

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Інформаційна система автоматизації складського
обліку та управління виробництвом крафтових
м'ясних виробів»**

Виконав студент групи ІПЗ-22-2
ПАЛАМАРЧУК Іван Сергійович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПАЛАМАРЧУК Іван Сергійович**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Інформаційна система автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів»,**

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **об'єктом дослідження є процеси складського обліку та управління виробництвом на малих і середніх підприємствах крафтового м'ясного виробництва, а предметом дослідження є методи та програмні засоби автоматизації складського обліку з партійним відстеженням та управління виробництвом крафтових м'ясних виробів.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	02.03.2026р.	02.03.2026р.
2	Локтікова Т.М.	02.03.2026р.	02.03.2026р.
3	Локтікова Т.М.	02.03.2026р.	02.03.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота освітнього ступеня «бакалавр» складається з інформаційної системи автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів і пояснювальної записки. Текстова частина пояснювальної записки викладена на 73 сторінках і містить 10 сторінок додатків, 29 ілюстрацій та 26 таблиць. Список використаних джерел містить 30 найменувань. Загальний обсяг становить 83 сторінки.

Метою роботи є розробка інформаційної системи автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів.

У роботі було досліджено предметну область, проаналізовано аналоги та на основі результатів дослідження визначено завдання та обґрунтовано вибір інструментарію для розробки. Також було спроектовано клієнт-серверну архітектуру системи, наведено основні варіанти використання, розроблено логіку взаємодії компонентів системи та побудовано схему бази даних PostgreSQL. Серверна частина була реалізована за допомогою фреймворку ASP.NET Core, а клієнтська частина за допомогою бібліотеки React та мови програмування TypeScript. Розгортання системи було реалізовано за допомогою інструменту Docker Compose.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, КРАФТОВІ М'ЯСНІ ВИРОБИ, ПАРТІЙНЕ ВІДСТЕЖЕННЯ, СКЛАДСЬКИЙ ОБЛІК, УПРАВЛІННЯ ВИРОБНИЦТВОМ, ASP.NET CORE, POSTGRESQL.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Паламарчук І.С.			Інформаційна система автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів	Літ.	Арк.	Аркуші
Перевір.		Локтікова Т.М.					3	83
Керівник		Локтікова Т.М.				Житомирська політехніка, група ІПЗ-22-2		
Н. контр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's degree thesis consists of an information system for the automation of warehouse accounting and production management for artisan meat products, together with an explanatory note. The explanatory note contains 73 pages of main text and includes 10 pages of appendices, 29 illustrations, and 26 tables. The list of references contains 30 entries. The total volume of the work is 83 pages.

The aim of the thesis is to develop an information system for the automation of warehouse accounting and production management of artisan meat products.

The subject area was investigated, and analogues were analyzed. Based on the research results, the problem statement was formulated, and the choice of development tools was justified. A client-server architecture for the system was designed, the main use cases were outlined, the logic of interaction between system components was developed, and a PostgreSQL database schema was constructed. The server side was implemented using the ASP.NET Core framework, and the client side using the React library and the TypeScript programming language. The system was deployed using Docker Compose.

KEYWORDS: ARTISAN MEAT PRODUCTS, ASP.NET CORE, BATCH TRACKING, INFORMATION SYSTEM, POSTGRESQL, PRODUCTION MANAGEMENT, WAREHOUSE ACCOUNTING.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту.....	13
1.3 Вибір архітектури системи	17
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	20
Висновки до першого розділу	23
РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ ТА УПРАВЛІННЯ ВИРОБНИЦТВОМ	25
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	25
2.2 Розробка бази даних	35
2.3 Проєктування та реалізація алгоритмів роботи системи.....	44
2.4 Реалізація функціоналу системи складського обліку та автоматизації виробництва крафтових м'ясних виробів	48
Висновки до другого розділу.....	51
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ	53
3.1 Порядок встановлення та налаштування параметрів системи.....	53
3.2 Структура інтерфейсу системи.....	56
3.3 Тестування системи	63
Висновки до третього розділу	68
ВИСНОВКИ.....	69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	70
ДОДАТКИ.....	73

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізольованість, довговічність транзакцій)

API – Application Programming Interface (програмний інтерфейс застосунку)

CQRS – Command Query Responsibility Segregation (розділення відповідальності команд і запитів)

ERP – Enterprise Resource Planning (планування ресурсів підприємства)

FEFO – First Expired, First Out (першим спливає термін, першим відвантажуюється)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

MRP – Material Requirements Planning (планування потреби в матеріалах)

ORM – Object-Relational Mapping (об'єктно-реляційне відображення)

REST – Representational State Transfer (передача репрезентативного стану)

WMS – Warehouse Management System (система управління складом)

БД – база даних

СКБД – система керування базами даних

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Крафтове м'ясне виробництво належить до сегмента харчової промисловості, де автоматизація бізнес-процесів стає важливим чинником підвищення ефективності та конкурентоспроможності підприємств [30]. Облік на малих і середніх підприємствах цього сектору здебільшого ведеться в електронних таблицях або в паперових журналах. Такий підхід ускладнює ведення партійного обліку по термінах придатності, який є важливим для виявлення браку або небезпеки. Також відсутність зв'язку між складським і виробничим обліками змушує менеджерів перевіряти наявність сировини вручну, що призводить до сповільнення виробничих процесів. Попри це малі підприємства уникають застосування комплексних систем через складність їхнього впровадження та надмірну функціональність [13]. Тому виникає потреба у компактній та функціонально достатній системі, яка дозволить малим підприємствам оптимізувати роботу без значних фінансових та часових витрат.

Метою випускної кваліфікаційної роботи бакалавра є спроектувати та розробити клієнт-серверну інформаційну систему автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів, яка охоплює процеси підприємства від закупівлі сировини до відвантаження готової продукції.

Для досягнення поставленої мети визначено наступні **завдання**: дослідити предметну область; проаналізувати аналоги; сформулювати функціональні та нефункціональні вимоги до системи; спроектувати архітектуру системи; реалізувати функціонал серверної та клієнтської частин; протестувати систему; розгорнути на віддаленому сервері.

Об'єктом дослідження є процеси складського обліку та управління виробництвом на малих і середніх підприємствах крафтового м'ясного виробництва.

Предметом дослідження є методи та програмні засоби автоматизації складського обліку з партійним відстеженням та управління виробництвом крафтових м'ясних виробів.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

Практичне значення роботи полягає в тому, що система дозволяє малим та середнім підприємствам м'ясного виробництва замінити облік в електронних таблицях на ведення складських залишків у реальному часі, забезпечити партійний облік із контролем термінів придатності, а також усунути ручну перевірку наявності продукції. Компактність робить систему привабливою для впровадження на підприємствах, що уникають громіздких та комплексних рішень.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези:

1) Паламарчук І.С., Лисогор Ю.І., Локтікова Т.М. Дослідження принципів побудови та проєктування системи управління виробництвом крафтових м'ясних виробів: тези доповідей VIII Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», м. Житомир, 2–3 грудня 2025 р. Житомир: «Житомирська політехніка», 2025. С. 63–64. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2026/01/povnyj-tekst-1.pdf> [27];

2) Паламарчук І.С., Локтікова Т.М. Побудова та проєктування інформаційної системи автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів: матеріали XVI Міжнародної наукової конференції студентів та молодих вчених «Сучасні інформаційні технології – 2026», м. Одеса, 14–15 травня 2026 р. Одеса: Наука і техніка, 2026. С. 277–279 [28].

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ

1.1 Постановка задачі

Крафтове виробництво м'ясних виробів належить до сегмента харчової промисловості та є одним із технологічно складних її напрямів з погляду управління запасами й виробничими процесами. Крафтові виробники працюють у сегменті малого та середнього бізнесу. Для них характерні широкий асортимент продукції, невеликі обсяги окремої партії та орієнтація на якість. Саме ця специфіка визначає вимоги до облікового рішення й відрізняє його від типових систем масового виробництва.

Виробничий цикл такого підприємства розпочинається з надходження замовлення від контрагента. На його основі обчислюють потребу в сировині та визначають терміни запуску виробничих операцій. Коректний розрахунок потреби можливий лише за наявності актуальних даних про залишки сировини на складах.

Після того як потребу визначено, підприємство закуповує сировину (туші, напівтуші, спеції) у постачальників. Доставлену сировину зважують і розміщують на складі з присвоєнням дати виготовлення, дати надходження та терміну придатності. Кожна одиниця сировини набуває власних часових характеристик.

Після закупівлі сировину розбирають, обвалюють та жилують. Особливість цих операцій полягає в тому, що з однієї вхідної одиниці отримують численні вихідні продукти, кожен з яких має власну масу, собівартість і термін придатності. У зв'язку з цим виникає необхідність розподілити вартість єдиної вхідної одиниці між усіма вихідними компонентами та забезпечити відповідність сум мас на вході так і на виході з урахуванням технологічних втрат.

Отримані після розбирання напівфабрикати проходять подальші стадії перероблення, склад яких залежить від типу кінцевого продукту. Далі готову продукцію передають на пакування, а завершальною ланкою циклу є реалізація

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

з підтвердженням операції в системі обліку. Паралельно з основним потоком, між описаними етапами або під час них, підприємство переміщує цінності між складами, коригує залишки та проводить інвентаризацію. Таким чином, підприємство одночасно оперує кількома потоками матеріальних цінностей, кожен з яких має власну часову та кількісну динаміку.

Описана послідовність процесів накладається на дві суттєві властивості галузі, а саме високу швидкість псування сировини та жорсткі санітарні й гігієнічні вимоги. Через це партійний облік із контролем термінів придатності стає обов'язковою умовою роботи. Для покращення ефективності обліку застосовують принцип FEFO, коли першою у виробництво подається партія з найкоротшим терміном придатності незалежно від дати її надходження.

Для обліку та автоматизації описаних процесів існують різні типи систем, які класифікуються за функціоналом, за призначенням, за користувачами, які їх використовують та за процесами, які вони автоматизують. Перший клас становлять системи управління складом (WMS, від англ. Warehouse Management System). Їх основним призначенням є автоматизація операцій із фізичного переміщення товарів: оприбуткування, розміщення, відбору та відвантаження. Такі системи відстежують залишки в розрізі партій, підтримують проведення інвентаризацій. Саме цей клас систем відповідає за точність складського обліку в режимі реального часу.

Проте складський облік охоплює лише одну сторону діяльності виробничого підприємства. Для планування потреби у сировині використовуються системи класу MRP (від англ. Material Requirements Planning). Їх логіка ґрунтується на тому, що потреба у сировині є похідною від потреби у кінцевому виробі. За допомогою розгортання специфікації продукту та з врахуванням поточних залишків система розраховує, що саме і коли потрібно закупити або виготовити для виконання виробничого плану [14].

На основі попередніх систем існують системи планування ресурсів підприємства (ERP, від англ. Enterprise Resource Planning). Вони є найбільш комплексним рішенням і поєднують в собі різні бізнес-процеси та модулі [24].

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Але для малих підприємств вони нерідко виявляються надмірними за обсягом функціоналу та вартістю впровадження [3].

З цього випливає, що для малого крафтового виробництва оптимальним є рішення, що поєднує функціонал складського обліку рівня WMS з елементами планування потреби сировини. Таке поєднання не навантажує підприємство зайвими модулями, що не відповідають його масштабу.

Після проведення аналізу предметної області можна сказати, що завдання полягає у розробці інформаційної системи, що включає розробку таких модулів як партійний облік, виробництво зі специфікаціями, закупівель, складські документи, автентифікація та рольова авторизація, управління довідниками, розбирання сировини на компоненти.

Модуль автентифікації та рольової авторизації забезпечує ідентифікацію користувачів за логіном і паролем та розмежування доступу до операцій відповідно до призначеної ролі. Кожен користувач бачить лише ті розділи системи та може виконувати лише ті дії, які відповідають його посаді на підприємстві. Паролі зберігаються у хешованому вигляді, а всі операції фіксуються в журналі аудиту.

Модуль управління довідниками містить фундаментальну інформацію, яку використовують інші модулі системи. Довідники охоплюють каталоги складів, одиниць вимірювання, продуктів із категоризацією, а також реєстр контрагентів. Записи довідників не будуть видалятися фізично, а будуть архівуватися для забезпечення коректного відображення документів за минулі періоди навіть після виведення запису з активного обігу.

Модуль партійного обліку реалізує відстеження кожної облікової одиниці товару від моменту надходження до списання. Кожній партії присвоюється номер, дата виробництва та термін придатності. Саме цей модуль забезпечує просте отримання відповіді на питання про те, яка сировина зараз наявна, в якій кількості та до якої дати вона придатна до використання.

Модуль складських документів автоматизує керування складськими залишками підприємства. Прибуткова накладна фіксує надходження сировини

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

від постачальника з розбиттям позицій на партії. Видаткова накладна оформлює відвантаження готової продукції клієнту з автоматичним підбором партій за принципом FEFO або ручним вибором. Документ переміщення між складами забезпечує облік внутрішнього руху запасів, а акт інвентаризації дозволяє звірити фактичні залишки з обліковими та автоматично сформувати рухи коригування.

Модуль закупівель буде реалізовано у спрощеній формі, що відповідає потребам малого підприємства. Менеджер з постачання формує замовлення постачальнику із переліком позицій та кількостей. Після фізичного отримання товару комірник оформлює прибуткову накладну з прив'язкою до відповідного замовлення, і система автоматично відстежує, скільки з замовленого вже отримано, а скільки ще очікується.

Модуль продажів працює таким чином, де кожне замовлення клієнта ініціює виробничий процес. Менеджер з продажів реєструє замовлення із зазначенням контрагента та переліку продуктів. Після виробництва та відвантаження до замовлення прив'язується відповідна видаткова накладна, що дозволяє відстежувати ступінь виконання кожного замовлення.

Модуль виробництва забезпечує запуск та контроль виконання виробничих завдань. Диспетчер створює виробниче замовлення на основі специфікації продукту, де описано перелік та норми витрат сировини на одиницю готового виробу. Система перевіряє наявність необхідних матеріалів, резервує їх на складі, а після завершення виробництва автоматично списує сировину та створює нову партію готової продукції з розрахованою собівартістю.

Модуль розбирання реалізує технологічний процес розбиття вхідного товару на декілька компонентів. Специфікація розбирання описує, які вихідні продукти отримуються з однієї вхідної одиниці та в яких нормативних частках. Замовлення на розбирання проходить той самий поетапний цикл, що й виробниче.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Безпосередніми користувачами виступають комірники, які оформлюють прибуткові та видаткові накладні, проводять інвентаризацію, технологи, які керують рецептурами, диспетчери виробництва, які створюють виробничі замовлення, менеджери з постачання та продажів, які ведуть відповідні замовлення, а також керівники підприємства, що відстежують залишки на складі та аналізують каталог продукції.

Фінальним результатом проєкту буде програмний продукт у вигляді клієнт-серверного застосунку з програмним інтерфейсом на бекенді та односторінковим вебінтерфейсом на фронтенді, розгорнутий за допомогою Docker Compose. Система повинна забезпечувати коректний партійний облік у режимі реального часу, гарантувати цілісність документального ланцюга та надавати інструменти для оперативного контролю термінів придатності наявних запасів.

1.2 Аналіз аналогів програмного продукту

Для подальшого аналізу вимог до систем складського обліку та автоматизації виробництва було оглянуто три аналоги, які мають попит на українському ринку або відповідають специфіці малих та середніх підприємств.

Першим об'єктом аналізу є веборієнтована ERP система Odoo [16]. Дана програма побудована за модульним принципом де кожен модуль покриває певний сегмент бізнесу, що робить систему гнучкою. З основних модулів програма має продажів, закупівель, склад, виробництво, фінансовий модуль та CRM. У контексті дослідження, аналіз охоплює модулі складу та виробництва.

Модуль "Склад" підтримує прибуткові та видаткові накладні, переміщення між складами та інвентаризацію (рис. 1.1). Система підтримує повний цикл документообігу зі статусами "Чернетка", "Підтверджено", "Готово". Скасування вже проведених документів реалізовано через зворотні проводки. Окремо варто відзначити функціонал партійного обліку. Партія в Odoo є повноцінним самостійним об'єктом бази даних із власним життєвим циклом. Завдяки цьому система забезпечує двосторонню простежуваність.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Попри модульність системи, базова конфігурація для виробничого підприємства потребує від двох до п'яти місяців з залученням сертифікованого партнера [8]. У системі є базовий функціонал створення специфікації на розбирання, але немає вбудованого встановлення коефіцієнтів цінності на кожен продукт та налаштування відсотків від сумарної вхідної ваги на кожен вихідний продукт. Це доводиться налаштовувати обхідними шляхами або встановлювати сторонні плагіни.

Product	Location	On Hand	Forecast	Route	Min	Max	To Order	Actions
[E-COM06] Corner Desk ...	WH/Stock	4.00	-1.00		0.00	0.00	1.00	Order Once Automate Snooze
[E-COM09] Large Desk	WH/Stock	1.00	-4.00		0.00	0.00	4.00	Order Once Automate Snooze
[FURN_9001] Flipover	WH/Stock	5.00	-6.00		0.00	0.00	6.00	Order Once Automate Snooze
[FURN_9666] Table	WH/Stock	2.00	-1.00		0.00	0.00	1.00	Order Once Automate Snooze
[FURN_7777] Office Chair	WH/Stock/Asse...	4.00	4.00	Buy	5.00	10.00	6.00	Order Once Automate Snooze
[FURN_8888] Office Lamp	WH/Stock/Asse...	8.00	8.00		10.00	10.00	2.00	Order Once
[FURN_8900] Drawer Black	WH/Stock/Asse...	12.00	12.00	Manufacture	0.00	0.00	0.00	
[FURN_9001] Flipover	WH/Stock/Asse...	5.00	5.00		0.00	0.00	0.00	Snooze
[E-COM06] Corner Desk ...	WH/Stock/Flat P...	4.00	4.00	Manufacture	0.00	0.00	0.00	Snooze
[E-COM09] Large Desk	WH/Stock/Flat P...	1.00	1.00		2.00	2.00	1.00	Order Once
[FURN_9666] Table	WH/Stock/Flat P...	2.00	2.00	Manufacture	5.00	10.00	8.00	Order Once Automate Snooze

Рис. 1.1. Вікно модуля “Склад” в Odoo

Отже, Odoo є гнучкою системою з широким охопленням функціоналу, однак її надмірна універсальність та тривалий термін впровадження є вагомими обмеженнями для малого бізнесу.

Наступним аналогом є хмарна система Katana MRP [11]. Дана програма є актуальним рішенням для управління виробництвом та запасами, орієнтоване на малий та середній бізнес. З сильних сторін є управління партіями та простежуваність (рис. 1.2). Вона забезпечує повний контроль партій від закупівлі сировини до відвантаження [10]. Присутній автоматичний обрахунок собівартості. Також доступне створення багаторівневих специфікацій із чітким відображенням напівфабрикатів.

Однак критичним обмеженням є відсутність підтримки специфікацій розбирання. Крім цього, система не має української локалізації. Таким чином, Katana є зручним та швидким у розгортанні рішенням з партійним обліком та

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

модулем виробництва, але обмежена в інших функціях.

Name	Variant code	Batch number	In stock	1 ↑ Expiration date	2 ↑ Created date
Chocolate Yum Bars / 2 oz single / Almond Nougat	CYB-AN-02	Batch-003	90 pcs	2020-07-24	2020-07-03
Caramel Cream Bar	CARAMEL-BAR	07-001	50 pcs	2020-07-31	2020-07-03
Caramel Cream Bar	CARAMEL-BAR	07-002	50 pcs	2020-08-01	2020-07-03
Chocolate Yum Bars / 2 oz single / Hazelnut Glaze	CYB-HG-02	Batch-005	50 pcs		2020-07-03
Chocolate Yum Bars / 3 oz single / Hazelnut Glaze	CYB-HG-03	Batch-006	30 pcs		2020-07-03
Chocolate Yum Bars / 3 oz single / Almond Nougat	CYB-AN-03	Batch-004	20 pcs		2020-07-03

Рис. 1.2. Вікно з таблицею партій в Katana MRP

Також існує платформа Microsoft Dynamics 365 Business Central [15]. Дане програмне забезпечення забезпечує управління складом із підтримкою партійного обліку, автоматичного підбору за терміном придатності, розмежування доступу на основі ролей, управління виробництвом (рис. 1.3). Разом із тим платформа має обмеження. По-перше, у базовій системі відсутній нативний інструмент специфікацій розбирання. Він може бути підключений тільки за допомогою сторонніх розширень. Крім того, не варто оминати проблему впровадження системи. Через свою комплексність та складність цей процес може тривати від 3 до 9 місяців та потребує значних фінансових витрат.

