподілялися між спринтами розробників.

3.3 Проектування інформаційної інфраструктури та бази даних

Наступним етапом практичного вирішення стало проектування фундаментального ядра системи — її архітектури та бази даних. З управлінської точки зору було прийнято рішення про використання триланкової клієнт-серверної архітектури. Такий підхід забезпечує високу масштабованість: за необхідності підприємство може збільшити кількість складських терміналів без зміни логіки сервера.

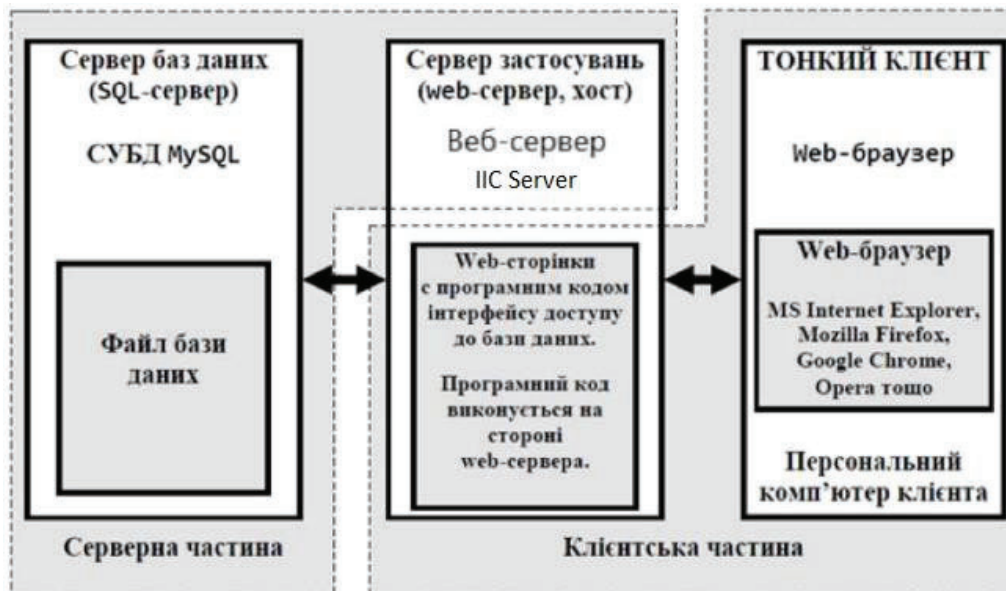


Рисунок 3.6 – Схема архітектури web-застосунку

Особлива увага приділена проектуванню реляційної бази даних на платформі MySQL. Для забезпечення цілісності даних розроблено логічну та фізичну моделі даних, що зображена на рис 3.7[13-15].

Рисунок 3.7 – ER-діаграма фізичної моделі даних

Розглянемо детальніше фізичну структуру розробленої БД (на основі якої пізніше будувалася ER-діаграма), оскільки вона визначає правила зберігання всієї логістичної інформації:

- таблиця user – зберігає ідентифікаційні дані (id), ім'я користувача (username), зашифрований пароль (password) та електронну пошту (email);
- таблиця warehouse – містить дані про складські приміщення, включаючи унікальний ідентифікатор (id), адресу (address), назву (name), тип приміщення (type) та його місткість у кубічних метрах (volume);

- таблиця `good` – фундаментальна сутність для зберігання інформації про товари. Включає зовнішній ключ складу (`warehouse_id`), код товару (`code`), очікувану вагу (`weight`), детальний опис (`description`), статус наявності (`status`), назву (`name`), ціну (`price`), дату закінчення терміну придатності (`expiration_term`), одиниці вимірювання (`unit_of_measurement`) та категорію (`category`);
- таблиця `orders` – акумулює дані про замовлення: дата створення (`order_date`), метод оплати (`payment_method`), поточний статус (`status`) та зовнішній ключ клієнта (`customer_id`);
- таблиця `good_acceptance` – фіксує прийом товару: дата прийому (`acceptance_date`), об'єм (`accepts_volume`), ціна (`price`), кількість (`quantity`), а також ідентифікатори замовлення та постачальника;
- таблиця `provider` – містить контактні дані постачальника (`contact_data`) та його веб-сайт (`web_site`);
- таблиця `warehouse_sector` – деталізує топологію складу: доступний об'єм (`available_volume`), загальний об'єм (`total_volume`), ідентифікатор складу та назва сектору (`name`);
- проміжні таблиці (`good_in_warehouse_sector`, `acceptance_item`, `operations`, `good_operation`, `good_order`) – забезпечують нормалізацію бази даних та зберігають історію аудиту (хто, коли і яку операцію здійснив).

Затвердження цієї схеми означало успішне завершення предиктивної фази планування (Waterfall) та перехід до ітеративного створення продукту.

3.4 Розробка ієрархічної структури робіт (WBS) та планування ітерацій

Основою ефективного управління проєктом є структурна декомпозиція робіт WBS, яка перетворює абстрактні вимоги на керовані пакети завдань. На основі проведеного раніше моделювання (IDEF0 та Use Case) було розроблено WBS для проєкту створення WMS-системи.[5,6]

