

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи відстеження залишків на складі
автозапчастин»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Богдан ДЗЯДЕВИЧ
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

_____ Богдан ДЗЯДЕВИЧ

Керівник: _____ Вікторія КОРЕЦЬКА

канд. пед. наук, доц.

Рецензент: _____

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ

« ____ » _____ 2026 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Дзядевичу Богдану Олександровичу

1. Тема кваліфікаційної роботи: «Розробка системи відстеження залишків на складі автозапчастин»

керівник кваліфікаційної роботи канд. пед. наук, доцент Вікторія КОРЕЦЬКА,
затверджені наказом Державного університету інформаційно-комунікаційних
технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про складський облік та управління запасами, опис процесів обліку автозапчастин, технічна документація з Java, Spring Boot, React, PostgreSQL, JWT та Docker Compose.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі складського обліку автозапчастин та існуючих програмних рішень.

2. Проєктування системи відстеження залишків на складі автозапчастин.

3. Програмна реалізація та опис функціонування вебсистеми APW Inventory.

4. Тестування системи відстеження залишків на складі автозапчастин.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Архітектура системи.
6. Структура бази даних.
7. Діаграма класів.
8. Екранні форми.

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02-28.02.2026	
2	Аналіз та дослідження існуючих аналогів	01.03-07.03.2026	
3	Формування вимог до системи відстеження залишків автозапчастин	09.03-17.03.2026	
4	Проектування архітектури, бази даних та інтерфейсу системи	18.03-25.03.2026	
5	Програмна реалізація вебсистеми APW Inventory	25.03-17.04.2026	
6	Тестування системи	18.04-25.04.2026	
7	Оформлення роботи: вступ, висновки, реферат	26.04-02.05.2026	
8	Розробка демонстраційних матеріалів	03.05-10.05.2026	
9	Попередній захист роботи	11.05-22.05.2026	

Здобувач вищої освіти

(підпис)

Богдан ДЗЯДЕВИЧ

Керівник
кваліфікаційної роботи

(підпис)

Вікторія КОРЕЦЬКА

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 110 стор., 24 табл., 47 рис., 21 джерело.

Мета роботи – підвищення оперативності контролю складських залишків автозапчастин за рахунок розробки вебсистеми з каталогом товарів, обліком складських операцій, контролем критичних залишків, ролями доступу та звітністю.

Об'єкт дослідження – процес обліку та відстеження залишків автозапчастин на складі.

Предмет дослідження – програмні засоби автоматизації обліку товарів, складських операцій, контролю залишків і формування звітності у вебсистемі.

Короткий зміст роботи: У роботі розглянуто предметну галузь складського обліку автозапчастин, визначено основні проблеми ручного та частково автоматизованого введення залишків, а також проаналізовано програмні засоби Odoo Inventory, Zoho Inventory, Sortly та inFlow Inventory. Спроектовано архітектуру вебсистеми, структуру бази даних, ролі користувачів, функціональні та нефункціональні вимоги, сценарії використання й логіку складських операцій. Реалізовано систему APW Inventory із серверною частиною на Java та Spring Boot, базою даних PostgreSQL, механізмом авторизації на основі JWT, клієнтською частиною на React і TypeScript, а також модулями каталогу товарів, залишків, транзакцій, звітності та експорту CSV. Проведено функціональне, інтеграційне, безпекове та нефункціональне тестування системи.

Сферою використання системи є автоматизація складського обліку в магазинах автозапчастин, сервісних центрах і на складах малого або середнього підприємства.

КЛЮЧОВІ СЛОВА: СКЛАДСЬКИЙ ОБЛІК, АВТОЗАПЧАСТИНИ, ЗАЛИШКИ, SKU, JAVA, SPRING BOOT, REACT, POSTGRESQL, JWT

ЗМІСТ

ВСТУП	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ	11
1.1 Особливості процесу обліку залишків на складі автозапчастин	12
1.2 Проблеми ручного та частково автоматизованого обліку автозапчастин	16
1.3 Аналіз існуючих програмних засобів для складського обліку	19
1.4 Порівняльна характеристика аналогів систем відстеження залишків	23
1.5 Огляд інформаційних технологій для розробки вебсистеми складського обліку	26
1.6 Постановка задачі бакалаврської роботи	30
2 ПРОЄКТУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ЗАЛИШКІВ НА СКЛАДІ АВТОЗАПЧАСТИН	33
2.1 Визначення функціональних та нефункціональних вимог до системи	35
2.2 Моделювання варіантів використання системи	40
2.3 Проєктування архітектури програмного забезпечення	46
2.4 Проєктування структури бази даних	49
2.5 Проєктування логіки складських операцій	53
2.6 Проєктування інтерфейсу користувача	56
3 РЕАЛІЗАЦІЯ СИСТЕМИ ВІДСТЕЖЕННЯ ЗАЛИШКІВ НА СКЛАДІ АВТОЗАПЧАСТИН	60
3.1 Реалізація серверної частини системи	61
3.2 Реалізація авторизації та розмежування прав доступу	65
3.3 Реалізація модуля каталогу автозапчастин	69
3.4 Реалізація модуля обліку залишків та складських операцій	70
3.5 Реалізація модуля звітності та експорту даних	71
3.6 Реалізація клієнтської частини системи	73
3.7 Опис основних екранних форм вебсистеми	77

4 ТЕСТУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ЗАЛИШКІВ НА СКЛАДІ АВТОЗАПЧАСТИН	86
4.1 Обґрунтування вибору видів тестування	86
4.2 Розробка тест-кейсів для перевірки функціональних можливостей системи	87
4.3 Перевірка нефункціональних вимог до системи	89
4.4 Аналіз результатів тестування	91
ВИСНОВКИ	93
ПЕРЕЛІК ПОСИЛАНЬ	96
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (<i>Презентація</i>)	98
ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ	102

ВСТУП

У сучасних умовах ефективного управління складськими запасами є одним із важливих чинників стабільної роботи підприємств, що займаються продажем, постачанням або обслуговуванням автомобільних запчастин. Для таких підприємств особливе значення має точність інформації про наявність товарів, їх кількість на складах, рух між складськими приміщеннями, своєчасне виявлення критичних залишків та формування звітності для прийняття управлінських рішень. Автозапчастини як предмет обліку мають низку специфічних характеристик: артикул або SKU, бренд, категорію, постачальника, закупівельну та роздрібну ціну, сумісність із певними моделями автомобілів, мінімальний допустимий залишок, а також розміщення на одному або декількох складах. Через це облік автозапчастин є складнішим, ніж звичайний облік однотипних товарів.

На практиці малі та середні склади автозапчастин часто використовують ручний облік, електронні таблиці або універсальні облікові програми, які не завжди достатньо адаптовані до потреб конкретної предметної галузі. Такий підхід може призводити до помилок під час внесення даних, несвоєчасного оновлення залишків, дублювання товарних позицій, відсутності повної історії складських операцій, складності контролю критичних залишків і недостатнього розмежування прав доступу між працівниками. У результаті підприємство може зіткнутися з ситуаціями, коли товар фактично відсутній на складі, хоча відображається як доступний у документах, або навпаки — товар є в наявності, але не врахований у системі. Такі помилки безпосередньо впливають на якість обслуговування клієнтів, швидкість виконання замовлень і фінансові показники підприємства.

Актуальність теми бакалаврської роботи зумовлена потребою у створенні спеціалізованої вебсистеми, яка дозволяє автоматизувати процес відстеження залишків автозапчастин, забезпечити централізоване зберігання даних, облік складських операцій, контроль мінімальних залишків, розмежування прав доступу

користувачів і формування звітності. Використання такої системи дозволяє зменшити кількість помилок у складському обліку, підвищити оперативність отримання інформації про залишки, спростити контроль руху товарів і забезпечити прозорість дій користувачів у системі.

Існуючі програмні рішення для складського обліку, зокрема ERP- та WMS-системи, мають широкий функціонал, проте часто є складними для впровадження, потребують додаткового налаштування, навчання персоналу або фінансових витрат, які можуть бути надмірними для малого чи середнього складу автозапчастин. Крім того, універсальні системи не завжди враховують специфіку автозапчастин, пов'язану з артикулом, брендом, категорією, постачальником і сумісністю з транспортними засобами. Тому доцільною є розробка окремої вебсистеми, орієнтованої саме на задачі обліку залишків автозапчастин.

Об'єктом дослідження є процес обліку та відстеження залишків автозапчастин на складі.

Предметом дослідження є програмні засоби автоматизації обліку товарів, складських операцій, контролю залишків і формування звітності у вебсистемі.

Метою бакалаврської роботи є підвищення точності та оперативності контролю складських залишків автозапчастин за рахунок розробки вебсистеми з каталогом товарів, ролями доступу, обліком складських операцій, контролем критичних залишків і формуванням звітів.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Дослідити особливості предметної галузі складського обліку автозапчастин і визначити основні проблеми, які виникають під час контролю залишків товарів.
2. Провести аналіз існуючих програмних рішень для складського обліку, визначити їх переваги, недоліки та обмеження щодо використання на складі автозапчастин.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Перший розділ бакалаврської роботи присвячено аналізу предметної галузі, у межах якої здійснюється розробка системи відстеження залишків на складі автозапчастин. Для обґрунтування доцільності створення програмного забезпечення необхідно визначити особливості складського обліку автозапчастин, основні проблеми, що виникають під час контролю залишків, а також вимоги до інформаційних систем, які використовуються для автоматизації таких процесів.

Складський облік автозапчастин має низку специфічних особливостей порівняно зі звичайним обліком товарів. Кожна товарна позиція може характеризуватися артикулом або SKU, брендом, категорією, постачальником, закупівельною та роздрібною ціною, мінімальним залишком, а також сумісністю з певними марками чи моделями автомобілів. Крім того, одна й та сама автозапчастина може зберігатися на різних складах або в різних складських зонах, що вимагає точного обліку кількості товарів у розрізі місць зберігання. Через це система обліку повинна забезпечувати не лише збереження інформації про товари, але й контроль їх руху, актуальності залишків і своєчасного виявлення дефіцитних позицій.

У межах даного розділу буде розглянуто типові підходи до організації складського обліку автозапчастин, зокрема ручний облік, використання електронних таблиць, універсальних ERP-систем і спеціалізованих inventory-рішень. Окрему увагу буде приділено проблемам, які виникають у разі недостатньої автоматизації: помилкам під час введення даних, дублюванню товарних позицій, несвоєчасному оновленню залишків, відсутності журналу складських операцій, складності контролю критичних запасів і недостатньому розмежуванню прав доступу між користувачами.

Також у розділі буде проведено аналіз існуючих програмних продуктів, які можуть застосовуватися для вирішення задач складського обліку. Доцільно розглянути як універсальні системи управління запасами, так і рішення,

орієнтовані на малий та середній бізнес. Порівняння таких систем дозволить визначити їхні переваги та недоліки, а також обґрунтувати актуальність розробки власної вебсистеми для відстеження залишків автозапчастин.

Для наочності результати аналізу предметної галузі та аналогів доцільно супроводити структурною схемою процесу складського обліку, прикладами інтерфейсів існуючих програмних рішень і порівняльною таблицею їх функціональних можливостей. Це дозволить не лише описати проблему теоретично, але й показати, які саме вимоги повинна враховувати майбутня система.



Рис. 1.1 Узагальнена схема процесу обліку залишків автозапчастин на складі

1.1 Особливості процесу обліку залишків на складі автозапчастин

Для складу автозапчастин облік залишків не зводиться лише до підрахунку кількості товарів. На практиці важливо точно визначити, яка саме деталь є в наявності, на якому складі вона розміщена, до яких автомобілів підходить, хто є її постачальником, яка її закупівельна та роздрібна ціна, а також чи не досягла вона

критичного мінімального залишку. Саме тому процес обліку автозапчастин потребує більш детальної структури даних, ніж облік звичайних товарів широкого вжитку.

Особливість автозапчастин полягає в тому, що схожі за назвою товари можуть суттєво відрізнятися за характеристиками. Наприклад, масляний фільтр, гальмівні колодки або ремінь ГРМ можуть мати однакове призначення, але різні артикули, виробників, розміри, сумісність із моделями автомобілів і ціну. Через це назви товару недостатньо для точного обліку. У системі необхідно використовувати унікальний ідентифікатор товарної позиції, зокрема SKU або артикул, який дозволяє відрізнити одну запчастину від іншої навіть у випадку подібних назв.

На складі автозапчастин велике значення має правильне ведення каталогу товарів. Каталог повинен містити не лише назву деталі, а й додаткові характеристики: категорію, бренд, постачальника, одиницю виміру, закупівельну ціну, роздрібну ціну, мінімальний залишок і опис сумісності з транспортними засобами. Така деталізація потрібна для швидкого пошуку товару, уникнення дублювання позицій і зменшення кількості помилок під час виконання складських операцій.

Окремою проблемою є облік товарів у розрізі складів. Якщо підприємство має декілька складів або хоча б декілька зон зберігання, загальної кількості товару вже недостатньо. Одна й та сама автозапчастина може бути доступна на центральному складі, але бути відсутньою в магазині або сервісній зоні. Тому система повинна показувати не тільки загальний залишок, а й кількість конкретної позиції на кожному складі окремо. Це дає змогу швидше приймати рішення щодо продажу, переміщення або поповнення товару.

Процес обліку залишків складається з кількох послідовних дій. Спочатку товар надходить від постачальника. Після цього працівник перевіряє, чи існує така позиція в каталозі. Якщо товар уже заведений у систему, оновлюється його кількість на відповідному складі. Якщо позиції ще немає, вона створюється в каталозі з усіма необхідними характеристиками. Надалі з товаром можуть

виконуватися складські операції: надходження, списання, переміщення між складами, повернення або коригування за результатами інвентаризації.

Важливо, щоб кожна складська операція змінювала залишок товару не довільно, а за чіткими правилами. Наприклад, операція надходження збільшує кількість товару на складі, списання — зменшує, переміщення одночасно зменшує залишок на одному складі та збільшує на іншому, а коригування використовується тоді, коли фактична кількість товару не збігається з обліковою. При цьому система не повинна дозволяти списання більшої кількості товару, ніж є в наявності, оскільки це призводить до появи від'ємних залишків і робить облік недостовірним.

Основні об'єкти, які використовуються в процесі складського обліку автозапчастин, наведено в таблиці 1.1.

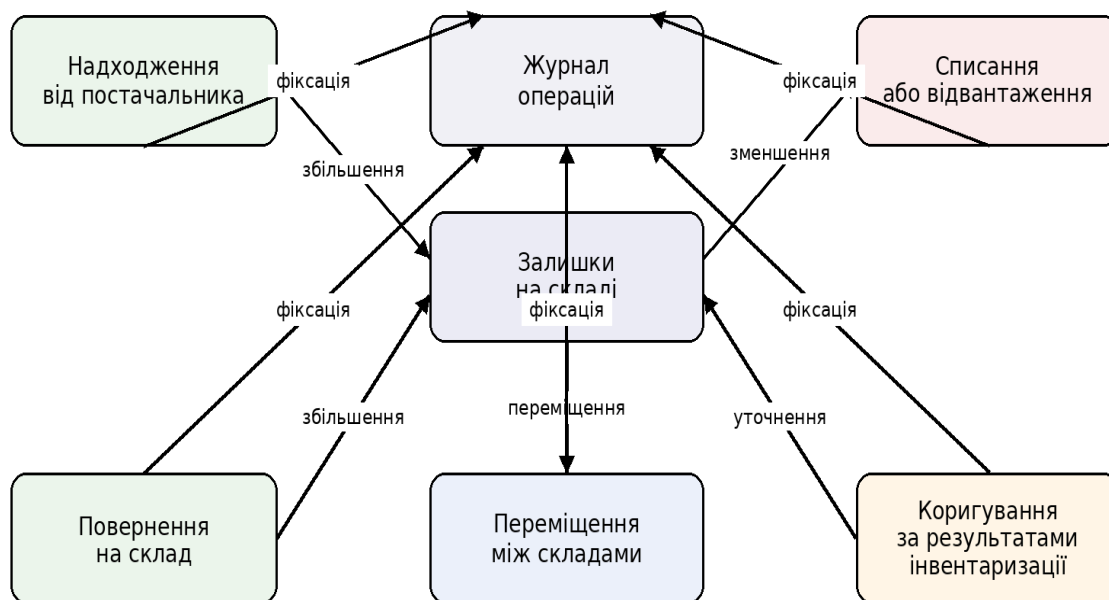
Таблиця 1.1

Основні інформаційні об'єкти складського обліку автозапчастин

Інформаційний об'єкт	Призначення в системі обліку
Товар	Містить основні дані про автозапчастину: SKU, назву, опис, ціну, одиницю виміру та мінімальний залишок
Категорія	Дає змогу згрупувати товари за типом: фільтри, підвіска, електрика, гальмівна система тощо
Бренд	Визначає виробника або торгову марку автозапчастини
Постачальник	Зберігає інформацію про джерело надходження товару
Склад	Визначає місце фізичного зберігання автозапчастин
Залишок	Показує кількість конкретного товару на конкретному складі
Складська операція	Фіксує рух товару: надходження, списання, повернення, переміщення або коригування
Користувач	Виконує дії в системі відповідно до своєї ролі та прав доступу

Наявність таких об'єктів дозволяє будувати систему не як просту таблицю товарів, а як повноцінну модель складського процесу. Наприклад, кількість товару не повинна зберігатися тільки в картці автозапчастини, оскільки одна й та сама позиція може бути розміщена на різних складах. Доцільніше зберігати залишок окремо - як зв'язок між товаром і складом. Це дозволяє отримувати точні дані про розміщення товарів і виконувати переміщення між складами без втрати інформації.

