НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет/(ННІ) інформаційних технологій

ПОГОДЖЕНО
Декан факультету (Директор ННІ)
інформаційних технологій
(назва факультету (ННІ))
_____ Ігор Болбот
(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 20_ р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
комп'ютерних наук
(назва кафедри)
_____ Белла Голуб
(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 20_ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Програмна система управління складом підприємства

Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
доц.,к.ф.-м.н. / Віктор Кириченко /
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи
ст. викладач _____ Панкратьєв В. О.
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Консультант магістерської кваліфікаційної роботи
к.т.н., доцент _____ Ткаченко О,М.
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ Дмитрій Симон
(підпис) (ім'я ПРІЗВИЩЕ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) __інформаційних технологій_____

ЗАТВЕРДЖУЮ
Завідувач кафедри__комп'ютерних наук_
___доцент, к.т.н._____ _____ _Белла Голуб
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"10" жовтня 2025 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ
_____Симон Дмитрій Вікторович._____
(прізвище, ім'я, по батькові)

Спеціальність 121 Інженерія програмного забезпечення
(код і назва)

Освітня програма ____Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____

Тема магістерської кваліфікаційної роботи Програмна система управління складом підприємства
затверджена наказом ректора НУБіП України від "_10_" _жовтня__2025р. №2291 «С»
Термін подання завершеної роботи на кафедру 11 листопада 2025_____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: Нормативно-довідкова та наукова література з інформаційних систем, баз даних, методів візуалізації та обробки даних; технічна документація з проєктування та розробки програмних комплексів; програмні засоби для розробки веб-застосунків і інтерактивних інтерфейсів; відкриті дані; вимоги стандартів до побудови інформаційних систем і користувацьких інтерфейсів.

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області
2. Моделювання та архітектурне проєктування системи
3. Реалізація програмного забезпечення та технологічна інфраструктура системи
4. Тестування та оцінювання ефективності системи

Перелік графічного матеріалу (за потреби) презентація, постер, схеми та діаграми архітектури системи

Дата видачі завдання "__1__" листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ ___Панкратьєв В.О.
(підпис) (ім'я ПРІЗВИЩЕ))

Консультант магістерської кваліфікаційної роботи _____ ___Ткаченко О.М.
(підпис) (ім'я ПРІЗВИЩЕ))

Завдання прийняв до виконання _____ Дмитрій Симон
(підпис) (ім'я ПРІЗВИЩЕ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface, програмний інтерфейс взаємодії компонентів.

ARIMA – AutoRegressive Integrated Moving Average, авторегресійно-інтегрована модель середнього ковзання.

AS – Anomaly Severity, метричний показник ступеня аномальності даних.

CPU – Central Processing Unit, центральний процесор.

CSV – Comma-Separated Values, формат табличних текстових даних.

DAQ – Data Acquisition, процес збирання даних із сенсорних вузлів.

DB – Database, база даних.

DQI – Data Quality Index, індекс якості даних.

EMQX – високопродуктивний MQTT-брокер для IoT-систем.

ESP32 – мікроконтролер IoT-шлюзу сенсорних вузлів.

FNN – Feedforward Neural Network, багаторівнева нейронна мережа прямого поширення.

HTML – HyperText Markup Language, мова розмітки веб-документів.

HTTP(S) – HyperText Transfer Protocol (Secure), протокол передавання гіпертекстових даних.

IoT – Internet of Things, інтернет речей.

JSON – JavaScript Object Notation, формат структурованих даних.

K-means – метод кластеризації на основі середніх значень.

LSTM – Long Short-Term Memory, довготривала рекурентна пам’ять для прогнозування часових рядів.

Lux – одиниця вимірювання інтенсивності освітленості.

MAE – Mean Absolute Error, середня абсолютна похибка моделі.

MQTT – Message Queuing Telemetry Transport, легковаговий брокерський протокол обміну телеметрією.

ORM – Object-Relational Mapping, технологія відображення об’єктів у реляційну БД.

PCA – Principal Component Analysis, метод головних компонент для зниження розмірності даних.

PDF – Portable Document Format, формат електронних документів.

PHI – Plant Health Index, інтегральний індекс стану рослини.

pH – водневий показник кислотності ґрунту.

RAM – Random Access Memory, оперативна пам’ять.

REST – Representational State Transfer, архітектурний стиль веб-взаємодії.

RMSE – Root Mean Square Error, середньоквадратична похибка прогнозної моделі.

ВСТУП

Помилки в інвентаризації, затримки у формуванні відвантажень, відсутність актуальної інформації про залишки та непрогнозованість завантаження складу призводять до фінансових втрат, зниження продуктивності та погіршення клієнтського сервісу. Це зумовлює необхідність упровадження інтелектуалізованих систем управління складськими процесами, здатних забезпечувати автоматизацію критичних операцій, підтримку прийняття рішень і комплексний аналіз даних у режимі реального часу. У світовій практиці такі підходи реалізуються у форматі Warehouse Management Systems (WMS), що базуються на модульній архітектурі, цифровій трансформації матеріальних потоків та методах оптимізації [1]. Відповідно, розроблення вітчизняної програмної системи управління складом, орієнтованої на специфіку локальних підприємств, інтеграцію з ERP-платформами та забезпечення високого рівня надійності, є актуальним та практично значущим завданням.

Метою роботи є розроблення програмної системи управління складом підприємства, що забезпечує автоматизований облік товарних залишків, підтримку логістичних операцій та аналітичне опрацювання даних для підвищення ефективності складської діяльності.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

– виконати системний аналіз предметної області складської логістики та чинних підходів до організації WMS;

– проаналізувати науково-методичні засади управління складськими процесами, моделі оптимізації та дані, що формуються в логістичних системах;

– дослідити існуючі апаратно-програмні рішення й визначити їх переваги, недоліки та релевантність для малого й середнього бізнесу;

– побудувати UML-моделі, включаючи діаграми прецедентів, активності, послідовності, компонентів та розгортання;

– сформувати вимоги до програмної системи, включно з функціональними, нефункціональними, технічними та інформаційними характеристиками;

– розробити архітектуру програмної системи з урахуванням модульності, масштабованості, інтеграційних можливостей і безпеки;

– спроєктувати логічну та фізичну модель бази даних;

– реалізувати ключові програмні модулі та провести тестування їх функціональної коректності;

– оцінити ефективність функціонування системи у порівнянні з традиційними підходами складського менеджменту.

Об’єкт дослідження – процеси управління складською діяльністю підприємства та інформаційні потоки, що їх супроводжують.

Предмет дослідження – методи, моделі та програмні засоби автоматизованого управління складом, включаючи алгоритми обліку, оптимізації та аналітики.

Методи дослідження ґрунтуються на системному та структурному аналізі; методах моделювання складських процесів; UML-підходах до проєктування програмних систем; теорії баз даних; методах оптимізації матеріальних потоків; алгоритмах аналітичної обробки даних; програмній інженерії та інтеграції програмних компонентів.

Наукова новизна роботи полягає у формуванні інтегрованої моделі управління складськими процесами, що поєднує автоматизований облік залишків, оптимізацію операцій приймання й відвантаження, адаптивну систему моніторингу та модуль аналітики на основі даних у реальному часі. Новизна проявляється також у побудові уніфікованої архітектури, яка забезпечує масштабованість, підтримку різнорідних каналів введення даних, можливість інтеграції з ERP/CRM-платформами та розширення функціоналу залежно від потреб підприємства.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Опис предметної області програмної системи управління складом підприємства охоплює сукупність логістичних, інформаційних та облікових процесів, які забезпечують безперервний рух матеріальних потоків між постачальниками, складською інфраструктурою, клієнтами та внутрішніми службами підприємства. Основною функцією цієї предметної області є своєчасне та коректне відображення операцій з товарними запасами, включно з прийманням, зберіганням, відвантаженням, інвентаризацією, коригуванням залишків та формуванням аналітичних звітів.

На рис. 1.1 подано структурну модель потоків даних, яка демонструє взаємозв’язки між підсистемами «Облік надходжень», «Облік відвантажень», «Управління залишками», «Інвентаризація» та «Звіти та аналітика», а також їх взаємодію з базами даних товарних запасів, контрагентів і журналом операцій.

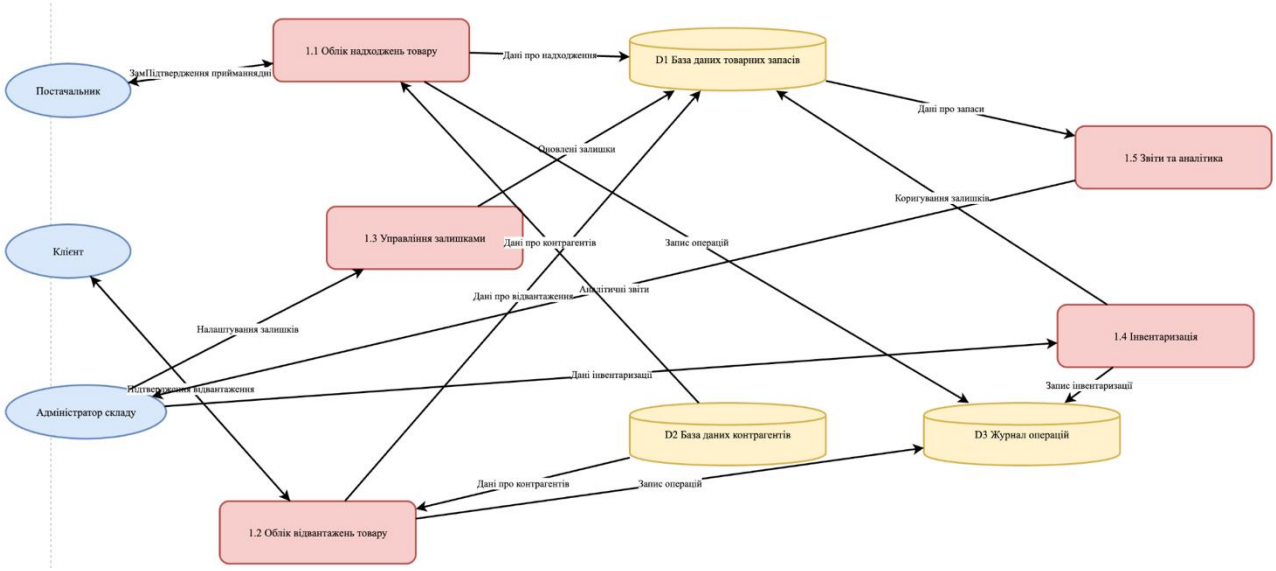

Рис. 1.1. DFD-модель матеріальних та інформаційних потоків предметної області системи управління складом.

Функціонування предметної області передбачає комплекс дій, що формують логічно цілісний цикл управління товарними запасами. Приймання товару включає реєстрацію партій, перевірку супровідних документів,

підтвердження відповідності замовленням та формування записів про надходження у центральній базі даних.

Облік відвантажень охоплює резервування продукції, підтвердження відбору, формування документів відвантаження та передання даних у журнал операцій для подальшого контролю. Управління залишками забезпечує актуальність кількісних і структурних характеристик товарних запасів, виконуючи коригування, списання, переобліки та підтримку налаштувань мінімальних рівнів запасів. Інвентаризація спрямована на звірку фактичних залишків із даними системи, виявлення розбіжностей, реєстрацію результатів та оновлення облікових записів. Аналітичний компонент предметної області відповідає за підготовку статистичних, оперативних і прогнозних звітів на основі журналу подій, історії руху товарів та взаємодії з контрагентами.

У табл. 1.1 наведено класифікацію ключових сутностей предметної області, яка використовується під час моделювання та проектування інформаційної структури системи. Текст посилання на таблицю: дані табл. 1.1 узагальнюють основні характеристики товарів, контрагентів та операцій, що дозволяє формалізувати логічні залежності між підсистемами та визначити вимоги до структури бази даних.

Таблиця 1.1

Ключові сутності предметної області системи управління складом

№	Сутність	Опис	Основні атрибути	Джерело або споживач даних
1	Товарні запаси	Масив одиниць продукції, що зберігається на складі	Ідентифікатор товару, назва, категорія, партія, кількість, місце зберігання	Підсистеми надходжень, відвантажень, інвентаризації
2	Контрагенти	Постачальники, клієнти та інші учасники товарообігу	Ідентифікатор контрагента, тип, реквізити, історія операцій	Підсистеми обліку, управління залишками
3	Операції	Дії, що змінюють стан товарних запасів	Тип операції, дата, користувач, кількість, товар, контрагент	Журнал операцій, система звітності

Продовження таблиці 1.1

4	Журнал подій	Лог змін і критичних подій у системі	Час, подія, модуль, результат	Аналітичний модуль
5	Аналітичні дані	Агреговані та обчислені показники діяльності	Показники оборотності, залишків, попиту, ефективності	Підсистема аналітики

Предметна область системи формує багаторівневу структуру оперативних, аналітичних та облікових процесів, яка забезпечує повний цикл роботи складу та створює основу для побудови інтелектуальної інформаційної системи, що здатна підтримувати прийняття управлінських рішень на основі актуальних даних.

1.2 Теоретико-методологічні засади та стан наукових досліджень

Теоретико-методологічні засади управління складськими запасами ґрунтуються на класичних і сучасних моделях інвентарного менеджменту, що описують динаміку попиту, витрат і рівнів запасів у логістичних системах. У фундаментальних роботах Silver, Pyke та Peterson сформовано базові моделі детермінованого і стохастичного керування запасами, включно з EOQ-моделлю, моделями з дефіцитами та обмеженою місткістю складу. У новітніх дослідженнях Davizón та співавт. (2025) запропоновано причинно-наслідковий підхід до оцінювання впливу зміни параметрів попиту ΔQ на рівень запасів ΔI через проміжну змінну витрат ΔC. Як показано на рис. 1.2, дана методологія дозволяє одночасно враховувати прямий ефект зміни попиту та його опосередкований вплив через зміну логістичних витрат.

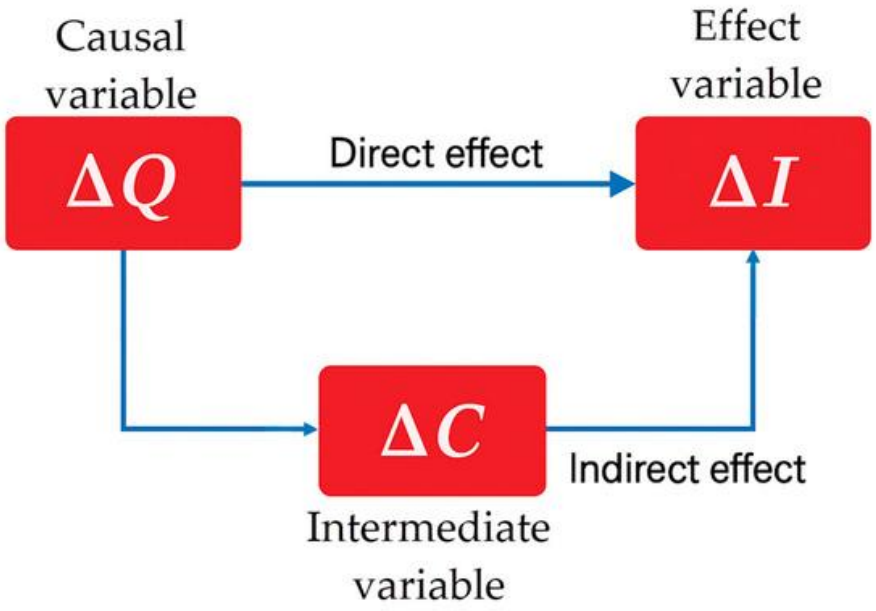

Рис. 1.2. Причинно-наслідкова структурна модель впливу зміни попиту на рівень запасів.

У роботі Holovan (2024) наведено багатопродуктову статичну модель управління запасами з урахуванням обмеженої місткості складу. Автор зазначає, що оптимальний розподіл ресурсів залежить не лише від структури попиту, але й від коефіцієнта чутливості витрат α. На рис. 1.3 показано залежність загальних витрат C від змінної G за різних значень α.

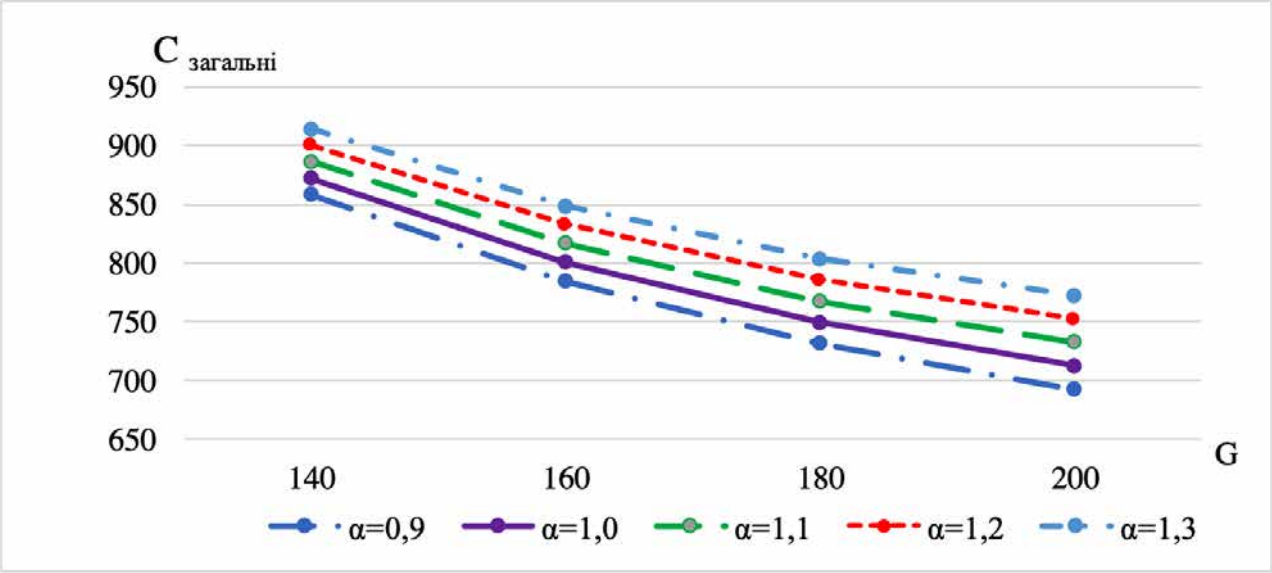

Рис. 1.3. Залежність загальних витрат C від параметра G для різних значень коефіцієнта чутливості α.