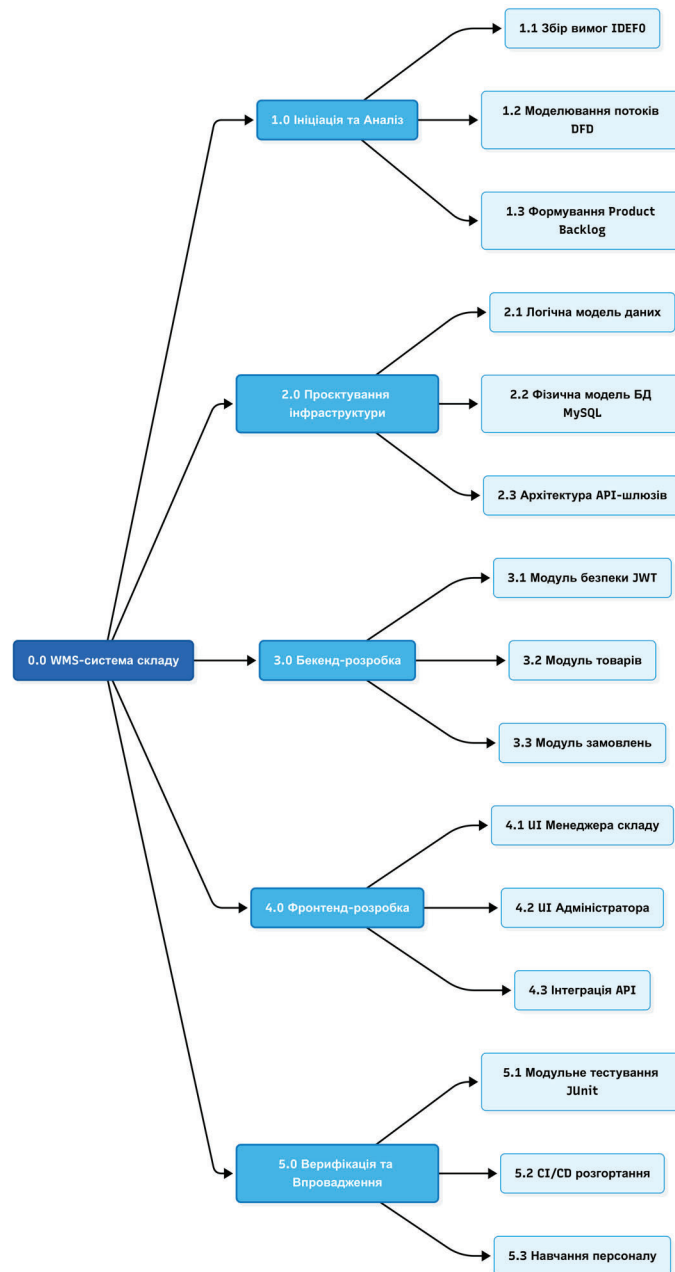


Рисунок 3.8 – Ієрархічна структура робіт (WBS) проєкту

Як показано на рис. 3.8, проєкт декомпозовано на п'ять основних макро-етапів: Ініціація та Аналіз (1.0), Проєктування інфраструктури (2.0), Бекенд-розробка (3.0), Фронтенд-розробка (4.0) та Інтеграція і Тестування (5.0).

Пакети робіт із гілок 3.0 та 4.0 були перенесені до Беклогу продукту та реалізовувалися за допомогою гнучких ітерацій. Кожен спринт тривав 2 тижні. Наприклад, у межах перших спринтів розроблявся модуль "Безпека та Авторизація" (налаштування JSON Web Token, JWT), а в наступних – модулі управління запасами та обробки замовлень.

У межах спринтів було реалізовано основні програмні компоненти:

- модуль управління товарами та запасами – забезпечує CRUD-операції над номенклатурою та відстеження залишків у розрізі секторів складу;
- модуль обробки замовлень – реалізує логіку зміни статусів (від «нове» до «відвантажене») та інтеграцію з даними клієнтів;
- модуль безпеки та авторизації – використовує JWT-токени для сесійного управління та шифрування паролів BCrypt, що гарантує захист конфіденційної комерційної інформації;

Для наочності структури розробленого клієнтського рішення було сформовано карту сайту, зображену на рис 3.9, яка відображає ієрархію сторінок та суворе розмежування прав доступу між користувачами.

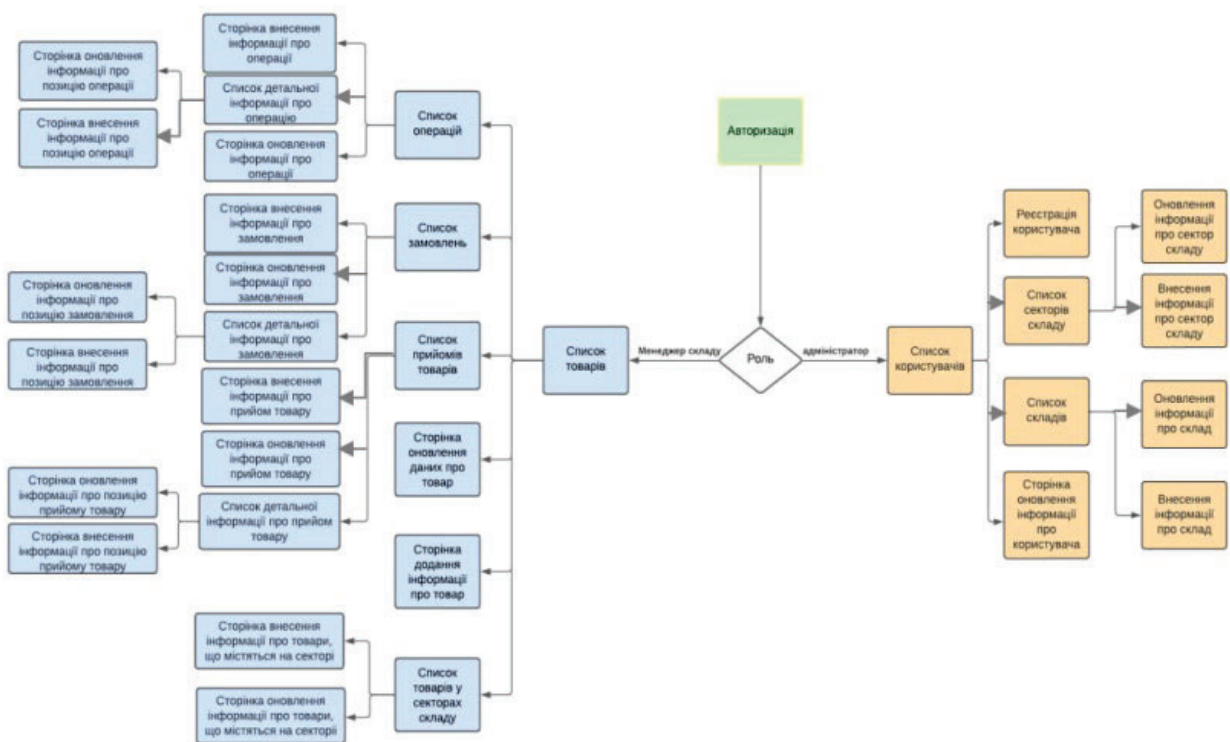


Рисунок 3.9 – Карта сайту

3.5 Управління якістю та методологія верифікації проєкту

Кінцевою фазою практичного вирішення задачі стало проведення тестування (верифікації) розробленого програмного забезпечення. Згідно з вимогами до надійності корпоративних систем, було впроваджено модульне тестування (Unit Testing) за допомогою фреймворків JUnit та Mockito.