Ще одним важливим елементом є контроль мінімальних залишків. Для складу автозапчастин відсутність популярної позиції може призвести до затримки ремонту, втрати продажу або невиконання замовлення клієнта. Тому для кожного товару доцільно встановлювати мінімальний залишок. Якщо фактична кількість стає меншою за це значення, система повинна відображати товар як позицію з низьким або критичним запасом. Такий механізм допомагає своєчасно планувати закупівлі й уникати ситуацій, коли потрібна запчастина закінчується непомітно для працівників.



Складські операції змінюють кількість товару на складі та обов'язково створюють запис у журналі операцій. Надходження і повернення збільшують залишок; списання зменшує його; переміщення змінює кількість по складах; коригування уточнює дані після інвентаризації.

Рис. 1.2 Схема основних складських операцій з автозапчастинами

1.2 Проблеми ручного та частково автоматизованого обліку автозапчастин

На невеликих складах і в магазинах автозапчастин облік товарів нерідко ведеться за допомогою паперових журналів, електронних таблиць або окремих файлів, які заповнюються вручну. Такий підхід може здаватися простим на початковому етапі, коли кількість товарних позицій невелика, а склад один. Проте зі збільшенням асортименту, появою декількох постачальників, різних складів, кількох відповідальних працівників і регулярних складських операцій ручний або частково автоматизований облік стає недостатньо надійним.

Основна проблема ручного обліку полягає в тому, що актуальність залишків повністю залежить від уважності працівника. Якщо під час надходження, списання або переміщення автозапчастини запис не було внесено одразу, фактична кількість товару на складі починає відрізнятися від облікової. У результаті працівник може бачити в таблиці, що певна деталь є в наявності, але фактично вона вже продана, переміщена або списана. Для складу автозапчастин це особливо критично, оскільки клієнту часто потрібна конкретна деталь із певним артикулом, брендом або сумісністю з автомобілем.

Частково автоматизований облік за допомогою Excel або Google Sheets зменшує кількість паперової роботи, але не вирішує всі проблеми. Електронні таблиці зручні для простого списку товарів, проте вони не завжди підходять для повноцінного обліку складських операцій. У таблиці складно одночасно підтримувати каталог товарів, залишки по різних складах, історію операцій, ролі користувачів, контроль мінімальних залишків і звітність. Якщо файл редагують кілька працівників, зростає ризик випадкового видалення даних, дублювання записів або внесення змін без зрозумілої історії.

Для автозапчастин особливо небезпечним є дублювання товарних позицій. Одна й та сама деталь може бути записана в системі кількома різними способами: за артикулом, скороченою назвою, назвою бренду або описом сумісності. Наприклад, одна позиція може бути внесена як «Фільтр масляний Bosch», друга —

як «Bosch oil filter», третя — як «Масляний фільтр BOSCH 0986...». Якщо немає чіткого контролю унікального SKU або артикулу, система починає показувати декілька схожих записів замість однієї точної товарної позиції. Це ускладнює пошук, інвентаризацію та формування правильних залишків.

Ще однією проблемою є відсутність повної історії руху товару. У ручному обліку або звичайній таблиці часто фіксується лише поточна кількість, але не зберігається детальна інформація про те, як саме вона змінилася. Якщо залишок певної автозапчастини зменшився, складно визначити, що стало причиною: продаж, списання, переміщення на інший склад, повернення постачальнику або помилка під час інвентаризації. Без журналу складських операцій важко знайти відповідального користувача, перевірити дату зміни та відновити реальну картину руху товару.

Проблеми також виникають під час роботи з декількома складами. У таблицях часто ведеться або загальний залишок, або окремі аркуші для кожного складу. Обидва підходи мають недоліки. Якщо використовується загальний залишок, працівник не бачить, де саме знаходиться товар. Якщо для кожного складу створено окремий аркуш, виникає ризик, що під час переміщення товару кількість буде змінена тільки в одному місці. Наприклад, працівник може зменшити залишок на центральному складі, але забути збільшити його на складі магазину. У результаті система не відображає реального розміщення товару.

Окремо слід виділити проблему контролю критичних залишків. У ручному обліку відповідальний працівник має самостійно переглядати список товарів і визначати, які позиції потребують поповнення. Якщо асортимент складається з сотень або тисяч автозапчастин, такий контроль стає малоефективним. Працівник може вчасно не помітити, що популярна позиція закінчується, і замовлення постачальнику буде зроблено із запізненням. Для магазину або сервісного центру це означає втрату продажу, затримку ремонту або необхідність терміново шукати запчастину в іншого постачальника.



Рис. 1.3 Основні ризики ручного та частково автоматизованого обліку автозапчастин

Таблиця 1.2

Проблеми ручного та частково автоматизованого обліку автозапчастин

Проблема	Прояв у роботі складу	Наслідок для підприємства
Помилки ручного введення	Працівник неправильно вносить кількість, артикул, назву або ціну товару	Облікові дані не відповідають фактичним залишкам
Дублювання товарних позицій	Одна й та сама автозапчастина записується під різними назвами або артикулами	Ускладнюється пошук, інвентаризація та контроль залишків
Несвоєчасне оновлення залишків	Операція виконана фактично, але не внесена до таблиці або журналу	Працівники бачать застарілу інформацію про наявність товару
Відсутність обліку по складах	Зберігається лише загальна кількість товару без прив'язки до місця зберігання	Складно визначити, де саме знаходиться потрібна автозапчастина
Відсутність журналу операцій	Не фіксується історія змін залишку та відповідальний працівник	Неможливо швидко знайти причину розбіжностей
Слабкий контроль доступу	Усі користувачі можуть змінювати однакові дані	Зростає ризик випадкових або несанкціонованих змін
Відсутність автоматичного контролю критичних залишків	Працівник вручну перевіряє, які товари потрібно дозамовити	Популярні позиції можуть закінчитися непомітно
Складність формування звітності	Дані потрібно збирати вручну з різних таблиць або аркушів	Звіти формуються повільно і можуть містити помилки

1.3 Аналіз існуючих програмних засобів для складського обліку

Для обґрунтування доцільності розробки системи відстеження залишків на складі автозапчастин необхідно розглянути програмні засоби, які вже використовуються для автоматизації складського обліку. Такі системи можуть мати різний масштаб: від простих мобільних додатків для інвентаризації до комплексних ERP- і WMS-рішень, які охоплюють закупівлі, продажі, склади, логістику, фінанси та звітність. Проте для малого або середнього складу автозапчастин важливо не лише мати широкий набір функцій, а й отримати зрозумілий інструмент, який не потребує надмірно складного впровадження.

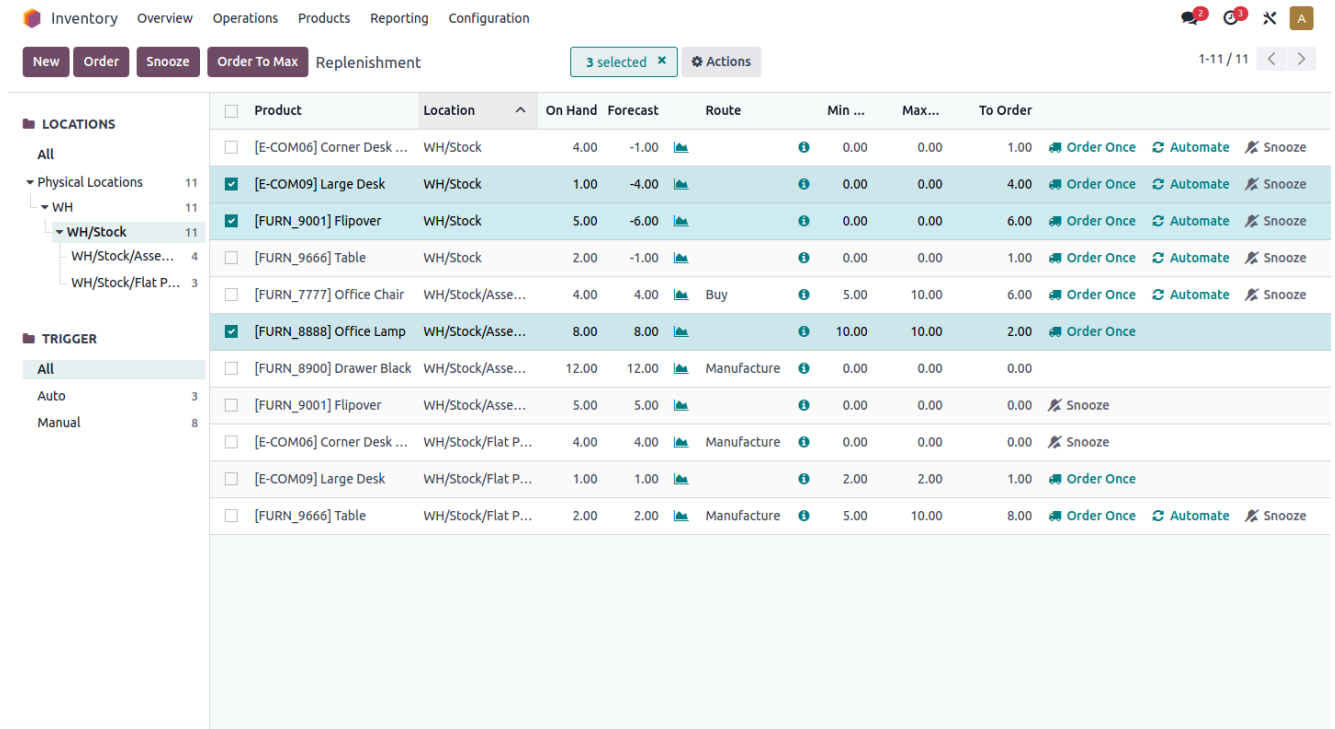
Під час аналізу аналогів доцільно враховувати такі критерії: наявність каталогу товарів, підтримка SKU або артикулів, облік залишків по складах, виконання складських операцій, контроль мінімальних залишків, формування звітів, робота з ролями користувачів, можливість експорту даних, а також придатність системи до задач саме складу автозапчастин. Окрему увагу слід звернути на те, чи дозволяє система зручно працювати з великою кількістю схожих товарних позицій, оскільки для автозапчастин це є типовою ситуацією.

Одним із відомих рішень для складського обліку є Odoo Inventory. Це модуль екосистеми Odoo, який поєднує можливості inventory-системи та складської системи управління. Odoo Inventory підтримує роботу з кількома складами, операції переміщення між складами, поповнення запасів, роботу зі штрихкодами, серійними номерами та партіями товарів [1].

Для підприємств, які мають складну структуру процесів, Odoo є досить потужним рішенням. Його перевагою є інтеграція з іншими модулями: продажами, закупівлями, виробництвом, бухгалтерією та CRM. Це дозволяє будувати єдину інформаційну систему підприємства, у якій складський облік пов'язаний з іншими бізнес-процесами.

Однак для малого складу автозапчастин Odoo може бути надмірно складним. Система має велику кількість модулів і налаштувань, тому для її повноцінного використання часто потрібні адаптація, налаштування

бізнес-процесів і навчання персоналу. Якщо підприємству потрібно лише вести каталог автозапчастин, контролювати залишки, виконувати складські операції та формувати базові звіти, використання комплексної ERP-системи може бути занадто об'ємним рішенням.



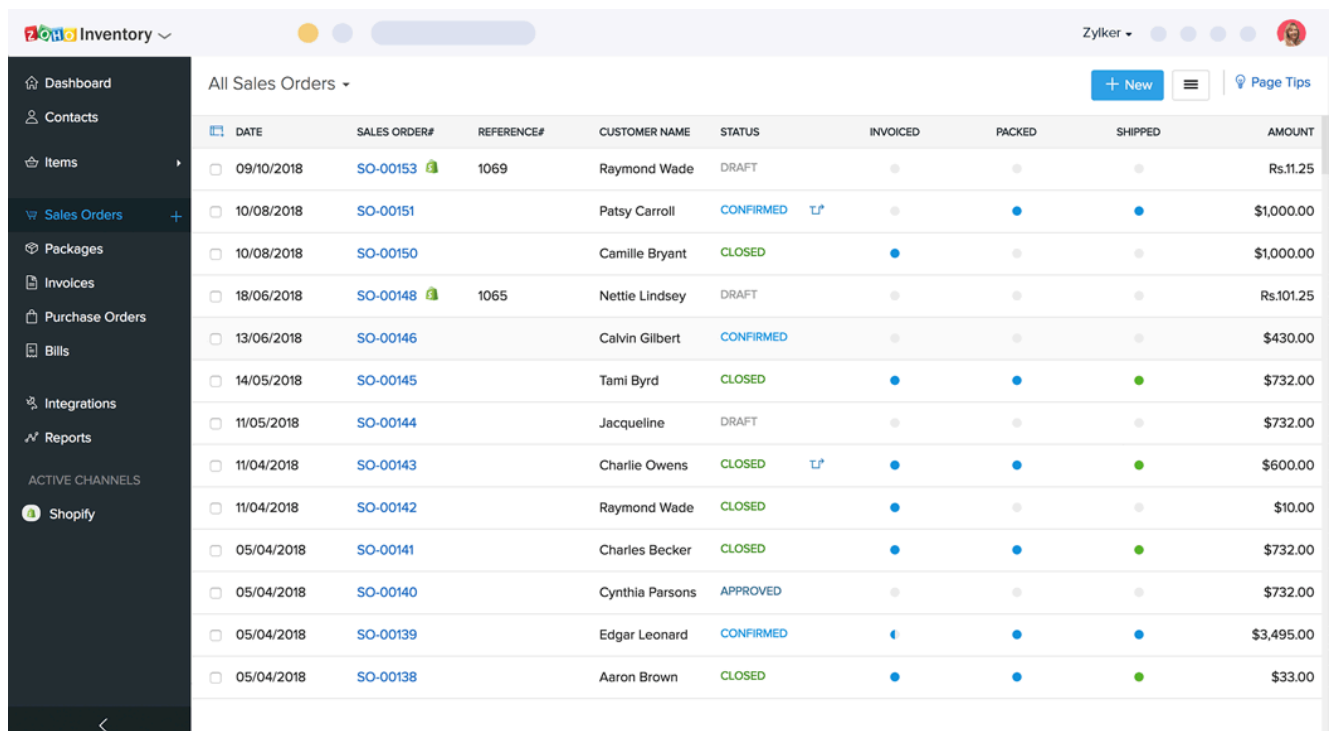
Product	Location	On Hand	Forecast	Route	Min ...	Max...	To Order	
[E-COM06] Corner Desk ...	WH/Stock	4.00	-1.00		0.00	0.00	1.00	Order Once Automate Snooze
[E-COM09] Large Desk	WH/Stock	1.00	-4.00		0.00	0.00	4.00	Order Once Automate Snooze
[FURN_9001] Flipover	WH/Stock	5.00	-6.00		0.00	0.00	6.00	Order 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. 1.4 Система Odoo Inventory

Іншим поширеним програмним продуктом є Zoho Inventory. Це хмарна система для управління запасами, замовленнями, складами та звітністю. Вона підтримує відстеження запасів, штрихкоди та RFID, роботу з партіями і серійними номерами, повідомлення про низький залишок, коригування запасів, звіти, а також ролі та дозволи користувачів [2].

Перевагою Zoho Inventory є те, що система орієнтована на бізнес-процеси, пов'язані з продажами, закупівлями та складом. Вона дає змогу працювати з товарами, відстежувати залишки, формувати звіти й отримувати повідомлення про необхідність поповнення запасів. Також у Zoho Inventory є складські звіти, які показують залишки товарів по складах і дозволяють фільтрувати дані за різними параметрами.

Водночас Zoho Inventory є хмарним сервісом, що може бути недоліком для підприємств, які хочуть мати локальне розгортання або повний контроль над власною базою даних. Для невеликого складу автозапчастин важливим може бути не лише доступ до готового SaaS-рішення, а й можливість адаптувати структуру даних під власні потреби: наприклад, зберігати сумісність із автомобілями, внутрішні категорії, спеціальні правила для мінімальних залишків або власну модель ролей.



The screenshot displays the Zoho Inventory web application. On the left is a dark sidebar with navigation links: Dashboard, Contacts, Items, Sales Orders (highlighted), Packages, Invoices, Purchase Orders, Bills, Integrations, Reports, and Active Channels (Shopify). The main area shows a table titled 'All Sales Orders' with columns: DATE, SALES ORDER#, REFERENCE#, CUSTOMER NAME, STATUS, INVOICED, PACKED, SHIPPED, and AMOUNT. The table contains 15 rows of data, including sales orders for Raymond Wade, Patsy Carroll, Camille Bryant, Nettie Lindsey, Calvin Gilbert, Tami Byrd, Jacqueline, Charlie Owens, Raymond Wade, Charles Becker, Cynthia Parsons, Edgar Leonard, and Aaron Brown.