Затуплене Вироб. Замовлення
3B3-000002 · Печиво з шоколадними дрібками

Опис: Печиво з шоколадними дрібками

Номер: 3B3-000002 | Кількість: 10

Опис 2: | Строк оплати: 19.01.2024

Тип джерела: Товар | Ідентифікатор виконавця: | Заблоковано: ☐

Номер джерела: T08-00008 | Дата останньої зміни: 19.01.2024

Опис пошуку: ПЕЧИВО З ШОКОЛАДНИМИ ДРІБКАМИ

Код товару	Строк оплати	Опис	Дата-Час початку	Дата-Час Кінець	Кількість	Код одиниці виміру	Закінчена кількість	Залишок	Собівартість одиниці
→ T08-00008	19.01.2024	Печиво з шоколадними дріб...	18.01.2024 12:08	18.01.2024 23:59	10	KG	0	10	32.50

План: 18.01.2024 12:08 | 18.01.2024 23:59

Облік: Облікова група товарів: 260000 | Регіон Код: | Загальна товарна група: ТОВАРИ | Код складу: ОСНОВНИЙ

Рис. 1.3. Вікно редагування виробничого замовлення

Порівняння функціоналу дозволяє виокремити набір функцій, який є спільний для всіх трьох систем. До нього належать управління складськими документами, журнал аудиту, партійний облік, ведення специфікацій, контроль термінів придатності та калькулювання собівартості. Оскільки ці можливості повторюються в усіх аналогах, вони фактично окреслюють ядро предметної області та мають бути реалізовані у запланованій системі.

Отже, після аналізу аналогів можна сказати, що програми цілком розв'язують задачі підприємства. Але варто звернути увагу, що Odoo та Microsoft Dynamics 365 Business Central є платформами, які орієнтуються на середній та великий бізнес, що проявляється у надлишковості функціоналу та, як наслідок, у тривалості й вартості впровадження, які є невиправданими для малих підприємств із вузькою спеціалізацією. З іншого боку Katana MRP вирізняється простотою та швидкістю впровадження, проте не забезпечує необхідної глибини для сегмента м'ясного виробництва. Тому варто знайти баланс між функціональністю та простотою, щоб програма була спроможною закрити більшість проблем підприємств харчової промисловості.

На основі проведеного аналізу можна сформулювати висновок щодо функціонального складу майбутньої програми. Система повинна включати повноцінний партійний облік із підтримкою принципу FEFO та автоматичним і ручним підбором партій, повний цикл складських документів, управління замовленнями на закупівлі та продажі, виробничі замовлення зі специфікаціями продуктів як для складання готового виробу, так і для розбирання вхідної сировини на вихідні компоненти, а також рольову авторизацію та журнал аудиту операцій. В результаті можна сформувати таблицю, яка наочно характеризує функціонал аналогів та функціонал, який запланований у системі (табл. 1.1).

Таблиця 1.1

Порівняння функціоналу аналогів

Функціональний критерій	Katana MRP	Odoo ERP	Microsoft Dynamics 365 Business Central	Запланована система
Управління складськими документами	Так	Так	Так	Так

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовження табл. 1.1

Управління партіями та їх простежуваність	Так	Так	Так	Так
Управління специфікаціями	Так	Так	Так	Так
Підтримка специфікацій розбирання	Ні	Частково	Ні	Так
Контроль термінів придатності та FEFO	Так	Так	Так	Так
Управління правами доступу на основі ролей	Так	Так	Так	Так
Калькулювання собівартості	Так	Так	Так	Так
Швидкість впровадження	4-6 тижнів	2-5 місяців	3-9 місяців	Швидке розгортання
Українська локалізація	Ні	Так	Так	Так
Розширена фільтрація даних з можливістю групування	Ні	Так	Так	Так

1.3 Вибір архітектури системи

З погляду масштабування та впровадження на інших платформах у майбутньому доцільно обрати клієнт-серверну модель, де бекенд та фронтенд розділені. Серверна частина відповідає за всю бізнес-логіку, керує даними та забезпечує їхню цілісність. Тоді як клієнтська частина відповідає виключно за відображення інформації та взаємодію з користувачем. Завдяки такому підходу з'являється перевага у можливості розробляти, тестувати та масштабувати обидві частини незалежно одна від одної.

Водночас використання монолітної архітектури є обґрунтованим через простоту розробки без додаткових налаштувань взаємодії. Але клієнт-серверна модель дає можливість вибору, в якому напрямку буде розроблятися проєкт у майбутньому. Наприклад, окремий мобільний застосунок чи інтеграція з пристроями в цеху, якщо така функціональність буде потрібна.

Взаємодія між клієнтом і сервером реалізована через REST API з використанням протоколу HTTP. Сервер надає набір кінцевих точок, що відповідають операціям над ресурсами системи. Клієнт надсилає асинхронні HTTP-запити та отримує відповіді у форматі JSON.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Серверна частина побудована на основі принципів чистої архітектури у поєднанні з предметно-орієнтованим проєктуванням [12, 25]. Предметно-орієнтоване проєктування є підходом до розробки програмного забезпечення, при якому головним орієнтиром стає предметна область задачі, а не технічні деталі реалізації [6, 26]. Вибір цього підходу зумовлений тим, що в програмі передбачається складна та критична логіка, яку важливо сконцентрувати в одному захищеному місці.

Чиста архітектура розв'язує цю проблему через суворе правило в якому залежності між шарами дозволені лише в одному напрямку, від зовнішніх до внутрішніх. Серверну частину системи поділено на чотири проєкти, взаємодію між якими проілюстровано на діаграмі компонентів (рис. 1.4).

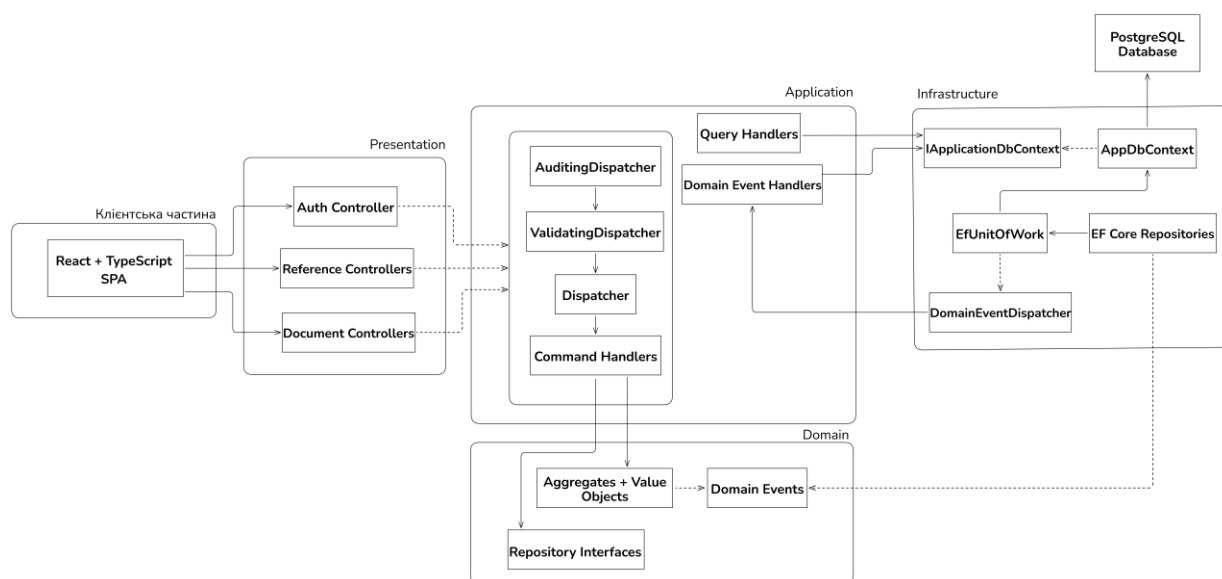


Рис. 1.4. Діаграма компонентів системи

Доменний шар є ядром і не залежить від жодного іншого. Він містить агрегати, об'єкти-значення, доменні події та інтерфейси репозиторіїв, але нічого не знає ані про базу даних, ані про HTTP.

Шар інфраструктури забезпечує технічну реалізацію доступу до даних. Репозиторії в цьому шарі абстрагують роботу з БД, що створює чистий інтерфейс для доступу та збереження агрегатів. Тут розміщені конфігурації для ORM. Це єдиний шар, який знає про PostgreSQL.

Шар застосунку застосовує бізнес-логіку з доменного шару, координує виконання операцій та взаємодію між ним та інфраструктурним шаром. В ньому

зосереджені обробники команд і запитів, обробники доменних подій та сервіси застосунку.

Шар представлення містить контролери, які перенаправляють запити до відповідних сервісів. Цей шар є тонким, він не містить бізнес-логіки, та відповідає виключно за HTTP-комунікацію.

Взаємодія з базою даних загалом буде відбуватися не напряму, а через ORM. Таблиці будуть генеруватися на основі доменних моделей у програмному коді за допомогою Entity Framework Core.

Всередині цієї архітектури реалізовано низку патернів, кожен з яких розв'язує конкретну проблему. Першим є патерн Агрегат разом із Репозиторієм.

Кожна ключова концепція предметної галузі оформлена як агрегат з кореневою сутністю. Усі операції зміни стану всередині агрегату проходять виключно через цей кореневий об'єкт, що гарантує консистентність даних. Доступ до агрегатів ззовні організовано через інтерфейси репозиторіїв, який оголошені у доменному шарі, а реалізований в інфраструктурному шарі.

З цього безпосередньо впливає наступне архітектурне рішення, яке розділяє відповідальності між командами та запитами [4]. Операції, що змінюють стан системи, реалізовано як команди. Вони проходять через агрегати та репозиторії, дотримуючись усіх бізнес-правил і транзакційності. Операції читання реалізовано як запити з прямим доступом до контексту бази даних через LINQ.

Такий поділ обґрунтований тим, що потреби двох типів операцій є принципово різними. Запис вимагає суворого дотримання бізнес-правил, тоді як читання потребує передусім швидкості й гнучкості у формуванні відповіді.

Усі команди та запити надходять до своїх обробників через єдину точку входу, який в системі називається центральним обробником, який реалізує інтерфейс IDispatcher. Даний клас побудовано за патерном Декоратор, де кілька класів послідовно обгортають один одного [29]. Завдяки цьому валідація, виконання операції та запис у журнал аудиту розділені між окремими

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

відповідальними класами, а розширення ланцюга новою поведінкою не потребує змін у наявному кодi.

Механізмом типобезпечності є строго типізовані ідентифікатори. Вони є звичайними обгортками над Guid. Це унеможливорює помилки передачі ідентифікатора неправильної сутності на рівні компілятора та усуває антипатерн Primitive Obsession.

Клієнтська частина буде реалізована як односторінковий застосунок. Даний підхід забезпечує плавний і швидкий інтерфейс, оскільки браузер завантажує застосунок один раз, а подальша навігація відбувається на клієнтській стороні. Структуру клієнтської частини організовано за доменним принципом де кожна функціональна область системи має власний ізольований модуль, що містить типи даних, функції звернення до API та компоненти інтерфейсу [23]. Сторінки є тонкими компонентами рівня маршрутизації, що лише компонують готові модулі, тоді як спільні примітиви інтерфейсу та допоміжні засоби виокремлено в окремий шар, який доступний усім частинам застосунку.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Система автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів має специфічні функціональні вимоги, що вимагає детального аналізу бібліотек та фреймворків, які найкраще відповідають вимогам реалізації фінального рішення. Варто врахувати суворий контроль партій та загальною кількістю залишків на складах, обробку документів з побічним ефектом у вигляді оновлення залишків, коли документ завершено, FEFO-підбір партій при списуванні для виробництва, розмежування прав доступу. Саме ці вимоги є критеріями вибору технологій для розробки.

Мовою програмування, якою буде реалізовано серверну частину буде C# з платформою ASP.NET Core [2]. Мова є статично типізованою, що обґрунтовується важливістю знаходження помилок на етапі розробки та

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

вимогою до надійності бізнес-обрахунків. Строгі типи є засобом захисту інваріантів на рівні компілятора. Порівняно з Java, яка є рівноцінною альтернативою в корпоративному сегменті, C# пропонує більш зрілу підтримку незмінних класів, природну інтеграцію типів-значень [1]. ASP.NET Core демонструє вищу пропускну здатність у незалежних бенчмарках [19].

Клієнтську частину системи розроблено на основі бібліотеки React [18]. Статична типізація коду забезпечується мовою програмування TypeScript [21]. Вона забезпечує узгодженість типів між API-відповідями та компонентами інтерфейсу ще на етапі компіляції. Для стилізації інтерфейсу застосовано CSS-фреймворк Tailwind CSS [20]. Бібліотека shadcn/ui буде слугувати основою для готових компонентів інтерфейсу та буде забезпечувати єдиний візуальний стиль [9]. Збірка інтерфейсу буде відбуватися за допомогою інструменту Vite [22].

Для впорядкованого обміну даними із сервером застосовано бібліотеку TanStack Query, яка зберігає вже отримані відповіді у тимчасовому сховищі та позбавляє від повторного завантаження тих самих даних. Перевірку правильності значень, що їх вводить користувач у форми, побудовано на поєднанні бібліотеки React Hook Form зі схемами правил, описаними засобами бібліотеки Zod. Переходи між розділами інтерфейсу забезпечує бібліотека React Router, через що адреса сторінки завжди відповідає поточному стану застосунку.

Для реалізації автентифікації та авторизації обрано вбудований механізм Cookie Authentication платформи ASP.NET Core. Підходи які розглядаються це автентифікація на основі токенів та сесійний підхід. Їхня ключова відмінність у наявності стану на сервері. Автентифікація на основі токенів не потребує серверного сховища сесій, що дає вигоду у розподілених системах із кількома серверами. Оскільки система обслуговується єдиним монолітним сервером, ця перевага не реалізується. Натомість серверне зберігання сесії забезпечує швидке відкликання доступу при зміні ролі чи деактивації користувача. З огляду на наявність рольового розмежування прав обрано сесійну автентифікацію.

Тип СКБД визначається характером даних у системі. Це повинна бути реляційна база із підтримкою ACID транзакцій. Операції проведення документів

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

відбуваються одночасно у кількох таблицях і повинні виконуватися як єдина атомарна операція. Нереляційні бази даних нехтують узгодженістю задля горизонтального масштабування, що є неприпустимим компромісом для облікової системи.

Серед альтернатив було проведено порівняльний аналіз трьох систем за критеріями, що є ключовими для даного проєкту (табл. 1.2).

Таблиця 1.2

Порівняння СКБД

Критерій	PostgreSQL 16	MySQL 8.4	MS SQL Server 2022
Відповідність ACID	Так	Так	Так
Ліцензія	Відкрита	Відкрита та Комерційна	Комерційна
CHECK-обмеження	Так	Частково	Так
Часткові індекси	Так	Так, але не повноцінно	Так
Ізоляція транзакцій без блокування читачів	Так	Так	Частково
Масштабованість	Висока	Середня	Висока
Розмір образу в Docker	~270 МБ	~350 МБ	~1,5 ГБ
Інтеграція з EF Core	Через Npgsql	Через Pomelo	Нативна
Покриваючі індекси	Так	Ні	Так
Підтримка складних типів даних	Jsonb, масиви, enum, range	Json	Json

В кінцевому результаті було обрано PostgreSQL [17]. Ця СКБД підтримує CHECK-обмеження на рівні бази даних, що дозволяє створювати бізнес-правила на рівні бази даних, а не лише на рівні програмного коду. Також аргументом стала підтримка часткових індексів, які дозволяють покращити продуктивність при великих обсягах даних. У додаток дана система БД має найкращу підтримку складних типів (масиви, перелічувані типи даних, JSON), що може бути використаним при масштабуванні функціоналу. Крім того, порівняно зі стороннім провайдером, який існує для MySQL, бібліотека Npgsql має повну інтеграцію з Entity Framework Core [7].

Система розгортається у Docker-контейнерах з використанням Docker Compose для координації сервісів [5]. Використання контейнеризації забезпечує ідентичне середовище на машині розробника та на цільовому сервері, що усуває клас проблем пов'язаних з сумісністю на різних комп'ютерах. Версії залежностей та змінні середовища зафіксовані в конфігурації, що робить розгортання відтворюваним.

Контейнери потребують середовища, у якому вони взаємодіятимуть із зовнішньою мережею. У ролі вебсервера було обрано зворотний проксі Caddy, який приймає звернення користувачів, самостійно отримує та оновлює сертифікати захищеного з'єднання і спрямовує запити до серверної частини. Саме рішення розгорнуто на безкоштовному рівні хмарної платформи Oracle Cloud. Таке поєднання задовольняє вимогу безперервної доступності системи для невеликого підприємства.

Для розробки серверної частини обрано Visual Studio через тісне поєднання з екосистемою платформи .NET і засоби покрокового налагодження, автоматичного доповнення коду та безпечного перейменування елементів коду. Клієнтську ж частину зручніше розробляти у легшому редакторі коду Visual Studio Code, який краще пристосований до екосистеми мови JavaScript.

Висновки до першого розділу

У розділі проведено аналіз предметної області, сформульовано задачу автоматизації складського обліку й управління виробничими процесами. Аналіз трьох наявних рішень, а саме Odoo, Katana MRP та Microsoft Dynamics 365 Business Central, показав, що вони або перевантажені зайвим функціоналом для малого бізнесу, або не забезпечують необхідної глибини в контексті специфіки виробництва м'ясних виробів.

На підставі цього аналізу було окреслено перелік модулів майбутньої системи, серед яких виокремлено партійний облік, складські документи, виробництво зі специфікаціями, розбирання сировини на компоненти,

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

закупівель, продажів, а також автентифікація з рольовою авторизацією та управління довідниками.

Також було обґрунтовано вибір клієнт-серверної архітектури, яка відкриває можливість незалежно розвивати клієнтську та серверну частини та за потреби додавати нові способи доступу до системи. Такий вибір зумовив подальші рішення щодо внутрішньої будови. Систему поділено на чотири шари зі суворим напрямком залежностей. Ключові концепції предметної області оформлено як агрегати з доступом через репозиторії. Поверх цього застосовано розділення команд і запитів, центральний обробник із ланцюгом декораторів та строго типізовані ідентифікатори, що разом утворюють основу для передбачуваної та розширюваної обробки операцій.

Взаємодію між цими частинами буде побудовано на основі REST API, а внутрішню логіку організовано за принципами предметно-орієнтованим проєктуванням. Вибір технологічного стека (ASP.NET Core, PostgreSQL, Entity Framework Core та Docker Compose) обґрунтовано вимогами до цілісності операції, безпеки та відтворюваності середовища розгортання.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ ТА УПРАВЛІННЯ ВИРОБНИЦТВОМ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Оскільки програма призначена для роботи на підприємствах користувачі, які взаємодіють з системою, безпосередньо пов'язані з виробничими процесами та мають відповідні обов'язки. Для відокремлення обов'язків та кращого розуміння функціональних вимог було проілюстровано діаграму варіантів використання (рис. 2.1).

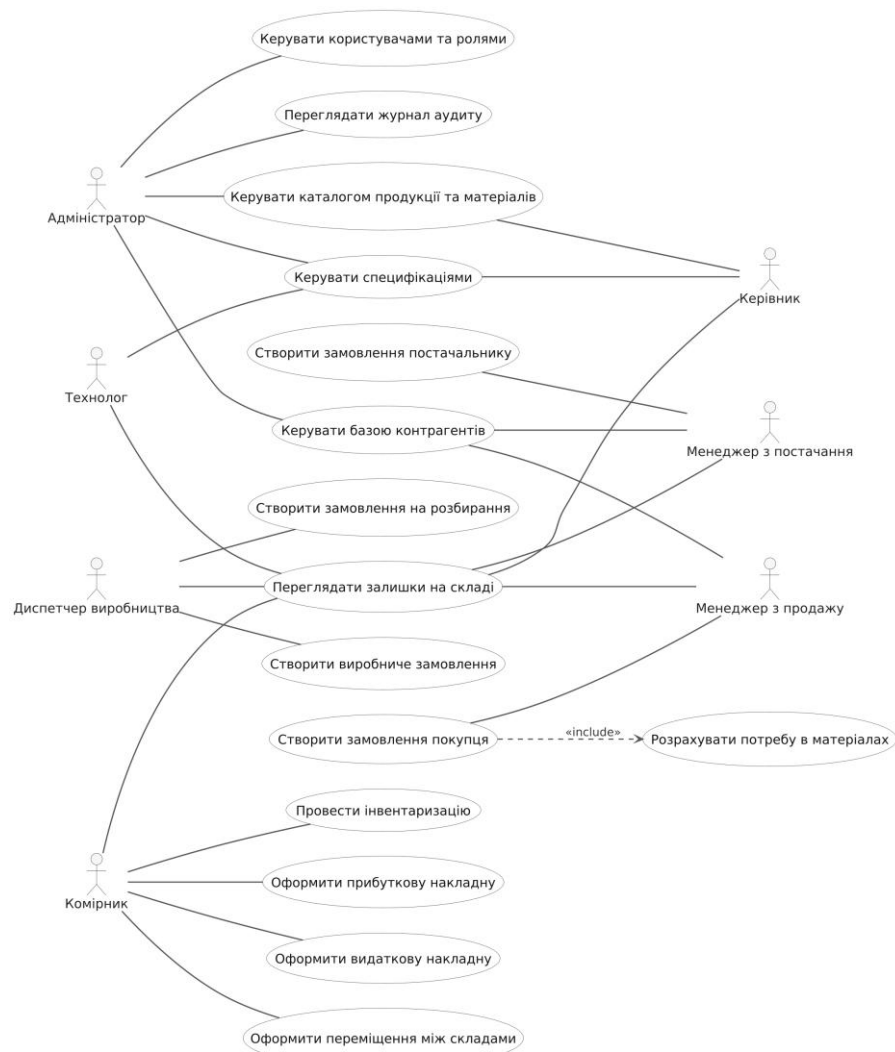


Рис. 2.1. Діаграма варіантів використання

Адміністратор є технічним наглядачем за системою. Йому доступні всі операції керування обліковими записами та ролями, перегляд журналу аудиту, а

також повний доступ до довідників. Технолог відповідає за керування специфікаціями продуктів та перегляд поточних залишків сировини для перевірки забезпеченості виробничих планів. Диспетчер виробництва створює виробничі замовлення та замовлення розбирання та контролює наявність матеріалів. Комірник є основним оператором складського документообігу. Він оформлює прибуткові та видаткові накладні, переміщення між складами та акти інвентаризації, а також переглядає поточні залишки. Менеджер з постачання формує замовлення постачальникам, веде базу контрагентів та моніторить залишки для визначення потреби в закупівлях. Менеджер з продажів створює замовлення та переглядає доступні залишки готової продукції. Керівник переглядає всю інформацію для управлінського контролю.