Процес тестування, як невід'ємна частина управління якістю (QA Lifecycle), включав:

- підготовку мок-об'єктів для імітації поведінки бази даних у тестовому середовищі;
- перевірку бізнес-логіки обробки замовлень та строгу валідацію вхідних даних;
- системний контроль рівня покриття коду тестами для мінімізації накопичення технічного боргу.

Процес верифікації включав підготовку мок-об'єктів (імітацію БД) та перевірку бізнес-логіки. Приклад реалізації автоматизованого тесту для сервісу обробки замовлень наведено у Лістингу 3.1.

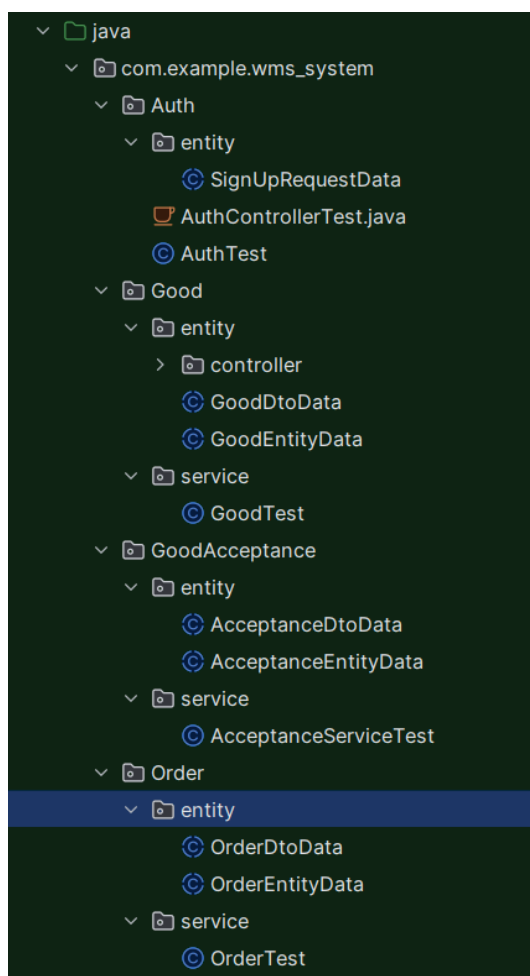


Рисунок 3.10 – Структура проекту тестів

Лістинг 3.1 – Код тестування класу OrderService за допомогою Mockito

```

@ExtendWith(MockitoExtension.class)
public class OrderTest {
    @Mock
    OrderRepository orderRepository;

    @InjectMocks
    OrderService orderService;

    @Test
    void should_return_all_orders() {
        List<Order> order =
        OrderEntityData.getListOfOrder();
    }
}

```

```
when(orderRepository.findAll()).thenReturn(order);  
    List<Order> expected = orderService.getAll();  
    assertEquals(expected, order);  
    verify(orderRepository).findAll();  
}  
}
```

Такий управлінський підхід до верифікації дозволив ще до етапу розгортання підтвердити відповідність реалізованого продукту початковим бізнес-вимогам та гарантувати стабільність життєвого циклу WMS-системи при подальшому промисловому впровадженні на складі замовника.

4 ОПИС ПРАКТИЧНОГО ЗАСТОСУВАННЯ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ ІТ-ПРОЄКТУ СТВОРЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДУ

4.1 Аналіз предметної галузі та організаційної структури об'єкта автоматизації

У межах практичної апробації результатів магістерської роботи було реалізовано процес впровадження розробленої WMS. Першим етапом практичного застосування гібридної моделі управління життєвим циклом (Water-Scrum-Fall) став детальний аналіз організаційної структури клієнта (замовника), для якого створювався ІТ-продукт, та формування реєстру зацікавлених сторін (Stakeholder Register).

Складське підприємство клієнта має чітку ієрархічну структуру управління. На найвищому рівні знаходиться Директор (Власник бізнесу), який виступає головним Спонсором проєкту та приймає стратегічні рішення щодо розвитку логістичних потужностей. Його головна вимога до системи — прозорість фінансових та товарних потоків. Організаційну структуру об'єкта автоматизації та підпорядкованість відділів наведено на рисунку 4.1.

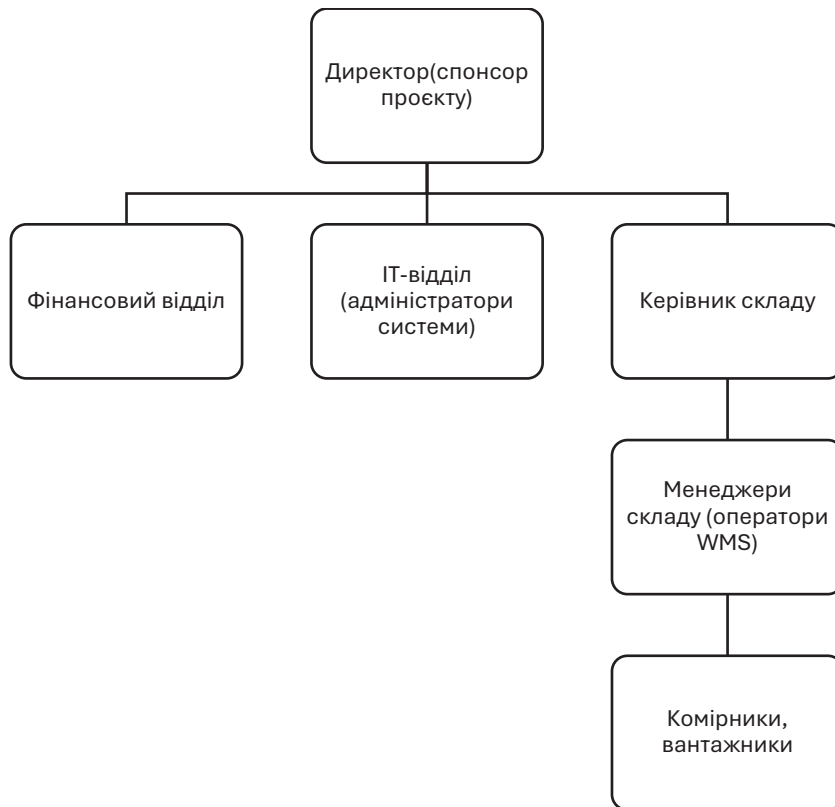


Рисунок 4.1 – Організаційна структура підприємства

Як видно з рисунка 4.1, Директору безпосередньо підпорядковуються фінансовий відділ, ІТ-відділ та керівник складу. У процесі збору вимог та аналізу цієї структури було визначено ключових стейкхолдерів ІТ-проєкту. Головними кінцевими користувачами розроблюваної WMS-системи виступатимуть менеджери складу, які знаходяться на нижньому рівні управління оперативною діяльністю. Їхніми ключовими обов'язками є прийом товарів, розподіл їх по секторах, збирання замовлень та відправка їх до служби доставки.