DATE	SALES ORDER#	REFERENCE#	CUSTOMER NAME	STATUS	INVOICED	PACKED	SHIPPED	AMOUNT
09/10/2018	SO-00153	1069	Raymond Wade	DRAFT				Rs.11.25
10/08/2018	SO-00151		Patsy Carroll	CONFIRMED				\$1,000.00
10/08/2018	SO-00150		Camille Bryant	CLOSED				\$1,000.00
18/06/2018	SO-00148	1065	Nettie Lindsey	DRAFT				Rs.101.25
13/06/2018	SO-00146		Calvin Gilbert	CONFIRMED				\$430.00
14/05/2018	SO-00145		Tami Byrd	CLOSED				\$732.00
11/05/2018	SO-00144		Jacqueline	DRAFT				\$732.00
11/04/2018	SO-00143		Charlie Owens	CLOSED				\$600.00
11/04/2018	SO-00142		Raymond Wade	CLOSED				\$10.00
05/04/2018	SO-00141		Charles Becker	CLOSED				\$732.00
05/04/2018	SO-00140		Cynthia Parsons	APPROVED				\$732.00
05/04/2018	SO-00139		Edgar Leonard	CONFIRMED				\$3,495.00
05/04/2018	SO-00138		Aaron Brown	CLOSED				\$33.00

Рис. 1.5 Система Zoho Inventory

Ще одним прикладом системи для інвентаризації є Sortly. Це програмний засіб, який більше орієнтований на просте відстеження запасів, обладнання, інструментів та інших об'єктів. Sortly підтримує сканування штрихкодів і QR-кодів, додавання фотографій до товарів, повідомлення про низький залишок і звітність [3].

Сильною стороною Sortly є простота використання. Такий інструмент може бути зручним для невеликих складів, сервісних команд або підприємств, де потрібно швидко бачити, які об'єкти є в наявності. Підтримка QR-кодів і

штрихкодів дає змогу прискорити перевірку товарів, а функція повідомлень про низький залишок допомагає контролювати запаси.

Недоліком Sortly для складу автозапчастин є те, що система більше схожа на універсальний інструмент інвентаризації, ніж на повноцінну спеціалізовану систему складських операцій. Вона добре підходить для простого обліку предметів, але для задач автозапчастин можуть знадобитися більш чіткі бізнес-процеси: надходження від постачальника, списання, повернення, переміщення, журнал операцій, ролі працівників, звіти щодо вартості залишків і контроль унікальності SKU.

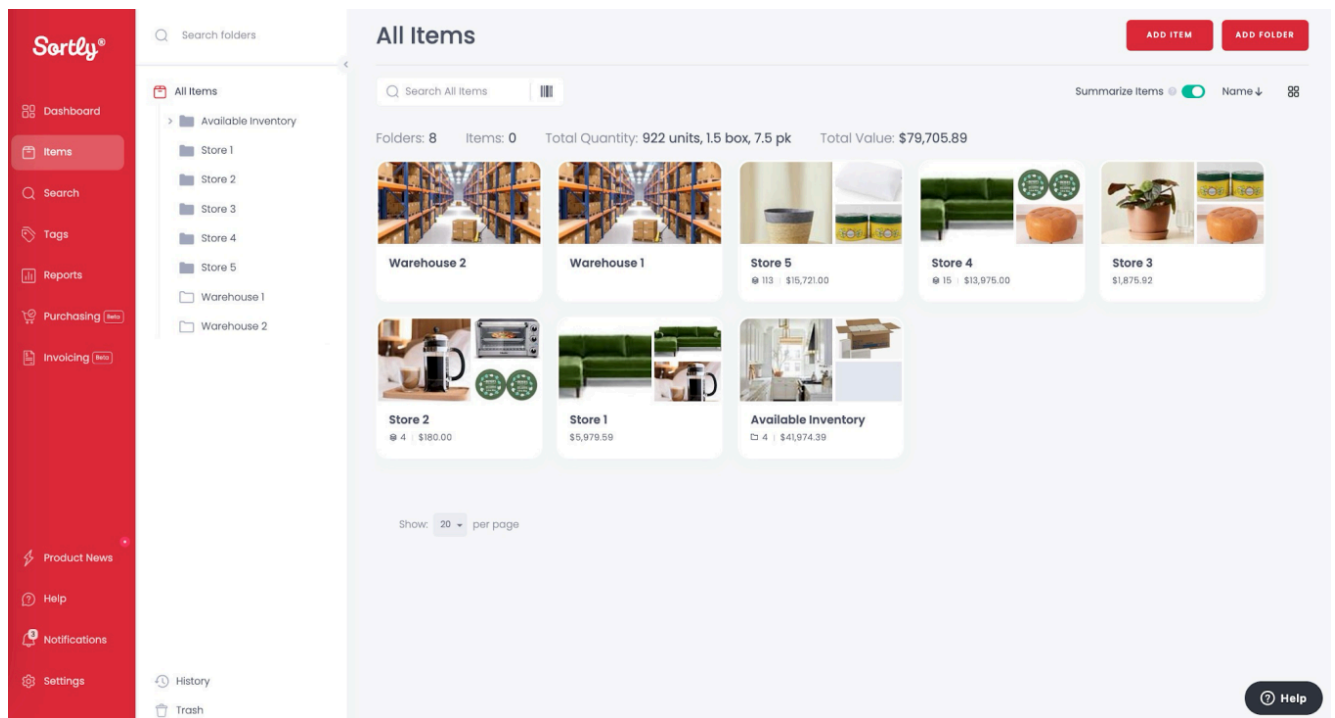


Рис. 1.6 Система Sortly

Для малого та середнього бізнесу також використовується inFlow Inventory. Це система, орієнтована на управління запасами, замовленнями, складом, штрихкодами та звітністю. Офіційні матеріали inFlow описують систему як інструмент для оптимізації контролю запасів, відстеження замовлень і оновлення інформації про склад у реальному часі [4].

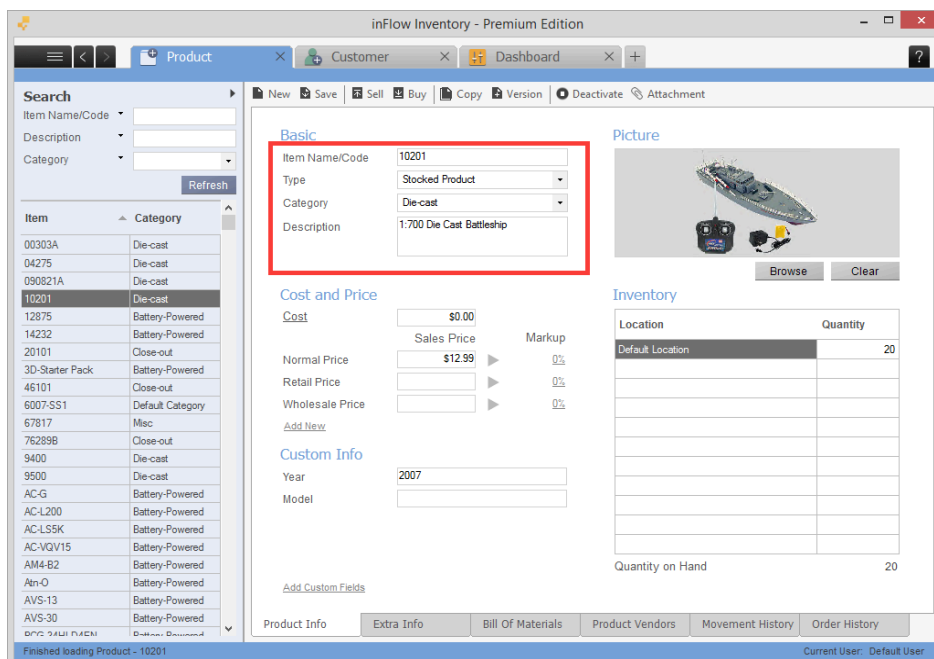


Рис. 1.7 Система inFlow Inventory

1.4 Порівняльна характеристика аналогів систем відстеження залишків

Після розгляду окремих програмних засобів для складського обліку доцільно узагальнити їхні можливості у вигляді порівняльної характеристики.

Для порівняння було обрано чотири програмні засоби: Odoo Inventory, Zoho Inventory, Sortly та inFlow Inventory. Вони належать до різних типів рішень. Odoo Inventory є частиною ширшої ERP-екосистеми, Zoho Inventory працює як хмарна система для управління запасами та замовленнями, Sortly орієнтований на просту інвентаризацію й візуальний облік, а inFlow Inventory спрямований на малий і середній бізнес, який потребує контролю товарів, складів і замовлень. Для зіставлення з ними додано розроблену систему APW Inventory, яка орієнтована саме на склад автозапчастин.

Порівняння здійснюється за критеріями, які є важливими для теми бакалаврської роботи: ведення каталогу товарів, підтримка SKU або артикулів, облік залишків по складах, складські операції, контроль мінімальних залишків, формування звітів, робота з ролями користувачів, можливість експорту даних, а також придатність системи до задач саме складу автозапчастин.

Таблиця 1.3

Порівняльна характеристика програмних засобів для складського обліку

Критерій порівняння	Odoo Inventory	Zoho Inventory	Sortly	inFlow Inventory	APW Inventory
Ведення каталогу товарів	+	+	+	+	+
Підтримка SKU / артикулу	+	+	+	+	+
Категорії товарів	+	+	+	+	+
Бренди товарів	+ / через налаштування	+ / через налаштування	частково	+ / через налаштування	+
Постачальники	+	+	частково	+	+
Облік залишків по складах	+	+	+ / за локаціями	+	+
Переміщення між складами	+	+	частково	+	+
Надходження товару	+	+	частково	+	+
Списання товару	+	+	частково	+	+
Коригування залишків	+	+	+	+	+
Повернення товару	+	+	частково	+	+
Контроль критичних залишків	+	+	+	+	+
Журнал складських операцій	+	+	частково	+	+
Ролі користувачів	+	+	залежить від тарифу	+	+
Формування звітів	+	+	+	+	+
Експорт даних	+	+	+	+	+
Локальне розгортання	можливе, але складне	-	-	обмежено	+
Орієнтація саме на автозапчастини	потребує адаптації	потребує адаптації	потребує адаптації	потребує адаптації	+
Простота впровадження для малого складу	середня	середня	висока	середня	висока
Основний недолік	складність ERP-рішення	залежність від хмарного сервісу	обмежені складські операції	залежність від комерційної моделі	потребує подальшого розвитку як MVP

Позначення в таблиці: «+» - функція підтримується; «-» - функція відсутня або не є типовою для рішення; «частково» - функція доступна, але не повністю відповідає задачам складського обліку автозапчастин; «потребує адаптації» - функція може бути реалізована через налаштування, але не є спеціалізованою для цієї предметної галузі.

З таблиці видно, що більшість розглянутих програмних засобів підтримують базові функції складського обліку: каталог товарів, SKU, залишки, звіти та експорт даних. Це означає, що такі можливості є необхідним мінімумом для сучасної системи управління запасами. Водночас для складу автозапчастин важливо не лише зберігати перелік товарів, а й працювати з більш деталізованою структурою: категоріями, брендами, постачальниками, сумісністю з автомобілями, мінімальними залишками та рухом товарів між складами.

Odoo Inventory має найбільш широкий набір можливостей, оскільки входить до комплексної ERP-платформи. Це робить систему функціонально сильною, але водночас складнішою для впровадження. Для малого складу автозапчастин така система може виявитися надмірною, оскільки потребує налаштування, адміністрування та адаптації бізнес-процесів. Її використання доцільне для підприємств, які хочуть автоматизувати не лише склад, а й закупівлі, продажі, бухгалтерію та інші процеси.

Zoho Inventory також має достатній набір функцій для обліку запасів, складів і замовлень. Її перевагою є готова хмарна інфраструктура, що дозволяє швидко почати роботу без розгортання власного сервера. Однак саме хмарна модель може бути недоліком для підприємств, яким потрібен локальний контроль над даними або можливість змінювати систему під власні внутрішні правила. Для складу автозапчастин це важливо, оскільки структура товару може містити специфічні поля, які не завжди зручно налаштовувати у готовому сервісі.

Sortly відрізняється простотою та зручністю візуального обліку. Він добре підходить для швидкої інвентаризації, роботи з фотографіями товарів, QR-кодами та локаціями. Проте для повноцінного обліку складських операцій автозапчастин його можливостей може бути недостатньо. У такій системі складно побудувати

повний цикл руху товару з надходженням, списанням, поверненням, переміщенням, коригуванням, журналом операцій і розмежуванням ролей у тому вигляді, який потрібний для спеціалізованого складського процесу.

inFlow Inventory є ближчим до задач малого та середнього бізнесу, оскільки орієнтований саме на управління запасами. Його можна розглядати як один із найбільш придатних аналогів для порівняння з розробленою системою. Він підтримує складський облік, штрихкоди, замовлення та звітність. Водночас inFlow залишається готовим комерційним продуктом із власними правилами використання, тарифами та обмеженнями. Для підприємства, якому потрібна система з можливістю локального розгортання й адаптації до власної структури автозапчастин, це може бути недоліком.

Розроблювана система APW Inventory не є повноцінною ERP- або WMS-платформою промислового рівня. Її доцільно розглядати як спеціалізовану вебсистему для малого або середнього складу автозапчастин. Її перевага полягає не в найбільшій кількості функцій, а в поєднанні саме тих можливостей, які потрібні для обраної предметної галузі: каталог автозапчастин із SKU, брендом, категорією, постачальником і сумісністю з автомобілями; облік залишків по складах; операції надходження, списання, переміщення, повернення та коригування; контроль критичних залишків; журнал транзакцій; ролі користувачів; звіти та експорт CSV.

1.5 Огляд інформаційних технологій для розробки вебсистеми складського обліку

Для розробки системи відстеження залишків на складі автозапчастин доцільно використовувати клієнт-серверну архітектуру, у якій логіка роботи системи поділяється між серверною частиною, клієнтським вебінтерфейсом і базою даних. Такий підхід добре підходить для складського обліку, оскільки декілька користувачів можуть працювати з єдиними даними, а основні правила обробки складських операцій виконуються централізовано на сервері.

У межах розроблюваної системи серверна частина відповідає за обробку запитів, перевірку прав доступу, виконання бізнес-логіки, роботу з товарами, складами, залишками, складськими операціями, користувачами та звітами. Клієнтська частина забезпечує зручну взаємодію користувача із системою через веббраузер. База даних використовується для збереження інформації про автозапчастини, довідники, склади, залишки, транзакції та користувачів.

Для серверної частини обрано мову програмування Java. Java є поширеною мовою для створення корпоративних і вебзастосунків, оскільки підтримує об'єктно-орієнтований підхід, має розвинену екосистему бібліотек і добре підходить для розробки серверної бізнес-логіки. Для проєкту складського обліку це важливо, оскільки система повинна коректно обробляти операції надходження, списання, переміщення, повернення та коригування залишків. У проєкті використовується Java 21, що є сучасною версією платформи Java для серверної розробки.

Основним фреймворком серверної частини є Spring Boot. Його використання дозволяє швидше створювати вебзастосунки на Java, налаштовувати REST API, підключати базу даних, реалізовувати безпеку та організовувати структуру проєкту. Для системи складського обліку це дає можливість розділити програмну логіку на контролери, сервіси, репозиторії та моделі даних.

Для взаємодії з базою даних у серверній частині використовується Spring Data JPA. Цей інструмент спрощує роботу з реляційними даними, оскільки дозволяє описувати сутності системи у вигляді Java-класів, а операції збереження, пошуку, оновлення та видалення виконувати через репозиторії. Для предметної галузі складського обліку такий підхід є зручним, оскільки основні сутності системи мають чітку структуру: товар, категорія, бренд, постачальник, склад, залишок, складська операція та користувач.

Для реалізації захисту системи використовується Spring Security у поєднанні з JWT-токенами. Система складського обліку містить дані, які не повинні бути доступні всім користувачам без обмежень. Наприклад, перегляд залишків, виконання складських операцій, видалення товарів або керування користувачами

повинні залежати від ролі працівника. Використання механізмів автентифікації та авторизації дозволяє обмежити доступ до захищених ресурсів і запобігти несанкціонованим діям.

Для збереження даних обрано PostgreSQL. Це об'єктно-реляційна система керування базами даних, яка добре підходить для зберігання структурованої інформації та підтримує роботу зі зв'язками між таблицями. Для системи обліку автозапчастин реляційна модель є доцільною, оскільки дані мають багато взаємозв'язків: товар належить до категорії, має бренд і постачальника, залишок пов'язаний одночасно з товаром і складом, а складська операція фіксує зміну кількості товару.

Важливим елементом роботи з базою даних є керування її структурою. Для цього використовується Flyway. Він дозволяє зберігати зміни структури бази даних у вигляді міграцій і застосовувати їх послідовно. Для дипломного проєкту це важливо, оскільки структура бази даних повинна бути відтворюваною: таблиці користувачів, товарів, складів, залишків і транзакцій мають створюватися однаково в різних середовищах. Міграції також спрощують подальший розвиток системи, коли потрібно додати нові поля або змінити обмеження в таблицях.