Для розкриття функціональної поведінки системи деталізовано чотири варіанти використання, що охоплюють найбільш критичні бізнес-процеси м'ясного виробництва. У табл. 2.1 представлено оформлення прибуткової накладної, якою комірник фіксує надходження товарів на склад.

Таблиця 2.1

Оформлення прибуткової накладної

Актор:	Комірник
Передумови:	Користувач автентифікований; постачальник та товари наявні в довідниках; склад існує
Основний потік:	1. Комірник створює новий документ, обирає постачальника та склад приймання 2. Для кожної товарної позиції комірник вказує продукт, кількість та ціну 3. За потреби позицію розбивають на декілька партій з різними датами виробництва або номерами партій постачальника 4. Комірник натискає «Провести» 5. Статус накладної змінюється на Проведено та залишки оновлюються
Альтернативні потоки:	А) Якщо на кроці 5 виявлено відсутність обов'язкових полів, система повертає повідомлення з переліком помилок. Документ залишається у статусі «Чернетка»

Наступний прецедент створення виробничого замовлення представлено в табл. 2.2. В його основі лежить списання сировини за принципом FEFO згідно зі специфікацією та формування партії готової продукції.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.2

Створення виробничого замовлення

Актор:	Диспетчер виробництва
Передумови:	Специфікація для продукту існує; диспетчер автентифікований
Основний потік:	1. Диспетчер створює виробниче замовлення, обирає продукт, планову кількість та дату запуску 2. Система розраховує потребу в сировині 3. Перевірка наявності необхідних матеріалів на складі 4. Диспетчер підтверджує запуск виробництва 5. Система автоматично підбирає партії для списання сировини 6. Система створює нову партію готової продукції
Альтернативні потоки:	А) На кроці 3 виявлено дефіцит компонентів Б) На кроці 5 диспетчер відхиляє автоматичний підбір та вручну обирає конкретні партії та склади для списання

У табл. 2.3 представлено оформлення видаткової накладної для відвантаження товару клієнту. Система підбирає партії автоматично, проте з можливістю ручного коригування.

Таблиця 2.3

Оформлення видаткової накладної

Актор:	Комірник
Передумови:	Клієнт наявний у базі контрагентів; товар є в наявності на складі
Основний потік:	1. Комірник створює видаткову накладну, обирає клієнта та склад відвантаження. Система перевіряє наявність товару на складі 2. Для кожної позиції вказує продукт та необхідну кількість 3. Комірник ініціює автоматичний підбір партій 4. Комірник переглядає підібрані партії та за потреби вносить ручні корективи для окремих позицій 5. Після підтвердження система проводить документ та оновлює залишки
Альтернативні потоки:	А) Якщо виявлено нестачу. Система повертає помилку. Документ залишається у статусі «Чернетка»

Останнім розглянутим прецедентом, який наведено у табл. 2.4, є проведення інвентаризації для звірення облікових залишків з фактичними. Система автоматично формує документи, що коригують залишки, якщо було виявлено розбіжності.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Проведення інвентаризації

Актор:	Комірник
Передумови:	Склад існує
Основний потік:	1. Комірник створює акт інвентаризації та обирає склад 2. Комірник вносить фактичні кількості за результатами фізичного перерахунку 3. Система розраховує розбіжності між обліковими та фактичними даними для кожної партії 4. Комірник підтверджує документ 5. Система формує коригуючі документи для оновлення залишків
Альтернативні потоки:	А) Якщо на кроці 4 розбіжностей не виявлено, акт все одно може бути проведений, але дочірні документи не створюються

Після огляду основних прецедентів можна детально описати функціональні вимоги системи.

Програма повинна забезпечувати автентифікацію користувачів за логіном і паролем та розмежування доступу до операцій відповідно до ролі. Для роботи з довідниковою інформацією передбачено ведення бази контрагентів, продукції із категоризацією та одиниць вимірювання. Окремою частиною довідника є звичайні специфікації та специфікації розбирання.

Зі складського модуля система повинна мати можливість оформлювати замовлення на закупівлю товарів, прибуткових накладних, видаткових накладних, внутрішніх переміщень між складами, а також актів інвентаризації по обраному складу. В кожному документі в якому передбачено оприбуткування товару на склад має бути можливість розподілити кожну позицію товару по партіях. В кожному документі де передбачено списання товару має бути можливість ручного підбору партій так і автоматичне списання по FEFO. Відображення актуальних залишків по кожному складу в реальному часі з урахуванням поточних резервацій є ще однією вимогою цього модуля.

Для виробничого модуля система має включати створення виробничих замовлень із прив'язкою до специфікації та перевіркою наявності матеріалів, списання фактичної сировини та формування партії готової продукції після

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

завершення. Аналогічні вимоги мають замовлення на розбирання, які моделюють обвалювання туші. Програма повинна мати обрахунок потреби при оформленні замовлення на продаж клієнту з інформуванням про дефіцит сировини та рекомендованих дій.

Поряд із функціональними вимогами важливою частиною є нефункціональні вимоги. Їх згруповано за категоріями та наведено у табл. 2.5.

Таблиця 2.5

Нефункціональні вимоги

Категорія	Вимоги
Продуктивність	Одночасна робота до 15 користувачів. Час розрахунку потреби в матеріалах не більше 10 секунд. Час відгуку на типові операції не більше 5 секунд. Час відгуку на складні аналітичні запити не більше 10 секунд
Надійність та доступність	Доступність системи не менше 97% загального часу. Середній час безвідмовної роботи не менше 10 робочих днів.
Безпека	Зберігання паролів у хешованому вигляді. Захист від SQL-ін'єкцій, міжсайтового скриптингу та підробки міжсайтових запитів. Розмежування доступу на рівні кінцевих точок програмного інтерфейсу. Ведення журналу аудиту операцій
Зручність використання	Час навчання звичайних користувачів не більше 3 робочих днів, адміністраторів не більше 1 робочого дня. Інтерфейс українською мовою. Відповідність сучасним принципам проектування користувацького досвіду

Також визначено вимоги до апаратного забезпечення. Мінімальні системні вимоги до серверного та клієнтського обладнання подано у табл. 2.6.

Таблиця 2.6

Апаратні обмеження

Компонент	Серверне обладнання	Робоче місце користувача
Диск	SSD від 100 ГБ	від 10–20 ГБ вільного місця
Оперативна пам'ять	Не менше 8 ГБ	Не менше 8 ГБ
Процесор	4–8 ядер з частотою від 3.0 ГГц	2 ядра, тактова частота від 2.0 ГГц
Операційна система	Linux (Ubuntu Server 20.04 і вище)	Windows, Unix, Android або iOS
Додатково	Пропускна здатність мережі від 1 Гбіт/с	Браузер Chrome, Firefox, Edge або Safari

Ядром підсистеми складського обліку є агрегат BatchStock, що інкапсулює кількісний стан конкретної партії на конкретному складі (рис. 2.2). Поточний залишок не зберігається як окреме поле бази даних, а обчислюється динамічно як різниця між надходженнями та списаннями на основі колекції об'єктів StockMovement. Кожен рух представляє атомарну зміну залишку з конкретним типом та посиланням на документ, через який було здійснено рух. Посилання на документ інкапсульовано в об'єкті SourceDocument, що в базі даних відображається на два стовпці без потреби в окремому зовнішньому ключі для кожного типу документа. Метод GetAvailableQuantity() повертає кількість вільного залишку з врахуванням резервацій.

Агрегат Batch є носієм ідентифікаційної та часової інформації про партію і зберігає три семантично різні дати: дату виробництва, дату оприбуткування на склад і системну мітку часу створення запису в базі. Дату терміну придатності інкапсульовано в окремому об'єкті ExpiryInfo, який містить логіку обчислення днів до закінчення термінів придатності.

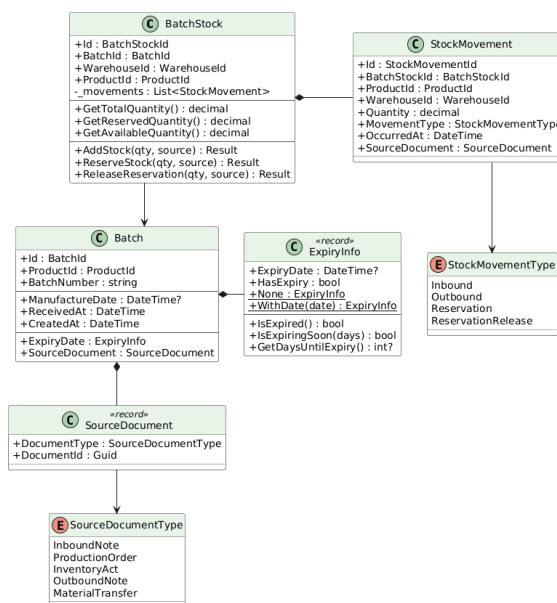


Рис. 2.2. Діаграма класів для обліку залишків по складах

Агрегат прибуткової накладної містить колекцію позицій документа, кожна з яких може бути розбита на одну або кілька партій через об'єкти IncomingBatch (рис. 2.3). Даний об'єкт є декларативним описом партії на момент введення даних і містить кількість, ціну, дату виробництва та термін придатності.

Метод Process() агрегату змінює статус документа та викликає доменну подію. Аналогічну структуру має агрегат видаткової накладної, але він містить список з об'єктами типу PickedBatch, що фіксують відібрані для відвантаження партії.

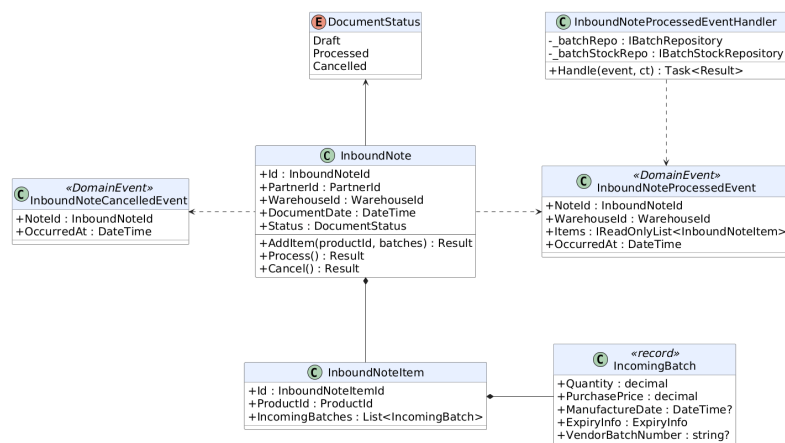


Рис. 2.3. Діаграма класів агрегату прибуткової накладної

Специфікації є описом рецепта товару від якого залежить поведінка виробничих документів (рис. 2.4). Специфікація виробництва містить поля OutputProductId та OutputQuantity, що разом визначають, який продукт і в якій базовій кількості є результатом рецептури, а також поле YieldPercentage з нормативним відсотком виходу готової продукції відносно сировини. Колекція Components зберігає інгредієнти зі своїми нормативними кількостями, а метод CalculateRequiredComponents() розраховує точну кількість кожного компонента для заданого обсягу виходу з урахуванням цього відсотка.

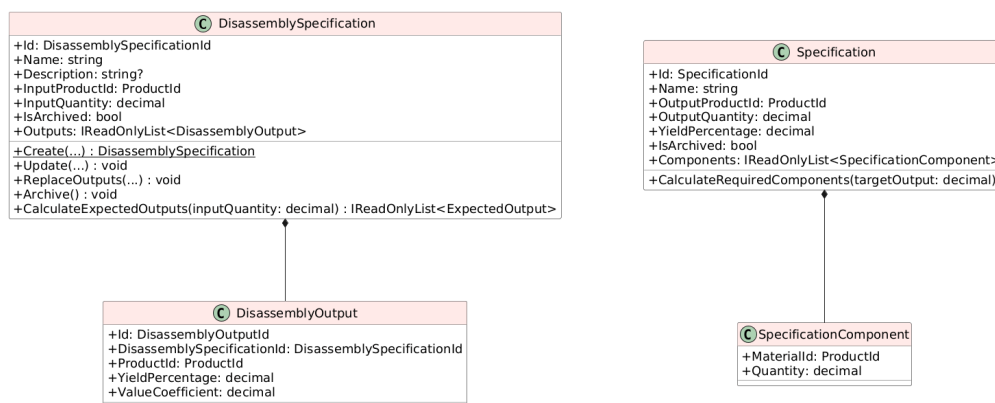


Рис. 2.4. Діаграма класів двох типів специфікацій

Специфікація для розбирання описує зворотний процес в якому з одного вхідного продукту отримують перелік виходів, кожен з яких має два нормативні поля. Поля InputProductId та InputQuantity описують вхідну сировину і базову

розрахункову кількість, а колекція Outputs перелічує продукти, що утворюються в результаті. Об'єкти цього списку мають поле YieldPercentage, яке задає очікувану частку вхідної маси, що переходить у цей відруб. Сума всіх коефіцієнтів може бути меншою за одиницю, оскільки решта є виробничими втратами. Друге поле коефіцієнт цінності, ValueCoefficient, яке визначає відносну вартість виходу і використовується для пропорційного розподілу сукупної собівартості сировини між вихідними компонентами.

Виробниче замовлення проходить через чотири стадії: «Чернетка», «Зарезервовано», «У виробництві» та «Завершено» (рис. 2.5). Ці статуси відображені у полі Status. За допомогою статичного методу Create() екземпляр класу створюється в статусі «Чернетка» та автоматично копіює дані інгредієнтів зі специфікації у поле Items. У полях PlannedOutputQuantity та ActualOutputQuantity зберігаються відповідно планова і фактична кількість готового продукту. Метод Reserve() перевіряє забезпеченість сировиною і блокує відповідні залишки через резервації. Після цього список з назвою Items буде містити заповнену добірку партій. Під час того коли документ у статусі «У виробництві» метод UpdateActualOutput() дозволяє коригувати фактичний вихід. Метод Complete() завершує документ, списує сировину, формує вихідну партію з полями OutputBatchNumber, OutputManufactureDate та OutputExpiryDate і записує в поле OutputUnitCost обчислену собівартість одиниці як суму матеріальних витрат і AdditionalCosts, поділену на фактичний вихід. Обчислювальна властивість Yield повертає відношення фактичного виходу до планового і стає доступною після завершення.

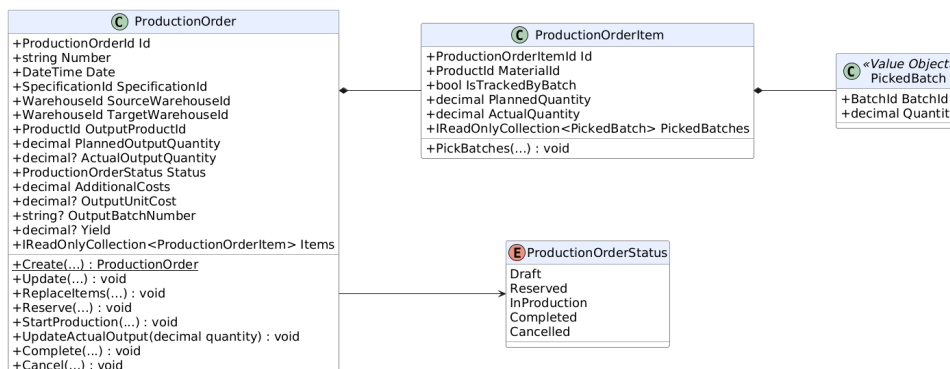


Рис. 2.5. Діаграма класів виробничого замовлення

Замовлення розбирання відзначається окремою логікою розподілу собівартості (рис. 2.6). Поля InputProductId та PlannedInputQuantity описують вхідну сировину, а список InputPicks містить фактично підібрані партії. Метод SetInputPicks() дозволяє обрати на розбирання декілька партій і перевіряє, щоб сума кількостей у підборі дорівнювала плановому входу. Метод Reserve() автоматично заповнює InputPicks за принципом FEFO. Список Outputs зберігає планові та фактичні кількості виходів, а також перевизначену собівартість. Метод UpdateActuals() використовується для коригування фактичних показників, а Complete() запускає розрахунок собівартості через клас DisassemblyCostAllocator.

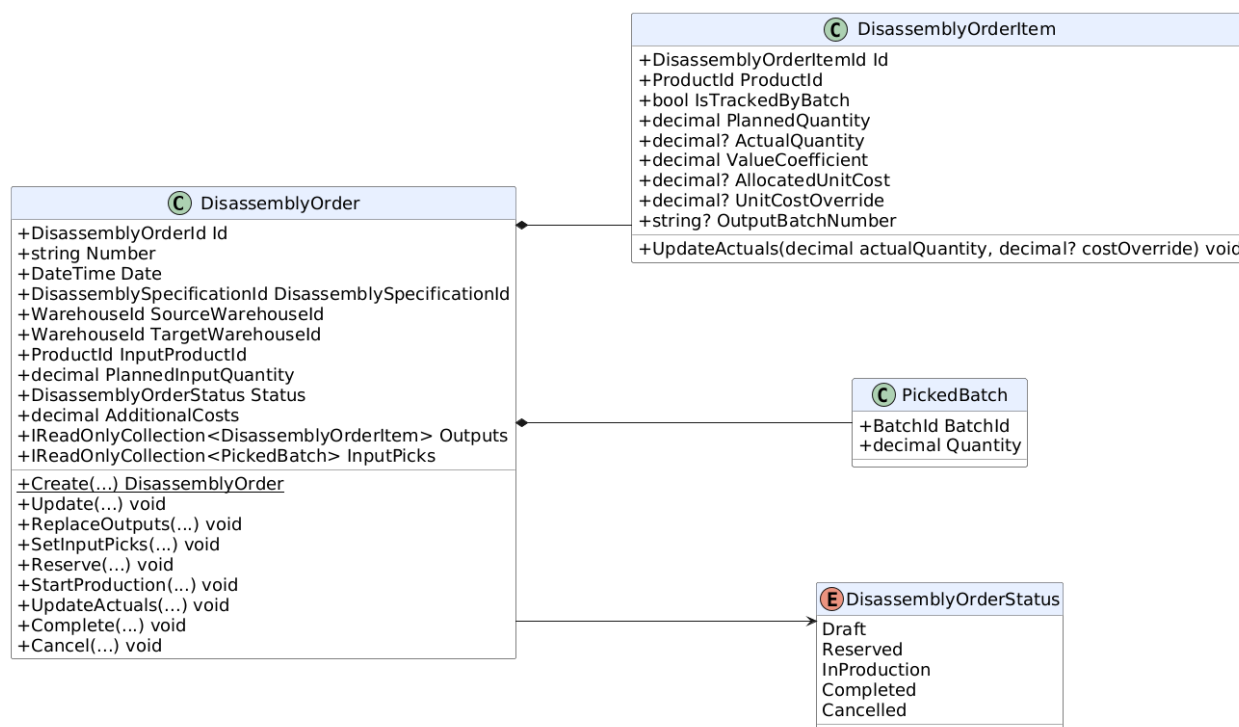


Рис. 2.6. Діаграма класів замовлення на розбирання

Центральною концепцією шару застосунку є поєднання архітектурного патерну розділення команд і запитів із ланцюгом обробників, який побудований за допомогою патерну «Декоратор» (рис. 2.7). Усі операції системи поділяються на команди, які змінюють стан і запити, які лише читають дані. Команди проходять через агрегати та репозиторії дотримуючись усіх бізнес-правил. Запити натомість звертаються безпосередньо до контексту бази даних для побудови агрегаційних виразів.