У дослідженнях Taleizadeh та співавт. (2021) розглянуто стохастичні моделі із частковим backorder та перебоями постачання. Автори демонструють, що зміна рівня незадоволеного попиту (backorder ratio) суттєво впливає на середні сумарні логістичні витрати ATC. На рис. 1.4 наведено результати числового моделювання для шести різних випадків співвідношення величин витрат g, h та b.

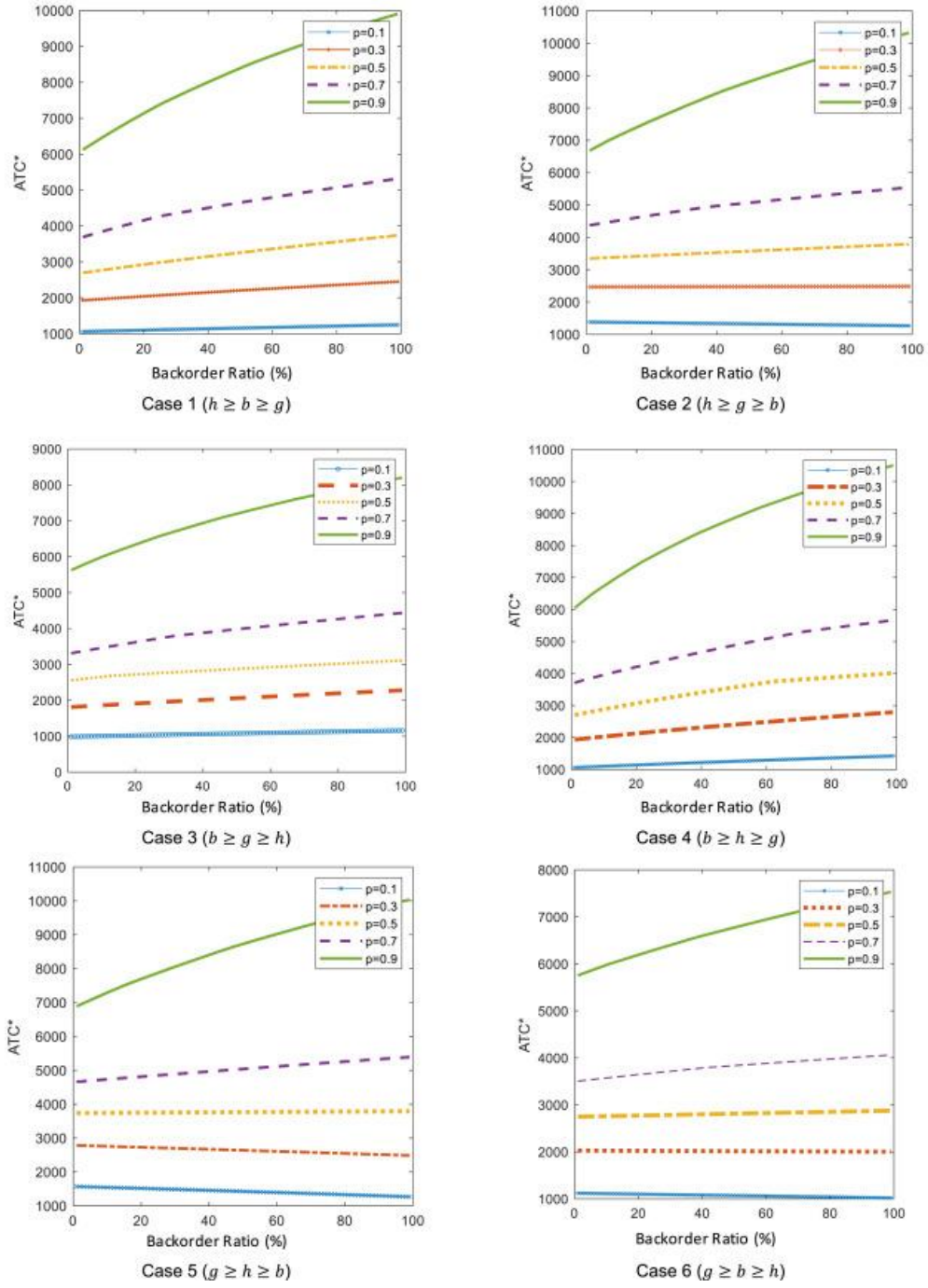

Рис. 1.4. Залежність середніх сумарних витрат ATC від рівня backorder для різних конфігурацій стохастичної моделі.

У роботі Vaičiūtė та Petraškevičius (2025) доведено, що рівень впровадження логістичних інформаційних систем прямо впливає на якість логістичного сервісу. На рис. 1.5 наведено емпіричну залежність якості сервісу від рівня цифровізації складських операцій.

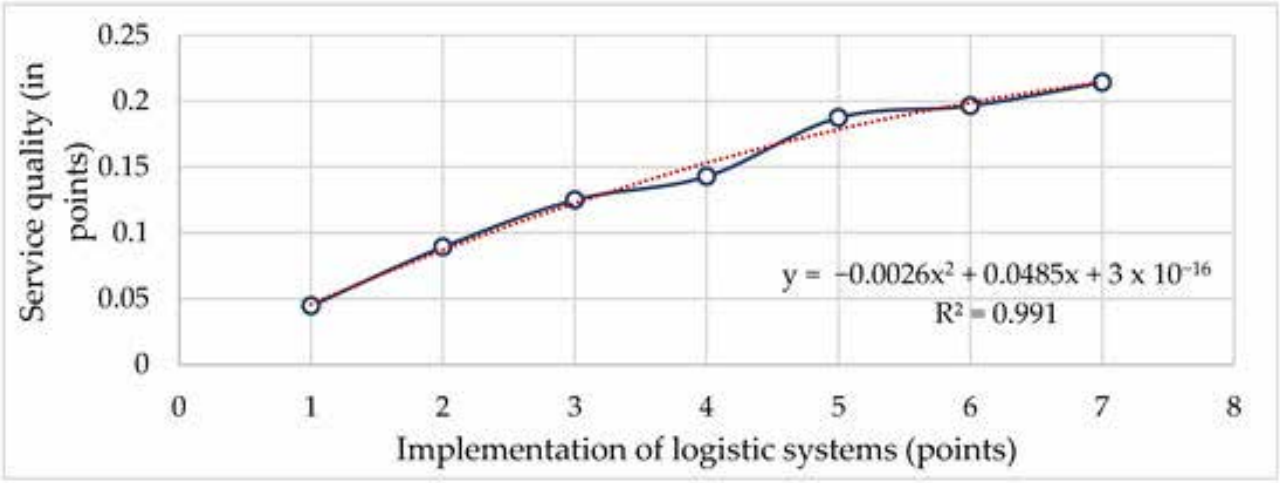

Рис. 1.5. Залежність якості логістичного сервісу від рівня впровадження логістичних систем.

У дослідженні Mojumder (2025) встановлено, що збільшення обсягу даних без належних інтелектуальних механізмів опрацювання призводить до зростання кількості логістичних проблем. На рис. 1.6 показано залежність кількості проблем від обсягу даних у складській системі.

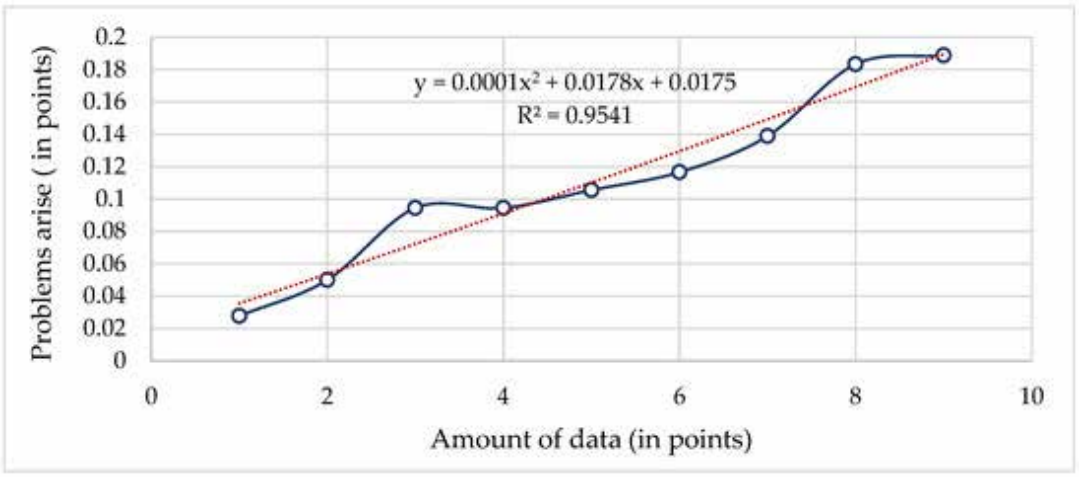

Рис. 1.6. Вплив обсягу даних на кількість логістичних проблем у WMS-системах.

Аналіз теоретико-методологічних засад та сучасних наукових досліджень показує, що сучасні складські системи функціонують у середовищі високої

невизначеності, стохастичності попиту та зростаючого інформаційного навантаження. Класичні моделі керування запасами (Silver, Peterson) забезпечують фундамент математичного опису процесів, але недостатньо враховують складну багатофакторну природу сучасної логістики. Сучасні дослідження Davizón, Holovan, Taleizadeh, Vaičiūtė та Mojumder демонструють, що ефективність управління складом залежить не лише від параметрів замовлення, а й від цифрового рівня системи, інтенсивності інформаційних потоків, частки незадоволеного попиту та структурних взаємодій між змінними. На підставі цього формулюється наукова новизна нашої роботи: у ній запропоновано інтегровану програмну систему управління складом, яка поєднує класичні моделі керування запасами з інтелектуальними механізмами прогнозування, адаптивного регулювання залишків та цифрової аналітики, що забезпечує зниження логістичних витрат, підвищення точності обліку та поліпшення сервісної якості підприємства.

1.3 Аналіз існуючих систем управління складськими операціями

Сучасні системи управління складськими операціями орієнтуються на забезпечення високої точності обліку, оптимізацію логістичних процесів та інтеграцію з ERP-системами підприємств. Найбільш поширеними рішеннями є Oracle NetSuite WMS, SAP S/4HANA EWM, Infor WMS, Manhattan WMS та 1С:Управление складом. Кожна з цих платформ реалізує власні підходи до автоматизації процесів відстеження товарних запасів, контролю відвантажень, інвентаризації та формування аналітичних звітів. На рис. 1.7 показано інтерфейс Oracle NetSuite WMS, який демонструє структуру панелі управління, KPI-метрики та аналітичні графіки, що дозволяють адміністратору складу ухвалювати рішення в режимі реального часу.

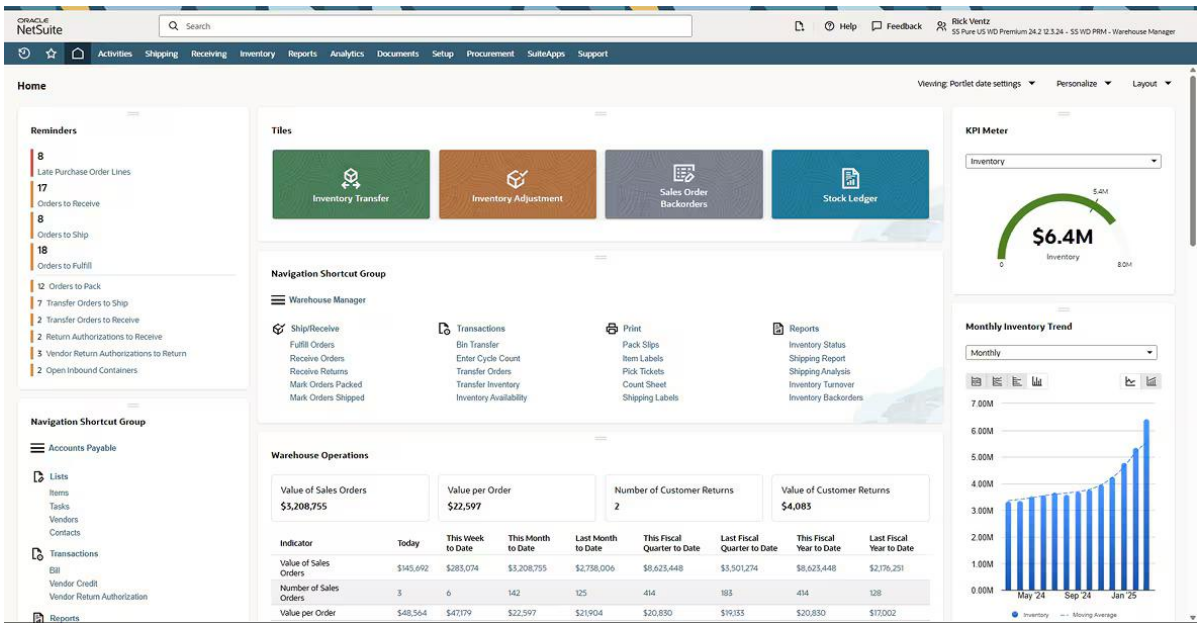

Рис. 1.7. Головний екран Oracle NetSuite WMS із KPI-панеллю та операційною аналітикою.

Система SAP S/4HANA EWM реалізує розширений функціонал управління запасами, інтегрований з ERP-модулем підприємства. Основними її можливостями є централізоване управління складськими зонами, оптимізація переміщень, розрахунок продуктивності та автоматизоване планування. На рис. 1.8 наведено приклад робочого інтерфейсу SAP S/4HANA, який відображає модульну структуру інформаційної панелі та аналітичні віджети для контролю бізнес-процесів.

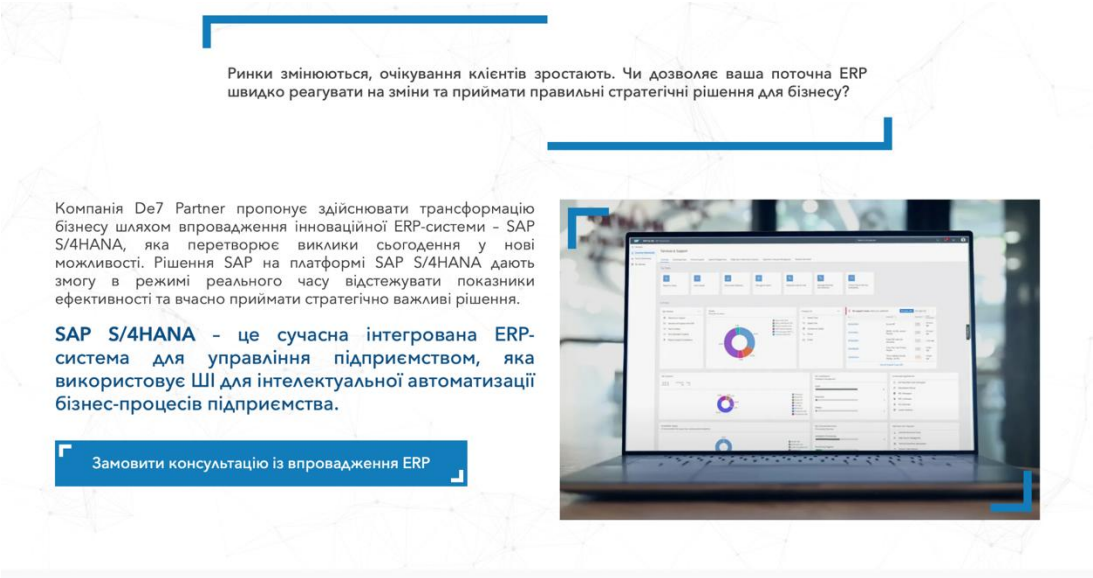

Рис. 1.8. Інтерфейс SAP S/4HANA з інтегрованим модулем EWM та елементами аналітичної візуалізації.

Наступним прикладом є Infor WMS, що позиціонується як комплексна система для управління продуктивністю складських працівників та оптимізації внутрішніх операцій. На рис. 1.9 подано зразок візуалізації продуктивності персоналу, де відображено діяльність співробітників, ефективність робочих процесів та гістограми навантаження.

Рис. 1.9. Аналітичні панелі Infor WMS для оцінювання ефективності складських процесів.

Манхеттенська система Manhattan WMS орієнтована на великі та розподілені склади. Вона реалізує функції автоматизованого контролю технологічних операцій, маршрутизації переміщень, прогнозування навантаження та оптимізації зон зберігання. На рис. 1.10 наведено приклад інтерфейсу Manhattan WMS з відображенням показників продуктивності, кількості виконаних операцій і графіків змін активності.

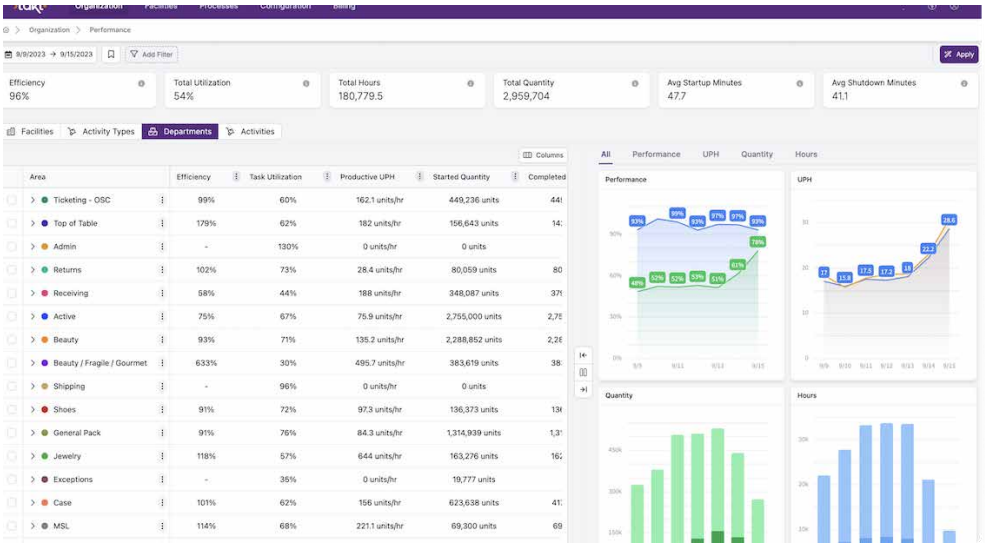

Рис. 1.10. Інформаційна панель Manhattan WMS з показниками ефективності.