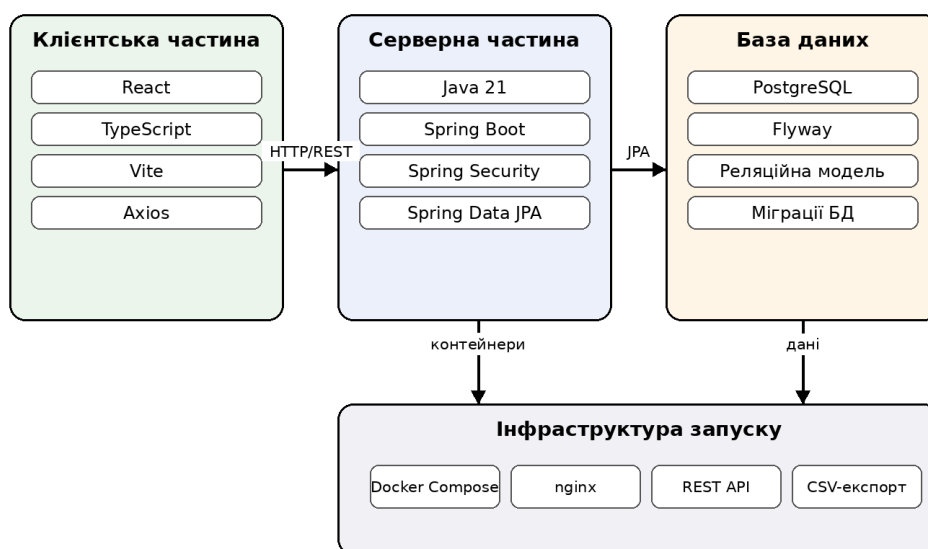
Клієнтська частина системи реалізується з використанням React. React дає змогу створювати вебінтерфейс на основі компонентів, що зручно для системи, яка має багато екранів і повторюваних елементів. Наприклад, таблиці товарів, форми додавання й редагування, фільтри, кнопки дій, блоки статистики та повідомлення можуть бути реалізовані як окремі компоненти. Це полегшує підтримку клієнтської частини і дозволяє поступово розширювати функціональність інтерфейсу.

Для підвищення надійності клієнтського коду використовується TypeScript. На відміну від звичайного JavaScript, TypeScript дозволяє явно описувати типи даних, що зменшує кількість помилок під час розробки. Для системи складського обліку це корисно, оскільки клієнтська частина працює з багатьма структурованими об'єктами: товарами, складами, залишками, користувачами,

ролями, транзакціями та звітами. Типізація допомагає уникати ситуацій, коли компонент отримує неповні або неправильні дані.

Для створення та запуску клієнтської частини використовується Vite. Він забезпечує швидке середовище розробки для сучасних frontend-проектів і підтримує шаблони для JavaScript та TypeScript. Для вебсистеми складського обліку це дає змогу швидко розробляти, перевіряти й оновлювати інтерфейс користувача під час реалізації сторінок товарів, залишків, складських операцій і звітів.

Для обміну даними між клієнтською та серверною частинами використовується Axios. Він застосовується для виконання HTTP-запитів до REST API, отримання даних із сервера, надсилання форм і передавання JWT-токена в заголовках запитів. У системі складського обліку через Axios можуть виконуватися запити на авторизацію, отримання каталогу товарів, створення складської операції, перегляд залишків, фільтрацію транзакцій і завантаження звітів.



Клієнтська частина надсилає запити до REST API. Серверна частина виконує бізнес-логіку, перевіряє доступ користувачів і працює з базою даних. PostgreSQL зберігає складські дані, а Docker Compose забезпечує запуск компонентів як пов'язаних сервісів.

Рис. 1.8 Стек технологій системи APW Inventory

Основні технології розробки системи APW Inventory

Технологія	Призначення в системі
Java 21	Реалізація серверної бізнес-логіки системи
Spring Boot	Створення REST API та організація backend-частини
Spring Data JPA	Робота з реляційними сутностями та репозиторіями
Spring Security	Реалізація автентифікації, авторизації та ролей доступу
JWT	Передавання інформації про авторизованого користувача між клієнтом і сервером
PostgreSQL	Збереження даних про товари, склади, залишки, транзакції та користувачів
Flyway	Керування міграціями структури бази даних
React	Побудова клієнтського вебінтерфейсу
TypeScript	Типізація frontend-коду та зменшення кількості помилок
Vite	Збирання та запуск клієнтської частини під час розробки
Axios	Обмін даними між frontend і backend через HTTP-запити
Docker Compose	Локальний запуск backend, frontend і бази даних як окремих сервісів

1.6 Постановка задачі бакалаврської роботи

Після розгляду особливостей складського обліку автозапчастин, проблем ручного та частково автоматизованого ведення залишків, а також аналізу існуючих програмних рішень можна сформулювати основну задачу бакалаврської роботи. Вона полягає у розробці вебсистеми, яка забезпечує централізований облік автозапчастин, контроль залишків по складах, виконання складських операцій, розмежування прав доступу користувачів і формування звітності.

Необхідність розробки такої системи пов'язана з тим, що склад автозапчастин має специфічну структуру товарів. Для кожної позиції важливо зберігати не лише назву та кількість, а й SKU, бренд, категорію, постачальника, закупівельну та роздрібну ціну, мінімальний залишок і сумісність із транспортними засобами.

Об'єктом дослідження є процес обліку та відстеження залишків автозапчастин на складі.

Предметом дослідження є програмні засоби автоматизації обліку товарів, складських операцій, контролю залишків і формування звітності у вебсистемі.

Метою бакалаврської роботи є підвищення точності та оперативності контролю складських залишків автозапчастин за рахунок розробки вебсистеми з каталогом товарів, ролями доступу, обліком складських операцій, контролем критичних залишків і формуванням звітів.

Для досягнення поставленої мети необхідно виконати такі задачі:

1. Дослідити предметну галузь складського обліку автозапчастин і визначити особливості товарів, які потрібно врахувати під час розробки системи.
2. Визначити проблеми ручного та частково автоматизованого обліку залишків автозапчастин.

Таблиця 1.5

Основні задачі бакалаврської роботи та очікувані результати

№	Задача бакалаврської роботи	Очікуваний результат
1	Дослідити предметну галузь складського обліку автозапчастин	Визначено специфіку обліку SKU, брендів, категорій, постачальників, складів і залишків
2	Проаналізувати проблеми ручного обліку	Сформовано перелік основних ризиків: помилки введення, дублювання товарів, відсутність журналу операцій
3	Розглянути існуючі програмні рішення	Визначено переваги й недоліки Odoo Inventory, Zoho Inventory, Sortly та inFlow Inventory
4	Визначити вимоги до системи	Сформовано функціональні та нефункціональні вимоги до вебсистеми
5	Спроектувати архітектуру системи	Побудовано клієнт-серверну архітектуру з frontend, backend і базою даних
6	Спроектувати базу даних	Визначено таблиці для товарів, складів, залишків, транзакцій, користувачів і довідників
7	Реалізувати серверну частину	Створено REST API для роботи з основними модулями системи
8	Реалізувати клієнтську частину	Створено вебінтерфейс для роботи з товарами, залишками, операціями та звітами
9	Реалізувати захист доступу	Забезпечено авторизацію, JWT-токени та ролі користувачів
10	Провести тестування	Перевірено основні функції системи, права доступу та коректність складських операцій

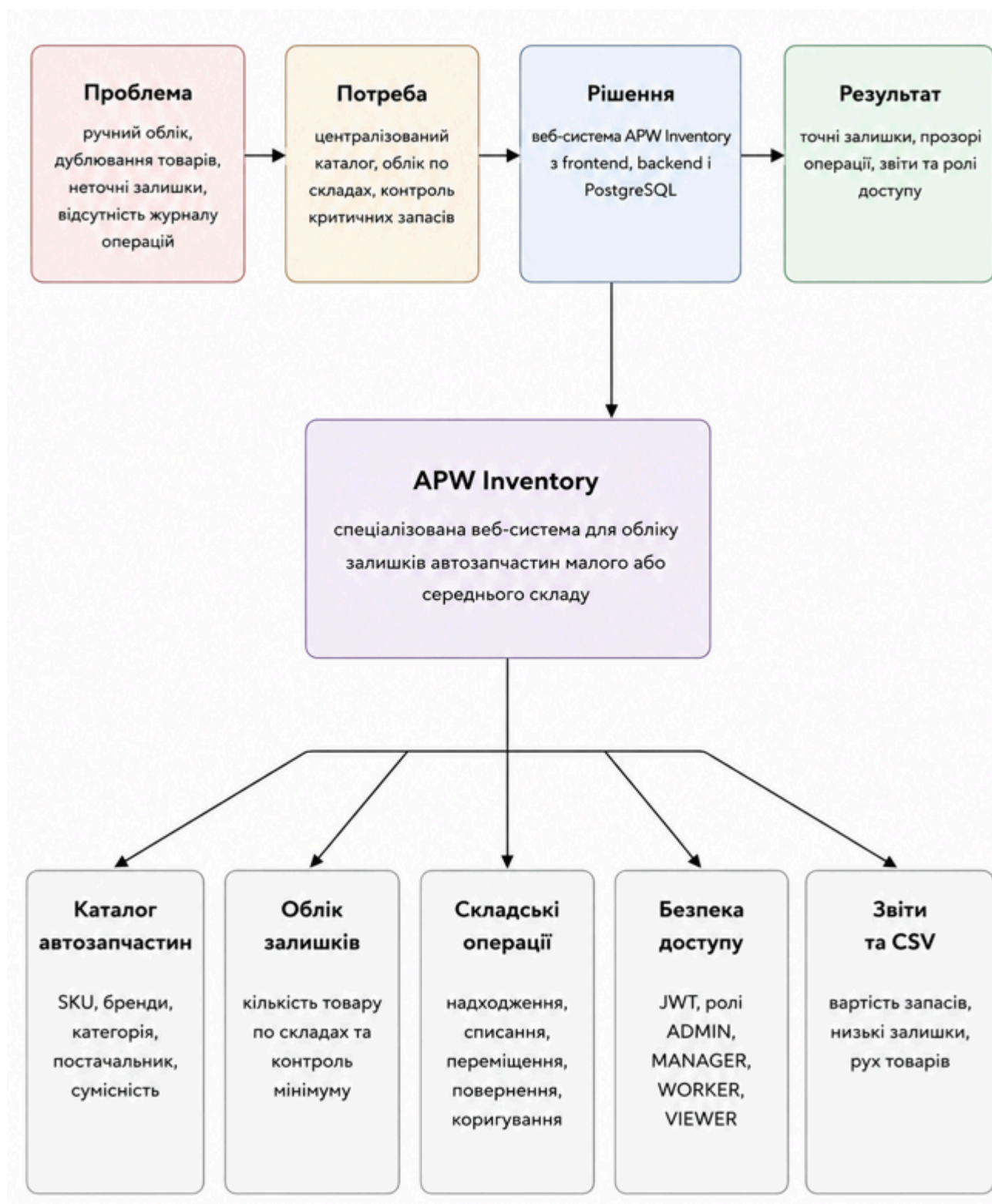


Рис. 1.9 Узагальнена постановка задачі розробки системи APW Inventory

2 ПРОЄКТУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ЗАЛИШКІВ НА СКЛАДІ АВТОЗАПЧАСТИН

Другий розділ бакалаврської роботи присвячено проєктуванню вебсистеми відстеження залишків на складі автозапчастин. На відміну від першого розділу, де було розглянуто предметну галузь, проблеми складського обліку, аналоги та технологічний стек, у цьому розділі необхідно перейти до побудови моделі майбутньої системи. Саме етап проєктування визначає, які функції повинна виконувати система, які ролі користувачів у ній передбачені, як буде організовано зберігання даних, яким чином компоненти системи взаємодіятимуть між собою та як користувач виконуватиме основні складські операції.

Система APW Inventory розглядається як клієнт-серверна вебсистема для малого або середнього складу автозапчастин. Її основним призначенням є централізоване ведення каталогу автозапчастин, контроль залишків по складах, фіксація складських операцій, виявлення критичних залишків, розмежування доступу між користувачами та формування звітності. З технічної точки зору система складається з клієнтської частини, серверної частини та бази даних. Клієнтська частина забезпечує роботу користувача через вебінтерфейс, серверна частина реалізує бізнес-логіку та API, а база даних зберігає інформацію про товари, склади, залишки, користувачів і складські транзакції.

Під час проєктування системи необхідно врахувати, що предметна галузь автозапчастин має власну специфіку. Товар у такій системі не може описуватися лише назвою та кількістю. Для коректного обліку потрібно зберігати SKU або артикул, категорію, бренд, постачальника, закупівельну й роздрібну ціну, одиницю виміру, мінімальний залишок і дані про сумісність із транспортними засобами. Крім того, залишок товару має обліковуватися не загально по системі, а в розрізі конкретних складів, оскільки одна й та сама автозапчастина може одночасно зберігатися в різних місцях.

Важливим елементом проектування є визначення ролей користувачів. У системі передбачено декілька рівнів доступу: адміністратор, менеджер, працівник складу та користувач із правами перегляду. Адміністратор має виконувати дії з керування користувачами, ролями та довідниками. Менеджер повинен мати доступ до каталогу товарів, залишків, звітів і контролю критичних запасів. Працівник складу має виконувати складські операції: надходження, списання, переміщення, повернення та коригування. Користувач із правами перегляду може отримувати інформацію про товари й залишки, але не повинен змінювати дані. Такий підхід дає змогу зменшити ризик випадкових або несанкціонованих змін у системі.

Окремої уваги потребує моделювання складських операцій. У системі повинні бути передбачені операції INCOME, OUTCOME, ADJUSTMENT, TRANSFER та RETURN. Кожна з них має змінювати залишок товару за певним правилом. Надходження збільшує кількість товару на складі, списання зменшує її, переміщення одночасно змінює кількість на двох складах, повернення фіксує надходження товару назад, а коригування використовується для уточнення залишку за результатами перевірки або інвентаризації. При цьому система не повинна дозволяти операції, що призводять до від'ємного залишку, оскільки це порушує достовірність складського обліку.

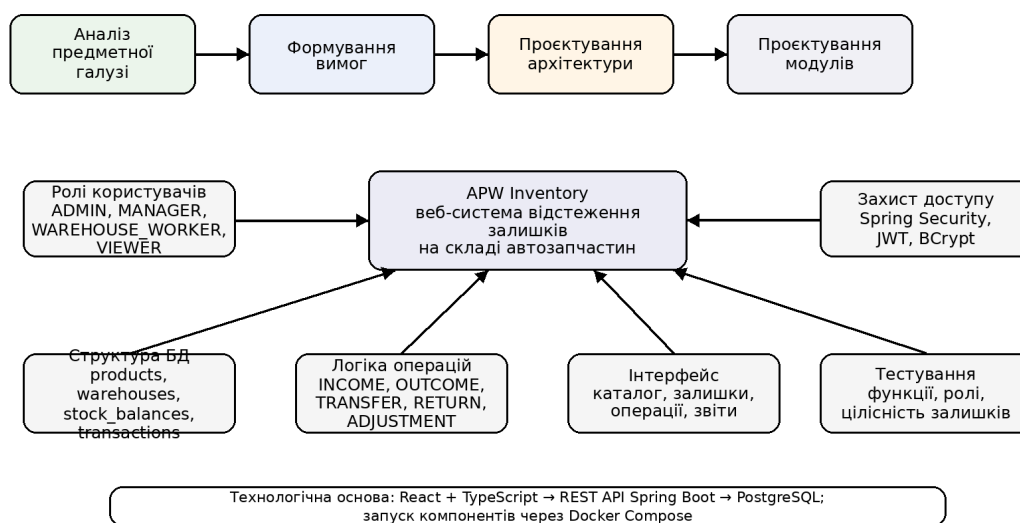


Рис. 2.1 Загальна схема проектування системи APW Inventory

2.1 Визначення функціональних та нефункціональних вимог до системи

Перед початком проєктування архітектури та структури бази даних необхідно визначити вимоги до системи відстеження залишків на складі автозапчастин. Вимоги описують, які задачі повинна виконувати система, які обмеження необхідно врахувати під час її роботи, які ролі користувачів мають бути передбачені та за якими критеріями можна буде перевірити якість розробленого програмного забезпечення.

Для системи APW Inventory вимоги формуються з урахуванням особливостей складського обліку автозапчастин. Система повинна забезпечувати ведення каталогу товарів, роботу з категоріями, брендами, постачальниками та складами, контроль залишків по складах, виконання складських операцій, формування журналу транзакцій, відображення товарів із критичними залишками, створення звітів та розмежування доступу між користувачами.

Вимоги до програмного забезпечення доцільно поділити на функціональні та нефункціональні. Функціональні вимоги описують дії, які система повинна виконувати для користувачів. Нефункціональні вимоги визначають показники якості роботи системи: швидкодію, безпеку, надійність, цілісність даних, зручність використання та умови приймання результатів розробки.