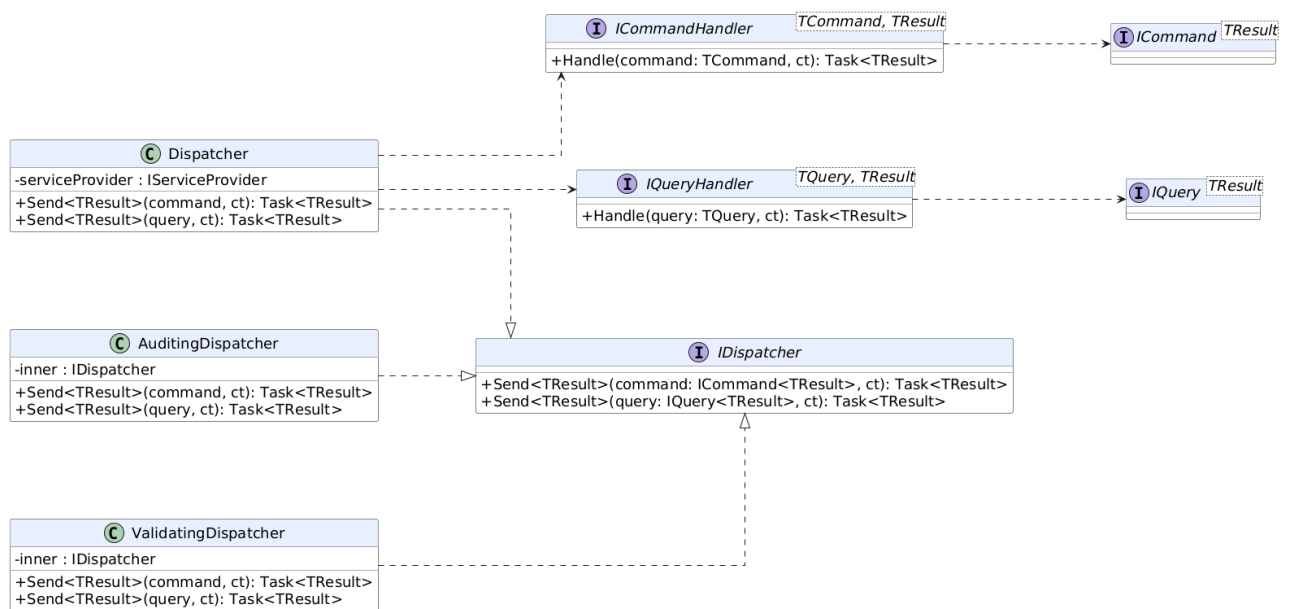


Рис. 2.7. Діаграма класів реалізації CQRS та ланцюг декораторів

Наведена діаграма показує, як описаний поділ команд і запитів реалізовано на рівні конкретних класів. Наприклад, команда `ProcessInboundNoteCommand` проходить через репозиторії та агрегати, які забезпечують дотримання усіх доменних правил. А запит `GetStockSummaryQuery` звертається напряму до `IApplicationDbContext` для агрегації та фільтрації. Інтерфейси `ICommandHandler` та `IQueryHandler` оголошені з generic-обмеженням, що гарантує типову узгодженість між командою та її обробником (рис. 2.8).

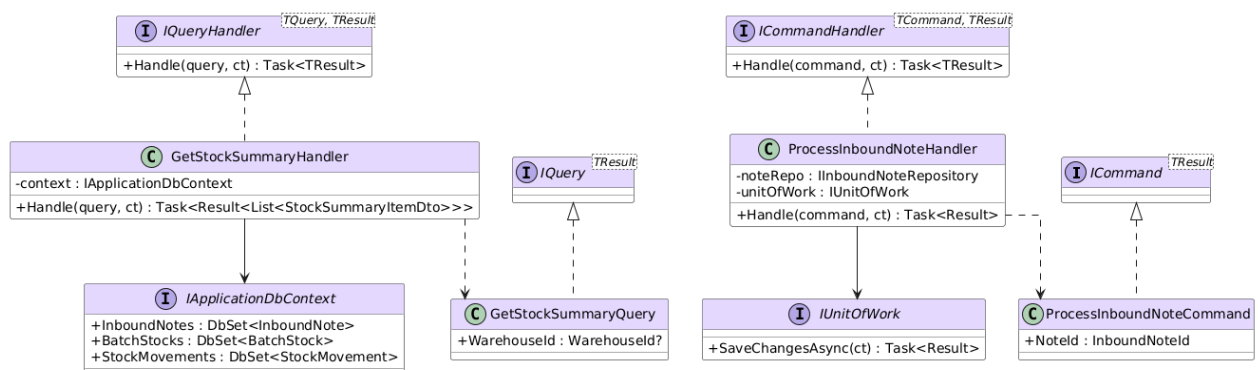


Рис. 2.8. Діаграма класів, які реалізують CQRS на практиці

Для ізоляції побічних ефектів від основної бізнес-логіки застосовано патерн доменних подій. Замість того щоб модель безпосередньо викликала код для роботи з базою даних, документи генерують події типу `IDomainEvent`. Обробники подій типу `IDomainEventHandler` виконують необхідні операції з залишками на складах.

Важливим обмеженням є заборона на виклик збереження змін усередині обробника події, оскільки транзакцією керує команда, яка гарантує атомарність усіх змін стану за одну операцію (рис. 2.9).

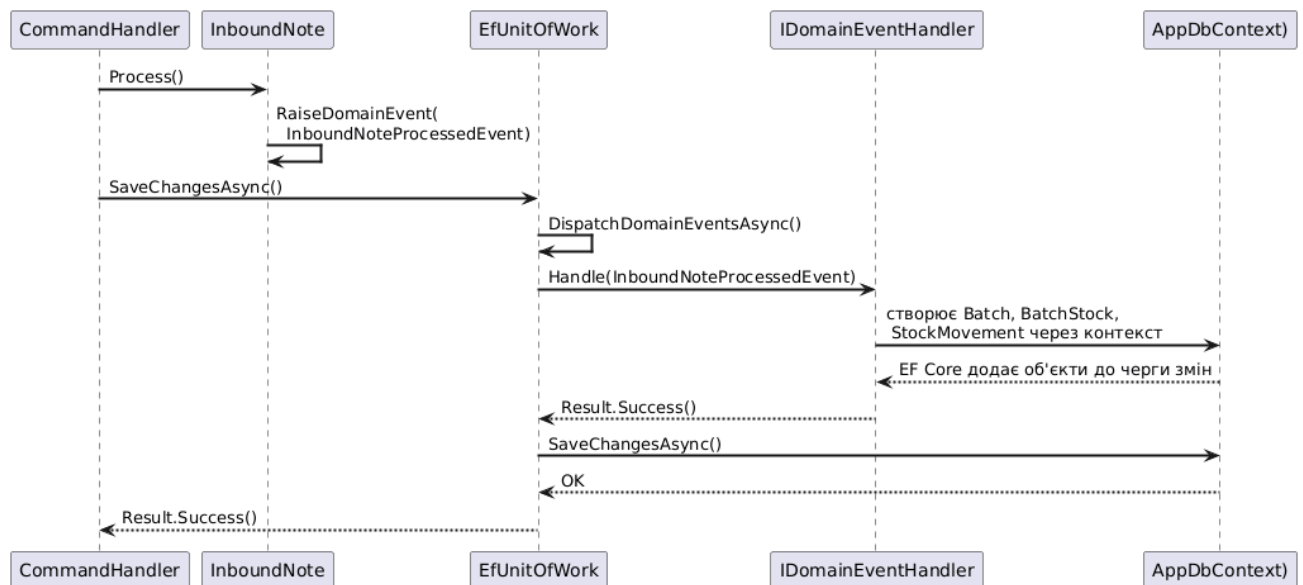


Рис. 2.9. Діаграма послідовності виклику подій та збереження даних в БД

2.2 Розробка бази даних

Структура бази даних системи формується автоматично за допомогою підходу Code First із застосуванням механізму міграцій Entity Framework Core. Суть цього підходу полягає в тому, що схема бази даних є похідною від доменної моделі, яка описана у програмному коді. Спочатку у вигляді класів та конфігурацій описуються сутності, їхні властивості й зв'язки між ними, а вже потім ORM генерує міграційні файли, які приводять реальну схему до відповідності з моделлю. Завдяки цьому доменна модель залишається єдиним джерелом істини, тоді як інфраструктурний шар лише відображає її на реляційну схему. Додатковою перевагою такого рішення є відтворюваність схеми, адже послідовність міграцій дозволяє розгорнути ідентичну базу на будь-якому середовищі. Для візуального представлення структури даних нижче наведено ER-діаграму системи (рис. 2.10). Окремо варто зазначити, що кожен міграційний файл фіксує лише різницю між поточною моделлю та її попереднім станом, тому схема розвивається поступово без написання SQL-скриптів вручну.

Структура індексів у БД має на меті максимізувати ефективність запитів агрегування залишків та пошуку за датою. Найважливішим є індекс у таблиці `stock_movements` для полів `product_id`, `warehouse_id`, `occurred_at` з включеними стовпцями `quantity` та `movement_type`. Такий покривальний індекс дозволяє базі даних виконувати агрегаційні запити для розрахунку поточних залишків без звернення до основних сторінок таблиці. Для таблиці `batches` створено індекс за полем `expiry_date` для ефективної вибірки партій з термінами, що спливають.

Часткові індекси використовуються для випадків коли запити звертаються тільки до підмножини рядків. Наприклад індекс для таблиці прибуткових накладних, який покриває записи зі статусом чернетки. Такий індекс покриває типовий сценарій пошуку накладних, що очікують на проведення без сканування всього масиву документів з всіма типами статусів.

Довідкові записи в системі ніколи не видаляються фізично. Замість видалення застосовано архівування, тобто кожен довідник несе ознаку `is_archived` та момент архівування `archived_at`. Архівований запис зберігається в базі заради цілісності історичних документів, проте перестає з'являтися в операційних списках та селекторах, бо відповідні запити явно додають умову на активність запису.

Структура індексів спроектована так, щоб пришвидшити найчастіші класи запитів, а саме агрегування залишків, пошук документів за станом та контроль унікальності. Окремий клас утворюють унікальні індекси, які боронять цілісність ключових ідентифікаторів предметної області. Зокрема номер кожного документа є унікальним у межах свого типу, тому база відхиляє спробу вставити повторний запис ще на рівні схеми бази даних, не покладаючись лише на код застосунку. Така перевірка водночас коректно розв'язує конкурентні вставки, коли два звернення намагаються створити однаковий запис одночасно.

Загалом база даних складається з понад двадцяти таблиць, тому для детальнішого огляду відібрано десять ключових, що несуть основне навантаження предметної області.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

Першою розглянемо таблицю products, яка зберігає всю номенклатуру в єдиному довіднику (табл. 2.7). Товар описується таким набором стовпців як зовнішні ключі категорій та одиниць вимірювання, поле, яке визначає режим обліку, поле кількості днів придатності, прапорець, який позначає архівування товару та дата, коли це відбулося.

Таблиця 2.7

Опис таблиці “products”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор товару
code	varchar(50)	Артикул, тобто унікальний код номенклатури
name	varchar(200)	Найменування товару
category_id	uuid	Посилання на категорію номенклатури
measure_id	uuid	Посилання на одиницю вимірювання
days_to_expire	integer	Термін придатності у днях для партійного обліку
is_tracked_by_batch	boolean	Ознака обліку за партіями
is_archived	boolean	Ознака архівування
archived_at	timestampz	Момент архівування
created_at	timestampz	Момент створення запису

Наступною є таблиця warehouses, яка описує фізичні місця зберігання запасів (табл. 2.8). З основних стовпців містить поля назви та адресу розташування.

Таблиця 2.8

Опис таблиці “warehouses”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор складу
name	varchar(200)	Назва складу
location	varchar(500)	Адреса розташування
is_archived	boolean	Ознака архівування
archived_at	timestampz	Момент архівування
created_at	timestampz	Момент створення запису

Таблиця partners об'єднує контрагентів обох типів, тобто постачальників та покупців, в одному довіднику (табл. 2.9). Як і решта довідників, партнер архівується замість видалення.

Таблиця 2.9

Опис таблиці “partners”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор контрагента
name	varchar(200)	Назва (ПІБ) контрагента
tax_code	varchar(50)	Податковий код
contact_phone	varchar(50)	Контактний телефон
notes	varchar(1000)	Довільні примітки
is_archived	boolean	Ознака архівування
archived_at	timestamptz	Момент архівування
created_at	timestamptz	Момент створення запису

Таблиця batches описує окрему партію, яка фіксує конкретну партію товару з її собівартістю, датами та номером, тому вона є носієм вартісної оцінки запасів (табл. 2.10). Пара стовпців source_document_type та source_document_id, як зазначалося раніше вказує, який саме документ створив партію. Також таблиця має прапорець, який позначає системну партію, яка створюється одна на один на склад для одного типу товару, якщо значення дорівнює true.

Таблиця 2.10

Опис таблиці “batches”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор партії
product_id	uuid	Зовнішній ключ на продукт
batch_number	varchar(100)	Номер партії
manufacture_date	timestamp	Дата виготовлення
received_at	timestamp	Дата надходження на склад
expiry_date	timestamp	Термін придатності
source_document_type	varchar(50)	Тип документа-джерела
source_document_id	uuid	Ідентифікатор документа, який породив партію
created_at	timestamp	Системна мітка створення запису

unit_cost	numeric(18,2)	Собівартість одиниці
is_system_batch	boolean	Ознака системної партії для обліку за товаром

Усі фактичні зміни запасів фіксуються в журналі рухів, що зберігається в таблиці stock_movements (табл. 2.11). Поточні залишки система не зберігає окремим числом, а обчислює як суму рухів за відповідним розрізом товару, складу й партії. Поле quantity є знаковим, тому надходження та зняття резерву записуються додатними значеннями, а списання й резервування від'ємними, а їхній тип фіксує стовпець movement_type, який набуває значень надходження, списання, резервування та зняття резерву. Завдяки полям source_document_type і source_document_id кожен рух прив'язаний до документа, що його спричинив.

Таблиця 2.11

Опис таблиці “stock_movements”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор руху
batch_stock_id	uuid	Зовнішній ключ на запис залишку партії
batch_id	uuid	Посилання на партію
product_id	uuid	Зовнішній ключ на продукт
warehouse_id	uuid	Зовнішній ключ на склад
movement_type	varchar(50)	Тип руху
quantity	numeric(18,3)	Кількість (від'ємна для списання)
occurred_at	timestamp	Дата здійснення руху
source_document_type	varchar(50)	Тип документа-джерела руху
source_document_id	uuid	Ідентифікатор документа-джерела руху

Надходження сировини й товарів оформлюється прибутковою накладною, яка зберігається в таблиці inbound_notes (табл. 2.12). Має поля позначення статусу документа, необов'язкове посилання на замовлення закупівлі, посилання на користувача, який провів, скасував та створив документ з відповідними полями дат виконання цих дій.

Таблиця 2.12

Опис таблиці “inbound_notes”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор накладної
number	varchar(50)	Номер документа
warehouse_id	uuid	Склад оприбуткування
supplier_id	uuid	Постачальник
purchase_order_id	uuid	Посилання на замовлення на закупівлю
status	varchar(30)	Статус документа
document_date	timestampz	Дата документа
created_by	uuid	Користувач, який створив документ
processed_at	timestampz	Момент проведення
cancelled_at	timestampz	Момент скасування
created_at	timestampz	Момент створення запису

Дзеркальним до надходження є відвантаження, яке оформлюється видатковою накладною в таблиці outbound_notes (табл. 2.13). Поле sales_order_id за аналогією з прибутковою накладною описує зв'язок виконання із замовленням на продаж.

Таблиця 2.13

Опис таблиці “outbound_notes”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор накладної
number	varchar(50)	Номер документа
warehouse_id	uuid	Склад відвантаження
receiver_id	uuid	Покупець
sales_order_id	uuid	Посилання на замовлення на продаж
status	varchar(30)	Статус документа
document_date	timestampz	Дата документа
created_by	uuid	Користувач, який створив документ
processed_at	timestampz	Момент проведення
cancelled_at	timestampz	Момент скасування
created_at	timestampz	Момент створення запису

Таблиця specifications зберігає в собі записи рецептур, які використовуються у виробничих замовленнях (табл. 2.14). Поля output_quantity та yield_percentage разом визначають нормативний вихід, тобто базовий обсяг виходу готового товару й частку придатного продукту з урахуванням технологічних втрат. Текстове поле description слугує для внесення технологічних приміток.

Таблиця 2.14

Опис таблиці “specifications”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор рецептури
name	varchar(200)	Назва рецептури
description	text	Опис рецептури
output_product_id	uuid	Готовий продукт на виході
output_quantity	numeric(18,4)	Базовий обсяг випуску
yield_percentage	numeric(5,4)	Коефіцієнт виходу як частка від нуля до одиниці
is_archived	boolean	Ознака архівування
created_by_id	uuid	Автор рецептури
created_at	timestampz	Момент створення запису

Виробничі замовлення описуються таблицею production_orders (табл. 2.15). Має поля статусу виробництва, поля, які фіксують користувача, який змінив статус та дату, номер, дату та собівартість готової партії, які заповнюються при завершенні виробництва, поля складів з яких списується сировина та оприбутковується готова продукція та посилання на замовлення продажу товару.

Таблиця 2.15

Опис таблиці “production_orders”

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор замовлення
number	varchar(50)	Номер документа
date	timestampz	Дата документа
specification_id	uuid	Рецептура по якій виконується виробництво

source_warehouse_id	uuid	Склад сировини
target_warehouse_id	uuid	Склад готової продукції
output_product_id	uuid	Готовий продукт
planned_output_quantity	numeric(18,3)	Планова кількість виготовлення
actual_output_quantity	numeric(18,3)	Фактична кількість виготовлення
additional_costs	numeric(18,2)	Додаткові витрати
output_unit_cost	numeric(18,4)	Собівартість одиниці виготовленого продукту
status	varchar(30)	Стадія виробничого циклу
created_at	timestampz	Момент створення запису

Таблиця sales_orders зберігає замовлення покупців (табл. 2.16). Містить такі поля як статус замовлення, номер документа, необов'язкові примітки до документа, склад відвантаження, посилання на контрагента, дата по яку потрібно відвантажити товар.

Таблиця 2.16

Опис таблиці "sales_orders"

Назва стовпця	Тип даних	Опис
id	uuid	Унікальний ідентифікатор замовлення
number	varchar(50)	Номер документа
order_date	timestampz	Дата замовлення
customer_id	uuid	Покупець
warehouse_id	uuid	Склад відвантаження
due_date	timestampz	Бажана дата виконання
status	varchar(30)	Статус замовлення
notes	varchar(1000)	Примітки
created_by	uuid	Користувач, який створив документ
created_at	timestampz	Момент створення запису

Отже, спроектована база даних відображає систему обліку, у якій номенклатура, склади та контрагенти працюють як довідникова інформація, документи фіксують господарські операції, а незмінний журнал рухів запасів виступає єдиним джерелом істини про залишки й собівартість. Застосування

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

підходу Code First з винесеними конфігураціями забезпечило чисту доменну модель і відтворювану схему. Система індексів і архівування замість видалення дозволила поєднати швидкодію типових запитів зі збереженням повного аудиторського сліду. Тому структура є достатньо гнучкою для подальшого розширення новими типами документів без зміни ядра обліку.

2.3 Проєктування та реалізація алгоритмів роботи системи

Центральним для всієї системи є алгоритм проведення складського документа, оскільки саме навколо нього будується решта обробки. Кожен документ проходить життєвий цикл від чернетки до проведеного стану, і лише в момент проведення він впливає на залишки. Коли користувач натискає кнопку проведення, система спершу перевіряє повноту й коректність даних, тобто наявність обов'язкових полів, відповідність кількостей і доступність складу. Якщо хоча б одна умова не виконується, операція припиняється, документ залишається чернеткою. У разі успішної перевірки статус документа змінюється на проведений, після чого агрегат не звертається до бази даних напряму, а лише породжує доменну подію, яка сповіщає про те, що сталося.

Така побудова ізолює побічні ефекти від основної бізнес-логіки. Замість того щоб документ сам формував записи рухів, цю роботу виконують окремі обробники доменних подій, які реагують на факт проведення. Ключовим тут є порядок виконання, адже спершу команда зберігає зміну стану документа через єдину транзакцію, і вже після цього центральний обробник передає накопичені події іншим обробникам.

У випадку з прибутковою накладною система створює нові партії та записує надходження з додатними значеннями кількості. Тоді як обробник видаткової операції формує рухи списання з від'ємними значеннями.

З цього загального алгоритму впливає логіка підбору партій для списання. Для видаткових операцій, переміщень, актів списання та споживання сировини у виробництві постає питання, з яких саме партій забирати товар. Система пропонує два шляхи розв'язання цієї задачі. Користувач може вручну вказати

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

потрібні партії ще у чернетці, і тоді алгоритм лише перевіряє, що сума підібраних кількостей дорівнює потребі позиції. Якщо ж підбір не задано, система виконує його автоматично за принципом «першим завершується термін, першим відвантажується», який скорочено позначають як FEFO (рис. 2.11).