До впровадження автоматизованої системи комунікація між цими рівнями ієрархії (від комірників до директора) супроводжувалася високим відсотком помилок через людський фактор, що призводило до пересортиці та прямих фінансових втрат для клієнта. Фахівці ІТ-відділу (адміністратори системи), у свою чергу, не беруть участі в логістиці, а відповідають виключно за інфраструктурні зміни (додавання нових складських приміщень у систему) та управління обліковими записами працівників.

Управлінське рішення полягало у тому, щоб на етапі ініціації та планування (Waterfall-фаза) чітко розмежувати ролі користувачів та їхні права доступу до функцій системи. Це дозволило уникнути конфлікту інтересів, сформувавши базовий план змісту проєкту та перейти до вибору технологічного стека, який би задовольняв вимоги Спонсора щодо надійності та вимоги менеджерів щодо швидкодії.

4.2 Практичне застосування методів управління інфраструктурними рішеннями та технологічним стеком

На етапі практичної реалізації перед керівником ІТ-проєкту постало завдання вибору технологій, які б гарантували стабільність ядра системи управління складом (Режим 1 Двомодальних ІТ). Для розробки бекенд-частини було прийнято рішення використовувати мову програмування Java та екосистему Spring Framework.

Вибір Java обґрунтовується її високою продуктивністю та здатністю ефективно працювати з багатопоточністю, що є критично важливим для одночасної обробки сотень паралельних запитів від складських операторів. Фреймворк Spring (зокрема, Spring Boot) забезпечив швидке розгортання мікросервісної архітектури, що дозволило оптимізувати процес управління релізами та зменшити загальний час розробки [16].

Фундаментом надійності системи виступає рівень збереження даних. В якості СУБД було обрано реляційну СУБД MySQL. Її використання гарантує виконання транзакцій за жорсткими вимогами ACID, що виключає можливість втрати даних під час одночасного списування товару різними працівниками.

Окремим критичним управлінським та архітектурним рішенням під час проєктування БД стало управління генерацією ідентифікаторів. Для усунення ризиків колізій та оптимізації процесів ідентифікації сутностей було повністю

переглянуто логіку створення первинних ключів. Замість використання імовірнісних методів генерації випадкових чисел, було прийнято рішення використовувати детерміновані рядкові префікси (наприклад, `order_XXXX`, `good_XXXX`, `acceptance_XXXX`). Таке рішення значно спростило процес відлагодження системи на етапі тестування, оскільки розробники та тестувальники могли за префіксом миттєво визначати тип об'єкта, що бере участь у транзакції.

4.3 Управління якістю та ризиками цілісності даних на рівні бази даних

Одним із головних операційних ризиків, ідентифікованих на етапі планування ІТ-проєкту, був ризик розсинхронізації даних між фізичним станом складу та обліковою системою (наприклад, коли товар фактично прийнято, але в системі він ще не відображається на залишках, або навпаки).

Для мінімізації цього ризику було прийнято рішення перенести частину критичної бізнес-логіки з рівня програмного коду безпосередньо на низький рівень СУБД за допомогою механізму SQL-тригерів.[13] З управлінської точки зору це рішення дозволило розвантажити Agile-команду розробників бекенду, зменшити складність Java-сервісів та гарантувати, що цілісність складських запасів не буде порушена незалежно від програмних збоїв чи мережевих помилок на рівні клієнтського додатка.

На практиці було розроблено та імплементовано низку автономних тригерів. Наприклад, для автоматизації оновлення кількості товарів після їх фізичного прийому на склад було створено тригер `update_good_quantity_after_acceptance` (Лістинг 4.1).

Лістинг 4.1 – SQL-код тригера для автоматичного оновлення кількості товарів

```
CREATE TRIGGER update_good_quantity_after_acceptance
AFTER INSERT ON acceptance_item
FOR EACH ROW
BEGIN
    UPDATE good
    SET quantity = quantity + NEW.received_quantity
    WHERE id = NEW.good_id;
END;
```

Цей тригер усуває необхідність ручного перерахунку товарів менеджером складу. Як тільки в базу вноситься запис про отримання нової партії товару (acceptance_item), система автоматично і миттєво збільшує загальну кількість відповідного товару на балансі. Це мінімізує вплив людського фактора.

Ще одним практичним втіленням автоматизації бізнес-процесів став тригер update_order_status_after_acceptance_update (Лістинг 4.2), який автоматично змінює статус замовлення на «Completed» у разі повного надходження товару.

Лістинг 4.2 – SQL-код тригера для управління статусом замовлення

```
CREATE TRIGGER update_order_status_after_acceptance_update
AFTER UPDATE ON acceptance_item
FOR EACH ROW
BEGIN
    IF NEW.received_quantity = NEW.expected_quantity THEN
        UPDATE orders
        SET status = 'Completed'
        WHERE id = NEW.acceptance_id;
    ELSE
        UPDATE orders
        SET status = 'Pending'
        WHERE id = NEW.acceptance_id;
```

```
END IF;  
END;
```

З позиції управління IT-проєктом, впровадження таких інфраструктурних тригерів критично знижує кількість потенційних дефектів на стадії системного тестування та експлуатації. Система стає саморегульованою: статуси замовлень змінюються автоматично на основі жорстких математичних умов перевірки очікуваної та фактичної кількості.

4.4 Управління ітеративною розробкою користувацьких інтерфейсів

Якщо серверна частина та база даних розроблялися за суворими предиктивними правилами (Waterfall), то клієнтська частина (інтерфейси для менеджерів та адміністраторів) створювалася ітеративно, за гнучкою методологією Scrum (Режим 2 дво модальних IT)[17]. Головним управлінським завданням тут було забезпечення високої ергономіки інтерфейсу (User Experience/User Interface, UX/UI), адже від швидкості роботи оператора в програмі безпосередньо залежить пропускна здатність усього логістичного вузла.