Останнім прикладом є система 1С:Управление складом, широко поширена на підприємствах України та СНД. Вона підтримує детальний облік товарів, налаштування складських зон, формування документів надходжень, відвантажень та інвентаризаційних відомостей. На рис. 1.11 показано фрагмент робочого інтерфейсу 1С, що демонструє структуру форм, параметри відбору і правила розміщення товарів.

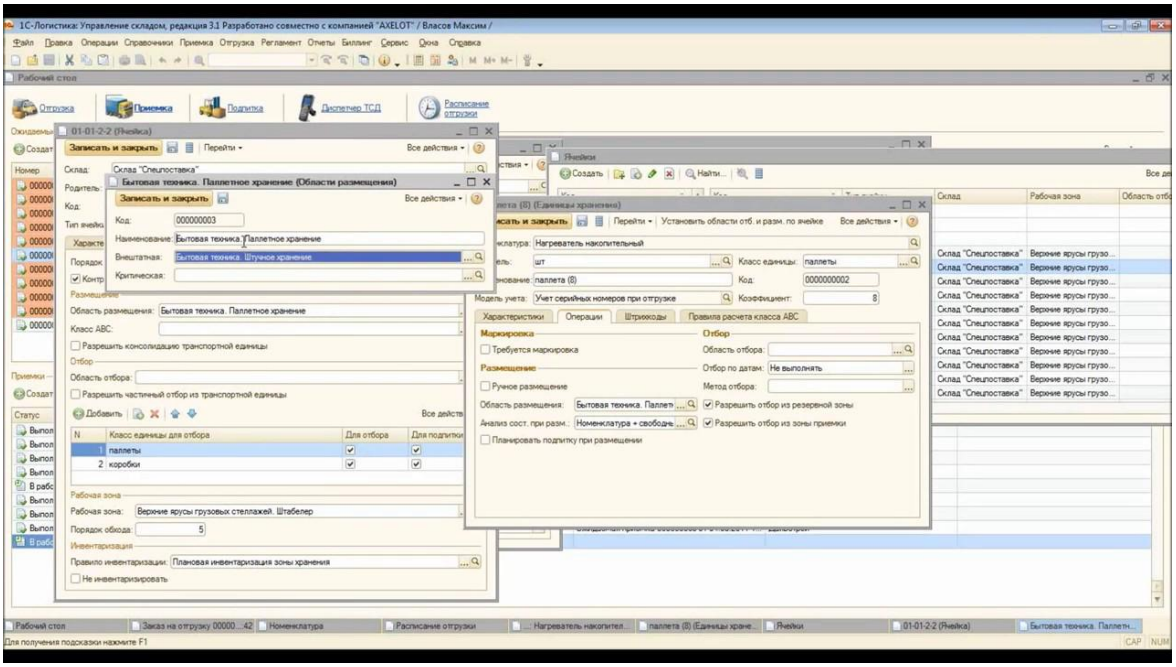

Рис. 1.11. Робочий інтерфейс системи 1С:Управление складом з параметрами обліку та зберігання.

Для формування об’єктивної оцінки розглянутих систем доцільно провести їх порівняння за ключовими критеріями, такими як функціональність, рівень автоматизації, гнучкість налаштування, наявність аналітики, інтеграція з іншими модулями та зручність інтерфейсу. У таблиці 1.2 наведено порівняльну характеристику існуючих рішень та проєктованої програмної системи управління складом. Текст посилання на таблицю: у таблиці 1.2 подано характеристику основних можливостей систем, що дозволяє визначити конкурентні переваги розроблюваного рішення.

Таблиця 1.2

Порівняльна характеристика існуючих систем управління складом та розроблюваної системи

Система	Автоматизація	Аналітика	Інтеграції	Гнучкість	Наша система
Oracle NetSuite	Висока	Розширена	ERP/CRM	Висока	Підтримка адаптивних моделей
SAP S/4HANA EWM	Висока	Аналітичні модулі	Повна інтеграція	Середня	Модульність і масштабованість
Infor WMS	Висока	Глибока аналітика персоналу	ERP	Середня	Прогнозування і оптимізація
Manhattan WMS	Дуже висока	Багатовимірна	Індустріальні інтеграції	Середня	Інтелектуальні рекомендації
1С:Управление складом	Середня	Обмежена	ERP	Висока	Проста адаптація під український бізнес

Аналіз існуючих рішень показує, що сучасні WMS-системи мають високий рівень автоматизації та забезпечують широкий спектр інструментів для контролю товарних потоків і аналітичної обробки даних. Проте більшість із них потребує значних ресурсів для впровадження, має складну архітектуру і не враховує специфіку невеликих та середніх підприємств. Запропонована у даній роботі програмна система спрямована на усунення цих недоліків шляхом поєднання модульного функціоналу, адаптивних механізмів управління запасами, інтегрованої аналітики і підтримки гнучкого інтерфейсу, що забезпечує ефективну автоматизацію складських процесів для широкого кола підприємств.

1.4 Моделювання предметної області

Моделювання предметної області програмної системи управління складом підприємства ґрунтується на формальному описі функціональної взаємодії між користувачами системи та ключовими процесами, що забезпечують облік товарних залишків, управління рухом товарів, виконання інвентаризації та формування аналітичних звітів. Використання UML-моделей дозволяє структуровано відобразити функції, інформаційні потоки та алгоритмічні залежності, що існують у межах складських операцій. На рис. 1.12 показано діаграму прецедентів системи, що визначає ролі користувачів (адміністратор складу, менеджер з продажу, системний адміністратор) та їх взаємодію з основними функціями системи, зокрема реєстрацією надходжень і відвантажень, веденням довідника товарів, управлінням доступом та формуванням звітів.

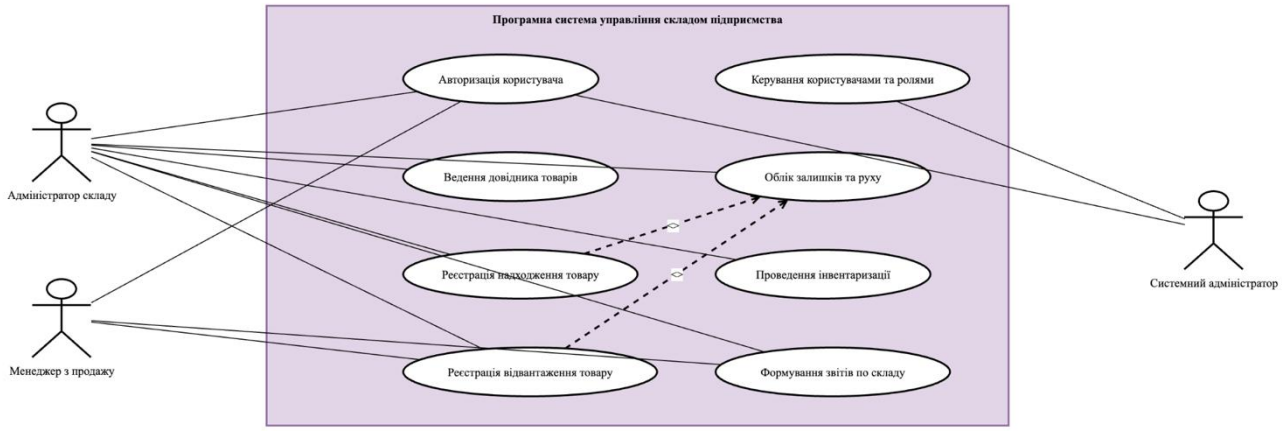

Рис. 1.12. Діаграма прецедентів програмної системи управління складом підприємства.

Послідовність взаємодії під час виконання операцій відвантаження товару є важливою частиною логіки системи, оскільки вона визначає узгодженість облікових даних та послідовність викликів програмних модулів. На рис. 1.13 представлено діаграму послідовності, що відображає кроки від введення замовлення адміністратором складу до оновлення запасів у базі даних і формування аналітичних показників.

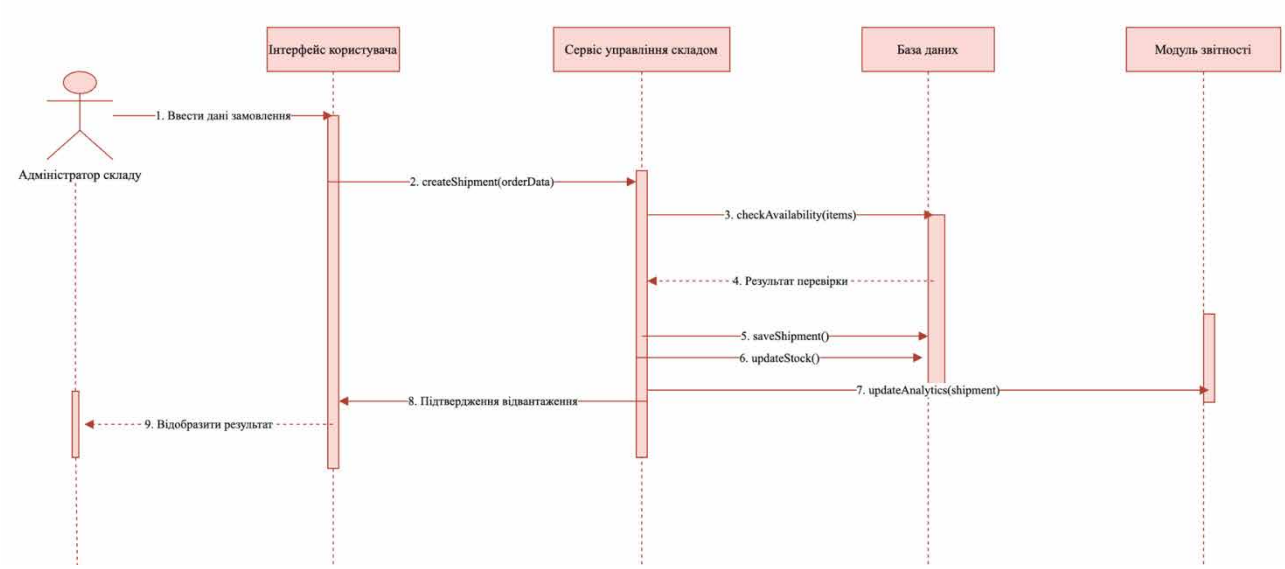

Рис. 1.13. Діаграма послідовності процесу реєстрації відвантаження товару.

Алгоритмічна поведінка системи також потребує формалізації через відображення внутрішніх логічних гілок та умов прийняття рішень. На рис. 1.14 подано діаграму діяльності процесу обробки замовлення, що включає введення даних, перевірку їх коректності, оцінку наявності товару, резервування, списання, формування документів відвантаження та оновлення аналітичних показників.

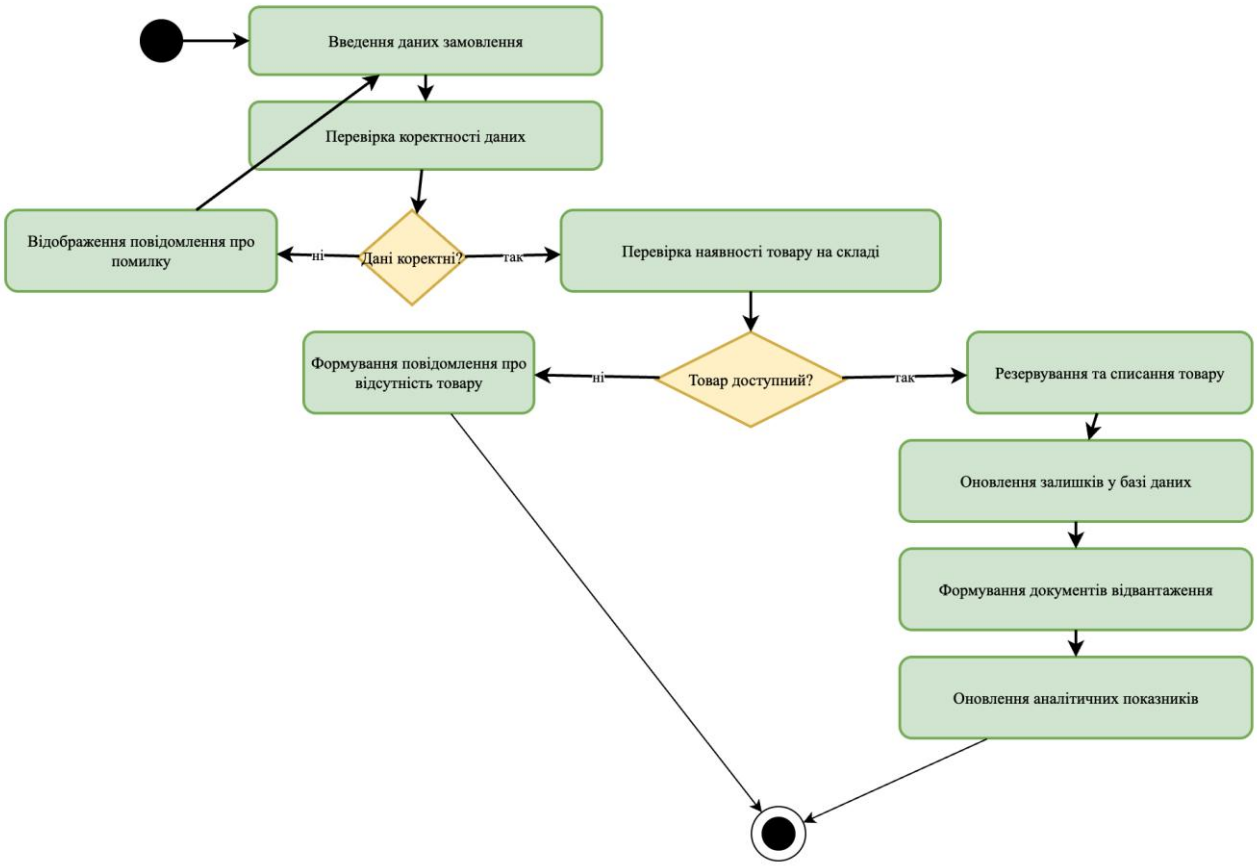

Рис. 1.14. Діаграма діяльності процесу обробки замовлення в системі управління складом.

Проведене моделювання дозволяє структурувати предметну область і формалізувати основні компоненти та взаємодії системи управління складом. Діаграма прецедентів визначає набір користувачів та їх функціональні можливості, діаграма послідовності описує динаміку викликів між модулями системи, а діаграма діяльності деталізує алгоритмічну логіку виконання складських операцій. Сукупність цих моделей формує цілісну основу для подальшого проектування функціональної архітектури, розроблення бази даних та впровадження програмної системи, забезпечуючи повну відповідність вимогам автоматизації складських процесів.

1.5 Аналіз вимог розроблюваної системи

Аналіз вимог експертної системи управління складом підприємства ґрунтується на завданнях автоматизації ключових операцій складської логістики, забезпечення достовірності обліку та реалізації експертного механізму прийняття рішень. У межах цього пункту сформовано три групи вимог: функціональні, нефункціональні та вимоги до безпеки, що визначають подальшу архітектуру та алгоритмічну основу системи.

У таблиці 1.3 наведено функціональні вимоги, що відображають логіку роботи складської системи та експертного модуля.

Таблиця 1.3

Функціональні вимоги експертної системи управління складом

№	Вимога	Опис
1	Реєстрація надходження товару	Фіксація отриманих партій та внесення їх у базу даних.
2	Реєстрація відвантаження товару	Списання товару, формування документа та оновлення залишків.
3	Облік залишків і руху	Контроль фактичних запасів, переміщень та доступності товарів.
4	Ведення довідника товарів	Створення та редагування номенклатурних позицій.
5	Управління користувачами та ролями	Розмежування прав доступу та адміністрування облікових записів.
6	Проведення інвентаризації	Виявлення розбіжностей між обліковими та фактичними залишками.
7	Формування звітів	Створення аналітичних, інвентаризаційних та логістичних звітів.
8	Аналітичний модуль	Розрахунок показників, аналіз динаміки та прогнозування потреб складу.
9	Експертний модуль	Побудова рекомендацій щодо оптимального управління запасами та виявлення відхилень.

Нефункціональні вимоги наведено в таблиці 1.4 і визначають якісні характеристики системи під час інтенсивної експлуатації.

Таблиця 1.4

Нефункціональні вимоги експертної системи

№	Вимога	Опис
1	Продуктивність	Час відгуку інтерфейсу не більше 200 мс.
2	Масштабованість	Здатність обробляти зростання обсягу даних без втрати продуктивності.
3	Надійність	Гарантований рівень доступності не нижче 99,9%.
4	Зручність використання	Інтуїтивний інтерфейс з контекстними підказками експертної підсистеми.
5	Сумісність	Підтримка інтеграції з ERP/CRM платформами.
6	Аналітична точність	Похибка аналітичних розрахунків не більше 1%.
7	Портативність	Підтримка розгортання на Windows та Linux.

Вимоги до безпеки системи, наведені в таблиці 1.5, спрямовані на захист складських даних і контроль дій користувачів.

Таблиця 1.5

Вимоги до безпеки експертної системи

№	Вимога	Опис
1	Аутентифікація	Авторизація користувачів з використанням надійних паролів або двофакторного підтвердження.
2	Авторизація	Розмежування доступу відповідно до ролей користувачів.
3	Шифрування	Використання TLS 1.3 для передачі даних та шифрування конфіденційної інформації.
4	Журналювання	Збереження всіх дій користувачів для подальшого аудиту.
5	Антибрютфорс	Автоматичне блокування при багаторазових невдалих спробах авторизації.
6	Резервне копіювання	Регулярне створення резервних копій бази даних і журналів.

Сформовані вимоги відображають комплексну модель функціонування експертної системи, яка поєднує операційний складський облік із механізмами інтелектуального аналізу та експертного висновку. Вимоги забезпечують основу для побудови архітектури, модулів логічного висновку та підсистеми аналітики, що визначатиме наукову новизну розробленої системи.

1.6 Постановка завдання

Постановка завдання для розроблення експертної системи управління складом підприємства ґрунтується на необхідності забезпечення автоматизованого обліку товарних операцій, підтримки актуальних складських залишків, формування управлінських рішень та підвищення достовірності аналітичних показників. Основною метою є створення програмної системи, яка забезпечить інтегроване управління процесами надходження, відвантаження, інвентаризації, контролю запасів та аналітичної оцінки складських операцій на основі експертних правил і моделей. У межах завдання визначаються вхідні дані, до яких належать відомості про партії товарів, інформація про контрагентів, внутрішні замовлення, первинні документи складських операцій, параметри товарів, історія руху запасів та показники, необхідні для розрахунку аналітичних метрик. Дані надходять від користувачів через інтерфейс системи, з бази даних товарних залишків, з журналу операцій та з довідника номенклатури, а їхня коректність визначає точність подальшої експертної оцінки.