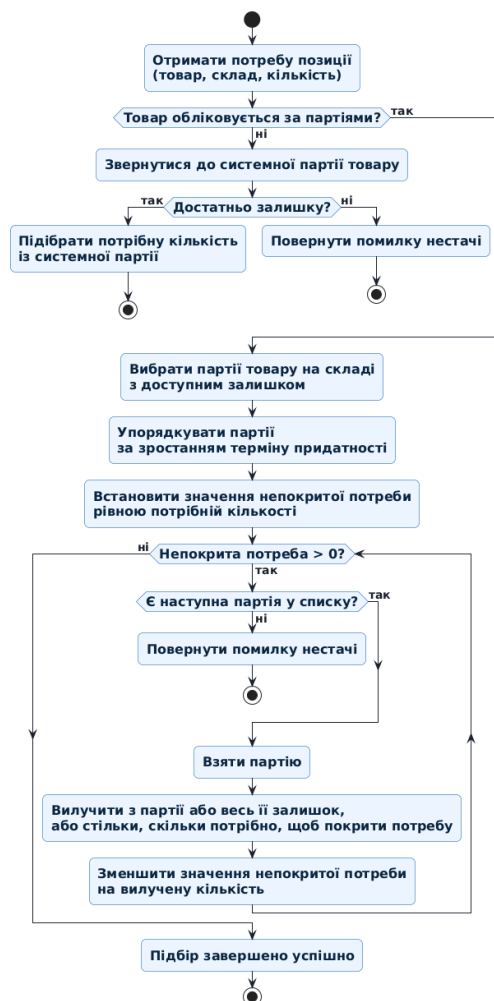


Рис. 2.11. Діаграма діяльності алгоритму автоматичного підбору партій за принципом FEFO

Спочатку система визначає всі партії потрібного товару на заданому складі, які мають доступний залишок, і впорядковує їх за зростанням дати завершення терміну придатності, а якщо партії однакових термінів, то за датою виготовлення. Системні партії, які використовуються для обліку товарів без партійного поділу, з цього переліку виключаються. Далі алгоритм рухається списком від найближчої до прострочення партії й послідовно списує з них їхні залишки поки лічильник потреби у товарі не буде дорівнювати нулю. Якщо ж перелік партій вичерпується раніше, ніж задоволено всю потребу, алгоритм

повертає помилку нестачі, а документ лишається непроведеним. Для товарів, що обліковуються на рівні номенклатури без партій, цей перебір не потрібен, оскільки весь їхній залишок зосереджено в єдиній системній партії, до якої звертаються напрямку.

Як вже відомо, система обчислює поточну кількість залишку продукції на складі як підсумок всіх складських операцій. Фактичний залишок на складі дорівнює різниці між сумою надходжень і сумою списань за відповідним розрізом товару, датою, складом та партій. Однак для прийняття рішення про відвантаження важливий не фізичний, а вільний залишок. Тому перед відвантаженням система додатково враховує кількість товару яка ще не зайнята іншими замовленнями, тобто обчислює різницю між сумою резервувань і сумою їх скасування. Саме на цьому алгоритмі спирається перевірка забезпеченості перед проведенням.

Алгоритм виконання виробничого замовлення проходить чотири стани. На стадії чернетки замовлення лише копіює зі специфікації перелік потрібних компонентів, перераховуючи їхні кількості під плановий обсяг випуску з урахуванням відсотка виходу. Коли користувач переводить замовлення у стан резервування, система перевіряє наявність сировини й закріплює її через рухи резервування, причому конкретні партії або підбираються вручну, або визначаються алгоритмом FEFO. Початок виробництва змінює стан і сигналізує, що робота триває. У момент завершення система перетворює резервування на фактичне списання сировини, підсумовує її собівартість як суму добутків підібраних кількостей на собівартість відповідних партій, додає до неї накладні витрати й ділить результат на фактичний обсяг випуску і в кінці отримуємо собівартість одиниці готового продукту. Скасування замовлення на будь-якій з проміжних стадій знімає раніше встановлені резервації та повертає сировину назад.

Зворотним до виробництва є алгоритм розбирання, який має власну логіку розподілу вартості. Тут одна вхідна партія сировини розкладається на кілька виходів, а сукупна собівартість входу має бути поділена між ними. Оскільки

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

вихідні компоненти мають різну цінність, розподіл виконується пропорційно добутку фактичної кількості на коефіцієнт цінності кожного виходу. Отриманий результат система пропонує як підказку, проте залишає за користувачем право перевизначити собівартість конкретного виходу вручну.

Алгоритм аналізу потреби в матеріалах працює під час оформлення замовлення на продаж і відповідає на питання, чого і скільки бракує для його виконання та що з цим робити (рис. 2.12). Спершу система розкладає готові товари замовлення на складники на основі їхніх специфікацій і обчислює потрібну кількість кожного компонента. Далі для кожного компонента визначається підсумкова забезпеченість, причому до уваги беруться не лише власні залишки на складах, а й товар у дорозі, який виводиться із незакритих замовлень на закупівлю. Окремо враховується попит інших підтверджених замовлень на продаж, який віднімається від доступної кількості, аби та сама сировина не вважалася покритою одразу для кількох замовлень. Дефіцит обчислюється як перевищення сукупної потреби над сумою власного залишку й товару в дорозі. Залежно від того, чи існує для дефіцитного компонента звичайна специфікація, специфікація розбирання або жодної з них, система рекомендує відповідні дії.

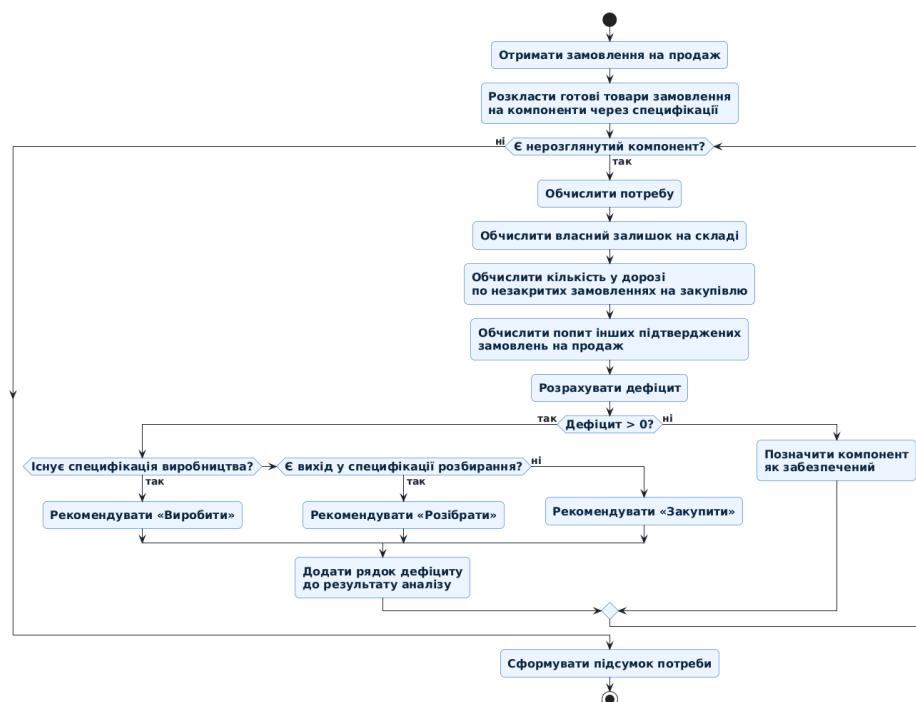


Рис. 2.12. Діаграма діяльності алгоритму аналізу потреби в матеріалах

Таким чином, описані алгоритми утворюють узгоджену поведінкову модель системи, у якій загальний механізм проведення документів через доменні події слугує спільною основою, підбір партій за принципом FEFO та обчислення вільного залишку забезпечують коректну роботу зі складом, а спеціалізовані алгоритми виробництва, розбирання й аналізу потреби розв'язують виробничі задачі.

2.4 Реалізація функціоналу системи складського обліку та автоматизації виробництва крафтових м'ясних виробів

Розроблене програмне забезпечення реалізовано у вигляді клієнт-серверного застосунку, де серверна частина побудована на платформі ASP.NET Core, а клієнтська як односторінковий застосунок на основі бібліотеки React з використанням мови програмування TypeScript. Взаємодія здійснюється через REST API, а збереження даних забезпечує PostgreSQL з використанням інструменту об'єктно-реляційного відображення Entity Framework Core.

Усі команди й запити проходять через центральний обробник. Даний обробник автоматично знаходить зареєстрований сервіс і викликає його, що значно автоматизує процес написання коду. Складність цієї автоматизації в тому, що на момент виклику команда відома обробнику лише через узагальнений інтерфейс `ICommand<TResult>`, тоді як для її виконання необхідно отримати з контейнера залежностей конкретний типізований обробник `ICommandHandler<TCommand, TResult>`. Оскільки фактичний тип команди стає відомим лише під час виконання, клас `Dispatcher` застосовує механізм рефлексії, за допомогою якої він методом `MakeGenericType` конструює закритий узагальнений тип обгортки під конкретну пару «тип команди – тип результату» та створює її екземпляр. Усередині обгортки параметри типів уже визначені, що дає змогу звернутися до контейнера залежностей і викликати обробник у звичайний типізований спосіб. Створені обгортки кешуються в словнику, тому операції рефлексії виконуються лише один раз для кожного типу команди, а всі подальші виклики зводяться до віртуального виклику методу (Додаток А).

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Сам об'єкт типу Dispatcher є лише ядром ланцюга, навколо якого нашаровано додаткову поведінку за патерном «Декоратор». Складання цього ланцюга відбувається в єдиному місці під час налаштування залежностей. Спочатку береться ядро, далі його обгортає диспетчер валідації, а зовні розташовується диспетчер аудиту, тому кожен запит проходить шари у зворотному порядку від зовнішнього до внутрішнього:

```
services.AddScoped<IDispatcher>(sp =>
{
    IDispatcher pipeline = sp.GetRequiredService<Dispatcher>();
    pipeline = new ValidatingDispatcher(pipeline,
    sp.GetRequiredService<CommandValidatorResolver>());
    pipeline = new AuditingDispatcher(
        pipeline,
        sp.GetRequiredService<IAuditNumberResolver>(),
        sp.GetRequiredService<ICurrentUser>(),
        sp.GetRequiredService<IAuditLogWriter>(),
        sp.GetRequiredService<ILogger<AuditingDispatcher>>());
    return pipeline;
});
```

Окремої уваги заслуговує механізм впливу команд на склад, де доменні події поєднано з керуванням транзакцією, яке реалізовано у класі EfUnitOfWork (Додаток А). Метод DispatchDomainEventsAsync послідовно передає доменні події обробникам, які створюють рухи запасів, при цьому після кожного проходження повторно перевіряє контекст на наявність нових подій, оскільки обробка однієї події може породити наступні. Якщо будь-який обробник повертає помилку, виконання припиняється. Лише після успішної обробки всіх подій викликається єдиний SaveChangesAsync, який гарантує узгодженість даних.

Поряд з обробкою команд є ще один наскрізний механізм, а саме перевірка передумов перед архівацією довідників. Метод EvaluateAsync в класі BusinessRuleEvaluator отримує контекст і перебирає всі зареєстровані правила з перевіркою, чи порушено умову, і на першому порушенні повертає невдалий результат з кодом цього правила (Додаток А). Наприклад, ProductHasStockRule забороняє архівувати товар з залишком, при цьому сам клас знає лише власний код помилки та делегує перевірку класу читання. Кожне правило реалізоване як незалежний клас, який отримується з контейнера залежностей, тому додавання

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

нової причини для заборони архівації потребує лише написання одного нового класу без змін у класі `BusinessRuleEvaluator` чи обробнику команди.

Головним аспектом системи є облікова логіка, а якщо конкретніше, то реалізація автоматичного підбору партій за принципом FEFO. Логіка визначення з яких саме партій списувати товар, скільки списати та чи взагалі вистачає товару зосереджена у класі `StockAvailabilityChecker` (Додаток А). Метод `AllocateForConsumptionAsync` бере зі сховища всі партії товару на складі разом з їхніми залишками. Далі відсіює ті, у яких сплив термін придатності, і впорядковує решту за зростанням дати завершення терміну, а коли терміни збігаються, додатково за датою виготовлення, тож першими в черзі опиняються партії, що псуються найшвидше. Для кожної партії він одразу обчислює доступний залишок станом на дату документа методом `GetAvailableQuantity`.

Клас `SystemBatchService` реалізує облік товарів без партій через єдину системну партію на товар (Додаток А). Метод `GetOrCreateAsync` шукає системну партію товару у сховищі, а коли її ще немає, створює і намагається зберегти. Метод `ApplyInboundAsync` оприбутковує надходження з перерахунком середньозваженої собівартості. Метод `ResolveForOutboundAsync` повертає партію для списання.

Другим обчислювальним вузлом облікової логіки є розподіл собівартості під час розбирання. Клас `DisassemblyCostAllocator` відповідає за рознесення вартості однієї вхідної партії сировини між кількома відрубами (Додаток А). Він спершу рахує сумарну вагу всіх виходів, де вага кожного дорівнює добутку фактичної кількості на коефіцієнт цінності. Потім проходить виходи по черзі й кожному виділяє частку загальної вартості пропорційно його вазі, округлюючи до копійок, а наприкінці ділить цю частку на кількість виходу й дістає собівартість за одиницю.

Отже, реалізовані механізми демонструють, що архітектурні принципи системи забезпечують стійкість роботи системи та автоматизований підхід до написання нового коду. Центральний обробник із рефлексією та ланцюгом декораторів забезпечує однакову обробку всіх операцій, доменні події

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

гарантують атомарність змін стану, а обробник правил та спеціалізовані алгоритми підбору за FEFO й розподілу собівартості реалізують облікову логіку без зайвої складності.

Доменна модель використовує строго типізовані ідентифікатори замість простих значень, тому для них задано перетворювач, який під час запису згортає ідентифікатор у звичайне значення бази, а під час читання відновлює назад. Крім того, в конфігурації рухів запасів StockMovementConfiguration посилання на документ відображено парою стовпців прямо в рядку руху через механізм складеної властивості:

```
builder.Property(m => m.ProductId)
    .HasConversion(
        id => id.Value,
        value => new ProductId(value))
    .IsRequired();
builder.ComplexProperty(m => m.SourceDocument, sd =>
{
    sd.Property(d => d.DocumentType)
        .HasConversion<string>()
        .HasMaxLength(50)
        .HasColumnName("source_document_type")
        .IsRequired();

    sd.Property(d => d.DocumentId)
        .HasColumnName("source_document_id")
        .IsRequired();
});
```

За допомогою перетворювача значень ідентифікатор лишається типобезпечним у коді, але в базі зберігається як звичайний унікальний ключ, а складена властивість дає змогу прив'язати будь-який рух до документа будь-якого типу без окремого зовнішнього ключа на кожен тип документа. У цій самій конфігурації оголошено й покривний індекс за товаром і складом із включеними кількістю та типом руху, що пришвидшує обчислення залишків.

Висновки до другого розділу

В межах розділу було побудовано діаграму варіантів використання і деталізовано основні прецеденти складського та виробничого документообігу. З них виведено повний перелік функціональних і нефункціональних вимог та апаратні обмеження системи.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрему увагу приділено об'єктній структурі системи. Через діаграми класів описано агрегати обліку залишків, документів, специфікацій та виробничих замовлень.

Структура бази даних сформована за підходом Code First. Завдяки цьому доменна модель лишилася чистою, а схема відтворюваною на будь-якому середовищі. Систему індексів спроектовано під найчастіші запити агрегування залишків і пошуку за станом документів. Замість фізичного видалення довідників застосовано архівування для збереження цілісності історії.

Також спроектовано та реалізовано алгоритми, які наповнюють описану структуру конкретною поведінкою. Центральним серед них є проведення документа через доменні події, навколо якого побудовано підбір партій за принципом найближчого терміну придатності, обчислення вільного залишку, алгоритми виробництва, розбирання з розподілом собівартості за коефіцієнтом цінності та аналізу потреби в матеріалах. Усі ці механізми втілено в коді засобами платформи ASP.NET Core, бібліотеки React та системи керування базами даних PostgreSQL.

Таким чином, результатом розділу є цілісний проєкт системи, у якому архітектурні рішення, об'єктна модель, структура даних та алгоритми узгоджені між собою й підпорядковані єдиній обліковій логіці.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ

3.1 Порядок встановлення та налаштування параметрів системи

Інформаційна система складається з трьох логічно відокремлених частин, кожна з яких відповідає за свою ділянку роботи. Першою є односторінковий застосунок на бібліотеці React, що виконується у браузері користувача. Другою виступає програмний інтерфейс на платформі ASP.NET Core, який містить усю бізнес-логіку. Третьою є реляційна база даних PostgreSQL, де зберігаються залишки, документи та довідники.

Взаємодія між цими частинами відбувається за певною послідовністю. Браузер користувача завантажує інтерфейс і через бібліотеку Axios надсилає запити за протоколом HTTPS. Ці запити приймає окремий вузол, що віддає статичні файли інтерфейсу та водночас перенаправляє звернення до програмного інтерфейсу на бекенд. Бекенд, своєю чергою, звертається до бази даних мовою SQL, отримує чи змінює дані та повертає відповідь у зворотному напрямку (рис. 3.1).

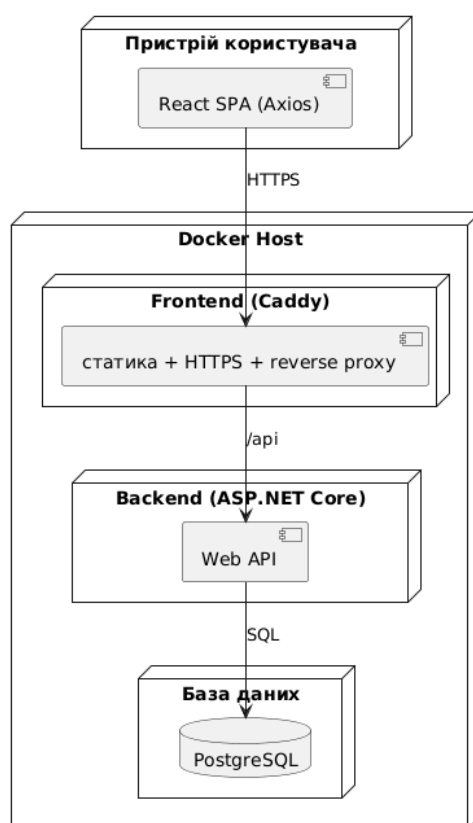


Рис. 3.1. Діаграма розгортання системи

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

Три частини мають різну природу і традиційно вимагали б різного оточення для запуску. Для бекенда потрібен встановлений набір засобів розробки .NET, для фронтенду потрібне середовище Node.js та інструмент для збирання Vite, а для бази даних потрібен окремо налаштований сервер СКБД PostgreSQL. Якщо встановлювати все це вручну на кожному комп'ютері, то зросте і кількість кроків, і ймовірність помилки через відмінності у версіях чи операційних системах. Саме цю проблему усуває контейнеризація.

Контейнер є ізольованим середовищем, усередині якого вже міститься застосунок разом з усіма потрібними йому залежностями. Завдяки цьому застосунок поводить себе однаково незалежно від того, на якому комп'ютері його запускають. У даній системі кожна з трьох частин загорнута у власний контейнер, а для їхнього спільного запуску використано засіб Docker Compose. Він описує весь набір контейнерів в одному файлі та підіймає їх однією командою.

Даний конфігураційний файл оголошує три служби, що відповідають трьом частинам системи. Служба бази даних спирається на готовий образ PostgreSQL шістнадцятої версії та зберігає свої дані у спеціальному сховищі, щоб вони не зникали після перезапуску. Служба серверної частини запускає сервер на платформі ASP.NET Core, який під час старту автоматично застосовує міграції та підключається до бази через змінну середовища. Служба клієнтської частини так само збирається та підіймає вебсервер Caddy, що віддає браузеру статичні файли та перенаправляє API-звернення до сервера.

Для створення образів контейнерів застосовано підхід багатоетапної збірки. Суть підходу полягає у тому, що процес створення образу поділено на кілька послідовних етапів, причому до кінцевого образу потрапляє лише результат останнього з них. Цей спосіб дозволяє відокремити важкі засоби збірки від легкого готового продукту. Щоб зібрати застосунок потрібен повний набір засобів розробки. Проте для того, щоб уже зібраний застосунок виконувався, ці засоби більше не потрібні. Якби вони лишалися в кінцевому образі, то він був би невинновдано громіздким, а отже повільнішим у завантаженні та менш

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

захищеним. Як для серверної, так і для клієнтської частини спочатку встановлюються залежності, компілюється готовий проєкт, готові файли копіюються в окремий образ. Через це кінцеві образи не містять ані вихідного коду, ані засобів збірки, що дозволяє оптимізувати їх розмір.