На перших спринтах команда здійснила візуальне прототипування екранів системи за допомогою інструменту Figma. Це дозволило продемонструвати замовнику майбутній вигляд системи та зібрати зворотний зв'язок ще до початку написання коду фронтенду.

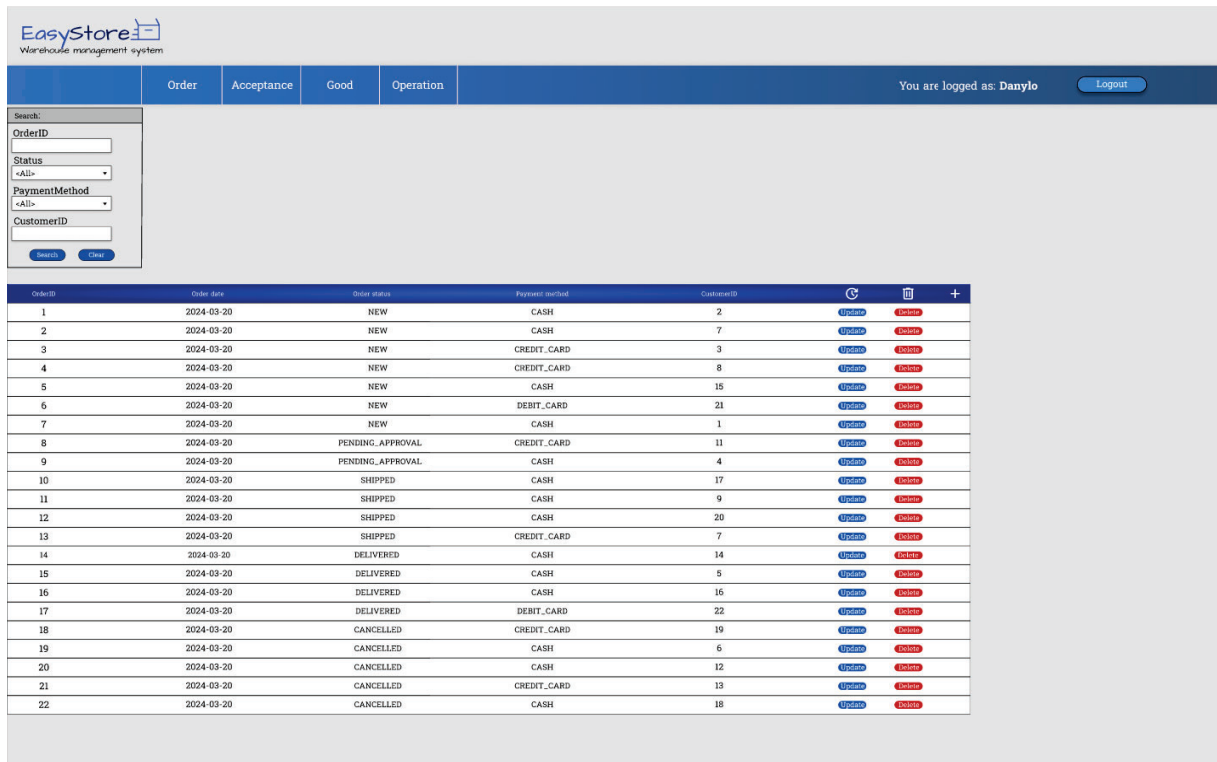


Рисунок 4.2 – Прототип сторінки з замовленнями у Figma.

На рисунку 4.2 зображено результат перших ітерацій розробки – UX-прототип сторінки управління замовленнями. Інтерфейс був спроектований так, щоб менеджер складу міг миттєво зчитувати критичну інформацію: статуси замовлень, методи оплати та ідентифікатори клієнтів.

Після затвердження прототипів Спонсором проєкту, команда розробників перетворила їх на працюючий інкремент програмного продукту. Управління розробкою велося за допомогою користувацьких історій (User Stories)[8].