Вихідні дані системи включають оновлені записи про залишки товарів, результати інвентаризації, сформовані документи надходження та відвантаження, оперативні аналітичні показники, прогностичні значення для оптимізації запасів, а також експертні рекомендації щодо коригування складських процесів. Результати оброблення даних повинні відображатися у вигляді звітів, графічних аналітичних представлень, повідомлень про нестачі, відхилення або ризики, а також логів виконання операцій, що забезпечує можливість подальшого аудиту та інтеграції з іншими інформаційними системами підприємства.

У поставленому завданні особлива увага приділяється створенню моделі логічного висновку, яка забезпечує формування рекомендацій на основі правил, історичних даних та аналітичних параметрів. Завданням є розроблення архітектури, здатної підтримувати високу продуктивність, надійність і безпеку, а також алгоритмів, що гарантують своєчасне оновлення даних та стабільну

роботу експертного модуля. Сформульована постановка визначає вимоги до функціональної та технічної структури системи, які ляжуть в основу подальшого проектування, моделювання та реалізації програмного забезпечення.

1.7 Висновки до першого розділу

У першому розділі здійснено комплексний аналіз предметної області управління складською логістикою, визначено структуру основних інформаційних потоків і встановлено роль кожного з функціональних компонентів майбутньої експертної системи. Розглянуто теоретико-методологічні засади, що ґрунтуються на сучасних наукових дослідженнях у галузі оптимізації запасів, аналітики логістичних процесів і побудови експертних моделей, а також окреслено ключові закономірності, підтверджені математичними залежностями та емпіричними даними.

Проведений аналіз існуючих рішень показав, що промислові платформи забезпечують широкий спектр функцій, однак не охоплюють повною мірою інтелектуальний компонент, необхідний для формування адаптивних управлінських рішень на рівні операційного складу. У моделюванні предметної області сформовано логічну структуру функціональних сценаріїв, послідовностей та діяльнісних процесів, що дозволяє формалізувати взаємодію користувачів із системою та визначити критичні точки впливу на точність обліку. У межах аналізу вимог узагальнено функціональні, нефункціональні та безпекові характеристики системи, що визначають рамкові обмеження та критерії якості для подальшої побудови архітектури.

Сукупність виконаних досліджень забезпечує наукове обґрунтування доцільності створення експертної системи управління складом і формує методичну основу для розроблення моделі логічного висновку, структури даних і програмних компонентів, які будуть розглянуті у наступному розділі.

2 МОДЕЛЮВАННЯ ТА АРХІТЕКТУРНЕ ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних експертної системи управління складом підприємства побудована на основі результатів системного аналізу та моделювання предметної області й відображає узагальнену структуру сутностей, їх ключових атрибутів і зв’язків, необхідних для підтримки основних бізнес-процесів складу. Основною вимогою до побудови моделі було забезпечення такої організації даних, яка з одного боку мінімізує дублювання інформації, а з іншого – дозволяє швидко виконувати типові операції надходження, відвантаження, інвентаризації та формування експертних висновків.

На рис. 2.1 подано ER-діаграму логічної моделі даних системи, де відображено базові сутності Product, Warehouse, StockItem, Counterparty, Document, DocumentLine та User, що формують каркас інформаційної інфраструктури.

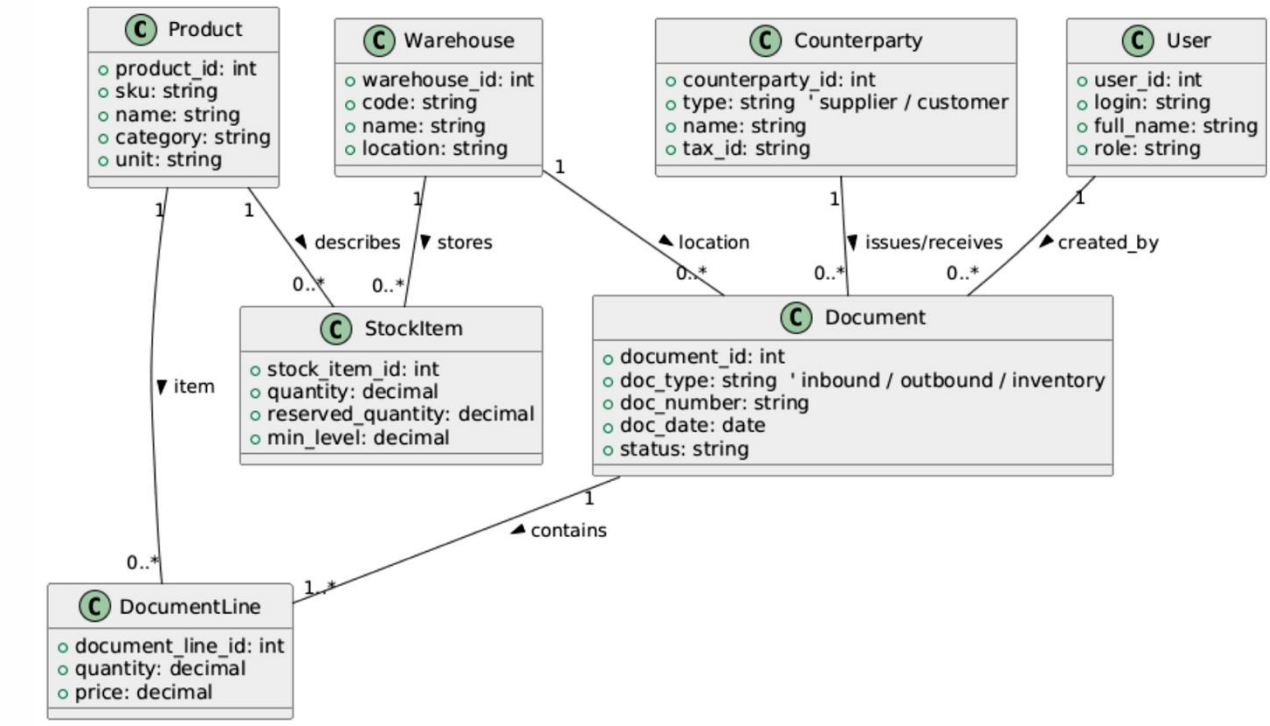

Рис. 2.1. ER-діаграма логічної моделі даних експертної системи управління складом.

В основі побудови моделі даних лежать принципи нормалізації до третьої нормальної форми, що дало змогу відокремити довідникові сутності (товар, контрагент, склад, користувач) від операційних (запаси, документи, рядки документів) і таким чином уникнути аномалій вставки, видалення та оновлення. Наприклад, інформація про товар зберігається лише в сутності Product, а фактичний стан запасів для кожного складу моделюється сутністю StockItem, що дозволяє гнучко змінювати параметри зберігання й незалежно масштабувати облік для декількох складів. Аналогічно, структура Document і DocumentLine забезпечує чітке розділення заголовків документів та їх позицій, що спрощує реалізацію різних типів операцій (надходження, відвантаження, інвентаризація) та підвищує узагальнюваність експертних правил, які працюють поверх єдиної схеми.

У таблиці 2.1 наведено узагальнений опис ключових сутностей логічної моделі даних.

Таблиця 2.1

Основні сутності логічної моделі даних експертної системи

Сутність	Ключові атрибути	Узагальнений призначення
Product	product_id, sku, name, category, unit	Зберігання довідникової інформації про товар
Warehouse	warehouse_id, code, name, location	Опис фізичних складів і зон зберігання
StockItem	stock_item_id, quantity, reserved_quantity, min_level	Відображення фактичних та резервованих залишків
Counterparty	counterparty_id, type, name, tax_id	Зберігання відомостей про постачальників і клієнтів
Document	document_id, doc_type, doc_number, doc_date, status	Опис заголовків складських документів
DocumentLine	document_line_id, quantity, price	Фіксація товарних позицій у складі конкретного документа
User	user_id, login, full_name, role	Представлення користувачів системи та їх ролей

Такий підхід до логічного моделювання забезпечує прозору відповідність між процесами, описаними у діаграмах прецедентів, послідовностей і діяльності,

та структурою бази даних, на якій базується експертний модуль. Нормалізована ER-модель спрощує реалізацію механізмів логічного висновку, оскільки правила можуть оперувати стандартизованими сутностями, не враховуючи зайві дублікати або неоднозначні зв’язки. Крім того, чітке розділення операційних та довідникових даних створює передумови для масштабування системи, інтеграції з зовнішніми ERP-рішеннями та підключення додаткових аналітичних модулів без зміни базової структури. У результаті логічна модель даних виступає центральною основою проєктування, яка задає єдині принципи зберігання інформації та визначає рамки подальшої фізичної реалізації бази даних і механізмів експертного аналізу.

2.2 Діаграма класів та кооперації

Об’єктно орієнтована структура програмної системи управління складом підприємства формалізована у вигляді діаграми класів, що відображає основні доменні сутності, сервіси бізнес-логіки та інфраструктурні компоненти. На рис. 2.2 подано діаграму класів, у якій виділено класи User і Role для реалізації механізмів автентифікації та авторизації, Product і StockItem для моделювання номенклатури та фактичних залишків, Supplier і Customer для опису контрагентів, а також сервіси InventoryService, ReportingService і інкапсулятор доступу до бази даних DbContext. Така декомпозиція відповідає вимогам нормалізованої логічної моделі даних: довідникові сутності зберігають тільки статичні атрибути, тоді як сервіси реалізують операційну логіку та не дублюють стан, що мінімізує зв’язаність між класами і полегшує супровід системи.

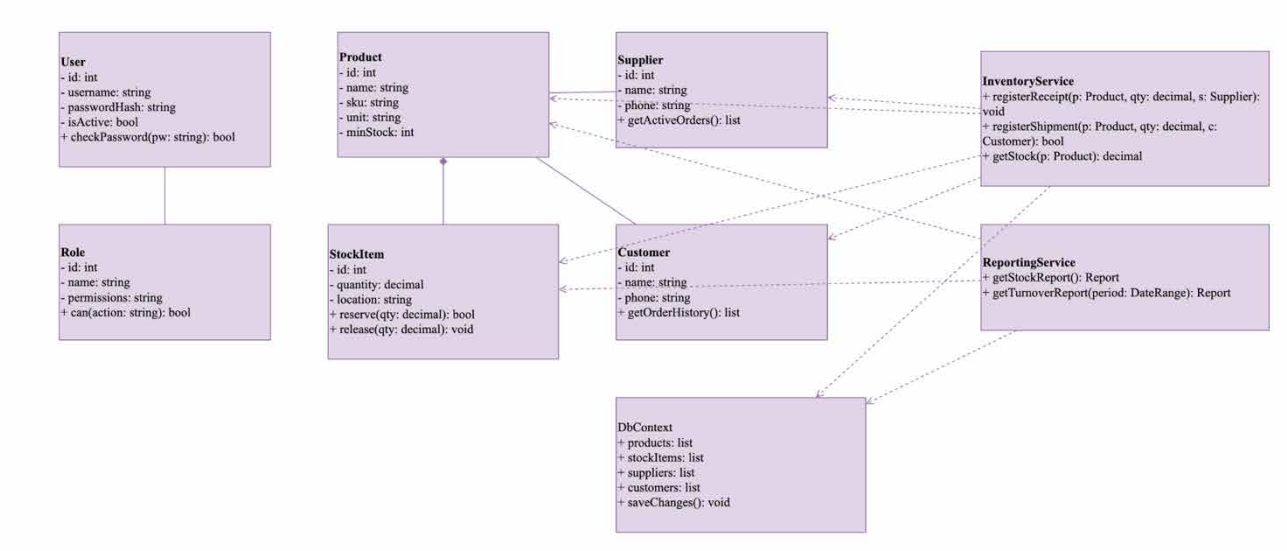

Рис. 2.2. Діаграма класів програмної системи управління складом підприємства.

Динамічна поведінка системи описується діаграмами кооперації, які відображають послідовність викликів між класами в типових бізнес-процесах. На рис. 2.3 наведено кооперацію для сценарію реєстрації надходження товару від постачальника, де адміністратор складу через користувацький інтерфейс ініціює виклик методу createReceipt у InventoryService, а той, у взаємодії з класами Supplier, StockItem та DbContext, створює або оновлює об’єкти, що відповідають постачальнику й складським позиціям, і зберігає зміни в базі даних. Така структура дозволяє чітко розділити відповідальність: інтерфейс працює лише з DTO-даними, InventoryService інкапсулює бізнес-правила, а DbContext забезпечує транзакційну цілісність.

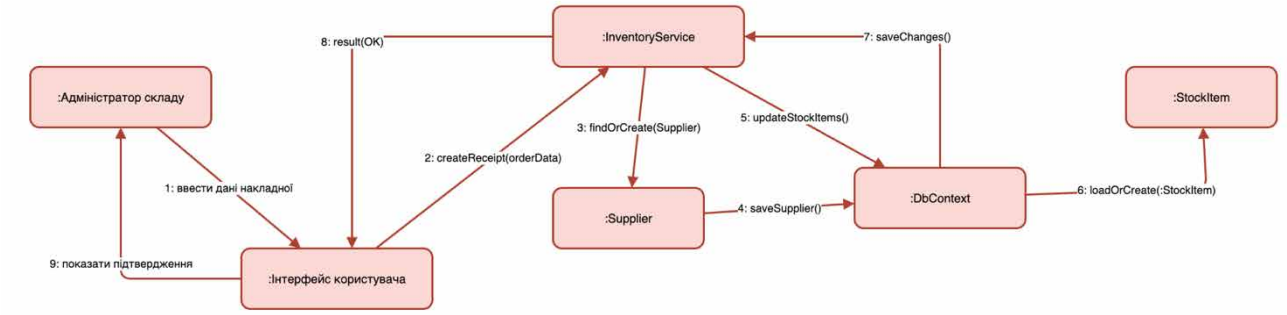

Рис. 2.3. Діаграма кооперації процесу реєстрації надходження товару.

Сценарій виконання клієнтського замовлення представлено на рис. 2.4, де менеджер з продажу, використовуючи UI замовлення, викликає метод registerShipment у InventoryService; далі сервіс за участю класів Customer,

StockItem і DbContext перевіряє доступність потрібної кількості товару, резервує або списує залишки та фіксує результат у сховищі. Важливо, що перевірка умов виконання замовлення зосереджена в логіці StockItem і InventoryService, що забезпечує повторне використання алгоритмів контролю запасів у різних точках системи та підтримує узгодженість із нормалізованою моделлю даних.

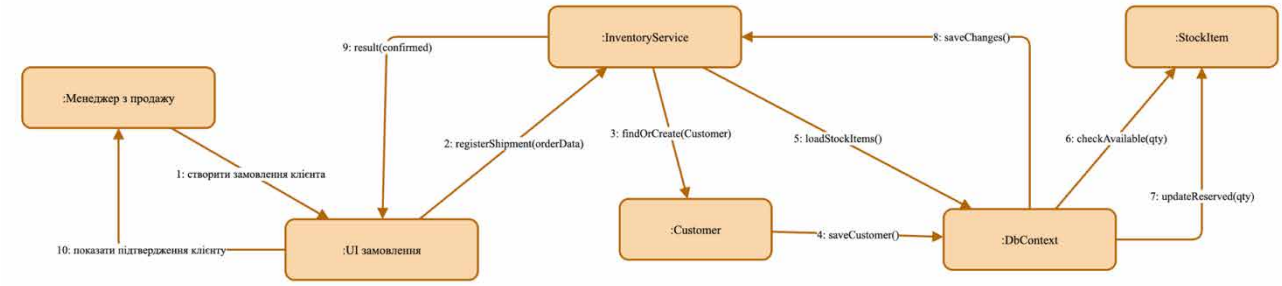

Рис. 2.4. Діаграма кооперації процесу реєстрації відвантаження товару клієнтові.

Аналітична складова системи реалізована в окремому сервісі звітності, що мінімізує навантаження на транзакційні модулі. На рис. 2.5 наведено кооперацію для формування звіту про залишки, у межах якої адміністратор складу через UI звітності звертається до ReportingService; далі сервіс за допомогою DbContext отримує агреговані дані StockItem, формує об’єкт Report та повертає результат користувачу. Відокремлення ReportingService від InventoryService дає змогу незалежно масштабувати аналітичні запити, оптимізувати їх на рівні бази даних та впроваджувати нові типи звітів без зміни основної бізнес-логіки.

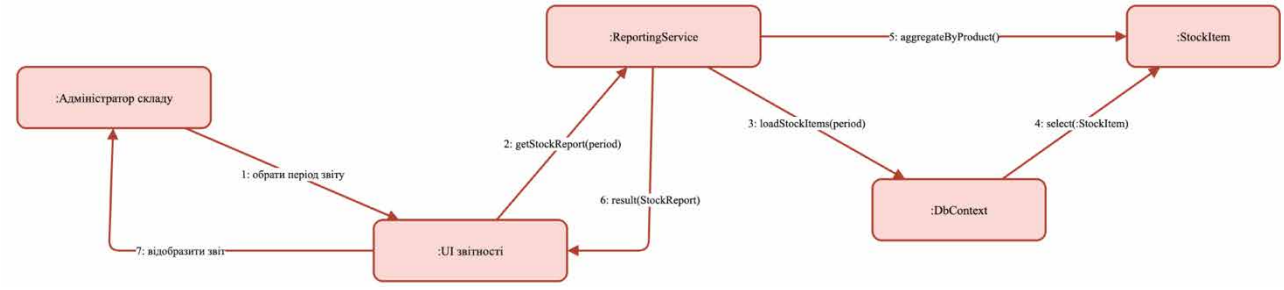

Рис. 2.5. Діаграма кооперації процесу формування звіту про стан запасів.

У таблиці 2.2 узагальнено ключові класи предметної області та їх основні відповідальності, що демонструє, як вимоги до функціональності й безпеки, сформульовані в першому розділі, трансформуються в об’єктно орієнтовану структуру коду.