Усі запити браузера спершу потрапляють на окремий вебсервер Caddy. Він виконує одразу три завдання. По-перше, він віддає браузеру готову розмітку сторінок, стилі та скрипти односторінкового застосунку. По-друге, він працює як зворотний проксі, тобто перехоплює звернення, адресовані до програмного інтерфейсу за шляхом “/api”, і передає їх бекенду. По-третє, він забезпечує захищене з’єднання за протоколом HTTPS.

Захищене з’єднання забезпечується автоматичним отриманням сертифіката за допомогою служби Let's Encrypt без додаткових налаштувань. Для локального запуску, коли публічного домену немає, Caddy використовує внутрішній самопідписаний сертифікат, тож захищене з’єднання працює у локальній мережі також.

База даних навмисно не публікує порти назовні. До неї мають доступ лише сусідні контейнери у спільній мережі, тоді як для решти інфраструктури вона лишається невидимою. Це підвищує захищеність системи, бо зловмисник не зможе звернутися до бази напряму.

Усі параметри, які залежать від конкретного оточення, передаються через змінні середовища. До таких параметрів належать рядок підключення до бази даних, облікові дані для доступу до неї та адреса публічного домену. Завдяки цьому той самий образ можна без жодних змін запускати і на комп’ютері розробника, і на робочому сервері, а чутливі дані не потрапляють ані до вихідного коду, ані до готового образу.

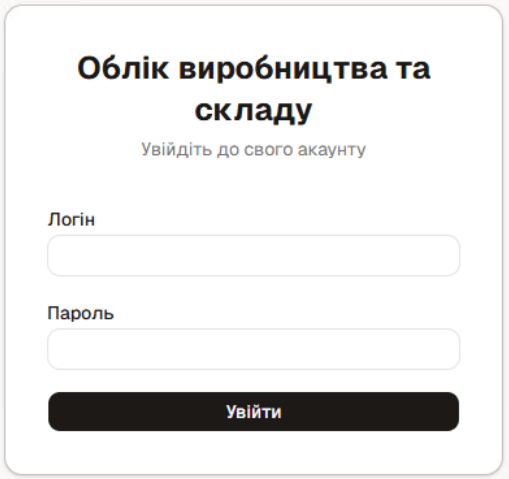
Docker файли дозволяють автоматизувати створення бази даних, застосування міграції для неї, збирання клієнтської частини та генерацію сертифікатів для забезпечення безпечного з’єднання. Урешті вебсервер Caddy починає віддавати інтерфейс та перенаправляти звернення до бекенда.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2 Структура інтерфейсу системи

Інтерфейс системи побудовано як єдиний застосунок, що відкривається у вікні браузера. Робота починається з форми входу, бо доступ до даних мають лише зареєстровані працівники (рис. 3.2). Поки облікові дані не введено правильно, далі система не пропускає, а після успішної перевірки одразу відкривається головне робоче вікно.

Після входу користувач потрапляє до головного вікна, яке має сталу побудову на всіх сторінках. Угорі розташовано смугу заголовка з назвою системи, іменем працівника та його роллю, а також кнопкою виходу. Ліворуч міститься бічна панель навігації, де пункти меню згруповано за змістом роботи, а саме «Огляд», «Склад», «Виробництво», «Закупівлі та продажі», «Довідники» та «Система». Решту простору займає робоча область, у якій і відкривається вибраний розділ. Важливо зауважити, що наповнення бічного меню залежить від прав конкретного працівника.



Облік виробництва та складу

Увійдіть до свого акаунту

Логін

Пароль

Увійти

Рис. 3.2. Форма входу користувача

Першим у меню стоїть розділ «Огляд», що відкриває інформаційну панель з ключовими показниками роботи складу (рис. 3.3). У верхньому ряду розміщено картки зі зведеними числами, серед яких загальна вартість всіх товарів на складах, кількість позицій у наявності, а також суми приходу та видатку за поточний місяць. Нижче подано графік руху запасів за останній час і таблицю партій, термін придатності яких добігає кінця. Таким чином, керівник з першого погляду оцінює стан справ, не заглиблюючись в окремі документи.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

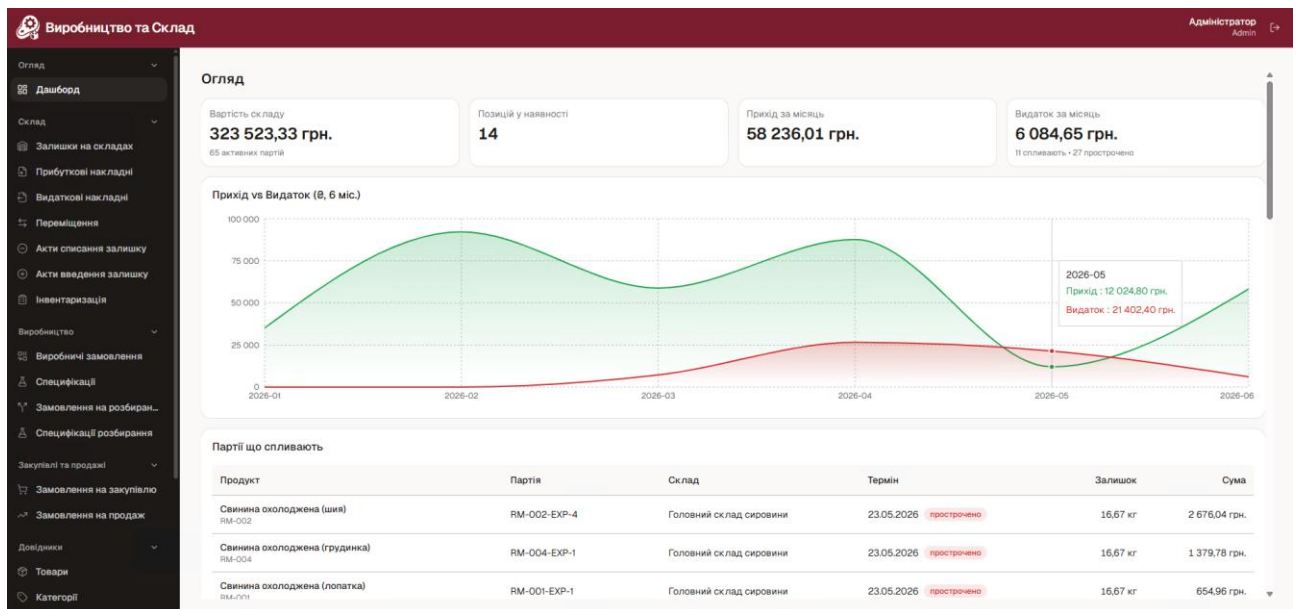


Рис. 3.3. Головне вікно системи з інформаційною панеллю огляду

Щоб дізнатися, скільки товару є на складах, працівник відкриває розділ «Залишки на складах» (рис. 3.4). Тут зібрано зведену таблицю з переліком товарів, їхньою кількістю та вартістю. Для товарів, що обліковуються за партіями, рядок можна розгорнути та побачити окремі партії з їхніми номерами та строками придатності. Над таблицею є засоби пошуку та фільтрації, які допомагають швидко знайти потрібну позицію серед великого переліку.

Виробництво та склад

Огляд

Дашборд

Склад

Залишки на складах

Прибуткові накладні

Видаткові накладні

Переміщення

Акти списання залишку

Акти виведення залишку

Інвентаризація

Виробництво

Виробничі замовлення

Специфікації

Замовлення на розбирання

Специфікації розбирання

Закупівлі та продажі

Замовлення на закупівлю

Замовлення на продаж

Додатки

Товари

Категорії

Одиниці виміру

Склади

Залишки на складах

Станом наСкладКатегорія

Пошук...

Код	Назва продукту	Категорія	Доступно	Резерв	Всього	Середня ціна	Сума
SCS-001	Напівтуша свиняча	Сировина м'ясна	160,00 кг	0,00 кг	160,00 кг	51,40 ₴	8 224,62 ₴
Код партіїСкладТермін придатностіДата отриманняДоступноЦінаСума							
SCS-001-EXP-0	Головний склад сировини	23.05.2026	31.12.2025 02:00:00	26,67 кг	152,58 ₴	4 069,31 ₴	
SCS-001-EXP-2	Головний склад сировини	23.05.2026	02.03.2026 02:00:00	26,67 кг	47,39 ₴	1 263,89 ₴	
SCS-001-EXP-4	Головний склад сировини	23.05.2026	19.04.2026 03:00:00	26,67 кг	20,35 ₴	542,73 ₴	
SCS-001-SOON-4	Головний склад сировини	17.06.2026	19.04.2026 03:00:00	26,67 кг	20,35 ₴	542,73 ₴	
SCS-001-FRESH-2	Головний склад сировини	05.09.2026	02.03.2026 02:00:00	26,66 кг	47,39 ₴	1 263,42 ₴	
SCS-001-FRESH-4	Головний склад сировини	05.09.2026	19.04.2026 03:00:00	26,66 кг	20,35 ₴	542,53 ₴	
PKG-001	Оболонка колагенова 45мм	Пакувальні матеріали	500,00 шт	0,00 шт	500,00 шт	125,92 ₴	62 960,00 ₴
PKG-002	Плівка вакуумна для упаковки	Пакувальні матеріали	500,00 уп	0,00 уп	500,00 уп	48,94 ₴	24 470,00 ₴
> RH-001	Свинина охолоджена (лопатка)	Сировина м'ясна	298,00 кг	0,00 кг	298,00 кг	121,26 ₴	36 135,88 ₴
> RH-002	Свинина охолоджена (шия)	Сировина м'ясна	228,62 кг	20,69 кг	249,31 кг	109,15 ₴	27 212,01 ₴
> RH-003	Сало свинне	Сировина м'ясна	140,09 кг	0,00 кг	140,09 кг	121,96 ₴	17 085,13 ₴
> RH-004	Свинина охолоджена (грудинка)	Сировина м'ясна	500,01 кг	0,00 кг	500,01 кг	120,73 ₴	60 366,31 ₴
> SAU-001	Ковбаса варена "Докторська" крафтова	Варені ковбаси	144,00 кг	0,00 кг	144,00 кг	130,62 ₴	18 809,20 ₴
> SAU-002	Ковбаса варена "Докторська" крафтова	Варені ковбаси	144,00 кг	0,00 кг	144,00 кг	130,62 ₴	18 809,20 ₴

Всього: 14 записівРезультат: 201 / 1< > < > < >

Рис. 3.4. Перегляд залишків на складах

Основну частину щоденної роботи комірника становлять складські документи, до яких належать прибуткові та видаткові накладні, переміщення між

складами, акти списання і введення залишку та інвентаризація. Усі ці розділи побудовано за єдиним зразком. Наприклад, прибуткові накладні відкриваються списком документів у вигляді таблиці з номером, датою, складом, контрагентом, сумою та статусом (рис. 3.5).

Якщо натиснути правою кнопкою миші на рядок окремого документа, з'являється контекстне меню дій, через яке документ можна провести, скасувати, відредагувати чи видалити залежно від його стану. Коли користувач натискає на рядок списку, праворуч висувається бічна панель з докладними відомостями про документ, і самий список при цьому лишається видимим.

Номер	Дата	Постачальник	Склад	Статус	Позицій	Сума
пн-123132	13.06.2026 01:25:10	Борочко - Балакун	Склад готової продукції	Проведено	1	51 949,71 грн.
IN-00015	31.05.2026 03:00:00	Скиба Пром	Головний склад сировини	Чернетка	2	5 965,50 грн.
IN-00016	08.05.2026 03:00:00	Коломієць Трейд	Головний склад сировини	Чернетка	2	2 765,20 грн.
IN-00011	19.04.2026 03:00:00	Балакун Збут	Головний склад сировини	Проведено	4	20 501,50 грн.
IN-00013	19.04.2026 03:00:00	Балакун Збут	Головний склад сировини	Проведено	1	1 628,00 грн.
IN-00012	19.04.2026 03:00:00	Балакун Збут	Головний склад сировини	Проведено	4	14 950,80 грн.
IN-00010	03.04.2026 03:00:00	Борочко - Балакун	Головний склад сировини	Проведено	4	14 204,60 грн.
IN-00009	03.04.2026 03:00:00	Борочко - Балакун	Головний склад сировини	Проведено	4	15 011,00 грн.
IN-00008	02.03.2026 02:00:00	Дмитришин, Петик and Компанієць	Головний склад сировини	Проведено	1	3 791,20 грн.
IN-00006	02.03.2026 02:00:00	Дмитришин, Петик and Компанієць	Головний склад сировини	Проведено	4	35 528,00 грн.
IN-00007	02.03.2026 02:00:00	Дмитришин, Петик and Компанієць	Головний склад сировини	Проведено	4	7 981,60 грн.
IN-00017	27.02.2026 02:00:00	Скиба Пром	Головний склад сировини	Проведено	1	2 400,00 грн.
IN-00018	27.02.2026 02:00:00	Скиба Пром	Склад готової продукції	Проведено	1	2 400,00 грн.
IN-00014	05.02.2026 02:00:00	Скиба Пром	Склад пакувальних матеріалів	Проведено	2	87 430,00 грн.

Рис. 3.5. Список прибуткових накладних з бічною панеллю

Для створення нового документа передбачено форму, що відкривається на всю робочу область (рис. 3.6). У її верхній частині заповнюють спільні поля документа, наприклад номер, дату, контрагента та склад, причому номер система пропонує підставити автоматично. Нижче розташовано таблицю позицій, до якої додають товари із зазначенням кількості та ціни, а підсумкова сума одразу перераховується внизу. Якщо товар обліковується за партіями, то рядок можна розгорнути та вказати конкретні партії. Після збереження форма закривається, а створений документ одразу з'являється у списку зі статусом чернетки й очікує остаточного проведення.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

←

Нова прибуткова накладна

Скасувати

Зберегти

Постачальник

Склад

Номер документа

Дата

Скиба Пром

Головний склад сировини

ПН-2026-0001

14.06.2026 18:12

Продукт	Кількість	Ціна	Загальна сума
Сало свинине	40 кг	31 грн.	1 240,00 грн.

Партії

Кількість

Номер партії

Дата виробництва

Термін придатності

36

ПН-2026-0001/1

Не вказано

20.06.2026

Кількість

Номер партії

Дата виробництва

Термін придатності

4

ПН-2026-0001/2

Не вказано

21.06.2026

+ Додати партію

Нітрита сіль

884 г

0,94 грн.

830,96 грн.

+ Додати позицію

Позицій: 2 Разом: 2 070,96 грн.

Рис. 3.6. Форма створення прибуткової накладної

У розділі «Виробництво» виготовлення продукції проходить кілька послідовних стадій. Через це виробниче замовлення має власну окрему сторінку, яка слугує робочим осередком для всього процесу. На ній угорі показано смугу стадій, що наочно веде замовлення від чернетки через резервування сировини та власне виробництво до завершення. Поряд зі смугою розташовано кнопки переходу між стадіями, причому система показує лише ті дії, які доречні на поточному кроці (рис. 3.7).

✓ Чернетка

✓ Зарезервовано

✓ У виробництві

✓ Завершено

ВИГОТОВЛЕННЯ ПРОДУКЦІЇ "Грудинка копчена крафтова"

за специфікацією "Рецепт: Грудинка копчена крафтова"

ДАТА ЗАМОВЛЕННЯ

22.05.2026 03:00:00

РОЗПОЧАТО

07.06.2026 18:34:28

ЗАКІНЧЕНО

07.06.2026 18:34:30

СКЛАД СИРОВИНИ

Головний склад сировини

СКЛАД ПРОДУКЦІЇ

Склад готової продукції

ЗА РЕЦЕПТОМ

40,00 кг

ФАКТИЧНО ОТРИМАНО

40,00 кг

% ВИХІД

100,0%

ЗАТРАТИ

6 271,84 грн.

ДОДАТКОВІ ВИТРАТИ

246,57 грн.

СОВІВАРТІСТЬ ОДИНИЦІ

162,97 грн.

ПАРТІЯ ПРОДУКЦІЇ

СМК-001-OUT-00005

СПИСОК СИРОВИНИ (4)

СИРОВИНА	ОД.	К-СТЬ	ЦІНА	СУМА	ПАРТІЇ
Сало свинине	кг	22,61	100,39	2 269,93	РН-001-SOON-4 6,95 кг до 17.06.2026 DT38-00001 15,96 кг до 05.08.2026
Свинина охолоджена (грудинка)	кг	24,58	155,26	3 815,74	РН-004-FRESH-1 10,06 кг до 05.09.2026 РН-004-FRESH-2 14,52 кг до 05.09.2026
Перець чорний мелений	г	0,60	100,21	59,83	
Часник сушений	г	0,75	167,57	126,35	
Всього				6 271,84 грн.	

ХРОНОЛОГІЯ

ДАТА	ПОДІЯ	ВИКОНАВЕЦЬ
07.06.2026 18:34:23	Створено	Адміністратор
07.06.2026 18:34:26	Зарезервовано	Адміністратор
07.06.2026 18:34:28	Виробництво почато	Адміністратор
07.06.2026 18:34:30	Оприбутковано	Адміністратор

Рис. 3.7. Сторінка виробничого замовлення

Щоб система знала з чого і в якій пропорції виготовляється продукт, поряд працює розділ специфікацій (рис. 3.8). Специфікація є рецептом, у якому перелічено складники, їхні кількості та відсоток виходу готового виробу. Саме з неї виробниче замовлення бере перелік потрібної сировини у момент створення, після чого працює вже зі своєю копією і не залежить від подальших змін рецепта.

Редагування специфікації

Назва *

Рецепт: Ковбаса сирокочена 'Салямі крафт'

Опис

Продукт виходу *

Ковбаса сирокочена 'Салямі крафт'

Продукт виходу не можна змінити після створення

Кількість виходу * Відсоток виходу *

1 кг 95 %

Компоненти *

+ Додати

Матеріал	Кількість	
Перець чорний мелений	0,0074	г
Свинина охолоджена (грудинка)	0,6409	кг
Свинина охолоджена (шия)	0,4914	кг
Сіль кухонна	0,0075	кг

Зберегти зміни Скасувати

Рис. 3.8. Форма редагування специфікації

Сторінка замовлення на розбирання побудована, так само як і у виробничого замовлення, з тією самою смугою стадій і робочим осередком, проте логіка тут зворотна (рис. 3.9). На вході стоїть один продукт, а на виході оприбутковується кілька, тому вартість сировини розподіляється між отриманими відрубками. Користувач при цьому вводить собівартість кожного виходу у гривнях за одиницю, а система підказує нормативне значення за рецептом.

Менеджер з продажів працює переважно у розділі «Закупівлі та продажі», де ведуть замовлення на закупівлю та замовлення на продаж. Замовлення відображає домовленість з контрагентом і поступово виконується через складські накладні, тому на його сторінці видно, яку частину позицій уже відвантажено чи отримано.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

✓

Чернетка

✓

Зарезервовано

3

У виробництві

4

Завершено

Додаткові витрати, грн.

Зберегти

ПРОДУКТ	ЗА РЕЦЕПТОМ	КІЛЬКІСТЬ	СОБІВАРТІСТЬ ЗА РЕЦЕПТОМ	СОБІВАРТІСТЬ	СУМА
Свинина охолоджена (шия)	12,50 кг	12,5	47,36 грн.	47,36	592,05 грн.
Свинина охолоджена (лопатка)	15,00 кг	15	56,84 грн.	56,84	852,55 грн.
Сало свинне	10,00 кг	11	37,89 грн.	43	473,00 грн.

Фактичний вихід: **77.0%** За рецептом 75.0%

РОЗБІР Напівтуша свиняча

за специфікацією Розбір: Напівтуша свиняча

📅

ДАТА ЗАМОВЛЕННЯ

14.06.2026 19:42:58

▶

РОЗПОЧАТО

14.06.2026 19:43:25

🏠

ЗАВЕРШЕНО

🏠

СКЛАД СИРОВИНИ

Головний склад сировини

🏠

СКЛАД ПРОДУКЦІЇ

Склад готової продукції

📊

ВХІД

50,00 кг

💰

ВАРТІСТЬ СИРОВИНИ

1 861,39 грн.

💰

ДОДАТКОВІ ВИТРАТИ

0,00 грн.

📊

РОЗПОДІЛЕНО НА ВИХОДИ

0,00 грн.

📊

КОЕФІЦІЄНТ ВИХОДУ (ФАКТ / РЕЦЕПТ)

75.0% / 75.0%

Рис. 3.9. Сторінка замовлення на розбирання

Окремої уваги заслуговує сторінка замовлення на продаж, бо саме на ній працює засіб планування потреби. Поки замовлення ще не виконане, система розкладає замовлені готові вироби на складники та звіряє їх з наявними залишками, відкритими закупівлями. За результатом вона показує, чого саме бракує, і поряд із кожною нестачею пропонує дію, а саме виробити, розібрати або закупити (рис. 3.10).

←

Замовлення PR-2026-0002

Підтверджено

Не відвантажено

Відвантажити за замовленням

Закрити

Скасувати

Клієнт

Скиба Пром

Склад відгрузки

Склад готової продукції

Дата

14.06.2026 19:34:22

Термін

18.06.2026

Сума

62 000,00 грн.