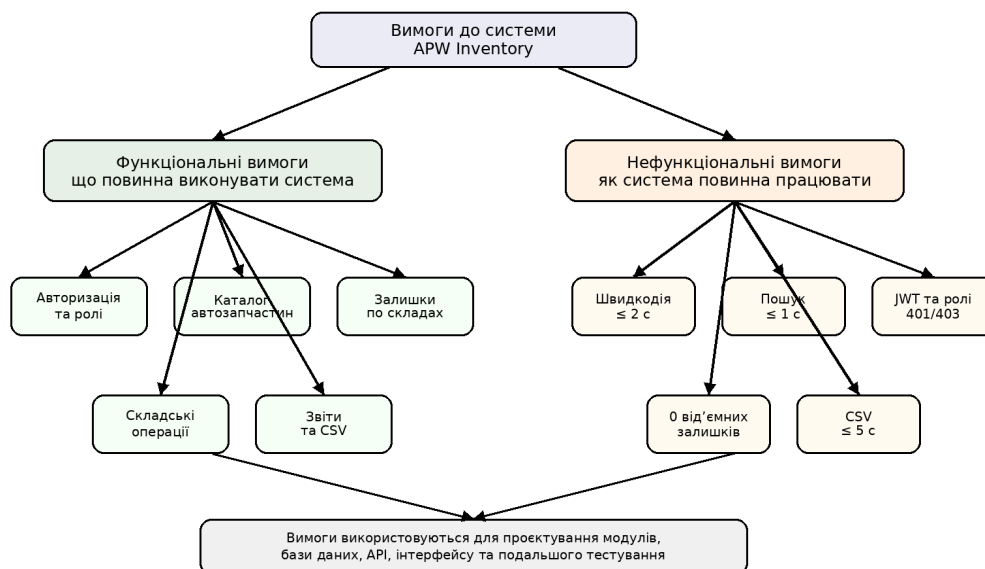


Рис. 2.2 Класифікація вимог до системи APW Inventory

Основними користувачами системи є адміністратор, менеджер, працівник складу та користувач із правами перегляду. Кожна роль має власний набір доступних дій. Такий поділ потрібний для того, щоб працівники могли виконувати лише ті операції, які відповідають їхнім обов'язкам. Наприклад, працівник складу повинен мати можливість фіксувати надходження, списання або переміщення товарів, але не повинен керувати обліковими записами інших користувачів. Користувач із правами перегляду може бачити товари й залишки, але не повинен змінювати дані.

Таблиця 2.1

Ролі користувачів системи APW Inventory

Роль користувача	Призначення ролі	Основні доступні дії
ADMIN	Адміністрування системи та керування користувачами	Керування користувачами, довідниками, товарами, складами, перегляд звітів і журналу операцій
MANAGER	Контроль товарів, залишків і звітності	Перегляд і редагування товарів, контроль критичних залишків, перегляд операцій, формування звітів
WAREHOUSE_WORKER	Виконання складських операцій	Надходження, списання, переміщення, повернення, коригування залишків
VIEWER	Перегляд інформації без права зміни даних	Перегляд товарів, залишків, складів і звітної інформації

Функціональні вимоги до системи сформовано відповідно до основних бізнес-процесів складу автозапчастин. До них належать авторизація користувачів, ведення каталогу автозапчастин, робота з довідниками, контроль залишків, виконання складських операцій, журналювання транзакцій, формування звітів і експорт даних. Узагальнений перелік функціональних вимог наведено в таблиці 2.2.

Функціональні вимоги до системи APW Inventory

Код вимоги	Функціональна вимога	Опис
F1	Авторизація користувачів	Система повинна забезпечувати вхід користувача за логіном і паролем
F2	Розмежування доступу	Система повинна підтримувати ролі ADMIN, MANAGER, WAREHOUSE_WORKER, VIEWER
F3	Керування користувачами	Адміністратор повинен мати можливість створювати, редагувати, активувати та деактивувати користувачів
F4	Ведення каталогу автозапчастин	Система повинна дозволяти створювати, переглядати, редагувати та видаляти товари
F5	Підтримка SKU	Кожна товарна позиція повинна мати унікальний SKU або артикул
F6	Робота з довідниками	Система повинна підтримувати категорії, бренди, постачальників і склади
F7	Облік залишків по складах	Система повинна відображати кількість кожного товару на конкретному складі
F8	Надходження товару	Система повинна збільшувати залишок товару після операції надходження
F9	Списання товару	Система повинна зменшувати залишок товару після операції списання
F10	Переміщення між складами	Система повинна зменшувати залишок на одному складі та збільшувати на іншому
F11	Повернення товару	Система повинна фіксувати повернення товару на склад
F12	Коригування залишків	Система повинна дозволяти коригувати залишок за результатами інвентаризації
F13	Заборона від'ємних залишків	Система не повинна дозволяти списання або переміщення кількості, більшої за доступну
F14	Журнал складських операцій	Після кожної успішної операції система повинна створювати запис у журналі транзакцій
F15	Контроль критичних залишків	Система повинна відображати товари, кількість яких менша або дорівнює мінімальному залишку
F16	Формування звітів	Система повинна формувати звіти про залишки, вартість товарів і рух запасів
F17	Експорт даних	Система повинна підтримувати експорт звітної інформації у форматі CSV
F18	Пошук і фільтрація	Система повинна дозволяти шукати товари за SKU, назвою, брендом або категорією

Особливу увагу у функціональних вимогах потрібно приділити складським операціям. У системі передбачено п'ять основних типів операцій: INCOME, OUTCOME, ADJUSTMENT, TRANSFER та RETURN. Кожна з них має власну логіку зміни залишку. Надходження збільшує кількість товару на складі, списання зменшує її, переміщення змінює залишки на двох складах, повернення збільшує кількість товару після повернення, а коригування використовується для уточнення фактичної кількості товару.

Таблиця 2.3

Логіка складських операцій у системі

Тип операції	Призначення	Вплив на залишок
INCOME	Надходження товару на склад	Збільшує залишок на вибраному складі
OUTCOME	Списання або видача товару	Зменшує залишок на вибраному складі
TRANSFER	Переміщення товару між складами	Зменшує залишок на складі-відправнику та збільшує на складі-одержувачі
RETURN	Повернення товару на склад	Збільшує залишок на вибраному складі
ADJUSTMENT	Коригування залишку	Встановлює або уточнює фактичну кількість товару

Окрім функціональних вимог, для системи необхідно визначити нефункціональні вимоги. Вони мають бути сформульовані так, щоб їх можна було перевірити під час тестування.

Тому замість загальних формулювань на зразок «система має бути швидкою» або «система має бути зручною» потрібно використовувати конкретні показники: час відкриття сторінок, час пошуку товару, допустиму кількість одночасних користувачів, час збереження операції, правила захисту API, заборону від'ємних залишків і збереження даних після перезапуску.

Нефункціональні вимоги до системи APW Inventory

Код вимоги	Нефункціональна вимога	Критерій приймання
NF1	Час відкриття сторінок «Товари», «Залишки», «Операції», «Звіти» має бути не більше 2 с	При 1000 товарних позицій кожна сторінка відкривається за ≤ 2 с
NF2	Пошук товару за SKU, назвою, брендом або категорією має виконуватися не більше 1 с	При 1000 товарних позицій результат пошуку повертається за ≤ 1 с
NF3	Збереження складської операції має виконуватися не більше 1,5 с	Операції INCOME, OUTCOME, TRANSFER, RETURN, ADJUSTMENT зберігаються за $\leq 1,5$ с
NF4	Система має підтримувати не менше 10 одночасних користувачів	10 користувачів одночасно переглядають залишки та виконують дозволені дії без помилок 5xx
NF5	Система не повинна дозволяти від'ємний залишок товару	У 100% випадків списання кількості, більшої за доступну, блокується
NF6	Доступ до захищених API має надаватися тільки з дійсним JWT-токеном	У 100% запитів без токена або з недійсним токеном сервер повертає 401 Unauthorized
NF7	Розмежування доступу має працювати для 4 ролей користувачів	Для ролей ADMIN, MANAGER, WAREHOUSE_WORKER, VIEWER заборонені дії повертають 403 Forbidden
NF8	Паролі користувачів мають зберігатися тільки у хешованому вигляді	У таблиці користувачів у 100% записів пароль не зберігається відкритим текстом
NF9	Мінімальна довжина пароля має становити 8 символів	Пароль коротший за 8 символів відхиляється у 100% випадків
NF10	Після кожної успішної складської операції має створюватися запис у журналі	У 100% успішних операцій створюється запис у журналі транзакцій
NF11	Експорт звіту у CSV має виконуватися не більше 5 с	CSV-файл до 1000 рядків формується за ≤ 5 с
NF12	Дані не повинні втрачатися після перезапуску системи	Після перезапуску Docker-контейнерів 100% збережених товарів, залишків і операцій доступні в системі

Сформульовані нефункціональні вимоги важливі для подальшого тестування. Наприклад, перевірка вимоги щодо заборони від'ємних залишків дозволяє переконатися, що система не допускає некоректних складських операцій. Перевірка часу пошуку та відкриття сторінок дає змогу оцінити швидкодію системи при роботі з великою кількістю товарів. Вимоги щодо JWT-токена, ролей доступу та хешування паролів пов'язані з безпекою системи та захистом складських даних.

Таблиця 2.5

Відповідність вимог модулям системи

Модуль системи	Пов'язані вимоги	Призначення модуля
Модуль авторизації	F1, F2, NF6, NF7, NF8, NF9	Вхід користувача, перевірка JWT, ролі та захист API
Модуль користувачів	F3, F2	Керування обліковими записами та ролями
Модуль товарів	F4, F5, F18	Ведення каталогу автозапчастин, пошук і фільтрація
Модуль довідників	F6	Категорії, бренди, постачальники, склади
Модуль залишків	F7, F13, F15, NF5	Контроль кількості товарів по складах і критичних залишків
Модуль складських операцій	F8, F9, F10, F11, F12, F14, NF3, NF10	Виконання операцій і журналювання транзакцій
Модуль звітності	F16, F17, NF11	Формування звітів і CSV-експорт
Модуль інтерфейсу	NF1, NF2, NF4	Відображення сторінок, таблиць, форм і фільтрів

2.2 Моделювання варіантів використання системи

Моделювання варіантів використання є важливим етапом проектування системи відстеження залишків на складі автозапчастин. Воно дозволяє визначити, які користувачі взаємодіють із системою, які дії вони можуть виконувати та які функціональні можливості повинні бути реалізовані в програмному забезпеченні. Для системи APW Inventory варіанти використання формуються відповідно до ролей користувачів, визначених у попередньому підрозділі: ADMIN, MANAGER, WAREHOUSE_WORKER та VIEWER.

Система призначена для автоматизації складських процесів, тому основними діями користувачів є вхід до системи, перегляд каталогу автозапчастин, пошук товарів, робота із залишками, виконання складських операцій, перегляд журналу транзакцій, контроль критичних залишків, формування звітів і, для адміністратора, керування користувачами. Кожна роль має власний рівень доступу, що дозволяє обмежити виконання критичних дій лише тими користувачами, які мають відповідні повноваження.

У системі передбачено загального актора «Користувач системи», від якого логічно походять окремі ролі. Усі авторизовані користувачі можуть входити до системи та переглядати доступну для них інформацію. Проте права на зміну даних залежать від ролі. Наприклад, користувач із роллю VIEWER може переглядати товари й залишки, але не може створювати складські операції або редагувати довідники. Працівник складу може виконувати операції з товарами, але не повинен керувати обліковими записами інших користувачів. Адміністратор має найширші права доступу, зокрема керування користувачами та довідниками.

Основні актори системи та їх призначення наведено в таблиці 2.6.

Таблиця 2.6

Актори системи APW Inventory

Актор	Призначення	Рівень доступу
Адміністратор	Відповідає за налаштування системи, користувачів, ролей, довідників і контроль основних даних	Повний доступ до адміністративних функцій
Менеджер	Контролює каталог автозапчастин, залишки, критичні запаси, звіти та журнал операцій	Доступ до управління товарами, залишками та звітністю
Працівник складу	Виконує складські операції з товарами: надходження, списання, переміщення, повернення, коригування	Доступ до операцій із залишками
Користувач із правами перегляду	Переглядає інформацію про товари, склади, залишки та звіти без права редагування	Обмежений доступ лише для перегляду

Для наочного представлення взаємодії користувачів із системою доцільно використати UML-діаграму варіантів використання. Вона показує основні ролі користувачів та функції, які доступні кожній ролі. Така діаграма потрібна для узгодження функціональних можливостей системи перед переходом до детального проектування архітектури, бази даних і програмних модулів.

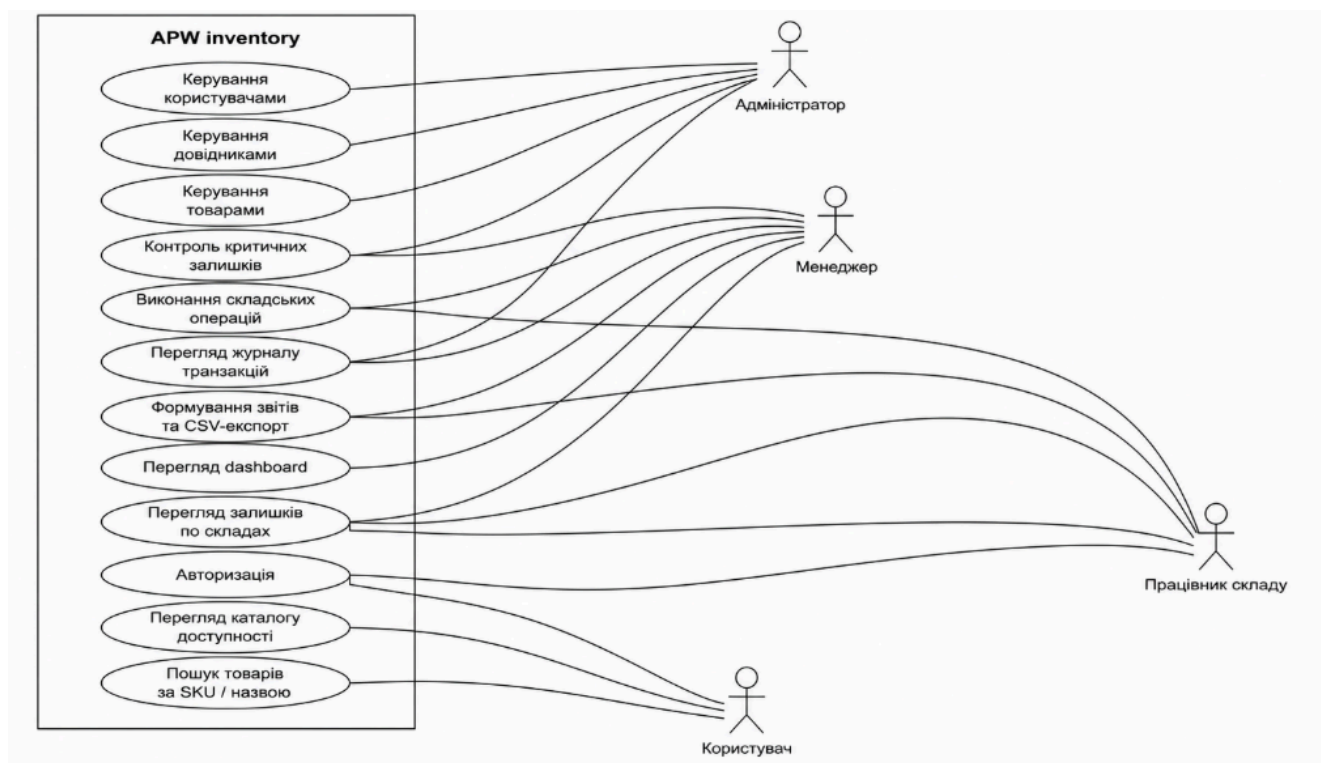


Рис. 2.3 Діаграма варіантів використання системи APW Inventory

Основним варіантом використання для всіх ролей є авторизація в системі. Перед початком роботи користувач вводить облікові дані, після чого система перевіряє логін і пароль. Якщо дані правильні, користувач отримує JWT-токен і доступ до функцій відповідно до своєї ролі. Якщо облікові дані неправильні або користувач неактивний, доступ не надається. Такий механізм потрібний для захисту складських даних і контролю дій користувачів.

Після входу до системи користувач може працювати з каталогом автозапчастин. Для менеджера та адміністратора передбачається можливість створення, редагування та видалення товарних позицій. Для працівника складу та користувача з правами перегляду основною дією є перегляд товарів і пошук

потрібної автозапчастини за SKU, назвою, брендом або категорією. Каталог є центральним елементом системи, оскільки всі складські операції виконуються щодо конкретних товарних позицій.

Окремий блок варіантів використання пов'язаний із довідниками. До довідників належать категорії, бренди, постачальники та склади. Їх ведення необхідне для структурованого опису товарів і правильного розподілу залишків. Адміністратор і менеджер можуть працювати з довідниками, тоді як інші користувачі переважно використовують ці дані лише під час перегляду або виконання операцій.

Найважливішими для предметної галузі є варіанти використання, пов'язані зі складськими операціями. Працівник складу повинен мати можливість створювати операції надходження, списання, переміщення, повернення та коригування. Під час виконання таких дій система перевіряє наявність товару, склад, кількість і допустимість операції. Наприклад, якщо працівник намагається списати більшу кількість товару, ніж є на складі, система повинна заблокувати таку дію. Це запобігає появі від'ємних залишків і підтримує цілісність облікових даних.

Ще одним важливим варіантом використання є перегляд журналу складських операцій. Журнал дозволяє бачити історію змін залишків: тип операції, товар, склад, кількість, дату виконання та користувача, який виконав дію. Доступ до журналу потрібний адміністратору та менеджеру для контролю правильності роботи складу, а також для пошуку причин розбіжностей між фактичними та обліковими залишками.