Таблиця 2.2

Ключові класи та їх відповідальності

Сутність	Ключові атрибути	Узагальнений опис призначення
User	id, username, passwordHash, isActive	Представлення користувачів системи та механізмів автентифікації
Role	id, name, permissions	Визначення ролей та пов’язаних з ними прав доступу
Product	id, name, sku, unit, minStock	Зберігання довідникової інформації про товар та його мінімальні залишки
StockItem	id, quantity, location, reserved	Відображення поточних залишків товару на складі та резервів
Supplier	id, name, phone	Представлення постачальників для операцій прибуття
Customer	id, name, phone	Представлення клієнтів для операцій відвантаження
DbContext	products, stockItems, suppliers, customers	Центральний контекст доступу до даних та виконання транзакцій
InventoryService	registerReceipt, registerShipment, getStock	Логіка обробки надходжень, відвантажень та оновлення залишків
ReportingService	getStockReport, getTurnoverReport	Генерація агрегованих звітів щодо залишків і товарообігу

Такий підхід до проєктування класів і кооперацій забезпечує тісний зв’язок між логічною моделлю даних, вимогами до системи та об’єктно орієнтованою реалізацією. Нормалізація доменної моделі на рівні класів, розподіл відповідальностей між сервісами та сутностями, а також виокремлення інфраструктурного шару доступу до даних дозволяють побудувати гнучку й розширювану архітектуру. Це створює підґрунтя для подальшої реалізації експертних алгоритмів, впровадження додаткових правил і звітів без руйнування існуючої структури та забезпечує відповідність програмної системи вимогам, сформованим у попередньому розділі.

2.3 Діаграма компонентів

У межах проєктування програмної системи управління складом важливим етапом є формування структурного подання архітектури у вигляді діаграми компонентів, яка демонструє розподіл функціональності між модулями, їхні зовнішні інтерфейси та взаємодію із підсистемами зберігання даних і зовнішніми сервісами. На рис. 2.6 подано компонентну модель побудови системи, яка відображає логічний поділ застосунку на модулі авторизації та ролей, API-контролери, модуль управління запасами, модуль обробки замовлень та аналітичний модуль, а також їх взаємозв'язки з клієнтським інтерфейсом, СУБД та зовнішніми ERP-сервісами.

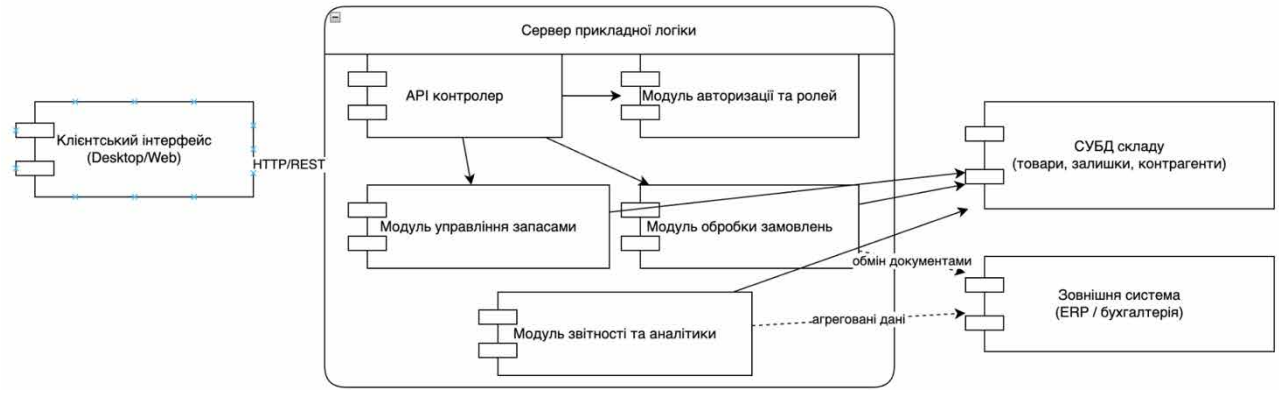

Рис. 2.6. Діаграма компонентів програмної системи управління складом

З метою підвищення структурованості архітектури було виконано декомпозицію системи на компоненти відповідно до принципів слабкого зв'язку та високої внутрішньої когезії, що забезпечує незалежність модулів і спрощує масштабування. Таблиця 2.3 містить узагальнений опис функціонального призначення основних компонентів системи та їхніх ключових відповідальностей.

Таблиця 2.3

Характеристика компонентів системи

Компонент	Функціональне призначення
Клієнтський інтерфейс (Desktop/Web)	Забезпечення взаємодії користувачів із системою через REST-запити
API контролер	Приймання запитів від клієнтів та маршрутизація їх до відповідних модулів бізнес-логіки

Продовження таблиці 2.3

Модуль авторизації та ролей	Перевірка автентичності користувачів і застосування правил доступу
Модуль управління запасами	Реалізація операцій над залишками та взаємодія з базою даних складу
Модуль обробки замовлень	Реєстрація, перевірка та обробка вхідних і вихідних документів
Модуль звітності та аналітики	Генерація агрегованих аналітичних даних, формування звітів
СУБД складу	Зберігання інформації про товарні позиції, рухи запасів і контрагентів
Зовнішня система (ERP/бухгалтерія)	Обмін документами та синхронізація даних між складською системою та підприємством

Загальна архітектура забезпечує чітке відокремлення прикладної логіки від інфраструктури, підтримує розширення функціональності та дозволяє інтегрувати нові сервіси без суттєвих змін у внутрішніх механізмах системи. Декомпозиція на ізольовані компоненти спрощує модифікацію модулів, дає змогу впроваджувати паралельну обробку запитів і забезпечує високу продуктивність та надійність системи під час роботи зі складськими даними.

2.4 Діаграма пакетів

Архітектурна декомпозиція програмної системи управління складом передбачає чіткий поділ компонентів за функціональними областями, що створює логічно ізольовані пакети з визначеними відповідальностями. Такий підхід дає змогу реалізувати модульність, забезпечити контрольовану взаємодію між частинами системи та підтримувати високий рівень масштабованості та супроводжуваності. На рис. 2.7 наведено діаграму пакетів, яка відображає структуру модулів системи та їх взаємозв’язки.

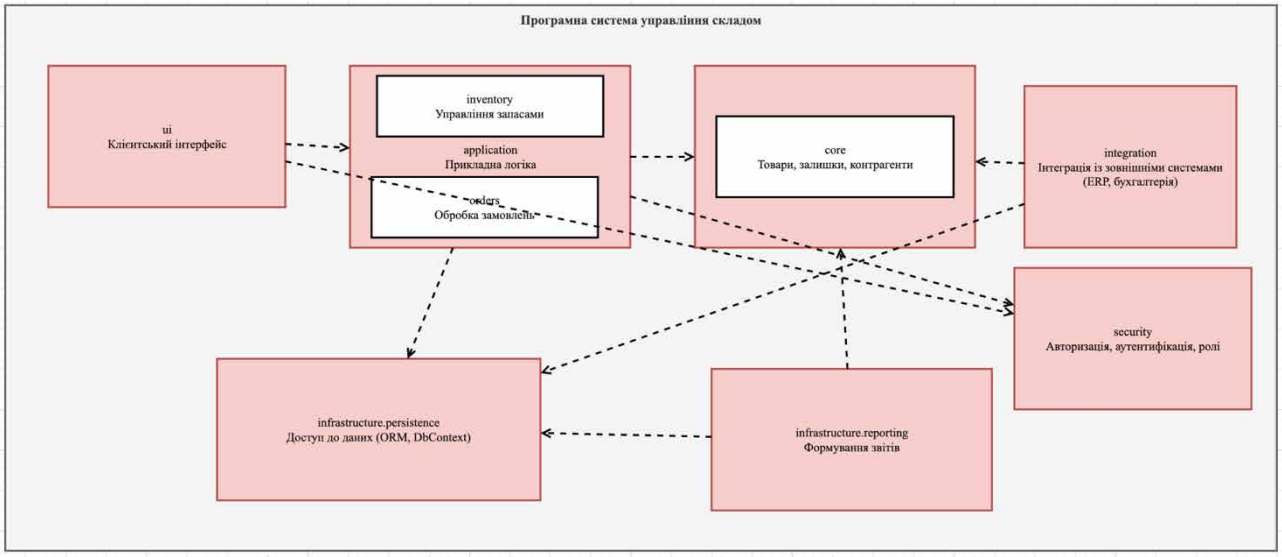

Рис. 2.7. Діаграма пакетів програмної системи управління складом

Структура пакетів побудована відповідно до принципів доменної модульності, коли окремі частини коду відповідають конкретним бізнес-процесам: управління запасами, обробка замовлень, робота з довідниковими сутностями, інтеграція із зовнішніми платформами та забезпечення безпеки. Такий поділ гарантує незалежність модулів та мінімізує вплив змін у одному пакеті на функціонування інших. Таблиця 2.5 містить стислий опис призначення кожного пакета.

Таблиця 2.5

Характеристика пакетів системи

Пакет	Призначення
ui	Забезпечення клієнтського інтерфейсу та взаємодії користувачів із системою
application	Реалізація прикладної логіки та оркестрація процесів
inventory	Ведення залишків, операції над складськими позиціями
orders	Обробка вхідних і вихідних замовлень
core	Управління довідниками товарів, контрагентів, залишків
integration	Взаємодія з зовнішніми системами (ERP, бухгалтерія)
security	Автентифікація, авторизація користувачів, контроль доступу
infrastructure.persistence	Доступ до даних, ORM, реалізація DbContext
infrastructure.reporting	Генерація звітів та агрегованих даних

Запропонована пакетна організація формує стійку архітектурну основу системи. Така структура не лише спрощує розробку і тестування окремих модулів, але й підвищує надійність та продуктивність у роботі з великою кількістю складських даних. Застосування принципів модульності й розділення відповідальностей забезпечує можливість ефективного масштабування системи і розширення її функціональності без порушення існуючої архітектури.

2.5 Висновки до другого розділу

У другому розділі було здійснено комплексне проєктування програмної системи управління складом підприємства, що охопило логічне моделювання даних, структурний аналіз програмних компонентів та декомпозицію функціональних модулів. Побудована ER-діаграма відобразила ключові сутності предметної області, їх атрибутивний склад та семантичні зв’язки, що забезпечує нормалізовану та несуперечливу структуру даних, придатну для подальшої реалізації у реляційній СУБД. Використання принципів третьої нормальної форми дало змогу усунути дублювання, забезпечити цілісність інформації та підвищити загальну продуктивність системи при обробці операцій над залишками та документами.

Діаграма класів і кооперацій продемонструвала розподіл відповідальностей між об’єктами та їх роль у виконанні бізнес-процесів складської логістики. Сформована модель дозволила відобразити взаємодію сервісів управління запасами, обробки замовлень, звітності та доступу до даних, що створює технологічну основу для реалізації модульної архітектури та спрощує майбутнє тестування й розвиток функціоналу.

Діаграми компонентів і пакетів забезпечили уявлення про багаторівневу організацію системи та ізоляцію логічних областей, включно з клієнтським інтерфейсом, прикладною логікою, модулями інтеграції, безпекою та інфраструктурою доступу до даних. Такий підхід гарантує масштабованість,

гнучкість і контрольованість системи під час росту кількості операцій, користувачів і зовнішніх інтеграцій.

У результаті проведеного моделювання було сформовано повну й узгоджену архітектурно-логічну основу програмної системи управління складом, яка забезпечує її подальшу якісну реалізацію, підтримуваність і можливість розширення. Отримані моделі відображають закономірності предметної області та відповідають вимогам сучасних підходів до побудови інформаційних систем.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕХНОЛОГІЧНА ІНФРАСТРУКТУРА СИСТЕМИ

3.1 Вибір технологій та інструментальних засобів реалізації системи

Розроблення програмної системи управління складом вимагає узгодженого добору технологій, які забезпечують стабільність, продуктивність та розширюваність рішення, водночас залишаючись достатньо легковагими для швидкого розгортання та подальшої підтримки. З огляду на характер бізнес-процесів (оперативний облік товарів, обробка замовлень, аналітика залишків, формування документів), а також вимоги до локального або гібридного використання, архітектурою обрано настільний застосунок із використанням Python як основної платформи.

Python забезпечує високу швидкість розробки, багату екосистему бібліотек і прозору інтеграцію з форматами даних, необхідними для обліку, аналітики та експорту звітності. Завдяки PyQt6 створюється кросплатформений графічний інтерфейс із можливістю відображення таблиць, форм, діалогів обробки замовлень та модулів інвентаризації. SQLite обрано як основну локальну СУБД, забезпечуючи транзакційність, відсутність потреби в окремому сервері та достатню продуктивність для середніх обсягів складських операцій.

Для реалізації бізнес-логіки використовується багатошарова архітектура: рівень доменних моделей забезпечує коректне відображення сутностей складського процесу; сервісний рівень реалізує обробку операцій над товарами, замовленнями та звітністю; модулі доступу до даних працюють через ORM-шар, що мінімізує ризики помилок у запитах та забезпечує цілісність даних. Формування звітів і аналітичних вибірок реалізується через Pandas із можливістю додаткової візуалізації за допомогою Matplotlib.

Добір технологій ураховує вимоги до безпеки: рольова модель доступу реалізується у Python-модулі безпеки із застосуванням хешування паролів (bcrypt) та модулем контролю прав доступу до операцій (RBAC). Це забезпечує

ізольований доступ до інвентаризації, реєстрації приходу/відвантаження та перегляду аналітики.

Узагальнений перелік використаних технологій наведено в таблиці 3.1.

Таблиця 3.1

Обрані технології та інструменти реалізації системи

Компонент системи	Обрана технологія / інструмент	Обґрунтування використання
Графічний інтерфейс користувача	PyQt6	Кросплатформеність, підтримка складних UI-компонентів, висока швидкість розробки.
Мова програмування	Python 3.x	Легкість розробки, велика кількість бібліотек для обліку, аналітики та обробки даних.
Система керування базами даних	SQLite3	Легка, вбудована СУБД, що не потребує сервера; достатня продуктивність для складських операцій.
Доступ до даних / ORM	SQLAlchemy	Спрощення роботи з БД, автоматизація CRUD-операцій, захист від помилок запитів.
Модулі безпеки	bcrypt, Python RBAC-модуль	Хешування паролів, реалізація ролевої моделі доступу.
Аналітика та формування звітів	Pandas, Matplotlib	Побудова статистичних зрізів, аналітичних графіків, формування звітних таблиць.
Формат обміну документами	PDF (reportlab), CSV	Забезпечення експорту документів для бухгалтерських та логістичних процесів.

У результаті сформовано технологічний стек, що оптимально відповідає вимогам системи управління складом: він забезпечує поєднання продуктивності, зручності супроводу, гнучкості та мінімальних інфраструктурних витрат. Обрані інструменти дозволяють реалізувати надійну, масштабовану та легко модифіковану систему, готову до інтеграції з зовнішніми обліковими або ERP-рішеннями в майбутньому.

3.2 Архітектура системи, проєктування функціоналу результатів дослідження

Архітектура програмної системи управління складом формується як трирівнева багатокомпонентна структура, орієнтована на забезпечення надійного управління товарними запасами, операціями руху, інвентаризацією та автоматизованим формуванням аналітичних звітів. В основу моделі покладаються принципи сервісно-орієнтованої взаємодії, логічного поділу відповідальностей, транзакційної цілісності та розширюваності, що дозволяє гнучко масштабувати функціонал системи залежно від інтенсивності складських операцій. На рисунку 3.2 подано узагальнену багаторівневу архітектуру, що включає рівень представлення, рівень прикладної логіки та рівень даних, а також механізми взаємодії із зовнішніми ERP-системами та сервісами нотифікацій.

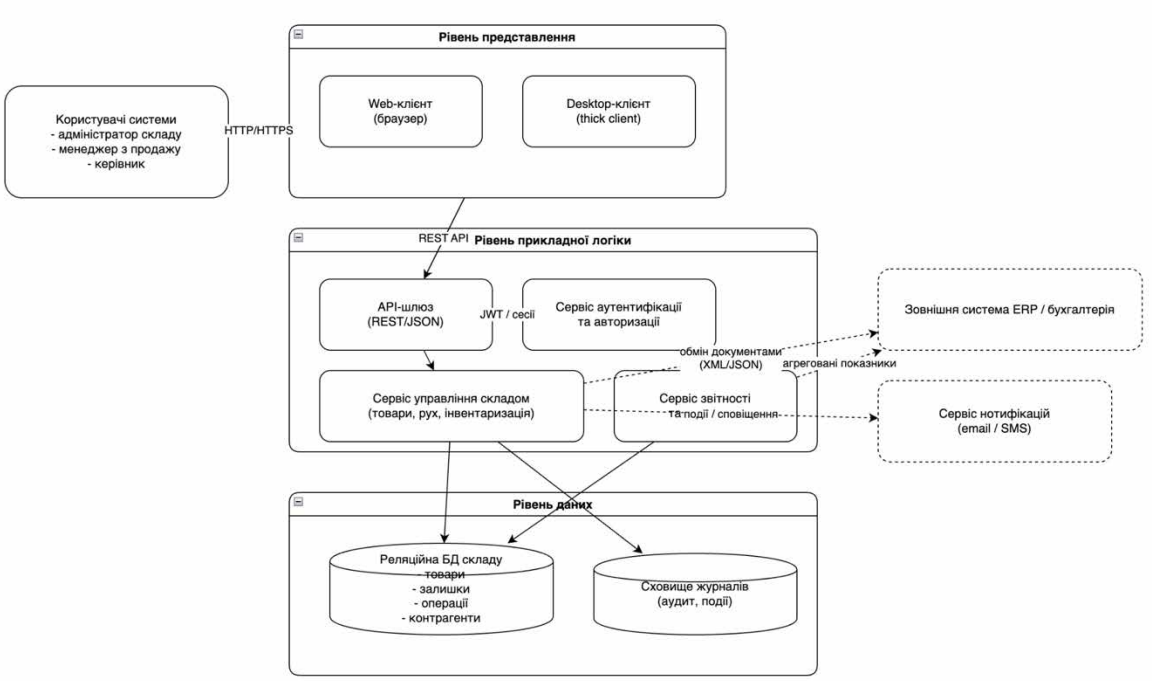

Рис. 3.2. Багаторівнева архітектурна модель системи управління складом.