ПЛАНУВАННЯ ПОТРЕБИ

Є дефіцит.

Створити закупівлю (1)

КОМПОНЕНТ	ПОТРЕБА	ДОСТУПНО	ДЕФИЦИТ	ДІЯ
Сало свинне RM-003	336,16 кг	64,55	271,60	Розібрати
Свинина охолоджена (грудинка) RM-004	344,11 кг	20,00	324,11	Закупити
Нітритна сіль SP-004	5,90 г	97,74		
Перець чорний мелений SP-002	8,53 г	109,64		

ПОЗИЦІЇ (1)

ТОВАР	КІЛЬКІСТЬ	ВІДВАНТАЖЕНО	ЦІНА	СУМА
Ковбаса варена 'Докторська' крафтова BAU-001	500,00 кг	0,00 / 500,00	124,00 грн.	62 000,00 грн.

Рис. 3.10. Замовлення на продаж з панеллю планування потреби

Щоб документи й замовлення мали з чим працювати, у системі ведуть довідники, до яких належать товари, категорії, одиниці виміру, склади та контрагенти. Усі вони влаштовані схоже, бо складаються зі списку записів і форми для створення чи редагування. Наприклад, на сторінці з товарами ліворуч розташовано дерево категорій для фільтрації, а основну частину займає таблиця товарів із пошуком (рис. 3.11).

Код	Назва	Категорія	Од. виміру	Термін придатності (дні)
BMK-001	Грудка колбасна крафтова	Колбасні вироби	кг	60
SAU-001	Колбаса варена "Докторська" крафтова	Варені колбаси	кг	45
SAU-002	Колбаса сирокочена "Салтис крафт"	Сирокочені колбаси	кг	45
CRC-001	Напівтуша свиняча	Сировина м'ясна	кг	30
SP-004	Нитратна сіль	Спеції та добавки	г	
PKG-001	Оболонка колбаскова 48мм	Пакувальні матеріали	шт	
SP-002	Перець чорний мелений	Спеції та добавки	г	
PKG-002	Плівка вакуумна для упаковки	Пакувальні матеріали	ун	
RM-003	Сало свиняче	Сировина м'ясна	кг	30
RM-004	Свинина охолоджена (грудка)	Сировина м'ясна	кг	90
RM-001	Свинина охолоджена (лопатка)	Сировина м'ясна	кг	90
RM-002	Свинина охолоджена (шані)	Сировина м'ясна	кг	30
SP-001	Сіль кухонна	Спеції та добавки	кг	

Рис. 3.11. Довідник товарів з деревом категорій

Розділ меню під назвою «Система» призначено для адміністратора. У ньому ведуть перелік користувачів, налаштовують ролі з доступом та переглядають журнал аудиту.

На сторінці редагування ролі права згруповано у вигляді таблиці дозволів (рис. 3.12). Прапорці задають, чи може роль переглядати та змінювати дані у певному модулі системи. Це дозволяє адміністратору швидко скласти потрібний набір повноважень.

← До ролей

Редагування ролі

Назва *

менеджер

Привілеї

Склад

Дія	Перегляд	Редагування
Залишки на складах	☒	☐
Прибуткові накладні	☒	☐
Видаткові накладні	☒	☐
Переміщення	☒	☐
Акти списання залишку	☒	☐
Акти введення залишку	☒	☐
Інвентаризація	☒	☐

Виробництво

Дія	Перегляд	Редагування
Виробничі замовлення	☐	☐
Специфікації	☐	☐
Замовлення на розбирання	☐	☐
Специфікації розбирання	☐	☐

Закупівлі та продажі

Рис. 3.12. Матриця дозволів ролі

На сторінці журналу аудиту розміщений список дій з фільтрами (рис. 3.13). Натискання на запис відкриває докладні відомості про конкретну зміну.

Журнал аудиту					Подія Редагування	
Період	3 дати	По дату	Дія	Об'єкт	Користувач	
Пошук...					ЧАС	14.06.2026 19:43:50
Перетягніть заголовки колонок сюди для групування					КОРИСТУВАЧ	admin
					ДІЯ	Редагування
					ОБ'ЄКТ	Замовлення на розбирання
					#	ІДЕНТИФІКАТОР P3-2026-0002
					РЕЗУЛЬТАТ	Успішно
					ТЕХНІЧНА КОМАНДА	UpdateDisassemblyActualsCommand
					Відкрити запис	
Час	Користувач	Дія	Об'єкт	Результат		
14.06.2026 19:43:50	admin	Редагування	Замовлення на розбирання P3-2026-0002	Успішно		
14.06.2026 19:43:26	admin	Запуск	Замовлення на розбирання P3-2026-0002	Успішно		
14.06.2026 19:43:17	admin	Резервування	Замовлення на розбирання P3-2026-0002	Успішно		
14.06.2026 19:43:11	admin	Створення	Замовлення на розбирання P3-2026-0002	Успішно		
14.06.2026 19:42:46	admin	Резервування	Замовлення на розбирання P3-2026-0001	Помилка		
14.06.2026 19:42:26	admin	Резервування	Замовлення на розбирання P3-2026-0001	Помилка		
14.06.2026 19:35:20	admin	Підтвердження	Замовлення на продаж PR-2026-0002	Успішно		
14.06.2026 19:35:10	admin	Створення	Замовлення на продаж PR-2026-0002	Успішно		
14.06.2026 19:27:39	admin	Резервування	Виробниче замовлення BP-2026-0003	Помилка		
14.06.2026 19:26:57	admin	Резервування	Виробниче замовлення BP-2026-0003	Помилка		
14.06.2026 19:26:53	admin	Редагування	Виробниче замовлення BP-2026-0003	Успішно		
14.06.2026 19:26:36	admin	Резервування	Виробниче замовлення BP-2026-0003	Помилка		
Всього: 233 записів					Рядків: 50 1 / 5	

Рис. 3.13. Журнал аудиту дій користувачів

3.3 Тестування системи

Якість програмного забезпечення забезпечується двома методами. Перший метод полягає у застосуванні автоматизованого тестування, яке перевіряє правильність обчислень та поведінку окремих складників без участі людини. Другий метод зводиться до виконання ручного тестування, під час якого реальні робочі сценарії проходилися через інтерфейс системи та звірялися з очікуваним результатом.

Юніт-тести перевіряють чисту доменну логіку та службові класи рівня застосунку, тобто ту частину системи, де зосереджені найважливіші обчислення. Передусім це розподіл партій за правилом найближчого терміну придатності, обчислення собівартості виробничого замовлення, розподіл собівартості замовлення розбирання, обчислення середньозваженої собівартості системної партії.

Інтеграційні тести підіймають увесь застосунок у пам'яті та звертаються до нього так, як це робив би клієнт. Написані тести підтверджують, що звернення без входу до системи отримує відмову в доступі, а звернення від користувача без

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволу на запис продуктів отримує відмову через брак прав. Також перевіряється успішне створення нового продукту та отримання його по ідентифікатору. Таким чином відстежується коректність роботи всієї послідовності від маршрутизації запиту через перевірку дозволів і обробку команди до фактичного запису в базу та подальшого читання з неї.

Ручне тестування доповнює автоматизоване там, де важливо переконатися у правильності цілісних робочих сценаріїв та поведінки інтерфейсу. Описані нижче перевірки зосереджені на бізнес-правилах, життєвих циклах документів та реакції інтерфейсу на дії користувача. В кожному випадку фактичний результат збігся з очікуваним, що підтверджує належну якість системи.

Першу перевірку присвячено повному життєвому циклу виробничого замовлення, оскільки саме він поєднує найбільшу кількість бізнес-правил в одному сценарії. У табл. 3.1 наведено умови, кроки та підсумок проходження замовлення від чернетки до завершеного випуску з обчисленою собівартістю одиниці.

Таблиця 3.1

Тест “Повний життєвий цикл виробничого замовлення”

Передумови	Користувача авторизовано з дозволом на запис виробничих замовлень. Існує специфікація, а на складі сировини достатньо матеріалів
Кроки	1. Створити виробниче замовлення на основі специфікації 2. Виконати резервування і перевірити перехід до стану зарезервовано 3. Запустити виробництво і перевірити перехід до стану у виробництві 4. Завершити замовлення з фактичним випуском 5. Перевірити стан завершено та обчислену собівартість випуску
Тестові дані	Плановий випуск: 100. Фактичний випуск: 95. Додаткові витрати задано
Очікуваний результат	Замовлення послідовно проходить усі стани, сировина резервується і списується, на склад продукції оприбутковується партія випуску з обчисленою собівартістю одиниці
Фактичний результат	Збігається з очікуваним

Наступну перевірку присвячено правилу, яке забезпечує цілісність довідкових даних. У табл. 3.2 показано спробу архівувати продукт із ненульовим залишком та реакцію системи, яка таку дію відхиляє і лишає продукт активним.

Таблиця 3.2

Тест “Блокування архівації продукту, що має залишок на складі”

Передумови	Користувача авторизовано як адміністратора. У системі існує продукт із ненульовим залишком на складі
Кроки	1. Перейти до довідника продуктів 2. Знайти продукт із залишком на складі 3. Викликати дію архівації для цього продукту 4. Зафіксувати реакцію системи
Тестові дані	Продукт із залишком більшим за нуль на будь-якому складі
Очікуваний результат	Система блокує архівацію та показує повідомлення про те, що продукт має залишок. Продукт залишається активним
Фактичний результат	Збігається з очікуваним

Близьким за змістом до попереднього є обмеження на зміну режиму обліку, тому що обидва правила захищають уже накопичені дані від суперечливих станів. Як описано у табл. 3.3, система не дозволяє перемкнути облік за партіями для продукту по якому вже відбулися рухи запасів.

Таблиця 3.3

Тест “Заборона зміни режиму обліку для продукту з рухами запасів”

Передумови	Користувача авторизовано як адміністратора. Існує продукт, по якому вже були рухи запасів
Кроки	1. Перейти до довідника продуктів 2. Відкрити форму редагування продукту з рухами 3. Спробувати змінити перемикач обліку за партіями на протилежне значення 4. Зберегти зміни та зафіксувати реакцію
Тестові дані	Продукт, що має чисту кількість руху більшу за нуль
Очікуваний результат	Система не дозволяє змінити режим обліку та показує відповідне повідомлення. Режим обліку лишається попереднім
Фактичний результат	Збігається з очікуваним

Окрему увагу приділено правильності підбору партій під час відвантаження. У табл. 3.4 відображено випадок, коли сума вручну підібраних партій не дорівнює кількості рядка, через що форма попереджає про розбіжність і не дозволяє провести документ.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.4

Тест “Перевірка відповідності суми підборів партій у накладній”

Передумови	Користувача авторизовано з дозволом на запис видаткових накладних. Існує продукт з обліком за партіями та залишком на складі
Кроки	1. Створити чернетку видаткової накладної 2. Додати рядок з продуктом та задати кількість 3. Вручну підібрати партії так, щоб їхня сума не дорівнювала кількості рядка 4. Зафіксувати реакцію форми
Тестові дані	Кількість рядка: 10. Сума підібраних партій: 7
Очікуваний результат	Форма показує живе попередження про розбіжність і не дозволяє провести документ, доки суму не вирівняно
Фактичний результат	Збігається з очікуваним

Окрім перевірки окремих правил, важливо переконатися в коректності операцій руху запасів. Першою з таких перевірок є переміщення між складами, яка описана у табл. 3.5.

Таблиця 3.5

Тест “Переміщення запасів між складами”

Передумови	Користувача авторизовано з дозволом на запис переміщень. На складі джерела є залишок продукту
Кроки	1. Створити чернетку переміщення між складами 2. Обрати склад джерела та склад призначення 3. Додати рядок з продуктом і кількістю в межах залишку 4. Провести документ та перевірити залишки на обох складах
Тестові дані	Кількість переміщення: 20. Залишок на складі джерела до проведення: 50
Очікуваний результат	Кількість списується зі складу джерела та оприбутковується на склад призначення, а загальний залишок по системі лишається незмінним
Фактичний результат	Збігається з очікуваним

Складнішим прикладом руху запасів є інвентаризація, адже вона не лише фіксує фактичні залишки, а й самостійно породжує коригувальні документи. У табл. 3.6 наведено сценарій, у якому за надлишком формується акт оприбуткування, а за нестачею акт списання.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.6

Тест “Інвентаризація з виявленням розбіжностей”

Передумови	Користувача авторизовано з відповідним дозволом. На складі є продукти з відомим обліковим залишком
Кроки	1. Створити акт інвентаризації для складу 2. Внести фактичні кількості, що відрізняються від облікових у більший та менший бік 3. Провести акт і зафіксувати породжені документи коригування
Тестові дані	Продукт із надлишком та продукт із нестачею відносно облікового залишку
Очікуваний результат	За виявленим надлишком система породжує акт оприбуткування, а за нестачею акт списання, після чого обліковий залишок вирівнюється з фактичним
Фактичний результат	Збігається з очікуваним

Поповнення складу новою партією перевірено на прикладі прибуткової накладної. У табл. 3.7 показано, що після проведення документа на складі з'являється партія із заданими номером та терміном придатності, а залишок продукту зростає на введену кількість.

Таблиця 3.7

Тест “Проведення прибуткової накладної зі створенням партії”

Передумови	Користувача авторизовано з дозволом на запис прибуткових накладних. Існує продукт з обліком за партіями
Кроки	1. Створити чернетку прибуткової накладної 2. Додати рядок з продуктом, кількістю та даними партії з терміном придатності 3. Провести документ та перевірити появу партії і залишку на складі
Тестові дані	Кількість: 40. Номер партії та дата придатності задані
Очікуваний результат	Після проведення на складі з'являється нова партія із заданими даними, а залишок продукту зростає на введену кількість
Фактичний результат	Збігається з очікуваним

Останню перевірку присвячено аналізу потреби. У табл. 3.8 наведено сценарій, у якому для замовлення продажу з недостатнім запасом сировини система обчислює дефіцит кожного компонента та визначає відповідну дію, а саме виробництво, розбирання або закупівлю.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

Тест “Аналіз потреби за замовленням продажу з підказкою дії”

Передумови	Користувача авторизовано з відповідними дозволами. Існує замовлення продажу, для якого є специфікація, а сировини на складі недостатньо
Кроки	1. Відкрити замовлення продажу 2. Запустити аналіз потреби 3. Оглянути перелік компонентів та обчислений дефіцит 4. Перевірити підказку дії для дефіцитного компонента
Тестові дані	Замовлення продажу, потреба якого перевищує наявний та очікуваний у дорозі залишок
Очікуваний результат	Система показує дефіцит та підказує дію виробити, розібрати або закупити залежно від наявних специфікацій
Фактичний результат	Збігається з очікуваним

У підсумку усі автоматизовані тести пройшли успішно, а в кожному ручному сценарії фактичний результат збігся з очікуваним. З цього випливає, що ключові бізнес-правила, життєві цикли документів та обмеження доступу працюють так, як задумано, і система придатна до використання за призначенням.

Висновки до третього розділу

У третьому розділі описано процес розгортання та тестування системи. Спершу розглянуто порядок встановлення. Систему поділено на клієнтський застосунок у браузері, програмний інтерфейс із бізнес-логікою та реляційну базу даних. Кожна частина має власне оточення, тому їх загорнуто в окремі контейнери, а спільний запуск налаштовано засобами Docker Compose.

Далі розглянуто інтерфейс, який відкривається у вікні браузера як єдиний застосунок. Робота починається з входу, після чого працівник потрапляє до головного вікна зі сталою структурою, а наповнення меню залежить від його прав. Наприкінці якість застосунку перевірено автоматизованими та ручними тестами. Оскільки фактичний результат збігся з очікуваним, ключові бізнес-правила, життєві цикли документів та обмеження доступу працюють коректно, а отже, система придатна до використання за призначенням.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання роботи спроектовано та розроблено клієнт-серверну інформаційну систему автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів, яка об'єднує складський облік, планування і контроль виробництва від закупівлі сировини до відвантаження готової продукції в межах єдиної системи.

Дослідження предметної області та огляд аналогів дали змогу визначити функціонал системи й сформувавши функціональні та нефункціональні вимоги, а також обґрунтувати вибір технологічних засобів. Серверну частину побудовано за принципами предметно-орієнтованого проектування, завдяки чому облікову логіку зосереджено в одному ядрі та відокремлено від технічних деталей.

Відповідно до поставлених завдань реалізовано серверну частину за допомогою платформи ASP.NET Core та клієнтську частину за допомогою бібліотеки React мовою програмування TypeScript. Система веде партійний облік з автоматичним та ручним підбором партій за принципом FIFO та оформлює повний цикл складських документів, до якого належать прибуткові й видаткові накладні, переміщення між складами та акти інвентаризації. Окрім цього, вона підтримує замовлення на закупівлю і продаж, виробничі замовлення зі специфікаціями, розбирання вхідної сировини, рольове розмежування доступу та журнал аудиту операцій.

В ході автоматизованого та ручного тестування фактичний результат збігся з очікуваним, що засвідчує коректність ключових бізнес-правил та життєвих циклів документів. Завершальним етапом стало розгортання системи засобами Docker Compose.

Результатом роботи є готова до експлуатації система, яка забезпечує контроль термінів придатності й відповідає сформованим вимогам щодо надійності та розмежування доступу. Закладена архітектура є відкритою для подальшого розвитку, зокрема для створення мобільного застосунку з можливістю сканування штрихкодів партій та реалізації повноцінного фінансового обліку.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Albahari J. C# 12 in a Nutshell: The Definitive Reference. Sebastopol : O'Reilly Media, 2023. 1070 p.
2. ASP.NET documentation. URL: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-10.0> (дата звернення: 16.06.2026).
3. Cheung T. ERP vs MRP: The Maintenance-First Guide to Choosing Your Manufacturing Stack // Factory AI. 2026. 23 Feb. URL: <https://f7i.ai/blog/erp-vs-mrp-the-maintenance-first-guide-to-choosing-your-manufacturing-stack> (дата звернення: 16.06.2026).
4. CQRS pattern // Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs> (дата звернення: 16.06.2026).
5. Docker Compose. URL: <https://docs.docker.com/compose/> (дата звернення: 16.06.2026).
6. Domain-Driven Design (DDD) // Sensidev. URL: <https://sensidev.com/glossary/domain-driven-design> (дата звернення: 16.06.2026).
7. Entity Framework documentation hub. URL: <https://learn.microsoft.com/en-us/ef/> (дата звернення: 16.06.2026).
8. Ingole A. How Long Does Odoo Implementation Take? // SDLC Corp. 2026. 27 Feb. URL: <https://sdlccorp.com/post/how-long-does-odoo-implementation-take/> (дата звернення: 16.06.2026).
9. Introduction – shadcn/ui. URL: <https://ui.shadcn.com/docs> (дата звернення: 16.06.2026).
10. Katana MRP review, pricing, features, pros & cons // Ciroapp. 2025. 17 Nov. URL: <https://ciroapp.com/software/katana-mrp> (дата звернення: 16.06.2026).
11. Katana. URL: <https://katanamrp.com/> (дата звернення: 16.06.2026).
12. Khononov V. Learning Domain-Driven Design. Sebastopol : O'Reilly Media, 2021. 342 p.

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Lauri K. The State of Cloud-Based ERP/MRP in Smaller Food Firms // Food Logistics. 2020. 24 Feb. URL: <https://www.foodlogistics.com/software-technology/emerging-technologies/article/21117299/mrpeasy-the-state-of-cloudbased-erpmrp-in-smaller-food-firms> (дата звернення: 16.06.2026).
14. Material Requirement Planning (MRP) // CIPS. URL: <https://www.cips.org/intelligence-hub/operations-management/material-requirements-planning> (дата звернення: 16.06.2026).
15. Microsoft Dynamics 365. URL: <https://www.microsoft.com/en/dynamics-365> (дата звернення: 16.06.2026).
16. Odoo. URL: https://www.odoo.com/uk_UA (дата звернення: 16.06.2026).
17. PostgreSQL: Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 16.06.2026).
18. React. URL: <https://react.dev/> (дата звернення: 16.06.2026).
19. Sirigiri K. .NET Core Vs. Java Spring Boot: A Review-Driven Performance Assessment for Cloud-Native API Architectures // International Journal of Intelligent Systems and Applications in Engineering. 2023. Vol. 11, no. 11s. P. 730–738. URL: <https://ijisae.org/index.php/IJISAE/article/view/7434> (дата звернення: 16.06.2026).
20. Tailwind CSS. URL: <https://tailwindcss.com/docs/installation/using-vite> (дата звернення: 16.06.2026).
21. TypeScript Documentation. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 16.06.2026).
22. Vite: Getting Started. URL: <https://vite.dev/guide/> (дата звернення: 16.06.2026).
23. Wieruch R. React Folder Structure Best Practices. 2026. 05 May. URL: <https://www.robinwieruch.de/react-folder-structure/> (дата звернення: 16.06.2026).
24. Власова А. Повний гайд по ERP-системах: що це, приклади та структура сучасних рішень // Brander. 2025. 22 серп. URL:

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://brander.ua/blog/povnyy-hayd-po-erp-systemakh-shcho-tse-prykłady-ta-struktura-suchasnykh-rishen> (дата звернення: 16.06.2026).