Основні варіанти використання системи APW Inventory

Код	Варіант використання	Актори	Очікуваний результат
UC1	Авторизація користувача	ADMIN, MANAGER, WAREHOUSE_WORKER, VIEWER	Користувач отримує доступ до системи відповідно до ролі
UC2	Перегляд каталогу автозапчастин	Усі ролі	Користувач бачить перелік товарів і основні характеристики
UC3	Пошук і фільтрація товарів	Усі ролі	Користувач знаходить товар за SKU, назвою, брендом або категорією
UC4	Створення та редагування товару	ADMIN, MANAGER	У каталозі створюється або змінюється товарна позиція
UC5	Керування довідниками	ADMIN, MANAGER	Оновлюються категорії, бренди, постачальники та склади
UC6	Перегляд залишків по складах	Усі ролі	Користувач отримує інформацію про кількість товарів на складах
UC7	Створення операції надходження	ADMIN, MANAGER, WAREHOUSE_WORKER	Залишок товару на складі збільшується
UC8	Створення операції списання	ADMIN, MANAGER, WAREHOUSE_WORKER	Залишок товару на складі зменшується, якщо кількість доступна
UC9	Переміщення товару між складами	ADMIN, MANAGER, WAREHOUSE_WORKER	Залишок зменшується на складі-відправнику та збільшується на складі-одержувачі
UC10	Повернення товару	ADMIN, MANAGER, WAREHOUSE_WORKER	Повернений товар знову враховується на складі
UC11	Коригування залишку	ADMIN, MANAGER, WAREHOUSE_WORKER	Облікова кількість уточнюється відповідно до фактичної
UC12	Перегляд журналу операцій	ADMIN, MANAGER	Користувач бачить історію складських транзакцій
UC13	Перегляд критичних залишків	ADMIN, MANAGER, WAREHOUSE_WORKER	Система показує товари, які потребують поповнення
UC14	Формування звітів	ADMIN, MANAGER	Система формує звітну інформацію про залишки та рух товарів
UC15	Експорт звітів у CSV	ADMIN, MANAGER	Користувач отримує CSV-файл із даними звіту
UC16	Керування користувачами	ADMIN	Адміністратор створює, редагує або деактивує користувачів

Сценарій варіанта використання UC8 «Створення операції списання»

сценарію	Опис
Назва варіанта використання	Створення операції списання
Основний актор	Працівник складу
Додаткові актори	Менеджер, адміністратор
Передумова	Користувач авторизований і має право виконувати складські операції
Основний сценарій	Користувач вибирає товар, склад, тип операції OUTCOME і кількість для списання
Перевірка системи	Система перевіряє, чи достатній залишок товару на вибраному складі
Успішний результат	Залишок товару зменшується, у журналі створюється запис Елемент про операцію
Альтернативний сценарій	Якщо кількість для списання більша за доступну, система блокує операцію
Результат помилки	Залишок не змінюється, користувач отримує повідомлення про недостатню кількість товару

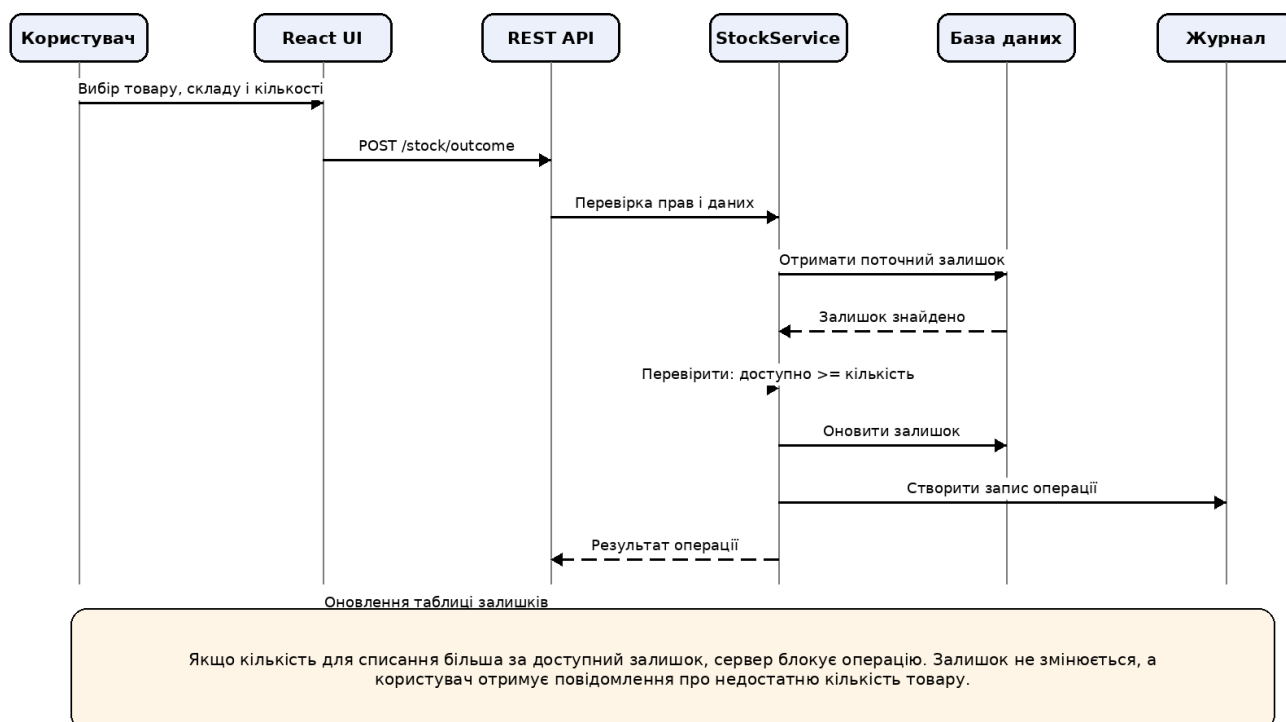


Рис. 2.4 Діаграма послідовності виконання операції списання товару

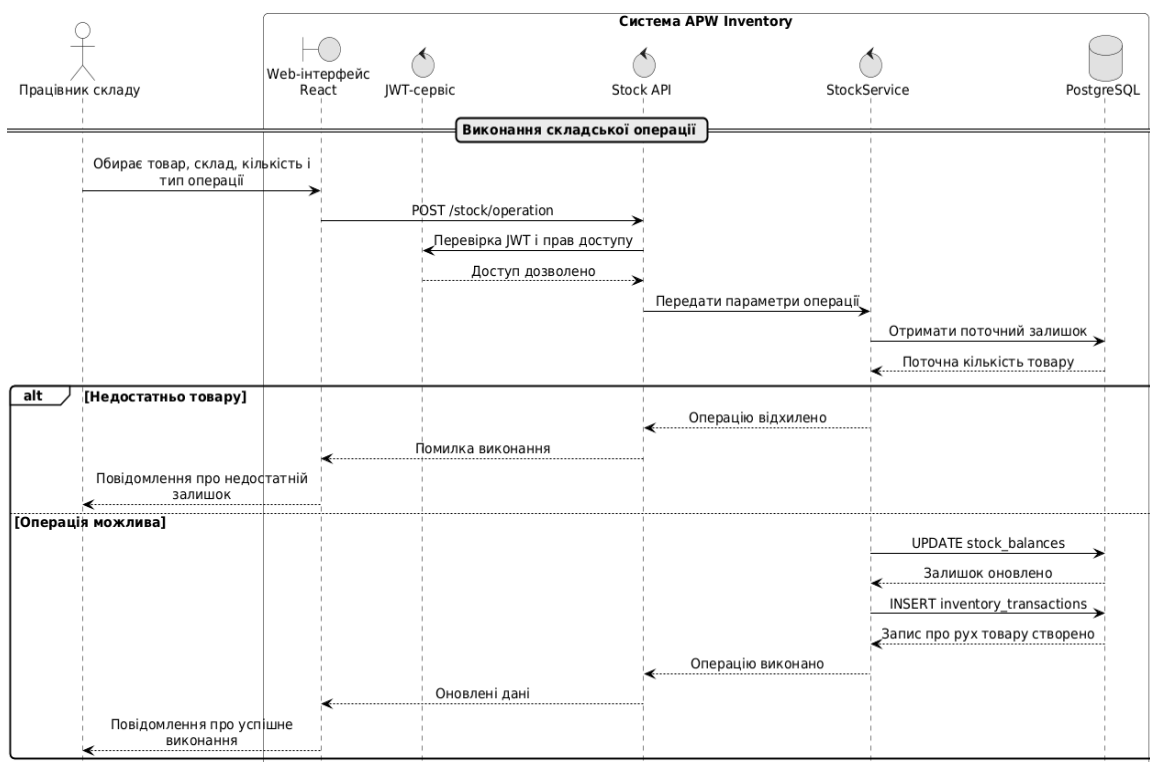


Рис. 2.5 Діаграма послідовності виконання складських операцій

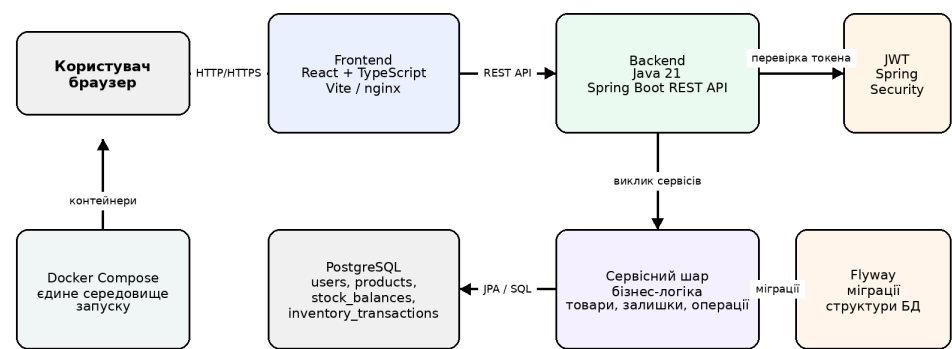
2.3 Проєктування архітектури програмного забезпечення

Архітектура системи APW Inventory проєктується як клієнт-серверна вебсистема, у якій окремо виділено клієнтську частину, серверну частину та базу даних. Такий підхід є доцільним для системи складського обліку, оскільки декілька користувачів можуть одночасно працювати з єдиною базою даних через вебінтерфейс, а основні правила обробки складських операцій виконуються централізовано на сервері. Це дозволяє уникнути ситуацій, коли різні користувачі працюють із розрізненими файлами або локальними копіями даних.

У системі APW Inventory клієнтська частина відповідає за відображення інтерфейсу користувача, введення даних, фільтрацію, перегляд таблиць, надсилання запитів до серверної частини та отримання результатів. Серверна частина виконує бізнес-логіку: перевіряє права доступу, обробляє запити, керує товарами, складами, залишками, користувачами, транзакціями та звітами. База даних використовується для постійного збереження інформації про всі основні об'єкти системи.

Загальна архітектура системи складається з таких основних рівнів:

- рівень клієнтського інтерфейсу;
- рівень REST API;
- рівень бізнес-логіки;
- рівень доступу до даних;
- рівень бази даних;
- рівень інфраструктури та розгортання.



Frontend відповідає за інтерфейс користувача, backend реалізує правила складського обліку, PostgreSQL зберігає дані, а Spring Security перевіряє авторизацію та ролі.

Рис. 2.6 Клієнт-серверна архітектура системи APW Inventory

Таблиця 2.9

Основні компоненти архітектури системи APW Inventory

Компонент системи	Технологічна основа	Призначення
Клієнтський інтерфейс	React, TypeScript, Vite	Відображення сторінок, форм, таблиць, фільтрів і взаємодія користувача із системою
API-клієнт	Axios	Надсилення HTTP-запитів до backend та отримання відповідей
REST API	Spring Web	Обробка запитів від клієнтської частини
Бізнес-логіка	Spring Boot Services	Виконання правил обліку товарів, залишків, операцій і звітності
Безпека	Spring Security, JWT, BCrypt	Автентифікація, авторизація, перевірка ролей і захист паролів
Доступ до даних	Spring Data JPA	Робота з сутностями та репозиторіями бази даних
База даних	PostgreSQL	Збереження товарів, складів, залишків, транзакцій, користувачів і довідників
Міграції БД	Flyway	Створення та оновлення структури бази даних
Розгортання	Docker Compose, nginx	Запуск frontend, backend і PostgreSQL як окремих сервісів

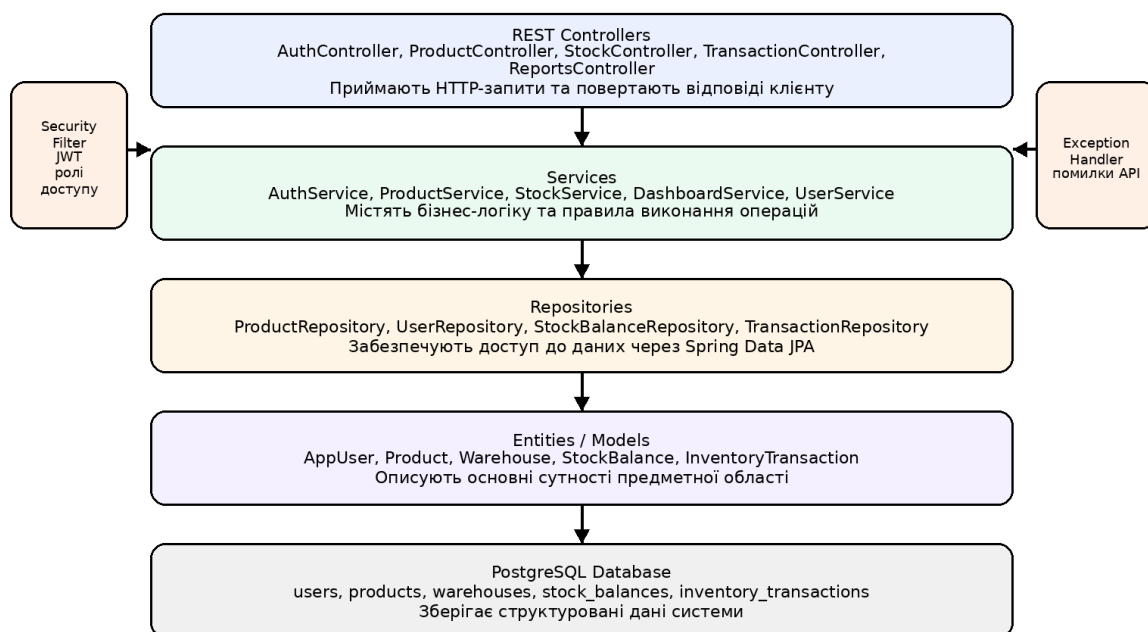


Рис. 2.7 Багаторівнева структура backend-частини системи APW Inventory

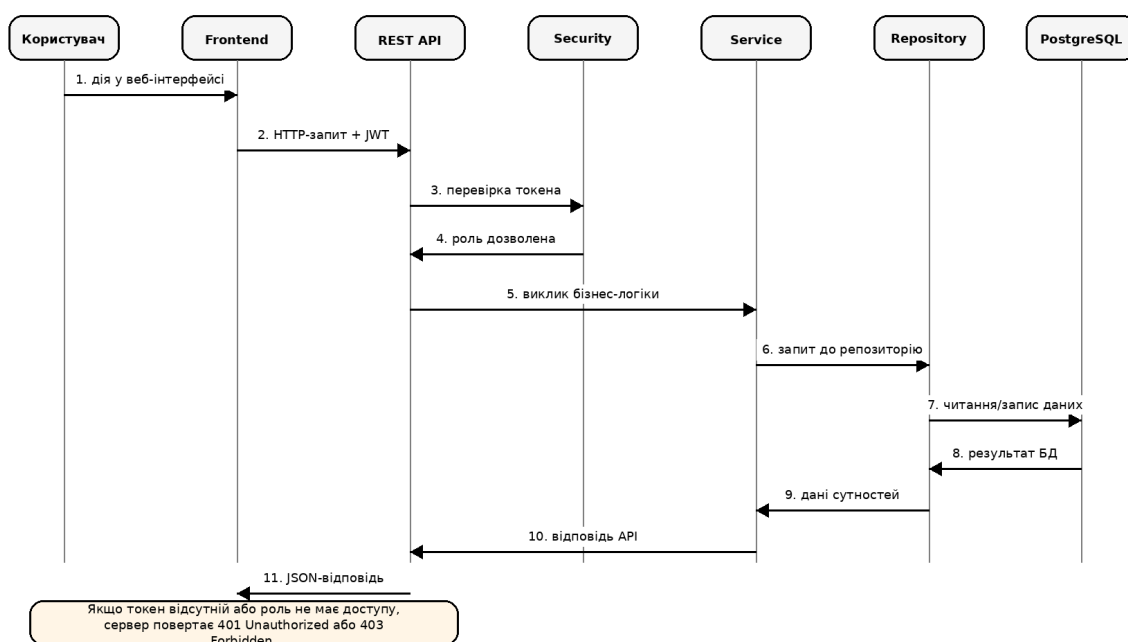


Рис. 2.8 Послідовність обробки запиту в системі APW Inventory

2.4 Проєктування структури бази даних

Проєктування бази даних є одним із ключових етапів створення системи відстеження залишків на складі автозапчастин. Саме база даних відповідає за збереження основної інформації про товари, склади, залишки, користувачів, довідники та складські операції. Якщо структура бази даних побудована неправильно, система не зможе коректно відображати фактичну кількість товарів, контролювати рух автозапчастин між складами та зберігати історію операцій.