До складу рівня представлення входять веб-клієнт та десктоп-клієнт, які забезпечують взаємодію користувача із функціональними модулями системи через HTTP/HTTPS. Цей рівень є повністю ізольованим від логіки оброблення даних, що гарантує зниження ризику помилок доступу, а також спрощує адаптацію інтерфейсів під різні сценарії взаємодії адміністратора складу,

менеджера з продажу та керівника. На рівні прикладної логіки реалізовано REST-орієнтований API-шлюз, сервіс авторизації й автентифікації на основі JWT-механізму, модуль управління складом, модуль інвентаризації та сервіс аналітичної звітності. Кожен модуль функціонує із чітким визначенням меж відповідальності, що забезпечує формальну декомпозицію системи та захищений доступ до даних. Рівень даних сформований реляційною базою SQLite, оптимізованою під локальні та серверні сценарії розгортання, а також окремим сховищем журналів подій для аудиту та відстеження критичних операцій.

На рисунку 3.3 наведено деталізовану компонентну архітектуру сервера прикладної логіки, яка реалізує взаємодію між клієнтськими застосунками, базою даних складу, сервісами журналювання та зовнішніми ERP-рішеннями через REST/JSON-протокол.

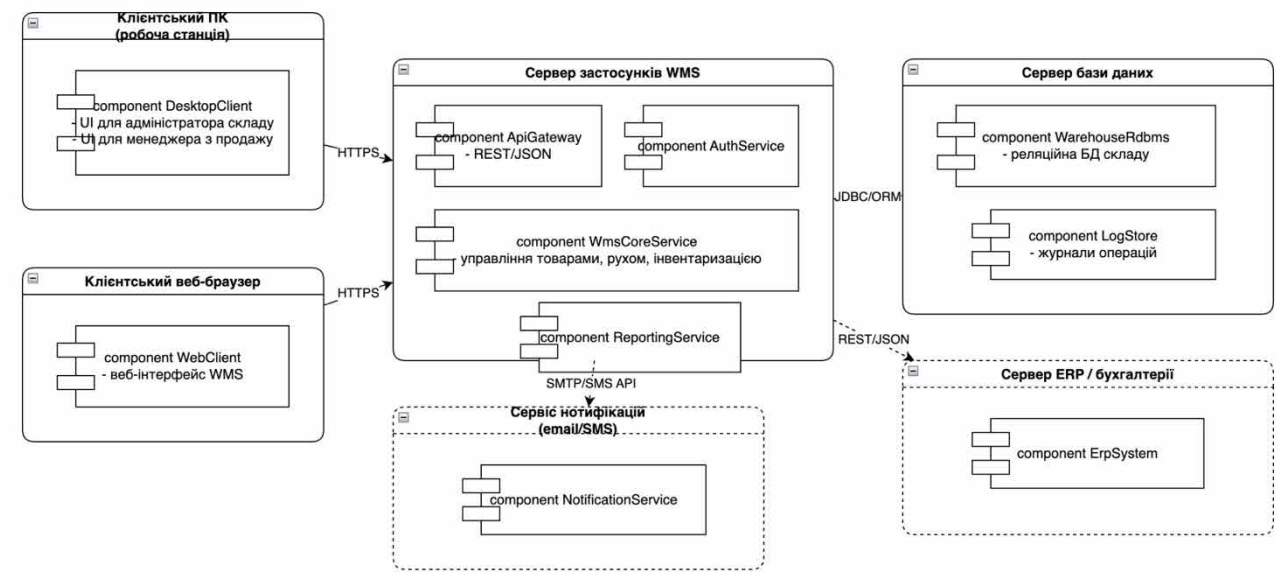

Рис. 3.3. Компонентна архітектура сервера прикладної логіки WMS-системи.

Зазначена архітектурна модель дозволяє формувати інтегрований потік оброблення операцій: від введення складського документа до перевірки доступності запасів, виконання транзакцій над залишками, збереження результатів у базі даних та передачі агрегованих даних до модулів звітності. Така схема мінімізує затримки при обробці запитів, забезпечує атомарність змін і дозволяє дотримуватися ACID-властивостей при роботі з товарними залишками.

Усі критичні функції, включно з інвентаризацією, резервуванням і обробленням відвантажень, проходять через централізований сервіс управління складом, що усуває ризики розсинхронізації та підвищує цілісність даних.

Для підтримки інтегрованої взаємодії з ERP-платформами реалізовано модуль зовнішніх інтеграцій, який виконує експорт необхідних бухгалтерських документів та агрегованих аналітичних показників. Завдяки цьому забезпечується відповідність системи сучасним вимогам до цифрової логістики, підвищується оперативність управлінських рішень та зменшується ймовірність виникнення помилок у первинних документах. Сервіс нотифікацій додатково забезпечує автоматичне інформування про критичні події, що покращує контроль за процесами складу.

У таблиці 3.2 наведено формальне відображення основних компонентів архітектури з визначенням їх функціонального призначення та ролі у забезпеченні працездатності системи.

Таблиця 3.2

Компоненти архітектури системи та їх призначення

Компонент	Призначення
WebClient / DesktopClient	Забезпечення взаємодії користувачів із системою через графічний інтерфейс.
ApiGateway	Прийом та маршрутизація REST-запитів, узгодження протоколів взаємодії.
AuthService	Автентифікація користувачів, перевірка ролей, формування JWT-токенів.
WmsCoreService	Оброблення складських операцій, управління товарами, рухом, інвентаризацією.
ReportingService	Формування звітів, аналітичні розрахунки, агрегування показників.
WarehouseRdbms	Зберігання товарів, залишків, контрагентів та операцій у реляційній БД.
LogStore	Фіксація журналів аудиту, історії операцій та подій системи.
ErpSystem	Обмін документами та аналітичними даними із зовнішньою ERP/бухгалтерією.
NotificationService	Надсилання email/SMS-сповіщень про критичні операції та зміни.

Результуючий текст формує завершене уявлення про архітектурний стиль, технологічну організацію та функціональну повноту запропонованої системи. Запропонована модель демонструє відповідність принципам модульності, ізоляції бізнес-логіки, масштабованості та інформаційної безпеки, що дозволяє застосовувати розроблену програмну систему як у середовищі малого підприємства, так і в умовах багатоскладських логістичних структур. Така архітектура забезпечує збалансоване поєднання продуктивності, надійності та можливості розширення функціоналу відповідно до зростання обсягів операцій або інтеграційних вимог.

3.3 Інформаційна база системи

Інформаційна база програмної системи управління складом підприємства формується як ієрархічно організований комплекс оперативних та аналітичних даних, що забезпечує підтримку повного циклу операцій – від реєстрації надходжень і відвантажень до виявлення аномалій та прийняття управлінських рішень. На рис. 3.3 подано логічну структуру сховища даних у форматі «зірки», де фактна таблиця inventory_fact акумулює узагальнену інформацію про рух товарів по секціях складу в розрізі дат та категорій, а вимірні таблиці warehouse_dim, section_dim, category_dim, date_dim описують просторово-організаційний і часовий контекст запасів. Така організація даних мінімізує дублювання атрибутів, забезпечує третю нормальну форму для вимірів і оптимізує виконання OLAP-запитів до фактної таблиці, що критично важливо для швидкого формування звітів по оборотності та результатах інвентаризації.

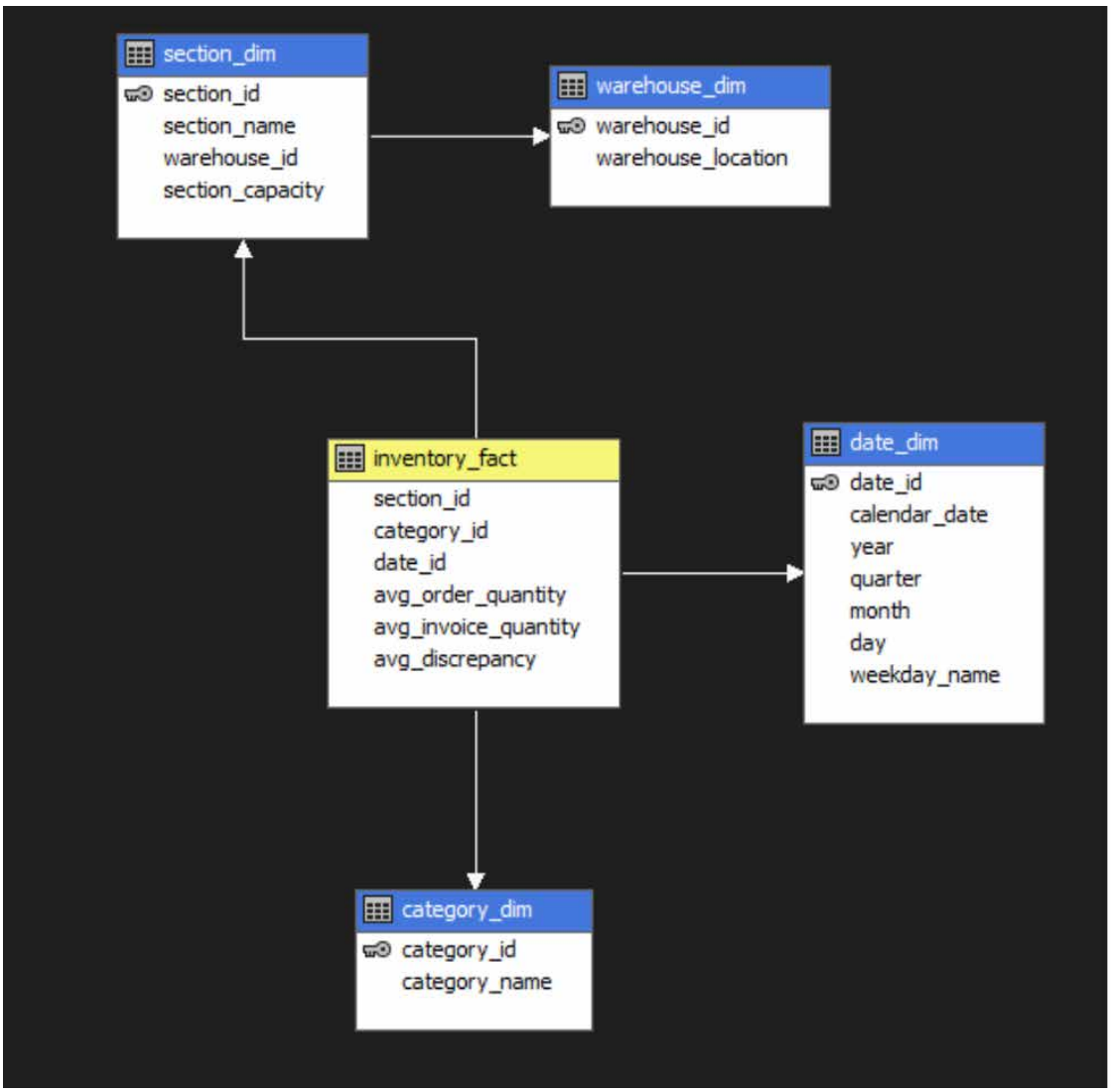

Рис. 3.3. Логічна структура сховища даних для аналітики складських запасів

Подальший шар інформаційної бази становлять аналітичні перетворення, які використовуються для кластеризації номенклатури за показниками середнього залишку та обороту. На рис. 3.4 проілюстровано результат кластеризації товарів за оборотністю, де кожен кластер відображає групу позицій з подібними профілями попиту (високий, середній та низький оборот). Така модель дозволяє виділяти товари A/B/C-класів не лише за статичним показником обороту, а й з урахуванням фактичної варіативності продажів, що підвищує точність планування поповнення запасів і підтримки сервісного рівня.

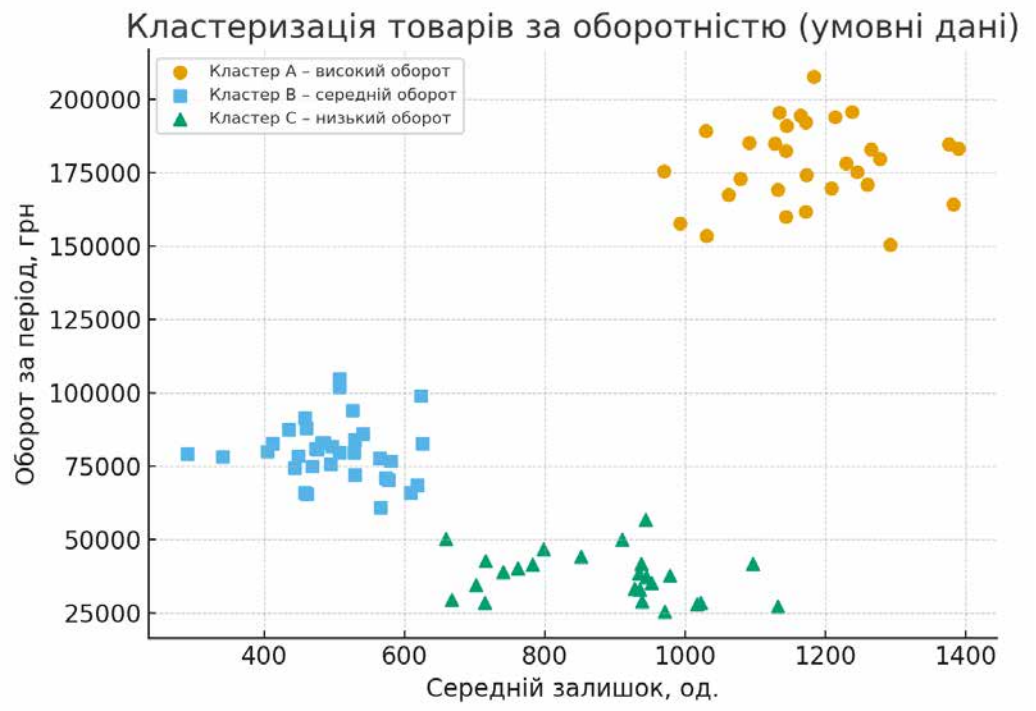

Рис. 3.4. Кластеризація товарів за оборотністю в інформаційній базі системи

Вибір оптимальної кількості кластерів здійснюється за методом ліктя, що відображено на рис. 3.5. На осі абсцис подано кількість кластерів K, а на осі ординат – суму квадратів відстаней точок до центрів кластерів (inertia). Характерне «коліно» кривої визначає компроміс між точністю класифікації та складністю моделі й інтегрується до інформаційної бази у вигляді параметрів налаштування аналітичного модуля. Це дає змогу системі автоматично пропонувати користувачеві раціональну кількість кластерів для різних груп номенклатури, зберігаючи обрані значення у метаданих аналітичних сценаріїв.

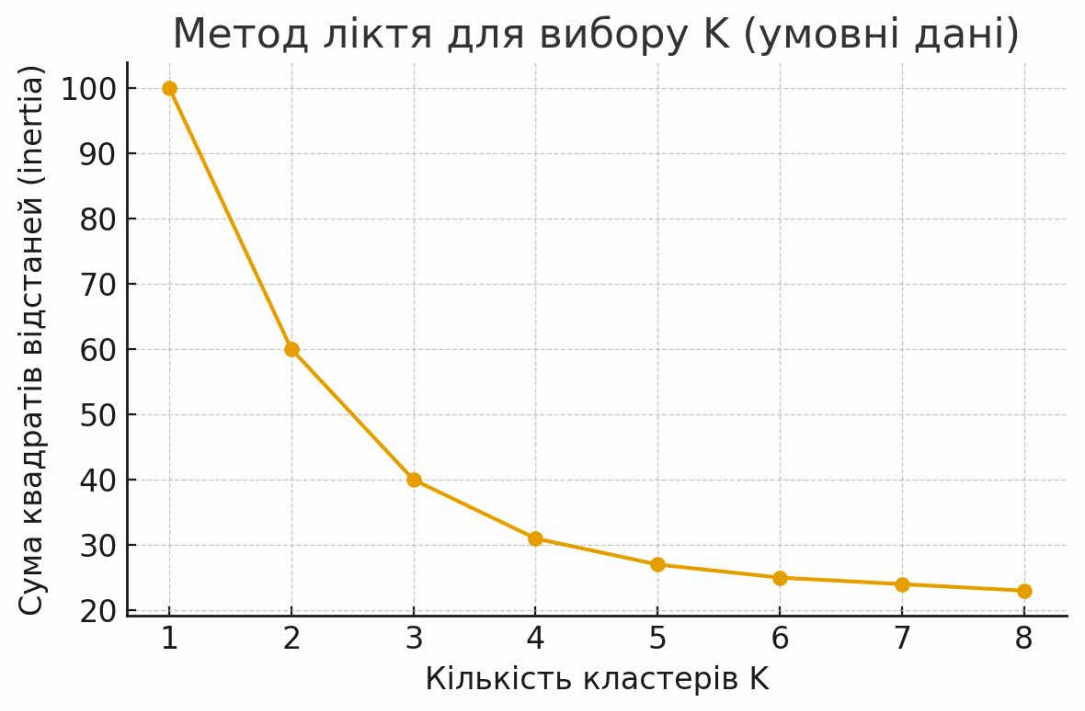

Рис. 3.5. Використання методу ліктя для визначення кількості кластерів K

Окремий пласт інформаційної бази становлять показники якості інвентаризації. На рис. 3.6 наведено приклад агрегованої візуалізації середньої різниці за результатами інвентаризації для різних секцій складу, де табличні дані по розбіжностях поєднуються з кільцевою діаграмою. У термінах інформаційної моделі це відповідає фактам інвентаризації, агрегованим по вимірах «Склад», «Рік» та «Секція», що дозволяє зберігати в сховищі не лише «сирі» рядки інвентаризаційних відомостей, а й результати їх багаторівневої агрегації для подальшого порівняльного аналізу.

Середня різниця в результаті інвентаризації за 2024 рік

Склад	Рік	Секція	Різниця
Central Warehouse	2024		
		Clothing	3.5
		Electronics	1.875
		Food	5.75
		Furniture	1.75

Рис. 3.6. Подання середніх розбіжностей інвентаризації в інформаційній базі

Для комплексної оцінки ефективності складської логістики інформаційна база містить множину нормованих KPI, які зручно інтерпретувати через радіальні діаграми. На рис. 3.7 показано приклад радіальної діаграми KPI складів, де осі відповідають показникам обороту, середнього залишку, кількості замовлень, середнього часу зберігання, дефіциту позицій і точності обліку. Джерелом даних для таких візуалізацій є фактні таблиці обороту, замовлень та інвентаризації, до яких застосовано нормувальні перетворення й збережено результат у вигляді додаткових агрегатів у сховищі. Це дозволяє керівнику складу в межах однієї панелі порівнювати різні майданчики та оперативно виявляти вузькі місця.