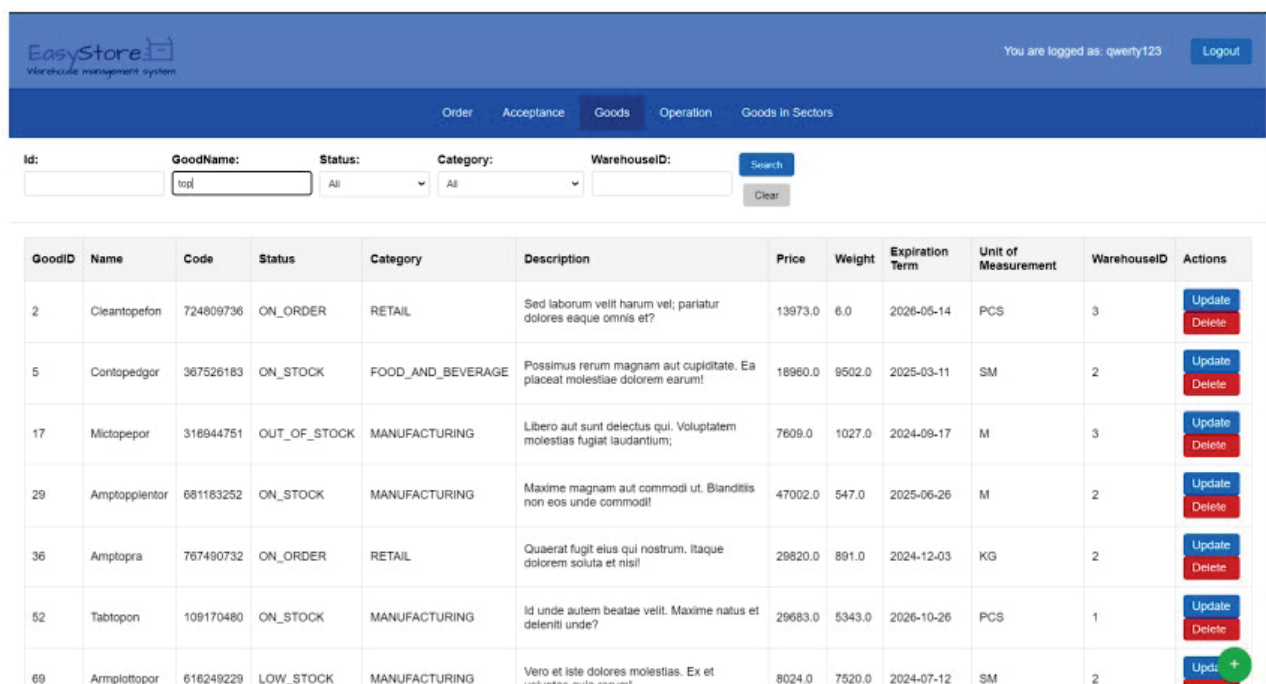


Рисунок 4.3 – Робочий інтерфейс сторінки списку товарів

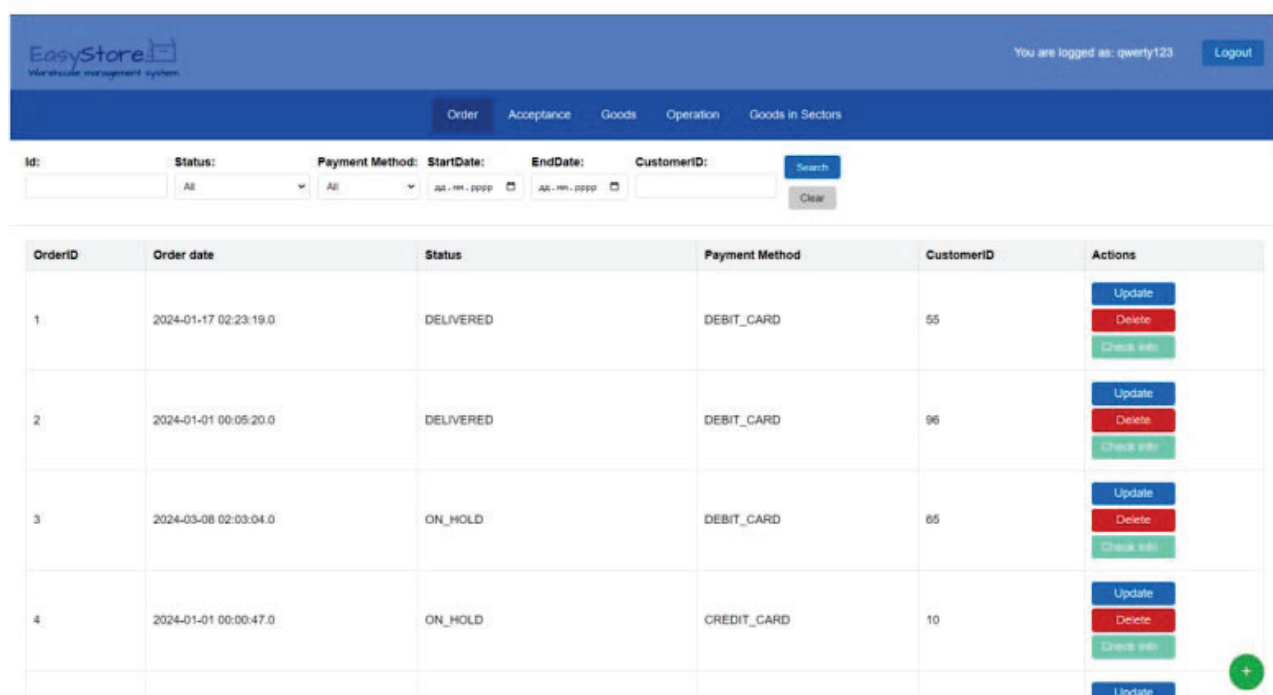


Рисунок 4.4 – Робочий інтерфейс сторінки списку замовлень

Як видно з рис. 4.2 та 4.3, готовий додаток повністю відтворює затверджену ергономіку. У наступних спринтах розроблялися модулі швидкого

редагування складських операцій. Зокрема, було спроектовано модальне вікно для зміни статусів замовлень.