Для системи APW Inventory обрано реляційну модель бази даних, оскільки предметна галузь має чітко структуровані сутності та зв'язки між ними. Наприклад, товар належить до певної категорії, має бренд і постачальника, залишок пов'язаний одночасно з товаром і складом, а складська операція фіксує зміну кількості конкретного товару на складі. Такий тип даних зручно зберігати у взаємопов'язаних таблицях із використанням первинних і зовнішніх ключів.

У системі передбачено такі основні сутності бази даних: користувачі системи, категорії автозапчастин, бренди, постачальники, склади, товари, залишки товарів по складах і складські транзакції. Кожна з цих сутностей відповідає окремій таблиці бази даних. Такий підхід дозволяє уникнути дублювання інформації, забезпечити цілісність даних і спростити подальшу підтримку системи.

Наприклад, назва бренду або постачальника не повинна щоразу вводитися вручну в картці товару. Доцільніше зберігати бренди й постачальників в окремих довідниках, а в таблиці товарів використовувати посилання на відповідні записи. Так само залишок товару не повинен зберігатися лише в картці автозапчастини, оскільки одна й та сама позиція може бути розміщена на різних складах.

Основні таблиці бази даних системи APW Inventory

Назва таблиці	Призначення
users	Зберігає інформацію про користувачів системи, їх ролі, логіни, паролі та стан активності
categories	Містить категорії автозапчастин, які використовуються для групування товарів
brands	Зберігає інформацію про бренди або виробників автозапчастин
suppliers	Містить дані про постачальників товарів
warehouses	Зберігає інформацію про склади або місця зберігання автозапчастин
products	Містить основні дані про автозапчастини: SKU, назву, опис, ціни, мінімальний залишок і сумісність
stock_balances	Фіксує кількість конкретного товару на конкретному складі
inventory_transactions	Зберігає історію складських операцій: надходження, списання, переміщення, повернення та коригування

Таблиця users використовується для збереження облікових записів користувачів. Вона містить дані, необхідні для авторизації та розмежування доступу: логін, хеш пароля, роль користувача та ознаку активності. У системі передбачено ролі ADMIN, MANAGER, WAREHOUSE_WORKER та VIEWER. Завдяки цьому можна обмежувати доступ до функцій відповідно до посадових обов'язків користувача.

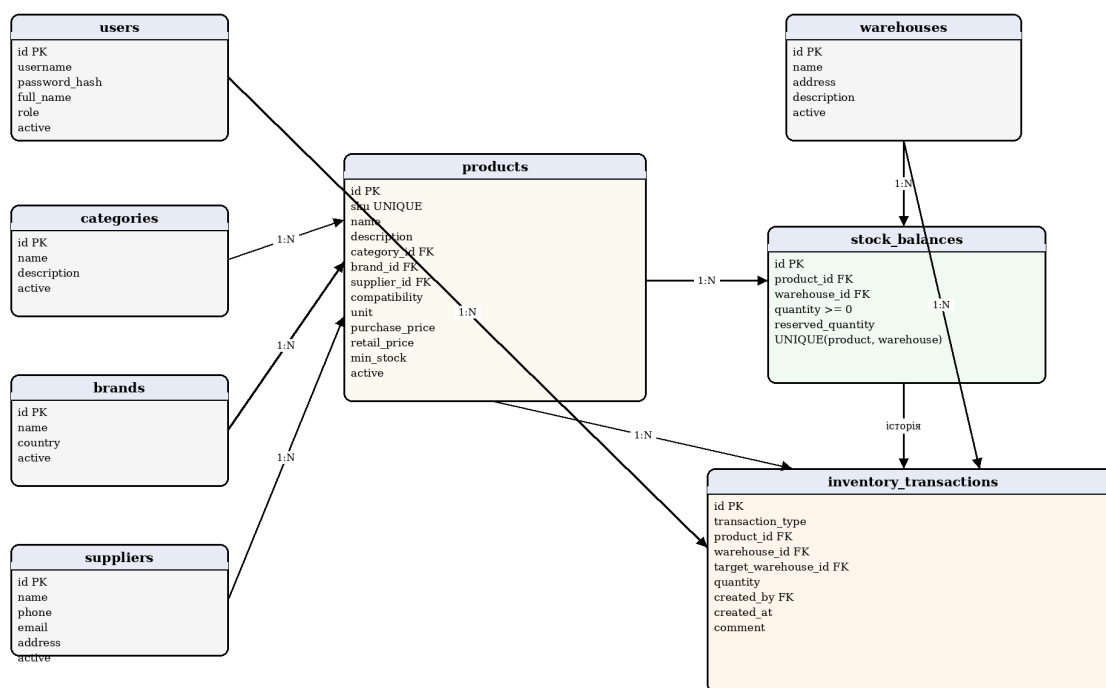
Таблиці categories, brands, suppliers та warehouses виконують роль довідників. Вони забезпечують структуроване зберігання повторюваних даних і використовуються під час створення або редагування товарів. Наприклад, автозапчастина може належати до категорії «Фільтри», мати бренд «Bosch» і постачальника, через якого вона надходить на склад. Винесення цих даних в окремі таблиці спрощує пошук, фільтрацію та підтримку єдиної структури каталогу.

Центральною таблицею предметної області є products. Вона зберігає інформацію про автозапчастини. До основних полів цієї таблиці належать SKU, назва товару, опис, категорія, бренд, постачальник, сумісність з автомобілями, одиниця виміру, закупівельна ціна, роздрібна ціна, мінімальний залишок та ознака

активності. Поле SKU має бути унікальним, оскільки саме воно дозволяє однозначно ідентифікувати товарну позицію в системі.

Таблиця `stock_balances` використовується для обліку залишків. Важливо, що кількість товару не зберігається безпосередньо в таблиці `products`, оскільки одна й та сама автозапчастина може бути розміщена на різних складах. Тому залишок формується як окремий запис, що пов'язує товар і склад. Такий підхід дозволяє отримати точну інформацію про те, скільки одиниць конкретної автозапчастини знаходиться на кожному складі.

Таблиця `inventory_transactions` відповідає за збереження історії складських операцій. Кожна операція, яка змінює залишок товару, повинна створювати запис у цій таблиці. Запис містить тип операції, товар, склад, кількість, дату виконання, користувача, який виконав дію, а також додаткову інформацію залежно від типу операції. Для переміщення між складами необхідно зберігати склад-відправник і склад-одержувач.



PK - первинний ключ; FK - зовнішній ключ; UNIQUE - обмеження унікальності; 1:N - зв'язок один-до-багатьох.

Рис. 2.9 ER-діаграма бази даних системи APW Inventory

Основні зв'язки між таблицями можна описати так: один користувач може виконати багато складських операцій; одна категорія може містити багато товарів; один бренд може бути пов'язаний із багатьма товарами; один постачальник може постачати багато товарів; один товар може мати залишки на різних складах; один склад може містити залишки багатьох товарів; одна складська операція пов'язана з конкретним товаром, складом і користувачем.

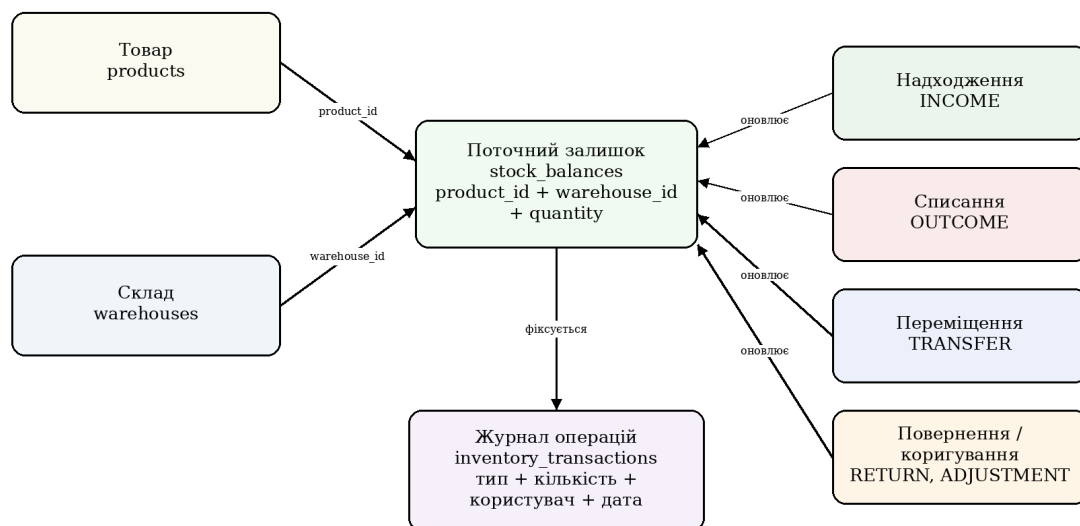
Така структура дозволяє відокремити довідкову інформацію від оперативних даних. Наприклад, дані про бренд або категорію зберігаються окремо, тоді як кількість товару на складі зберігається в таблиці залишків. Це робить базу даних більш гнучкою та зменшує ризик дублювання інформації.

Для забезпечення цілісності даних у базі даних потрібно передбачити обмеження. Зокрема, SKU товару повинен бути унікальним, кількість товару не повинна бути від'ємною, закупівельна та роздрібна ціна мають бути невід'ємними, а залишок товару повинен бути пов'язаний із реально існуючим товаром і складом. Такі обмеження допомагають запобігти некоректному стану системи навіть у випадку помилки під час виконання запиту.

Таблиця 2.11

Основні обмеження цілісності даних

Обмеження	Призначення
Унікальність SKU	Не дозволяє створити дві однакові товарні позиції з однаковим артикулом
Зовнішній ключ category_id	Гарантує, що товар належить до існуючої категорії
Зовнішній ключ brand_id	Гарантує, що товар пов'язаний з існуючим брендом
Зовнішній ключ supplier_id	Гарантує, що товар пов'язаний з існуючим постачальником
Зовнішній ключ product_id у таблиці залишків	Не дозволяє створити залишок для неіснуючого товару
Зовнішній ключ warehouse_id у таблиці залишків	Не дозволяє створити залишок для неіснуючого складу
Обмеження quantity ≥ 0	Запобігає появі від'ємного залишку товару
Обмеження для цін	Не дозволяє зберігати від'ємну закупівельну або роздрібну ціну
Обмеження ролі користувача	Дозволяє зберігати лише передбачені ролі користувачів



Поточний залишок відображає стан товару на складі, а журнал транзакцій зберігає історію всіх змін кількості.

Рис. 2.10 Логічна структура зберігання залишків і транзакцій

Таблица 2.12

Поля бази даних, для яких доцільно використовувати індекси

Таблица	Поле	Причина використання індексу
products	sku	Швидкий пошук товару за унікальним артикулом
products	name	Пошук товару за назвою або частиною назви
products	category_id	Фільтрація товарів за категорією
products	brand_id	Фільтрація товарів за брендом
stock_balances	product_id	Отримання залишків конкретного товару
stock_balances	warehouse_id	Отримання залишків по конкретному складу
inventory_transactions	transaction_type	Фільтрація операцій за типом
inventory_transactions	created_at	Формування звітів за період
inventory_transactions	product_id	Перегляд історії руху конкретного товару

2.5 Проєктування логіки складських операцій

Логіка складських операцій є однією з центральних частин системи APW Inventory, оскільки саме вона відповідає за зміну залишків автозапчастин на складах. Каталог товарів зберігає опис автозапчастини, таблиця залишків показує її поточну кількість на конкретному складі, а складські операції пояснюють, чому ця кількість збільшилася, зменшилася або була скоригована. Тому під час

проектування системи важливо визначити не тільки перелік можливих операцій, а й правила їх виконання.

У системі передбачено п'ять основних типів складських операцій: INCOME, OUTCOME, TRANSFER, RETURN та ADJUSTMENT. Операція INCOME використовується для надходження товару на склад і збільшує залишок відповідної автозапчастини. Операція OUTCOME призначена для списання або видачі товару зі складу, тому вона зменшує залишок. Операція TRANSFER використовується для переміщення товару між складами: кількість зменшується на складі-відправнику та збільшується на складі-одержувачі. Операція RETURN фіксує повернення товару на склад. Операція ADJUSTMENT застосовується тоді, коли потрібно уточнити облікову кількість після перевірки або інвентаризації.

Складська операція не повинна виконуватися як звичайне редагування числа в таблиці. Перед зміною залишку система має перевірити, чи існує товар, чи активний він, чи існує склад, чи має користувач право виконувати дію, а також чи достатньо товару для списання або переміщення. Після успішного виконання операції система створює запис у журналі транзакцій. Завдяки цьому зберігається не тільки поточна кількість товару, а й історія її зміни.

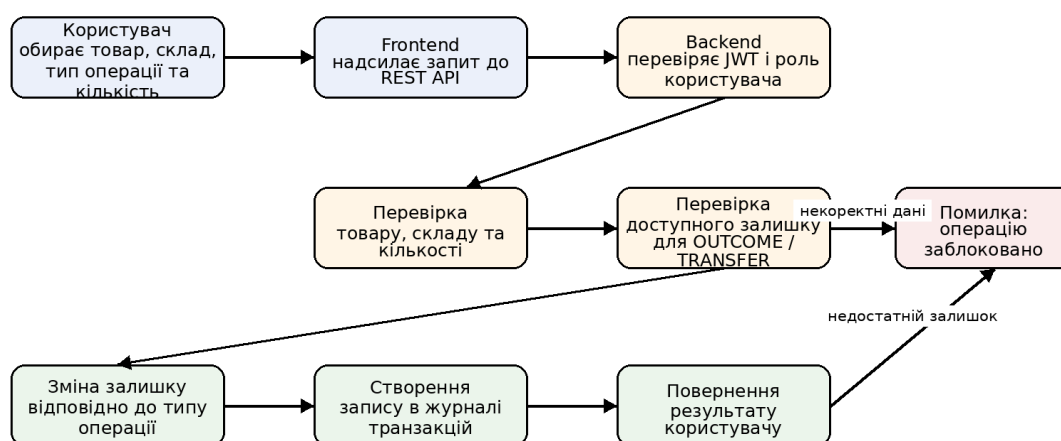


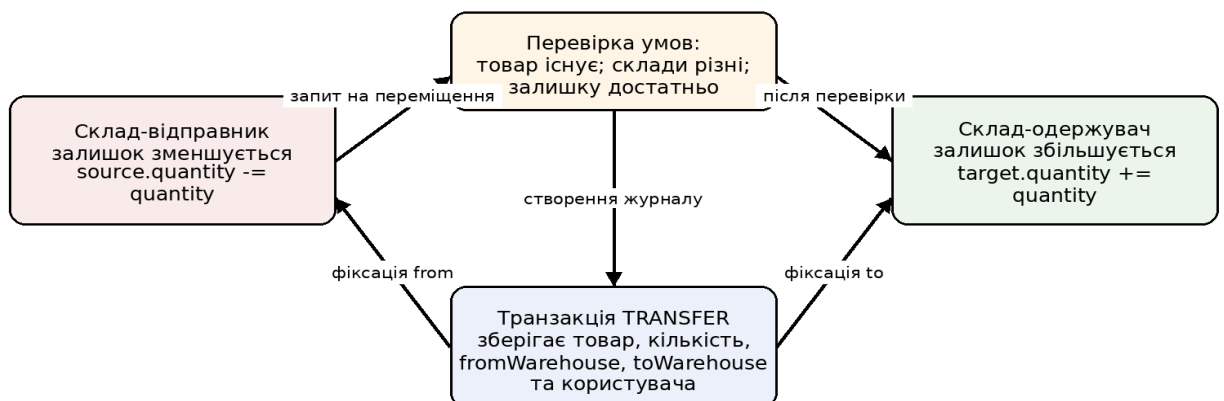
Рис. 2.11 Загальна логіка виконання складської операції в системі APW Inventory

Загальний сценарій виконання складської операції можна описати так. Спочатку користувач у вебінтерфейсі обирає товар, склад, тип операції та

кількість. Після цього клієнтська частина надсилає запит до серверної частини. Сервер перевіряє авторизацію користувача, його роль, коректність введених даних, наявність товару та складу. Якщо операція передбачає зменшення залишку, додатково перевіряється, чи достатня кількість товару доступна на складі. Якщо всі умови виконані, система змінює залишок і зберігає запис про операцію в журналі.

У розробленій системі ця логіка реалізована у сервісному класі StockService. Саме цей клас відповідає за виконання основних складських операцій і контроль коректності залишків. Нижче наведено фрагмент коду, який демонструє зміну залишків для надходження, списання, повернення та переміщення товару між складами.

```
@Transactional
public StockBalanceResponse income(StockOperationRequest request,
AppUser user) {
    requirePositive(request.quantity(), "Кількість надходження має бути
    більшою за нуль");
    Product product = activeProduct(request.productId());
```



Операція переміщення виконується як одна логічна дія: система не повинна списати товар з одного складу без додавання цієї самої кількості на інший склад.