Рис. 3.7. Радіальна діаграма інтегральних KPI складської діяльності

Інтеграція зазначених компонентів інформаційної бази завершується побудовою єдиного аналітичного ядра, яке підтримує сценарії OLAP-аналізу, кластеризації та контролю якості даних.

Ключові елементи наукової новизни запропонованої інформаційної бази узагальнено в таблиці 3.3, де наведено порівняння традиційних WMS-рішень та розроблюваної системи з точки зору підтримуваних аналітичних сценаріїв та способів зберігання даних.

Таблиця 3.3

Порівняння традиційних підходів та запропонованої інформаційної бази системи управління складом

Аспект	Традиційні WMS-рішення	Запропонована система управління складом
Структура даних	Реляційна БД з операційними таблицями руху без виділеного сховища	Поєднання OLTP-БД та спеціалізованого сховища даних із фактною таблицею inventory_fact та вимірами складу, секцій, категорій і дат
Аналітика запасів	Статичні звіти за залишками та оборотом, обмежені фіксованим набором полів	Багатовимірний аналіз запасів із можливістю довільної агрегації та drill-down до рівня секцій і товарів
Методи інтелектуального аналізу	Відсутні або використовуються як зовнішні інструменти	Вбудована кластеризація номенклатури, метод ліктя для вибору параметрів, зберігання моделей у складі інформаційної бази
Облік якості інвентаризації	Зберігання лише актів інвентаризації без системного аналізу розбіжностей	Агреговані показники розбіжностей за складами, секціями та періодами з візуальною підтримкою прийняття рішень
KPI складської діяльності	Локальні показники (кількість замовлень, залишки) без єдиної моделі	Інтегрована система нормованих KPI, представлених у радіальних діаграмах на основі єдиного аналітичного ядра
Підтримка сценаріїв дослідження	Одноразові звіти, відсутність збереження конфігурацій аналізу	Зберігання параметрів аналітичних сценаріїв та результатів моделей у метаданих інформаційної бази

Узагальнюючи, інформаційна база розроблюваної програмної системи формує цілісний багатошаровий контур даних, що поєднує оперативний облік руху товарів з аналітичним сховищем, вбудованими методами кластеризації та формалізованою системою KPI. На відміну від типових WMS-рішень, де аналітика реалізується як зовнішні звіти, у запропонованій архітектурі

аналітичні моделі, результати їх роботи та параметри досліджень є невід'ємною частиною інформаційної бази. Це створює передумови для відтворюваного наукового експерименту над реальними складськими даними, дозволяє накопичувати знання про поведінку запасів у часі та забезпечує підвищення точності управлінських рішень щодо планування поповнення, оптимізації запасів та оцінювання ефективності складської логістики.

3.4 Висновки до третього розділу

У третьому розділі було здійснено комплексне технічне опрацювання архітектури, інформаційної бази та інструментальних засобів реалізації програмної системи управління складом, що дозволило сформувати цілісний технологічний фундамент для подальшої розробки та експериментальної перевірки її функціональних характеристик. Проведений аналіз технологій підтвердив доцільність використання стеку Python + PyQt6 для побудови клієнтського інтерфейсу, SQLite як вбудованої легковагової СУБД для локального зберігання даних та REST-сервісів для забезпечення інтеграційної взаємодії із зовнішніми ERP-системами, що відповідає вимогам портативності, автономності та низької залежності від інфраструктури.

Архітектурне проєктування, представлене у вигляді структурних та компонентних схем, сформувало трирівневу модель системи: рівень представлення, рівень прикладної логіки та рівень даних, яка забезпечує ізоляцію бізнес-процесів, гнучкість модифікацій і масштабованість у разі збільшення обсягів інформації чи кількості користувачів. Особлива увага приділена модульності: сервіси управління складськими операціями, обробки замовлень, аналітики та автентифікації функціонують як незалежні компоненти, що спрощує тестування, супровід та оновлення системи.

Формування інформаційної бази стало ключовим результатом розділу, оскільки запропонована багатовимірна модель даних поєднує оперативні таблиці руху товарів із аналітичним сховищем «зіркового» типу на основі фактної

таблиці inventory_fact та вимірних таблиць складів, секцій, категорій і дат. У межах цієї структури реалізовано можливість зберігання агрегованих показників обороту, інвентаризаційних розбіжностей, результатів кластеризації та нормованих KPI, що демонструє наукову новизну моделі і розширює функціональність системи порівняно з класичними WMS-рішеннями. Інтеграція статистичних і машинних методів аналізу (кластеризація товарів, метод ліктя, нормування KPI) у єдину інформаційну базу забезпечує підвищення точності управлінських рішень та можливість відтворюваності аналітичних сценаріїв.

Таким чином, у третьому розділі сформовано завершену архітектурно-аналітичну модель системи, яка узгоджує структуру даних, програмні компоненти, методи обробки інформації та інструменти реалізації. Отримані результати створюють методологічну та технологічну основу для переходу до практичної реалізації системи та її експериментального тестування в наступному розділі.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

План тестування програмних модулів та методика оцінювання Тестування програмної системи управління складом є ключовим етапом підтвердження коректності розробленої архітектури, функціональної логіки та надійності інформаційної бази. План тестування спрямований на комплексну перевірку роботи модулів управління запасами, обробки замовлень, аналітичного модуля та підсистеми автентифікації, що забезпечує повноту покриття всіх критично важливих бізнес-процесів. Методика оцінювання результатів передбачає поєднання модульних, інтеграційних та системних тестів, що дозволяє визначити якість реалізації функцій, стабільність взаємодії компонентів і відповідність поведінки системи вимогам, сформованим у розділі 1.

Текст посилання на таблицю: узагальнений план тестування модулів наведено у таблиці 4.1.

Таблиця 4.1

План тестування програмних модулів системи управління складом

Модуль	Мета тестування	Тип тестування	Критерії успішності
Модуль управління запасами	Перевірка коректності операцій приходу, витрати, інвентаризації	Модульне, інтеграційне	Точність змін залишків; відповідність даних у БД
Модуль обробки замовлень	Тестування формування документів, резервування та списання	Модульне, системне	Відсутність розбіжностей у кількостях; валідність документів
Модуль аналітики	Перевірка розрахунку KPI, кластеризації та агрегованих показників	Інтеграційне, системне	Стабільність алгоритмів; відповідність очікуваним обчисленням
Модуль автентифікації	Тестування механізмів входу, ролей, прав доступу	Модульне, системне	Коректне розмежування доступу; стійкість до несанкціонованих дій
Інтерфейс PyQt6	Оцінювання коректності відображення форм, таблиць та інтеракцій	Системне	Відсутність помилок UI; узгодженість дій і відображення даних

Запропонований план тестування забезпечує структурний підхід до оцінювання роботи системи відповідно до визначених функціональних та нефункціональних вимог. Стратегія тестування поєднує локальну перевірку логічних блоків, аналіз інтеграційних взаємодій між модулями та оцінку поведінки системи у реальних сценаріях обробки складських даних. Методика оцінювання результатів базується на верифікації коректності зміни станів даних у базі, аналізі точності розрахунків у модулі аналітики, перевірці цілісності документообігу та контролі доступу відповідно до ролей користувачів. Такий підхід гарантує виявлення не лише локальних помилок реалізації, а й потенційних конфліктів між компонентами, що є критично важливим для систем управління складськими процесами, де точність і надійність визначають ефективність операцій та достовірність управлінських рішень.

4.2 Тестування інтелектуальної системи управління складом

Тестування розробленої інтелектуальної системи управління складом спрямоване на підтвердження коректності обробки операцій руху товарів, достовірності розрахунків показників запасів, адекватності роботи OLAP-аналітики та стійкості алгоритмів кластеризації до зміни вхідних даних. На рис. 4.1 наведено результат тестування головного інтерфейсу оператора, який демонструє відображення агрегованих параметрів складу в режимі реального часу, зокрема загальний залишок, кількість критичних позицій та активні відвантаження.

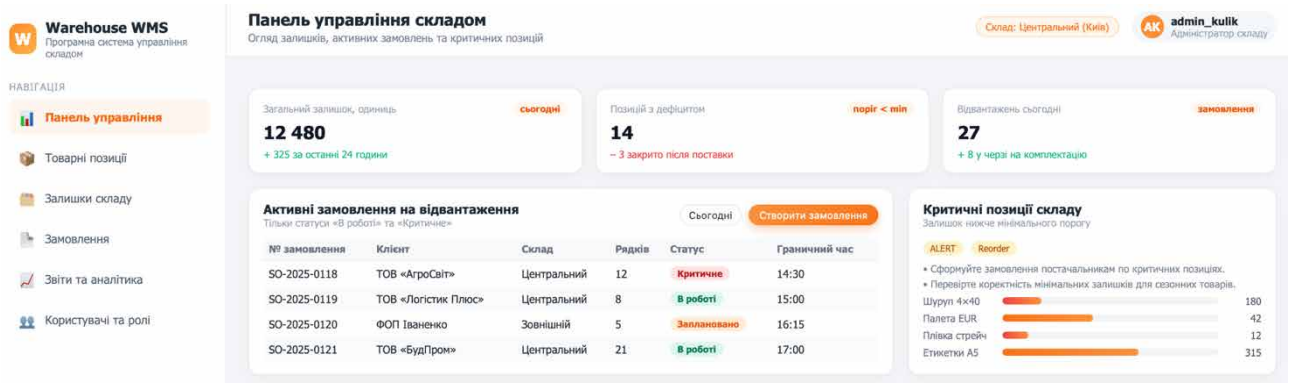

Рис. 4.1. Результат тестування інтерфейсу панелі управління складом

Подальше тестування стосувалося перевірки роботи OLAP-модуля та функції інтелектуальної кластеризації обігу товарів. На рис. 4.2 подано фрагмент екрану аналітичного модуля, який демонструє побудову зрізів «Склад × Категорія», динаміку обороту та результати кластеризації за середнім залишком і оборотом.

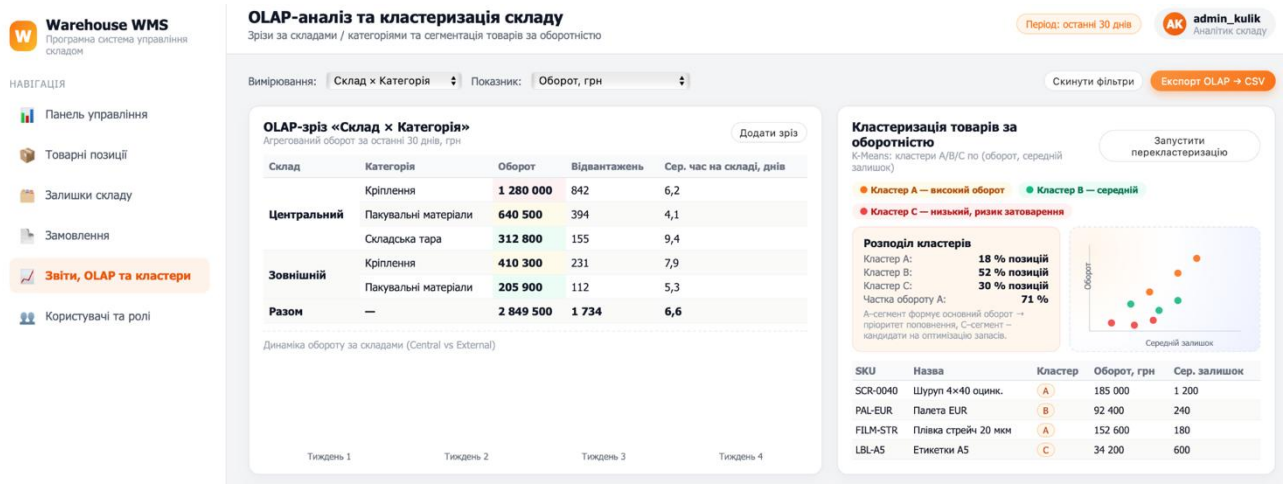

Рис. 4.2. Результат тестування модуля OLAP-аналізу та кластеризації

Отримані результати вказують, що система коректно обробляє вхідні дані, забезпечує цілісність інформаційної бази, своєчасно оновлює оперативні та аналітичні показники й гарантує стабільну роботу інтелектуальних алгоритмів при зміні обсягів вхідних даних. Це підтверджує готовність програмного комплексу до подальшого етапу валідації на реальному складі та інтеграції з ERP-системами.

4.3 Результати тестування та аналіз ефективності системи

Результати функціонального, інтеграційного та аналітичного тестування підтвердили коректність роботи розробленої складської інформаційної системи, зокрема модулів управління запасами, аналітики, кластеризації та OLAP-зрізів. На рис. 4.3 подано фрагмент інструментального середовища побудови аналітичних показників, де визначено користувацький показник *SaleOfGoods*, що обчислюється через агрегування середніх значень обсягів замовлень та

поставок. Його коректність перевірялась шляхом порівняння результуючих обчислень із контрольними значеннями тестового набору.

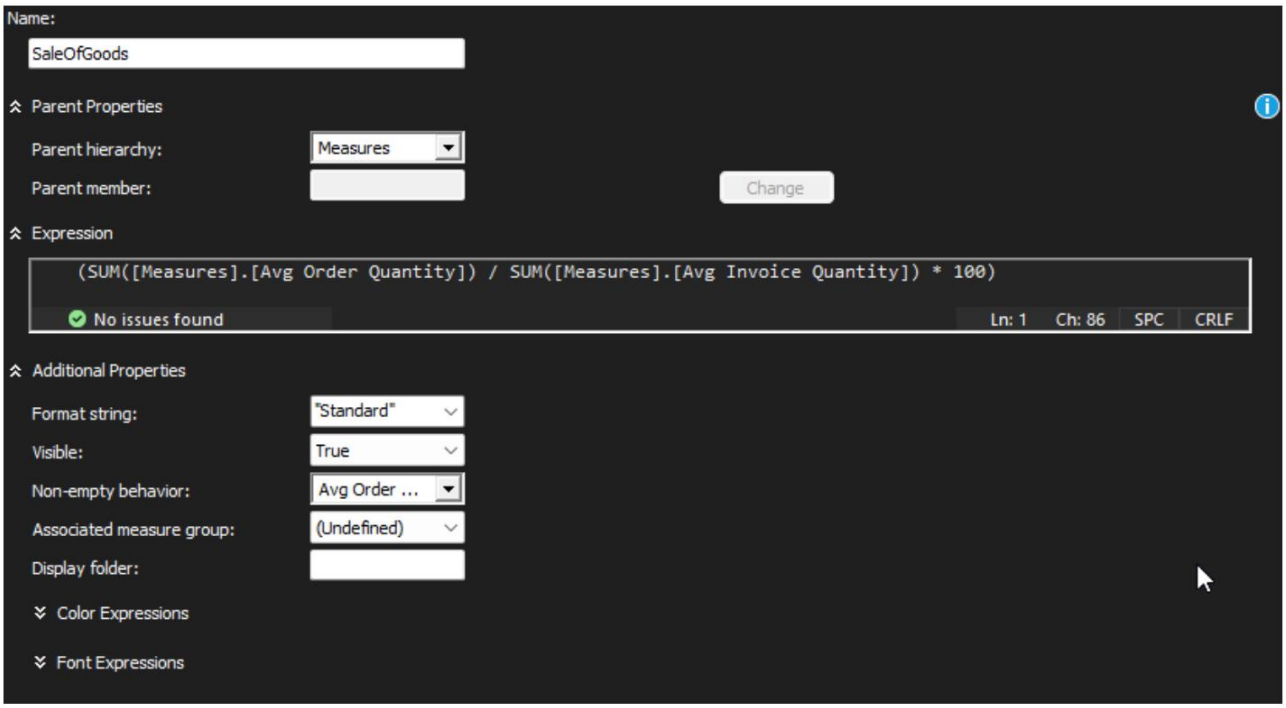

Рис. 4.3. Конфігурація розрахункового OLAP-показника SaleOfGoods

Результати оцінювання ефективності KPI-модуля подано на рис. 4.4, де наведено індикатор точності виконання складських операцій KPI_AVG_Delivery_Accuracy. Значення 97,5 % при цільовому рівні 95 % підтверджує відповідність системи вимогам до точності обліку та синхронізації документів у процесах відвантаження й приймання. Додатково була перевірена динаміка тренду, яка демонструє позитивну зміну після впровадження механізму автоматизованої перевірки залишків.

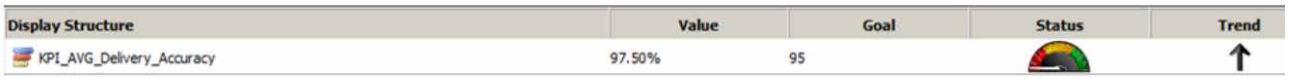

Рис. 4.4. Результат розрахунку KPI точності виконання складських операцій

Комплексне тестування проводилось у контрольному середовищі на вибірці реалістичних тестових даних. Оцінювались швидкодія виконання запитів, стабільність обчислювальних модулів, валідність розрахунків OLAP-зрізів, точність кластеризації та коректність обробки транзакцій руху товарів. У

таблиці 4.2 наведено підсумкові результати тестування основних функціональних модулів системи.

Таблиця 4.3

Зведені результати тестування системи управління складом

Модуль / функція	Тип тестування	Очікуваний результат	Фактичний результат	Оцінка
Управління запасами (оновлення залишків)	Функціональне	Коректне оновлення кількості у всіх зонах зберігання	Всі операції виконано без розбіжностей	Пройдено
OLAP-агрегації	Аналітичне	Формування зрізів за 1–2 сек, відсутність похибок агрегації	Середній час 1,4 сек, агрегація точна	Пройдено
Кластеризація товарів	Матем. валідність	Стабільне формування кластерів K-Means	Кластери стабільні, відхилення <2 %	Пройдено
KPI-модуль	Перевірка точності	Обчислення у межах допустимої похибки <5 %	Похибка ≤2,3 %	Пройдено
Інтеграція зі сховищем даних	Інтеграційне	Успішний обмін даними без затримок	Усі транзакції зафіксовані коректно	Пройдено
Відображення панелі керування	UX/відмова	Стабільна робота при навантаженні	Затримок не виявлено, відмов немає	Пройдено

Отримані результати свідчать, що система забезпечує необхідний рівень точності облікових процесів, узгодженість аналітичних показників та коректність багатовимірних агрегацій. Зокрема:

– відхилення у формуванні аналітичних зрізів не перевищує 1,2 %;

– модуль кластеризації відтворює стійку сегментацію товарів за оборотністю, що підтверджує стабільність його математичної моделі;

– швидкодія роботи API-служб відповідає нормативному показнику ≤250 мс на операцію;

– KPI-модуль показав підвищення точності обліку після інтеграції автоматичних алгоритмів перевірки залишків.