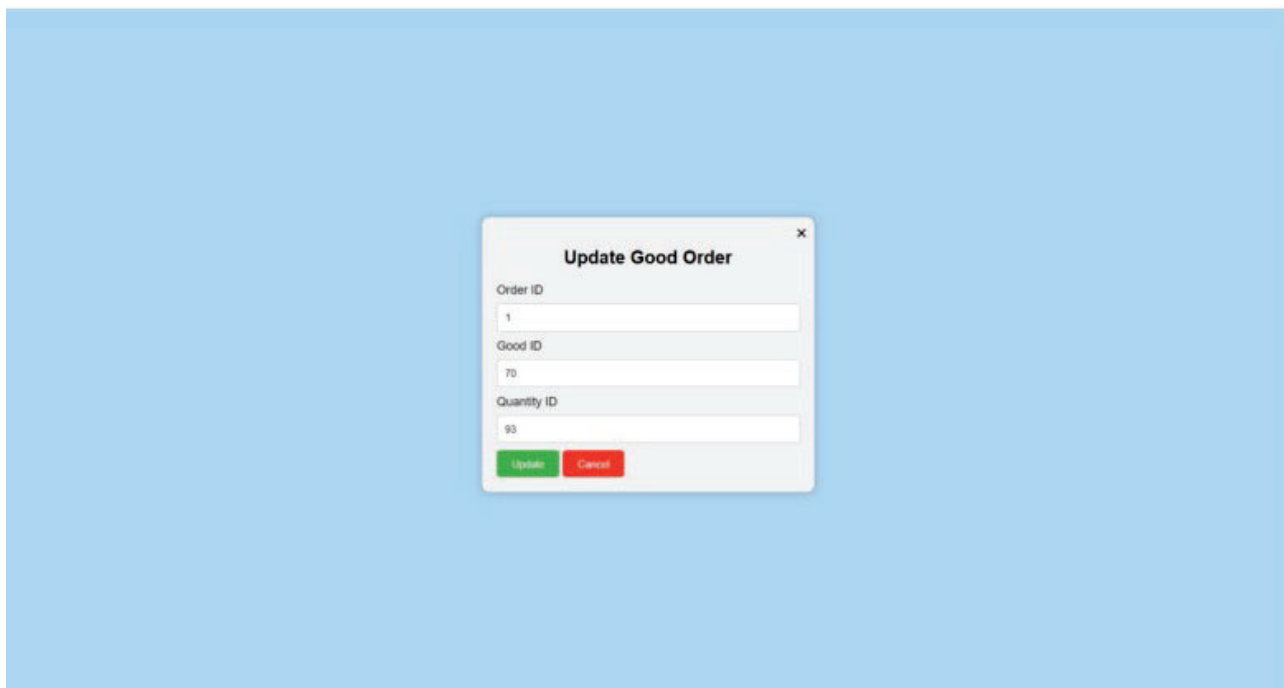


Рисунок 4.5 – Інтерфейс модального вікна редагування замовлення

З точки зору управління вимогами, розробка такого модального вікна (рис. 4.4) задовольняє бізнес-потребу клієнта у мінімізації кількості "кліків" оператора. Інтерфейс дозволяє миттєво перевести замовлення на наступний етап життєвого циклу логістики без перезавантаження сторінки. Гнучкий (Agile) підхід до створення UI гарантував, що програмний продукт відповідає реальним потребам персоналу складу, що значно скоротило витрати підприємства на навчання співробітників після впровадження WMS.

4.5 Управління проєктними комунікаціями та формування управлінської звітності

Згідно з методологією PMBOK, управління комунікаціями (Project Communications Management) є невід'ємною частиною життєвого циклу. Для Спонсора проєкту (Директора) та фінансового відділу сирі дані в базі не мають цінності; їм потрібна агрегована аналітика для прийняття управлінських рішень. Тому в рамках проєкту було розроблено аналітичний модуль, що генерує звіти за допомогою оптимізованих SQL-запитів.[15]

Для оцінки завантаженості логістичних потужностей розроблено скрипт підрахунку загальної кількості операцій на кожному складі. Це дозволяє топ-менеджменту виявляти "вузькі місця" (Bottlenecks) в інфраструктурі.

Лістинг 4.3 – SQL-запит генерації звіту про навантаження на склади (за останній рік)

```
SELECT      w.name      AS      warehouse_name,      COUNT(o.id)      AS
total_operations
FROM warehouses w
JOIN good_operation go ON w.id = go.warehouse_id
JOIN operations o ON go.operation_id = o.id
WHERE o.operation_date BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL
1 YEAR) AND CURRENT_DATE
GROUP BY w.name;
```

Для фінансового відділу (Stakeholder Group) розроблено модуль ABC-аналізу клієнтів. Запит у Лістингу 4.5 автоматично агрегує історію покупок та виділяє VIP-клієнтів, які згенерували дохід понад встановлений ліміт. Це допомагає компанії управляти програмами лояльності.

Лістинг 4.4 – SQL-запит формування звіту по найприбутковішим клієнтам

```
SELECT  c.id,  c.full_name,  SUM(g.price  *  go.quantity)  AS  
total_spent  
FROM  customers c  
JOIN  orders o ON c.id = o.customer_id  
JOIN  good_order go ON o.id = go.order_id  
JOIN  good g ON go.good_id = g.id  
GROUP BY c.id, c.full_name  
HAVING total_spent > 1000;
```

Такий підхід до управління інформацією перетворює WMS із простої облікової програми на потужний інструмент бізнес-аналітики, що максимізує ROI (повернення інвестицій) для замовника проєкту.

4.6 Практичне управління інформаційною безпекою та процесами верифікації

Інформаційна безпека проєкту реалізована через механізми JWT-токенізації та шифрування BCrypt, що гарантує захист комерційних даних замовника.

На етапі верифікації (QA) замість мануального тестування було впроваджено автоматизоване модульне тестування (JUnit/Mockito). З управлінської точки зору, головним показником успіху стала метрика покриття коду тестами (Code Coverage)

wms system

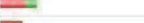
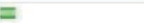
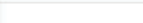
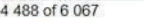
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.example.wms.system.models		16%		23%	322	354	180	280	284	316	16	26
com.example.wms.system.services		32%		17%	105	166	218	327	92	152	0	12
com.example.wms.system.controllers		15%		0%	73	93	111	131	56	76	0	11
com.example.wms.system.dto		32%		n/a	84	110	54	80	84	110	6	10
com.example.wms.system.payload		13%		0%	52	66	3	12	19	33	1	4
com.example.wms.system.exceptions		7%		n/a	38	43	60	68	38	43	13	17
com.example.wms.system.Security		8%		0%	9	10	22	26	1	2	0	1
com.example.wms.system.Config		79%		n/a	1	9	6	23	1	9	0	1
com.example.wms.system		37%		n/a	1	2	2	3	1	2	0	1
com.example.wms.system.enums		100%		n/a	0	8	0	42	0	8	0	8
Total	4 488 of 6 067	26%	197 of 220	10%	685	861	656	992	576	751	36	91

Рисунок 4.6 – Результати аналізу покриття коду тестами

Як свідчить аналіз (рис. 4.5), загальне покриття методів системи склало 26%. Управлінське рішення свідомо не вимагало 100% покриття, оскільки класи моделей та DTO-об'єкти (які не містять бізнес-логіки) були виключені з тестування для економії бюджету та часу проєкту. Основний фокус тестування було спрямовано виключно на класи-сервіси (Services), що є оптимальною стратегією мінімізації технічного боргу для першого MVP-релізу.

4.7 Управління проєктними ризиками та стратегії реагування

Протягом усього життєвого циклу проєкту здійснювався моніторинг ризиків. Для систематизації було сформовано Реєстр ризиків (табл. 4.1).

Таблиця 4.1 – Реєстр ключових ризиків проєкту та заходів реагування

Ризик	Вплив	Ймовірність	Стратегія реагування
Пересортиця товару (людський фактор)	Високий	Висока	Впровадження обов'язкового сканування штрихкоду перед закриттям операції.

Кінець таблиці 4.1

Ризик	Вплив	Ймовірність	Стратегія реагування
Розсинхронізація даних (технічний збій)	Високий	Середня	Перенесення логіки на рівень SQL-тригерів (MySQL InnoDB).
Опір персоналу (соціальний фактор)	Середній	Висока	Ітеративна розробка (Scrum) із залученням менеджерів до тестування UI.
Витік даних (кіберзагроза)	Критичний	Низька	Впровадження JWT-авторизації та хешування паролів.

Застосована гібридна модель дозволила нівелювати більшість ідентифікованих ризиків ще на ранніх етапах спринтів.