25. Мартін Р. Чиста архітектура: мистецтво розробки програмного забезпечення. Харків : Фабула, 2019. 416 с.
26. Мещерякова К. Що таке DDD: як працює та як впровадити // High Bar Journal. 2023. 07 верес. URL: <https://journal.gen.tech/post/sho-take-ddd> (дата звернення: 16.06.2026).
27. Паламарчук І. С., Лисогор Ю. І., Локтікова Т. М. Дослідження принципів побудови та проєктування системи управління виробництвом крафтових м'ясних виробів // Комп'ютерні технології: інновації, проблеми, рішення : тези доповідей VIII Всеукраїнської науково-технічної конференції, м. Житомир, 2–3 грудня 2025 р. Житомир : Державний університет «Житомирська політехніка», 2025. С. 63–64. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2026/01/povnyj-tekst-1.pdf> (дата звернення: 16.06.2026).
28. Паламарчук І. С., Локтікова Т. М. Побудова та проєктування інформаційної системи автоматизації складського обліку та управління виробництвом крафтових м'ясних виробів // Сучасні інформаційні технології – 2026 : матеріали XVI Міжнародної наукової конференції студентів та молодих вчених, м. Одеса, 14–15 травня 2026 р. Одеса : Наука і техніка, 2026. С. 277–279.
29. Швець О. Декоратор // Рефакторинг. Гуру. URL: <https://refactoring.guru/design-patterns/decorator> (дата звернення: 16.06.2026).
30. Як програмні комплекси ERP-рівня допомагають розширювати бізнес м'ясопереробки? // Business Evolution. URL: <https://bevol.com.ua/blog/yak-erp-dopomagaye-rozshyryuvaty-biznes-myasopererobky/> (дата звернення: 16.06.2026).

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

Dispatcher.cs

```

using System.Collections.Concurrent;
using Application.Shared;
using Microsoft.Extensions.DependencyInjection;

namespace Infrastructure.Dispatching;

public sealed class Dispatcher : IDispatcher
{
    private static readonly ConcurrentDictionary<Type, object> _commandWrappers =
new();
    private static readonly ConcurrentDictionary<Type, object> _queryWrappers = new();

    private readonly IServiceProvider _serviceProvider;

    public Dispatcher(IServiceProvider serviceProvider)
    {
        _serviceProvider = serviceProvider;
    }

    public Task<TResult> Send<TResult>(ICommand<TResult> command, CancellationToken ct
= default)
    {
        var wrapper = GetCommandWrapper<TResult>(command.GetType());
        return wrapper.Handle(command, _serviceProvider, ct);
    }

    public Task<TResult> Send<TResult>(IQuery<TResult> query, CancellationToken ct =
default)
    {
        var wrapper = GetQueryWrapper<TResult>(query.GetType());
        return wrapper.Handle(query, _serviceProvider, ct);
    }

    private static QueryHandlerWrapper<TResult> GetQueryWrapper<TResult>(Type
queryType)
    {
        return (QueryHandlerWrapper<TResult>) _queryWrappers.GetOrAdd(queryType,
CreateQueryWrapper, typeof(TResult));
    }

    private static CommandHandlerWrapper<TResult> GetCommandWrapper<TResult>(Type
commandType)
    {
        return (CommandHandlerWrapper<TResult>) _commandWrappers.GetOrAdd(commandType,
CreateCommandWrapper, typeof(TResult));
    }

    private static object CreateCommandWrapper(Type commandType, Type resultType)
    {
        Type wrapperType =
typeof(CommandHandlerWrapperImpl<, >).MakeGenericType(commandType, resultType);
        return Activator.CreateInstance(wrapperType)!;
    }

    private static object CreateQueryWrapper(Type queryType, Type resultType)
    {
        Type wrapperType =
typeof(QueryHandlerWrapperImpl<, >).MakeGenericType(queryType, resultType);
        return Activator.CreateInstance(wrapperType)!;
    }
}

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

internal abstract class CommandHandlerWrapper<TResult>
{
    public abstract Task<TResult> Handle(object command, IServiceProvider sp,
CancellationToken ct);
}

internal sealed class CommandHandlerWrapperImpl<TCommand, TResult> :
CommandHandlerWrapper<TResult>
where TCommand : ICommand<TResult>
{
    public override Task<TResult> Handle(object command, IServiceProvider sp,
CancellationToken ct)
    {
        var handler = sp.GetRequiredService<ICommandHandler<TCommand, TResult>>();
        return handler.Handle((TCommand)command, ct);
    }
}

internal abstract class QueryHandlerWrapper<TResult>
{
    public abstract Task<TResult> Handle(object query, IServiceProvider sp,
CancellationToken ct);
}

internal sealed class QueryHandlerWrapperImpl<TQuery, TResult> :
QueryHandlerWrapper<TResult>
where TQuery : IQuery<TResult>
{
    public override Task<TResult> Handle(object query, IServiceProvider sp,
CancellationToken ct)
    {
        var handler = sp.GetRequiredService<IQueryHandler<TQuery, TResult>>();
        return handler.Handle((TQuery)query, ct);
    }
}

```

StockAvailabilityChecker.cs

```

using Application.Services;
using Domain.BatchStocks;
using Domain.Products;
using Domain.Services;
using Domain.Shared;
using Domain.Shared.ValueObjects;

namespace Application.Services;

public class StockAvailabilityChecker : IStockAvailabilityChecker
{
    private readonly IBatchStockRepository _batchStockRepository;
    private readonly IProductRepository _productRepository;
    private readonly ISystemBatchService _systemBatchService;

    public StockAvailabilityChecker(
        IBatchStockRepository batchStockRepository,
        IProductRepository productRepository,
        ISystemBatchService systemBatchService)
    {
        _batchStockRepository = batchStockRepository;
        _productRepository = productRepository;
        _systemBatchService = systemBatchService;
    }

    public async Task<StockCheckResult> ValidatePickedBatchesAsync(
        WarehouseId warehouseId,

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        IEnumerable<(BatchId BatchId, ProductId ProductId, decimal Quantity)>
pickedBatches,
        DateTime asOf)
    {
        var shortages = new List<StockShortage>();
        var pickedList = pickedBatches.ToList();

        var groupedByBatch = pickedList
            .GroupBy(p => p.BatchId)
            .Select(g => new
            {
                BatchId = g.Key,
                ProductId = g.First().ProductId,
                TotalQuantity = g.Sum(x => x.Quantity)
            })
            .ToList();

        var batchIds = groupedByBatch.Select(g => g.BatchId).ToList();

        var batchStocks = await _batchStockRepository
            .GetBatchStocksByBatchIdsAsync(warehouseId, batchIds);

        var batchStockLookup = batchStocks.ToDictionary(bs => bs.BatchId);

        foreach (var pick in groupedByBatch)
        {
            if (!batchStockLookup.TryGetValue(pick.BatchId, out var batchStock))
            {
                shortages.Add(new StockShortage(
                    warehouseId,
                    pick.ProductId,
                    Required: pick.TotalQuantity,
                    Available: 0,
                    Missing: pick.TotalQuantity));
                continue;
            }

            var available = batchStock.GetAvailableQuantity(asOf);
            if (available < pick.TotalQuantity)
            {
                shortages.Add(new StockShortage(
                    warehouseId,
                    pick.ProductId,
                    Required: pick.TotalQuantity,
                    Available: available,
                    Missing: pick.TotalQuantity - available));
            }
        }

        return shortages.Count == 0
            ? StockCheckResult.Available()
            : StockCheckResult.Unavailable(shortages);
    }

    public async Task<Result<List<(BatchId BatchId, decimal Quantity)>>>
AllocateBatchesFEFOAsync(
        WarehouseId warehouseId,
        ProductId productId,
        decimal requiredQuantity,
        DateTime asOf)
    {
        var stocksWithBatches = await _batchStockRepository
            .GetBatchStocksForFEFOAsync(warehouseId, productId, asOf);
        var candidates = stocksWithBatches
            .Where(pair => !pair.Batch.ExpiryDate.IsExpired())
            .OrderBy(pair => pair.Batch.ExpiryDate.HasExpiry

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ? pair.Batch.ExpiryDate.ExpiryDate
        : DateTime.MaxValue)
    .ThenBy(pair => pair.Batch.ManufactureDate ?? DateTime.MaxValue)
    .Select(pair => new
    {
        BatchStock = pair.BatchStock,
        Batch = pair.Batch,
        Available = pair.BatchStock.GetAvailableQuantity(asOf)
    })
    .ToList();

var allocations = new List<(BatchId BatchId, decimal Quantity)>();
var remaining = requiredQuantity;

foreach (var candidate in candidates)
{
    if (remaining <= 0) break;

    var takeQuantity = Math.Min(candidate.Available, remaining);

    if (takeQuantity <= 0) continue;

    allocations.Add((
        candidate.Batch.Id,
        takeQuantity));

    remaining -= takeQuantity;
}

if (remaining > 0)
{
    var totalAvailable = candidates.Sum(c => c.Available);
    return Result<List<(BatchId, decimal)>>.Failure(
        StockErrors.Insufficient.With(
            ("required", requiredQuantity),
            ("available", totalAvailable),
            ("missing", requiredQuantity - totalAvailable)));
}
return Result<List<(BatchId, decimal)>>.Success(allocations);
}

public async Task<Result<List<(BatchId BatchId, decimal Quantity)>>>
AllocateForConsumptionAsync(
    WarehouseId warehouseId,
    ProductId productId,
    decimal requiredQuantity,
    DateTime asOf,
    CancellationToken ct = default)
{
    var product = await _productRepository.GetByIdAsync(productId, ct);
    if (product is null)
        return Result<List<(BatchId, decimal)>>.Failure(ProductErrors.NotFound);

    if (product.IsTrackedByBatch)
        return await AllocateBatchesFEFOAsync(warehouseId, productId,
        requiredQuantity, asOf);

    var batchIdResult = await
    _systemBatchService.ResolveForOutboundAsync(productId, warehouseId, ct);
    if (!batchIdResult.IsSuccess)
        return Result<List<(BatchId,
    decimal)>>.Failure(batchIdResult.Error!.Value);

    var batchStock = await
    _batchStockRepository.GetByBatchAndWarehouseAsync(batchIdResult.Value, warehouseId,
    ct);

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

var available = batchStock?.GetAvailableQuantity(asOf) ?? 0;

if (available < requiredQuantity)
{
    return Result<List<BatchId, decimal>>.Failure(
        StockErrors.Insufficient.With(
            ("required", requiredQuantity),
            ("available", available),
            ("missing", requiredQuantity - available)));
}

return Result<List<BatchId, decimal>>.Success([(batchIdResult.Value,
requiredQuantity)]);
}
}

```

SystemBatchService.cs

```

using Domain.Batches;
using Domain.BatchStocks;
using Domain.Shared;
using Domain.Shared.ValueObjects;

namespace Application.Services;

public interface ISystemBatchService
{
    Task<Result<Batch Batch, BatchStock Stock>> GetOrCreateAsync(
        ProductId productId,
        WarehouseId warehouseId,
        SourceDocument sourceDocument,
        DateTime receivedAt,
        CancellationToken ct);

    Task<Result> ApplyInboundAsync(
        ProductId productId,
        WarehouseId warehouseId,
        decimal quantity,
        decimal unitCost,
        SourceDocument source,
        DateTime occurredAt,
        CancellationToken ct);

    Task<Result<BatchId>> ResolveForOutboundAsync(
        ProductId productId,
        WarehouseId warehouseId,
        CancellationToken ct);
}

public class SystemBatchService : ISystemBatchService
{
    private readonly IBatchRepository _batchRepository;
    private readonly IBatchStockRepository _batchStockRepository;
    private readonly IUnitOfWork _unitOfWork;

    public SystemBatchService(
        IBatchRepository batchRepository,
        IBatchStockRepository batchStockRepository,
        IUnitOfWork unitOfWork)
    {
        _batchRepository = batchRepository;
        _batchStockRepository = batchStockRepository;
        _unitOfWork = unitOfWork;
    }
}

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public async Task<Result<(Batch Batch, BatchStock Stock)>> GetOrCreateAsync(
        ProductId productId,
        WarehouseId warehouseId,
        SourceDocument sourceDocument,
        DateTime receivedAt,
        CancellationToken ct)
    {
        var batch = await _batchRepository.GetSystemBatchByProductIdAsync(productId,
ct);

        if (batch is null)
        {
            var createResult = Batch.CreateSystemBatch(productId, sourceDocument,
receivedAt);
            if (!createResult.IsSuccess)
                return Result<(Batch, BatchStock)>.Failure(createResult.Error!.Value);

            batch = createResult.Value;
            await _batchRepository.AddAsync(batch, ct);

            var saveResult = await TrySaveWithUniqueRetryAsync(ct);
            if (!saveResult.IsSuccess)
                return Result<(Batch, BatchStock)>.Failure(saveResult.Error!.Value);

            if (saveResult.Value == SaveRetryOutcome.Conflict)
            {
                batch = await
_batchRepository.GetSystemBatchByProductIdAsync(productId, ct);
                if (batch is null)
                    return Result<(Batch, BatchStock)>.Failure(CommonErrors.Unmapped);
            }
        }

        var batchStock = await
_batchStockRepository.GetByBatchAndWarehouseAsync(batch.Id, warehouseId, ct);
        if (batchStock is null)
        {
            batchStock = new BatchStock(batch.Id, warehouseId, productId);
            await _batchStockRepository.AddAsync(batchStock, ct);

            var saveResult = await TrySaveWithUniqueRetryAsync(ct);
            if (!saveResult.IsSuccess)
                return Result<(Batch, BatchStock)>.Failure(saveResult.Error!.Value);

            if (saveResult.Value == SaveRetryOutcome.Conflict)
            {
                batchStock = await
_batchStockRepository.GetByBatchAndWarehouseAsync(batch.Id, warehouseId, ct);
                if (batchStock is null)
                    return Result<(Batch, BatchStock)>.Failure(CommonErrors.Unmapped);
            }
        }

        return Result<(Batch, BatchStock)>.Success((batch, batchStock));
    }

    public async Task<Result> ApplyInboundAsync(
        ProductId productId,
        WarehouseId warehouseId,
        decimal quantity,
        decimal unitCost,
        SourceDocument source,
        DateTime occurredAt,
        CancellationToken ct)

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {
        var getResult = await GetOrCreateAsync(productId, warehouseId, source,
occurredAt, ct);
        if (!getResult.IsSuccess)
            return Result.Failure(getResult.Error!.Value);

        var (batch, batchStock) = getResult.Value;
        var globalQty = await
_batchStockRepository.GetGlobalOnHandQuantityAsync(batch.Id, ct);

        var wacResult = batch.UpdateWeightedAverageCost(globalQty, quantity,
unitCost);
        if (!wacResult.IsSuccess)
            return wacResult;

        var addResult = batchStock.AddStock(quantity, source, occurredAt);
        return addResult.IsSuccess
            ? Result.Success()
            : Result.Failure(addResult.Error!.Value);
    }

    public async Task<Result<BatchId>> ResolveForOutboundAsync(
        ProductId productId,
        WarehouseId warehouseId,
        CancellationToken ct)
    {
        var batch = await _batchRepository.GetSystemBatchByProductIdAsync(productId,
ct);
        if (batch is null)
            return Result<BatchId>.Failure(BatchErrors.NoSystemBatch);

        var batchStock = await
_batchStockRepository.GetByBatchAndWarehouseAsync(batch.Id, warehouseId, ct);
        if (batchStock is null)
            return Result<BatchId>.Failure(BatchErrors.NoStockInWarehouse);

        return Result<BatchId>.Success(batch.Id);
    }

    private async Task<Result<SaveRetryOutcome>>
TrySaveWithUniqueRetryAsync(CancellationToken ct)
    {
        var result = await _unitOfWork.SavePendingChangesAsync(ct);
        if (result.IsSuccess)
            return Result<SaveRetryOutcome>.Success(SaveRetryOutcome.Saved);

        if (result.Error?.Code == CommonErrors.UniqueViolation.Code)
            return Result<SaveRetryOutcome>.Success(SaveRetryOutcome.Conflict);

        return Result<SaveRetryOutcome>.Failure(result.Error!.Value);
    }

    private enum SaveRetryOutcome
    {
        Saved,
        Conflict
    }
}

DisassemblyCostAllocator.cs

namespace Domain.Services;

public static class DisassemblyCostAllocator
{

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    public readonly record struct OutputInput(decimal ActualQuantity, decimal
ValueCoefficient);

    public static IReadOnlyList<decimal?> Allocate(
        decimal materialCost, decimal additionalCosts, IReadOnlyList<OutputInput>
outputs)
    {
        var totalCost = materialCost + additionalCosts;
        var result = new decimal?[outputs.Count];

        var produced = new List<int>();
        for (var i = 0; i < outputs.Count; i++)
            if (outputs[i].ActualQuantity > 0) produced.Add(i);

        if (produced.Count == 0) return result;

        var weights = produced.ToDictionary(i => i, i => outputs[i].ActualQuantity *
outputs[i].ValueCoefficient);
        var totalWeight = weights.Values.Sum();

        if (totalWeight <= 0)
        {
            var totalQty = produced.Sum(i => outputs[i].ActualQuantity);
            foreach (var i in produced)
                weights[i] = outputs[i].ActualQuantity;
            totalWeight = totalQty;
        }

        decimal allocatedSoFar = 0m;
        for (var k = 0; k < produced.Count; k++)
        {
            var i = produced[k];
            decimal allocated;
            if (k == produced.Count - 1)
                allocated = totalCost - allocatedSoFar;
            else
            {
                allocated = Math.Round(totalCost * weights[i] / totalWeight, 2,
MidpointRounding.AwayFromZero);
                allocatedSoFar += allocated;
            }
            result[i] = allocated / outputs[i].ActualQuantity;
        }

        return result;
    }
}

```

EfUnitOfWork.cs

```

using Domain.Shared;
using Domain.Shared.Events;
using Microsoft.EntityFrameworkCore;

namespace Infrastructure;

public class EfUnitOfWork : IUnitOfWork
{
    private readonly AppDbContext _context;
    private readonly IDomainEventDispatcher _dispatcher;

    public EfUnitOfWork(AppDbContext context, IDomainEventDispatcher dispatcher)
    {

```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _context = context;
        _dispatcher = dispatcher;
    }

    public async Task<Result> SaveChangesAsync(CancellationTokens ct = default)
    {
        var dispatchResult = await DispatchDomainEventsAsync(ct);
        if (!dispatchResult.IsSuccess)
            return dispatchResult;

        await _context.SaveChangesAsync(ct);
        return Result.Success();
    }

    public async Task<Result> SavePendingChangesAsync(CancellationTokens ct = default)
    {
        try
        {
            await _context.SaveChangesAsync(ct);
            return Result.Success();
        }
        catch (DbUpdateException ex)
        {
            var message = ex.InnerException?.Message ?? ex.Message;
            if (message.Contains("23505") || message.Contains("duplicate key",
StringComparison.OrdinalIgnoreCase))
                return Result.Failure(CommonErrors.UniqueViolation);

            return Result.Failure(CommonErrors.Unmapped);
        }
    }

    private async Task<Result> DispatchDomainEventsAsync(CancellationTokens ct)
    {
        while (true)
        {
            var entitiesWithEvents = _context.ChangeTracker
                .Entries<Entity>()
                .Where(e => e.Entity.DomainEvents.Count != 0)
                .Select(e => e.Entity)
                .ToList();

            if (entitiesWithEvents.Count == 0)
                break;

            foreach (var entity in entitiesWithEvents)
            {
                var events = entity.DomainEvents.ToList();
                entity.ClearDomainEvents();

                foreach (var domainEvent in events)
                {
                    var result = await _dispatcher.DispatchAsync(domainEvent, ct);
                    if (!result.IsSuccess)
                        return result;
                }
            }
        }
        return Result.Success();
    }
}

```

BusinessRuleEvaluator.cs

```
namespace Application.Shared.Rules;

using Domain.Shared;

public sealed class
BusinessRuleEvaluator<TContext>(IEnumerable<IBusinessRule<TContext>> rules)
{
    public async Task<Result> EvaluateAsync(TContext context, CancellationToken ct)
    {
        foreach (var rule in rules)
        {
            if (await rule.IsBrokenAsync(context, ct))
                return Result.Failure(rule.Code);
        }
        return Result.Success();
    }
}
```

ProductHasStockRule.cs

```
using Application.Shared.Rules;
using Domain.Products;
using Domain.Shared;

namespace Application.Products.Rules;

public sealed class ProductHasStockRule(IProductOperationalReadService read) :
IBusinessRule<ArchiveProductContext>
{
    public Error Code => ProductErrors.HasStock;
    public Task<bool> IsBrokenAsync(ArchiveProductContext ctx, CancellationToken ct)
=>
        read.HasStockAsync(ctx.ProductId, ct);
}
```

		Паламарчук І.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Локтікова Т.М.				83
Змн.	Арк.	№ докум.	Підпис	Дата		