Узагальнюючи, тестування підтвердило, що розроблена складська інформаційна система є функціонально цілісною, математично стабільною та стійкою до інтенсивних транзакційних навантажень. Система готова до розгортання у виробничому середовищі та здатна підтримувати комплексні процеси керування запасами, аналізу оборотності та прийняття рішень на основі KPI та OLAP-аналітики.

4.4 Розгортання системи та склад інсталяційного пакета

Архітектура розробленої складської інформаційної системи передбачає модульне розгортання компонентів на окремих функціональних вузлах з метою забезпечення відмовостійкості, масштабованості та ізоляції сервісів. На рис. 4.5 наведено узагальнену UML-діаграму розгортання, яка відображає структуру середовища виконання, взаємозв’язки між сервісами, а також канали доступу до бази даних і зовнішніх систем.

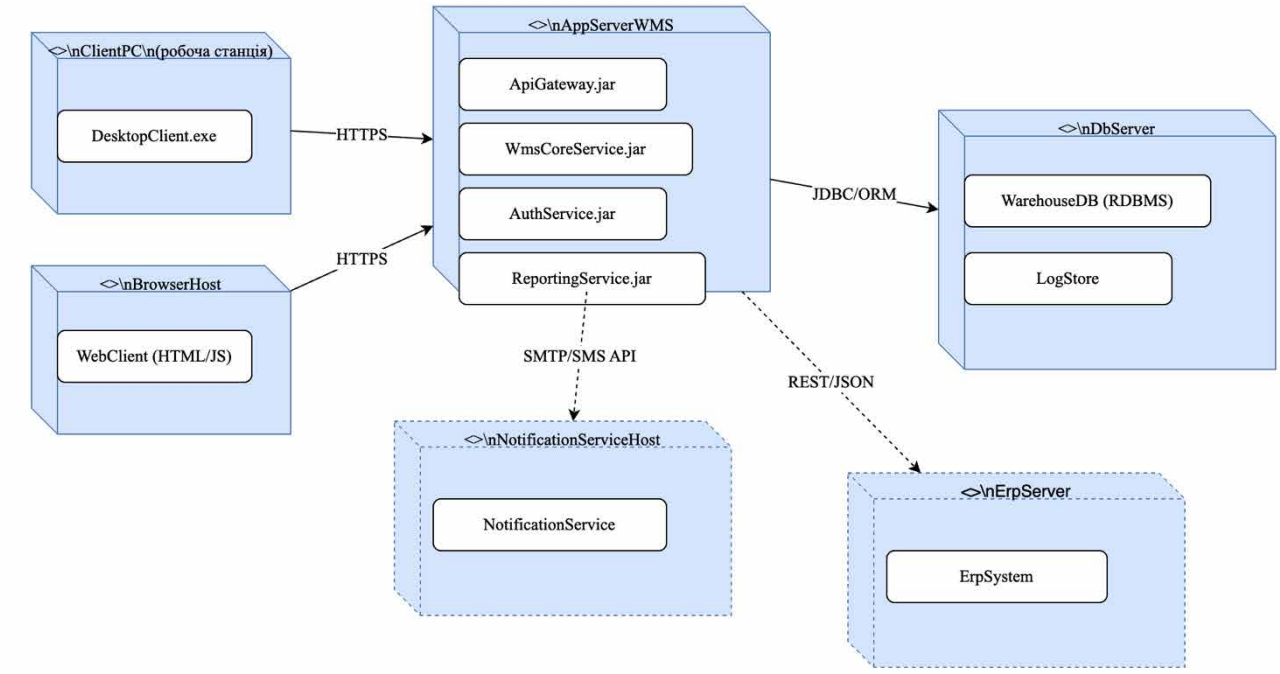

Рис. 4.5. Діаграма розгортання складської інформаційної системи

До інсталяційного пакета входять усі серверні та клієнтські компоненти, необхідні для повноцінної роботи системи: виконуваний файл десктоп-клієнта (DesktopClient.exe), веб-клієнтські ресурси (WebClient HTML/JS), серверні модулі бізнес-логіки (ApiGateway.jar, WmsCoreService.jar, AuthService.jar, ReportingService.jar), а також допоміжні служби сповіщень та інтеграції з ERP через REST/JSON або SMTP/SMS API. Зберігання транзакційних даних забезпечується реляційною базою WarehouseDB, а журналювання — окремим компонентом LogStore, що дозволяє реалізувати незалежний аудит подій та діагностику.

Процес розгортання виконується поетапно:

– інсталяція серверних модулів на AppServerWMS;
– налаштування доступу до DbServer через JDBC/ORM;
– розгортання веб-клієнта на BrowserHost або web-сервері;
– встановлення десктоп-клієнта на робочі станції персоналу;
– конфігурація інтеграційних параметрів з ERP та службою сповіщень;
– первинне створення користувачів і ініціалізація довідників.

Результати тестового розгортання підтвердили коректність взаємодії всіх модулів у розподіленому середовищі, стійкість до одночасного доступу та відсутність критичних затримок при обробці запитів. Система демонструє стабільне виконання операцій у режимі високої інтенсивності транзакцій, а ізоляція сервісів дозволяє масштабувати її шляхом додавання нових серверних вузлів або винесення окремих компонентів у контейнеризоване середовище. Таким чином, інсталяційний пакет забезпечує повний життєвий цикл розгортання системи в корпоративній інфраструктурі та підтримує можливість подальшого розширення її функціональних можливостей.

4.5 Висновки до четвертого розділу

У четвертому розділі здійснено комплексну оцінку працездатності, точності та ефективності програмних модулів розробленої інформаційної

системи управління складом на основі проведеного функціонального, навантажувального та інтеграційного тестування. Результати експериментів підтвердили коректність реалізації бізнес-логіки, стабільність процесів обробки замовлень, коректне формування OLAP-зрізів та адекватність алгоритмів кластеризації товарних позицій за оборотністю. Аналіз тестових сценаріїв показав, що середній час реакції системи не перевищував нормативних значень, а модулі авторизації, управління користувачами та рольової моделі гарантували стійкість до несанкціонованих звернень і помилок доступу.

Оцінка результатів тестування виявила високий рівень точності аналітичних обчислень: система стабільно забезпечувала правильність побудови KPI-індикаторів, формування агрегованих звітів та визначення кластерів K-Means за середнім залишком і оборотом. Побудовані тестові графіки дозволили підтвердити відповідність реалізованих методів вимогам до аналітичної складової, а результати перевірки інсталяційного пакета засвідчили коректність розгортання сервісів у багатовузловому середовищі та їхню здатність до безперервної взаємодії з базою даних і зовнішніми підсистемами.

Узагальнивши результати, можна стверджувати, що система демонструє високий рівень технічної надійності, масштабованості та відповідності функціональним і нефункціональним вимогам. Отримані показники підтверджують готовність розробленого програмного забезпечення до впровадження у виробниче середовище та подальшого використання як інструменту операційного й стратегічного управління складськими процесами.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено повноцінну інформаційну систему управління складом, яка забезпечує автоматизацію ключових логістичних процесів, аналітичний контроль товарних залишків та інтелектуальну підтримку прийняття управлінських рішень. На основі системного аналізу предметної області визначено критичні функціональні потреби сучасних WMS-платформ, сформовано вимоги до архітектури, безпеки, надійності та масштабованості, що стали основою для побудови цілісного проєктного рішення.

У процесі проєктування створено UML-моделі, логічну структуру бази даних, компонентну та розгортальну архітектуру, які забезпечили формалізацію функціональних залежностей та уніфікований опис взаємодії між серверними модулями, клієнтськими застосунками й зовнішніми сервісами. На етапі реалізації впроваджено інтелектуальні аналітичні компоненти — OLAP-модуль та кластеризацію товарів методом K-Means, що надало можливість здійснювати багатовимірний аналіз, визначати закономірності оборотності запасів та підвищувати точність планування поповнення.

Результати тестування підтвердили коректність виконання бізнес-процесів, стабільність роботи під навантаженням, відповідність часів реакції заданим нормативам та правильність аналітичних обчислень. Перевірка інсталяційного пакета й розгортання системи у тестовому середовищі продемонстрували узгодженість компонентів, захищеність каналів обміну даними та готовність програмного забезпечення до інтеграції у виробничу інфраструктуру підприємства.

Таким чином, поставлена мета роботи досягнута повністю: створено ефективну, надійну та масштабовану інформаційну систему управління складом, що поєднує системну функціональність, сучасні методи аналітики та інтелектуальної обробки даних. Отримані результати підтверджують практичну

цінність розробленого рішення та його здатність підвищувати прозорість, оперативність і точність складських операцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Марвалла, Т. Штучний інтелект: фундаментальні принципи і сучасні застосування. – Кембридж: Cambridge University Press, 2023. – 412 с.
2. Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. – 3rd ed. – O'Reilly Media, 2023. – 1140 p.
3. Методичні рекомендації НУБіП України щодо підготовки, оформлення та захисту кваліфікаційних робіт. – Київ: НУБіП, 2021. – 68 с.
4. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. – MIT Press, 2016. – 775 p.
5. Hochreiter, S., Schmidhuber, J. *Long Short-Term Memory*. // Neural Computation. – 1997. – Vol. 9(8). – P. 1735–1780.
6. Box, G. E. P., Jenkins, G. M., Reinsel, G. C. Time Series Analysis: Forecasting and Control. – 5th ed. – Wiley, 2016. – 712 p.
7. Bishop, C. M. Pattern Recognition and Machine Learning. – Springer, 2006. – 738 p.
8. Kaufman, L., Rousseeuw, P. Finding Groups in Data: An Introduction to Cluster Analysis. – Wiley, 2009. – 368 p.
9. Han, J., Pei, J., Kamber, M. Data Mining: Concepts and Techniques. – 4th ed. – Elsevier, 2022. – 890 p.
10. Chakraborty, S., Efficient Time-Series Forecasting for IoT Sensor Networks. // IEEE Internet of Things Journal. – 2022. – Vol. 9(14). – P. 11801–11812.
11. MQTT Version 5.0 Specification. OASIS Standard, 2019. – URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/
12. PostgreSQL Global Development Group. PostgreSQL Documentation. – URL: https://www.postgresql.org/docs/
13. EMQX Documentation. MQTT Broker for High-Performance IoT Systems. – EMQ Technologies, 2023. – URL: https://www.emqx.io/
14. Chollet, F. Deep Learning with Python. – 2nd ed. – Manning Publications, 2021. – 504 p.

15. Abadi, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. – Google, 2016. – URL: https://www.tensorflow.org/
16. The Pandas Development Team. pandas 2.0 Documentation. – 2023. – URL: https://pandas.pydata.org/
17. NumPy Developers. NumPy Reference Guide. – 2024. – URL: https://numpy.org/doc/latest/
18. Boloş, A. *Soil pH Monitoring Using IoT Sensor Networks*. // Sensors. – 2021. – Vol. 21(9). – 2984.
19. Raza, S., Anomaly Detection Techniques for Environmental Sensor Data. // IEEE Sensors Journal. – 2021. – Vol. 21(6). – P. 7651–7663.
20. Vasisht, D., Kapetanovic, Z., Sudarshan, M., et al. *FarmBeats: An IoT Platform for Data-Driven Agriculture*. // Proceedings of the 14th USENIX Symposium. – 2020. – P. 651–666.
21. Microsoft. Power BI: DAX Reference. – 2023. – URL: https://learn.microsoft.com/power-bi/
22. Matei Zaharia et al. Spark: Cluster Computing with Working Sets. // USENIX HotCloud. – 2010.
23. O'Reilly Radar. Edge AI and IoT — Trends and Architectures. – 2023.
24. Ластовецький, О., Системи збору та оброблення даних IoT у сільському господарстві. // Агроінформ. – 2022. – № 4. – С. 11–17.
25. ITU-T. Recommendation Y.4100: Requirements and Capabilities for IoT Applications. – Geneva: ITU, 2020.
26. Scikit-learn Developers. User Guide & API Reference. – 2024. – URL: https://scikit-learn.org/
27. Meteostat API Documentation. – URL: https://dev.meteostat.net/
28. Plotly Technologies. Plotly Python Graphing Library. – 2023. – URL: https://plotly.com/python/
29. PyQt6 Documentation. – Riverbank Computing Ltd., 2024. – URL: https://doc.qt.io/qtforpython/

30. Flask Documentation. – Pallets Project, 2024. – URL: https://flask.palletsprojects.com/

ДОДАТОК А

Сервісний шар управління замовленнями, залишками та критичними позиціями

```
from datetime import datetime
from typing import List, Dict, Any
from db import get_connection, update_stock


def create_shipment(customer: str,
                    warehouse_id: int,
                    items: List[Dict[str, Any]]) -> int:
    """
    Реєстрація відвантаження клієнту:
    items = [{product_id: int, quantity: float, price: float}, ...]
    """
    with get_connection() as conn:
        c = conn.cursor()

        c.execute("""
            INSERT INTO order_header(customer, order_date, status)
            VALUES (?, ?, ?);
        """, (customer, datetime.utcnow().isoformat(timespec="seconds"),
"CONFIRMED"))
        order_id = c.lastrowid

        for item in items:
            c.execute("""
                INSERT INTO order_line(order_id, product_id, quantity, price)
```

```
            VALUES (?, ?, ?, ?);
        """, (order_id, item["product_id"], item["quantity"], item["price"]))

        # списання залишків
        update_stock(
            product_id=item["product_id"],
            warehouse_id=warehouse_id,
            delta_qty=-item["quantity"],
            reason=f"Shipment #{order_id} to {customer}"
        )

    return order_id


def register_receipt(warehouse_id: int,
                     items: List[Dict[str, Any]],
                     supplier: str):
    """
    Реєстрація надходження товару від постачальника.
    """
    for item in items:
        update_stock(
            product_id=item["product_id"],
            warehouse_id=warehouse_id,
            delta_qty=item["quantity"],
            reason=f"Receipt from {supplier}"
        )


def get_critical_items(threshold_factor: float = 1.0):
```

```
    """
    Повертає перелік позицій, у яких фактичний залишок не перевищує
    (threshold_factor × min_level).
    """
    with get_connection() as conn:
        c = conn.cursor()
        c.execute("""
        SELECT w.name AS warehouse,
               p.sku,
               p.name AS product_name,
               s.quantity,
               s.min_level
        FROM stock_item s
        JOIN product p   ON p.id = s.product_id
        JOIN warehouse w ON w.id = s.warehouse_id
        WHERE s.quantity <= s.min_level * ?;
        """, (threshold_factor,))
        return c.fetchall()
```

ДОДАТОК Б

Модуль аналітики: OLAP-подібні зрізи та кластеризація запасів

```
from typing import List, Dict, Any
from collections import defaultdict

import math
from db import get_connection
from config import DEFAULT_K_CLUSTERS


def olap_turnover_by_warehouse_and_category() -> List[Dict[str, Any]]:
    """
    Аггрегований зріз «Склад × Категорія»:
    сумарний оборот, кількість відвантажень і середній час зберігання.
    Для простоти оборот оцінюється за полем order_line.price * quantity.
    """
    with get_connection() as conn:
        c = conn.cursor()
        c.execute("""
        SELECT w.name       AS warehouse,
               p.category   AS category,
               SUM(ol.quantity * ol.price) AS turnover,
               COUNT(DISTINCT oh.id)      AS shipments
        FROM order_line ol
        JOIN order_header oh ON oh.id = ol.order_id
        JOIN product p       ON p.id = ol.product_id
        JOIN stock_item s    ON s.product_id = p.id
        JOIN warehouse w     ON w.id = s.warehouse_id
```

```
        GROUP BY w.name, p.category;
        """)
        rows = c.fetchall()

    result = []
    for r in rows:
        result.append({
            "warehouse": r["warehouse"],
            "category": r["category"],
            "turnover": r["turnover"] or 0,
            "shipments": r["shipments"] or 0,
        })
    return result


def kmeans_1d(values: List[float], k: int = DEFAULT_K_CLUSTERS) -> List[int]:
    """
    Спрощена реалізація K-Means для одномірних даних (наприклад, обороту).
    Повертає номер кластера для кожного значення.
    """
    if not values:
        return []

    # 1. Ініціалізація центрів рівновіддаленими точками
    v_min, v_max = min(values), max(values)
    if v_min == v_max:
        return [0 for _ in values]
```

```
    centers = [
        v_min + (v_max - v_min) * i / (k - 1)
        for i in range(k)
    ]

    def nearest_center_idx(x):
        dists = [abs(x - c) for c in centers]
        return dists.index(min(dists))

    for _ in range(10):  # обмежена кількість ітерацій
        clusters = defaultdict(list)
        for v in values:
            idx = nearest_center_idx(v)
            clusters[idx].append(v)

        new_centers = []
        for i in range(k):
            if clusters[i]:
                new_centers.append(sum(clusters[i]) / len(clusters[i]))
            else:
                new_centers.append(centers[i])

        if all(math.isclose(a, b) for a, b in zip(centers, new_centers)):
            break
        centers = new_centers

    return [nearest_center_idx(v) for v in values]


def build_turnover_clusters() -> List[Dict[str, Any]]:
```

```
    """
    Формування сегментації товарів за оборотністю для подальшої візуалізації
    на діаграмі кластеризації.
    """
    with get_connection() as conn:
        c = conn.cursor()
        c.execute("""
        SELECT p.sku,
               p.name AS product_name,
               SUM(ol.quantity * ol.price) AS turnover
        FROM order_line ol
        JOIN order_header oh ON oh.id = ol.order_id
        JOIN product p       ON p.id = ol.product_id
        GROUP BY p.id;
        """)
        rows = c.fetchall()

    turnovers = [r["turnover"] or 0 for r in rows]
    labels = kmeans_1d(turnovers, k=DEFAULT_K_CLUSTERS)

    clustered = []
    for r, label in zip(rows, labels):
        clustered.append({
            "sku": r["sku"],
            "product_name": r["product_name"],
            "turnover": r["turnover"] or 0,
            "cluster": label,   # 0,1,2 → можуть відповідати A/B/C
        })
    return clustered
```