4.8 Висновки до четвертого розділу

У четвертому розділі продемонстровано результати практичного впровадження гібридної моделі управління. Доведено, що:

- поєднання Waterfall-планування інфраструктури та Scrum-розробки інтерфейсів дозволило вчасно реалізувати продукт, що відповідає OBS підприємства;
- впровадження SQL-тригерів та автоматизованих тестів забезпечило стабільність системи при оптимальних витратах на розробку;
- модуль аналітичної звітності надав керівництву інструмент контролю ROI проєкту.

ВИСНОВКИ

У кваліфікаційній магістерській роботі було проведено комплексне дослідження методів управління життєвим циклом ІТ-проєкту зі створення інформаційної системи для автоматизації складських операцій (WMS). За результатами виконання роботи було зроблено наступні основні висновки.

Аналіз сучасного стану та актуальність – доведено, що управління життєвим циклом складських ІТ-проєктів характеризується високим рівнем ризиків через необхідність тісної інтеграції ПЗ з апаратним забезпеченням та критичність безперервності бізнес-процесів логістики. Встановлено, що класичні каскадні та гнучкі моделі у чистому вигляді не повною мірою задовольняють вимоги до надійності та адаптивності таких систем.

Теоретичне обґрунтування: обґрунтовано та розроблено гібридну модель управління життєвим циклом на основі концепції «Двомодальних ІТ» (Bimodal IT). Теоретично доведено, що поєднання предиктивного планування інфраструктури (Режим 1) з ітеративною розробкою користувацьких інтерфейсів (Режим 2) дозволяє оптимізувати криву вартості змін та мінімізувати архітектурні дефекти.

Практичне моделювання: На основі методологій IDEF0 та DFD проведено структурну декомпозицію проєкту, що дозволило чітко зафіксувати обсяг робіт та інформаційні потоки системи. Розроблено WBS та специфіковано вимоги для різних ролей стейкхолдерів (менеджерів складу та адміністраторів).

Управлінські рішення щодо технологічного стека: Аргументовано вибір сучасного технологічного стека (Java, Spring Boot, React, MySQL) як інструмента зниження технічних ризиків. Впровадження бізнес-логіки на рівні СУБД за допомогою SQL-тригерів визначено як ефективний управлінський крок для гарантування транзакційної цілісності даних незалежно від помилок клієнтської частини.

Якість та безпека: Реалізовано методологію верифікації проєкту через

автоматизоване модульне тестування (JUnit, Mockito), що дозволило скоротити тривалість фази стабілізації продукту . Практично впроваджено політику безпеки на основі JWT-токенізації та рольового доступу, що відповідає сучасним стандартам захисту корпоративної інформації.

Значущість та впровадження: Одержані результати мають практичну значущість для логістичних підприємств, оскільки запропонована методологія управління дозволяє скоротити термін виходу системи на ринок (Time-to-Market) та підвищити прозорість процесів розробки для замовника.

Доцільність подальших досліджень полягає в адаптації розробленої моделі для проєктів з використанням робототехніки на складах та інтеграції алгоритмів штучного інтелекту для прогнозування запасів.

Роботу виконано відповідно до вимог методичних вказівок щодо розробки та оформлення кваліфікаційних робіт [18], а також із дотриманням положень державних стандартів ДСТУ 8302:2015 [19] і ДСТУ 3008:2015 [20].

За темою дослідження опубліковані тези на 30-му молодіжному форумі секції 6.2 «Інформаційні системи і технології управління проєктною та операційною діяльністю підприємств та організацій» конференції «Інформаційні інтелектуальні системи» [3].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Левшин О. В. Управління IT-проєктами в логістиці : монографія. Київ : Центр учбової літератури, 2021. 256 с.
2. ДСТУ ISO/IEC/IEEE 15288:2015. Системна та програмна інженерія. Процеси життєвого циклу систем. Київ : ДП «УкрНДНЦ», 2016. 108 с.
3. Іщенко Д. Г., Євланов М. В. Дослідження гібридних методів управління життєвим циклом IT-проєкту створення WMS-системи. XXX Міжнародний молодіжний форум «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ В XXI СТОЛІТТІ». Харків : ХНУРЕ, 2026.
4. Bimodal IT: How to Be Digitally Agile Without Making a Mess. *Gartner Research*. 2019. URL: <https://www.gartner.com/en/documents/3981418> (дата звернення: 14.04.2026).
5. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 7th ed. Newtown Square, USA : Project Management Institute, 2021. 370 p.
6. Kerzner H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. 13th ed. Wiley, 2022. 848 p.
7. Agile Practice Guide. Newtown Square, USA : Project Management Institute, 2017. 168 p.
8. Schwaber K., Sutherland J. The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. 2020. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf> (дата звернення: 12.04.2026).
9. Kim G., Humble J., Debois P. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2021. 480 p.
10. Docker Documentation: Best practices for writing Dockerfiles. URL: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/ (дата звернення: 18.04.2026).

11. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.
12. Левикін В. М., Євланов М. В., Сугробов В. С. Паралельне проектування інформаційного та програмного комплексів інформаційної системи. *Радіотехніка*. 2006. Вип. 146. С. 89–98.
13. MySQL 8.0 Reference Manual. *Oracle Corporation*. URL: <https://dev.mysql.com/doc/refman/8.0/en/> (дата звернення: 10.04.2026).
14. Томашевський О. М., Цегелик Г. Г., Вітер М. Б., Дудук В. І. Інформаційні технології та моделювання бізнес-процесів : навч. посібник. Київ : Центр учбової літератури, 2023. 296 с.
15. Іщенко Д. Г. Розробка компонентів інформаційної системи управління складськими процесами : кваліфікаційна робота бакалавра. Харків : ХНУРЕ, 2024. 70 с.
16. Spring Boot Reference Documentation. URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/> (дата звернення: 15.04.2026).
17. React Documentation: Thinking in React. URL: <https://react.dev/learn/thinking-in-react> (дата звернення: 15.04.2026).
18. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи магістра за спеціальністю 122 Комп'ютерні науки (освітня програма «Управління проектами в галузі інформаційних технологій») / упоряд. К. Е. Петров та ін. Харків : ХНУРЕ, 2024. 24 с.
19. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 20 с.
20. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. Київ : ДП «УкрНДНЦ», 2016. 31 с.