Рис. 2.12 Логіка переміщення автозапчастини між складами

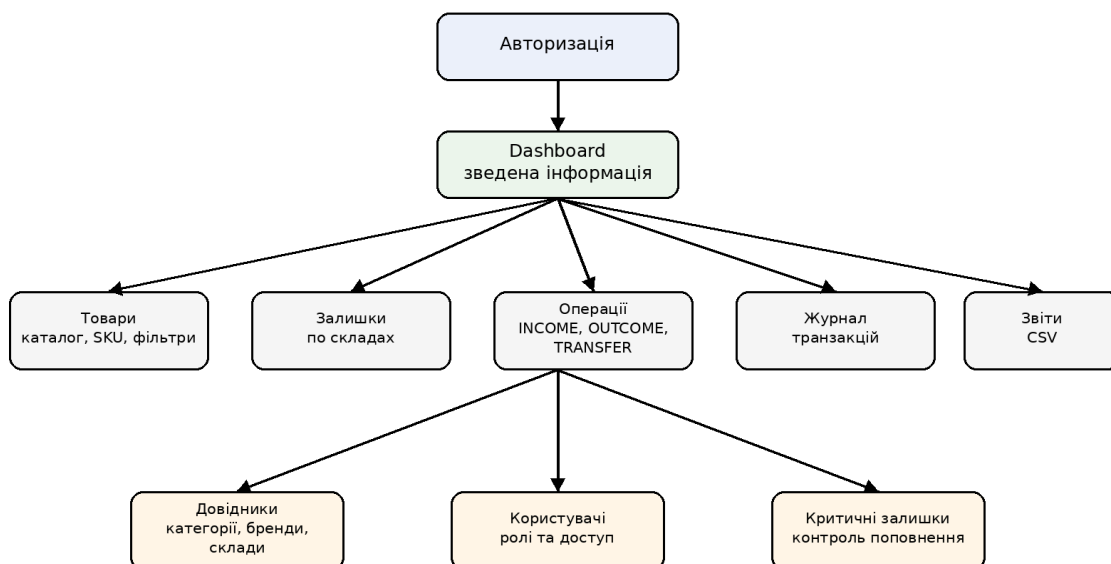
2.6 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача для системи APW Inventory виконується з урахуванням того, що програмний продукт призначений для щоденної роботи зі складськими даними. Користувачі системи повинні швидко знаходити потрібні автозапчастини, переглядати залишки по складах, виконувати складські операції, контролювати критичні запаси та формувати звіти. Тому інтерфейс має бути не лише візуально зрозумілим, а й логічно пов'язаним із реальними діями працівників складу.

Основною вимогою до інтерфейсу є швидкий доступ до ключових розділів системи. У роботі складу користувач не повинен витрачати багато часу на пошук потрібної функції, оскільки складські операції часто виконуються в умовах обмеженого часу: під час приймання товару, видачі автозапчастини, переміщення між складами або перевірки залишків. Саме тому інтерфейс системи доцільно будувати навколо основних робочих сторінок: панелі статистики, каталогу товарів, залишків, складських операцій, журналу транзакцій, звітів, довідників і керування користувачами.

Початковою сторінкою системи є сторінка авторизації. Вона повинна містити мінімальний набір елементів: поле для введення логіна, поле для введення пароля та кнопку входу. Після успішної авторизації користувач отримує доступ до системи відповідно до своєї ролі. Якщо облікові дані неправильні або користувач не має доступу, система повинна відобразити зрозуміле повідомлення про помилку. Такий підхід дозволяє відразу обмежити доступ до складських даних і не показувати внутрішні сторінки неавторизованим користувачам.

Після входу користувач переходить на головну панель системи. На цій сторінці доцільно розмістити коротку зведену інформацію про стан складу: загальну кількість товарних позицій, кількість складів, кількість товарів із критичним залишком, загальну вартість запасів і останні складські операції. Така панель дає змогу менеджеру або адміністратору швидко оцінити поточний стан складського обліку без переходу до окремих звітів.



Навігація будується навколо щоденних задач складу: пошуку товарів, контролю залишків, створення операцій, перегляду історії та формування звітів.

Рис. 2.13 Структура основних екранів системи APW Inventory

Однією з центральних сторінок є каталог автозапчастин. Він повинен відображати список товарів у табличному вигляді, оскільки саме таблиця є найбільш зручною формою для роботи з великою кількістю однотипних позицій. У таблиці доцільно показувати SKU, назву товару, категорію, бренд, постачальника, закупівельну та роздрібну ціну, мінімальний залишок і стан активності. Для швидкого пошуку товару потрібно передбачити пошуковий рядок і фільтри за категорією, брендом або постачальником.

Під час створення або редагування товару користувач повинен працювати з формою, яка відображає всі основні характеристики автозапчастини. До такої форми належать поля SKU, назва, опис, категорія, бренд, постачальник, сумісність із автомобілями, одиниця виміру, закупівельна ціна, роздрібна ціна та мінімальний залишок. Важливо, щоб обов'язкові поля були візуально зрозумілими, а некоректні значення не приймалися системою. Наприклад, ціна не повинна бути від'ємною, а SKU не повинен дублювати вже існуючу товарну позицію.

Окрема сторінка залишків повинна показувати кількість товарів у розрізі складів. Для складу автозапчастин це особливо важливо, оскільки одна й та сама позиція може бути наявна на одному складі та відсутня на іншому. У таблиці

залишків доцільно відображати товар, SKU, склад, поточну кількість, зарезервовану кількість, доступну кількість і мінімальний залишок. Позиції з низьким або критичним запасом бажано візуально виділяти, щоб користувач одразу бачив товари, які потребують поповнення.

Сторінка складських операцій повинна бути побудована так, щоб користувач міг швидко виконати одну з типових дій: надходження, списання, переміщення, повернення або коригування. Для цього доцільно використовувати форму, у якій користувач обирає тип операції, товар, склад, кількість і за потреби додає коментар. Якщо операція передбачає переміщення, форма повинна містити два склади: склад-відправник і склад-одержувач. При цьому інтерфейс не повинен дозволяти обрати один і той самий склад для обох полів.

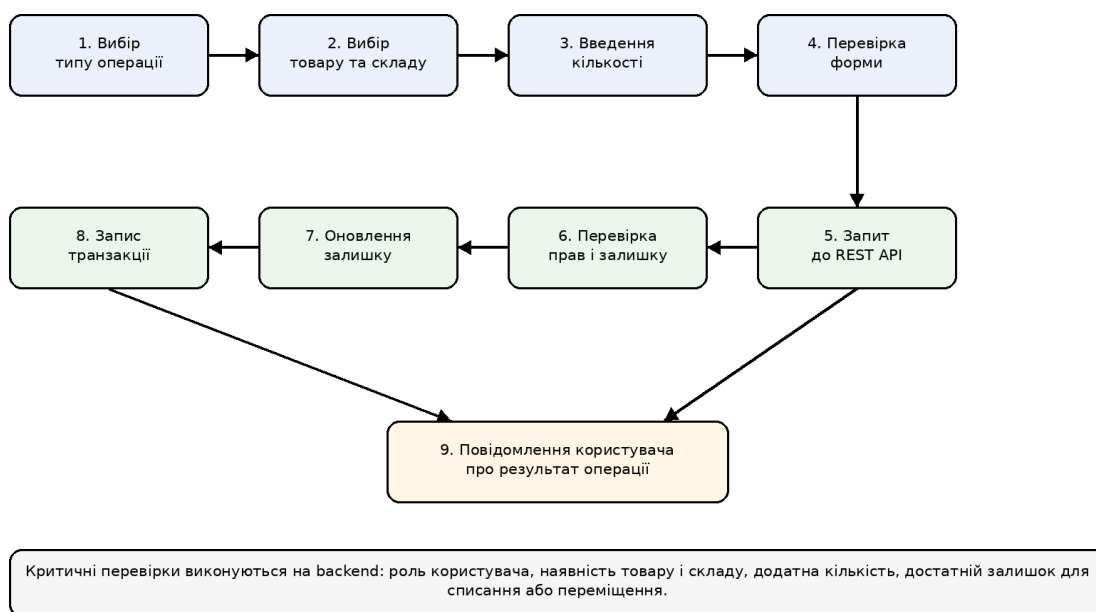


Рис. 2.14 Сценарій виконання складської операції через інтерфейс користувача

Основні екрани інтерфейсу системи APW Inventory

Екран системи	Призначення
Авторизація	Вхід користувача до системи за логіном і паролем
Dashboard	Короткий огляд стану складу, критичних залишків і останніх операцій
Товари	Перегляд, пошук, фільтрація, створення та редагування автозапчастин
Залишки	Перегляд кількості товарів на складах і контроль критичних запасів
Складські операції	Виконання надходження, списання, переміщення, повернення та коригування
Журнал операцій	Перегляд історії складських транзакцій
Звіти	Формування звітів і експорт даних у CSV
Довідники	Керування категоріями, брендами, постачальниками та складами
Користувачі	Керування обліковими записами та ролями користувачів

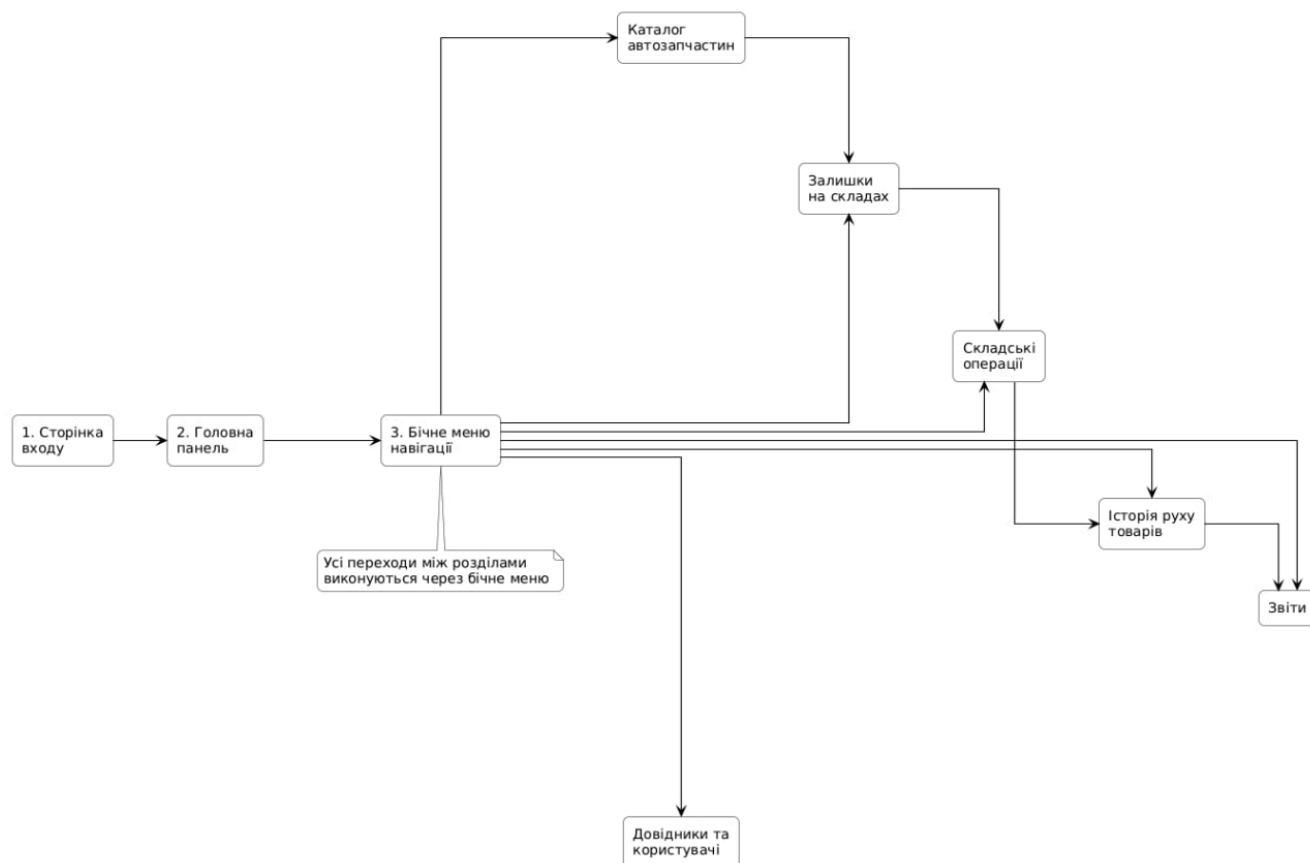


Рис. 2.15 Логіка навігації користувача в інтерфейсі APW Inventory

З РЕАЛІЗАЦІЯ СИСТЕМИ ВІДСТЕЖЕННЯ ЗАЛИШКІВ НА СКЛАДІ АВТОЗАПЧАСТИН

Третій розділ бакалаврської роботи присвячено опису реалізації вебсистеми APW Inventory, призначеної для відстеження залишків на складі автозапчастин. У попередньому розділі було визначено вимоги до системи, спроектовано її архітектуру, структуру бази даних, логіку складських операцій та інтерфейс користувача. На цьому етапі необхідно показати, як запроєктовані рішення були реалізовані у програмному продукті.

Розроблена система має клієнт-серверну архітектуру та складається з трьох основних частин: серверної частини, клієнтської частини та бази даних. Серверна частина реалізує бізнес-логіку складського обліку, обробляє HTTP-запити, перевіряє права доступу користувачів, виконує операції з товарами, складами, залишками, транзакціями та звітами. Клієнтська частина забезпечує взаємодію користувача з системою через вебінтерфейс. База даних використовується для збереження інформації про користувачів, автозапчастини, категорії, бренди, постачальників, склади, залишки та складські операції.

Backend-частина системи реалізована мовою Java з використанням фреймворку Spring Boot. Такий підхід дозволив побудувати серверну частину у вигляді окремих модулів, кожен із яких відповідає за конкретну групу функцій. У структурі backend можна виділити модулі авторизації, безпеки, користувачів, товарів, довідників, залишків, складських операцій, dashboard, звітності та обробки помилок. Завдяки цьому код системи має зрозумілий поділ відповідальності, а окремі частини можуть розвиватися незалежно одна від одної.

У серверній частині використано типову для Spring Boot багаторівневу структуру: контролери, сервіси, репозиторії, сутності та DTO-об'єкти. Контролери приймають запити від клієнтської частини та повертають відповіді. Сервіси містять основну бізнес-логіку, зокрема перевірку коректності складських операцій, зміну залишків і створення записів у журналі транзакцій. Репозиторії

відповідають за доступ до даних через Spring Data JPA. Такий поділ дозволяє відокремити логіку обробки запитів від правил предметної області та роботи з базою даних.

Особливе значення в реалізації системи має модуль складських операцій. Саме він забезпечує виконання операцій INCOME, OUTCOME, TRANSFER, RETURN та ADJUSTMENT. Під час виконання цих операцій система не просто змінює кількість товару, а перевіряє доступність залишку, правильність введеної кількості, існування товару й складу, а також створює запис у журналі операцій. Завдяки цьому забезпечується цілісність складських даних і неможливість появи від'ємного залишку.

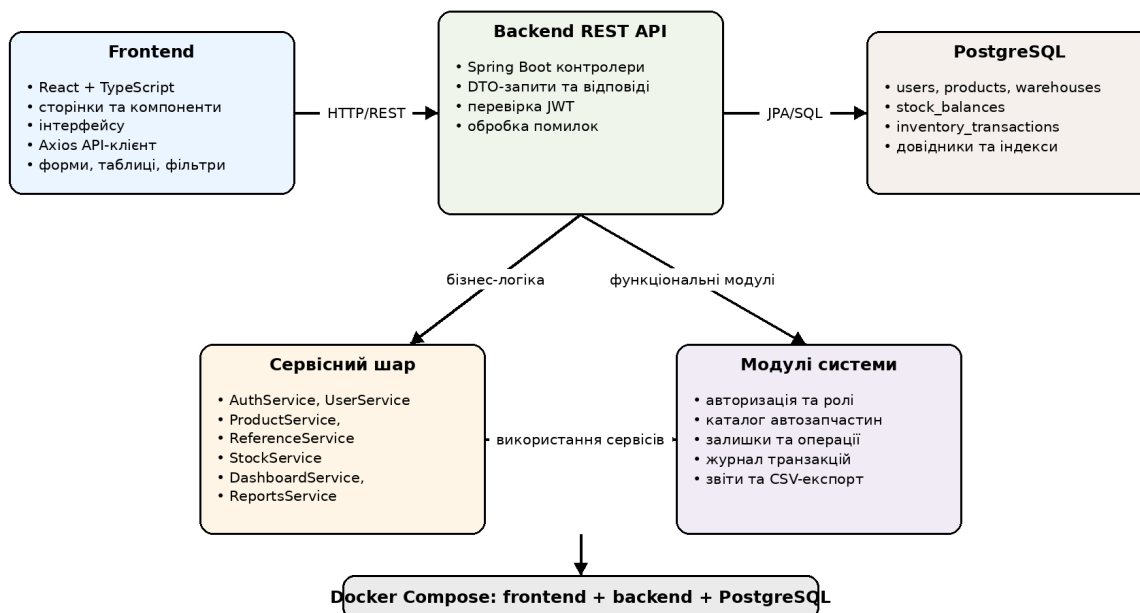


Рис. 3.1 Структура реалізації системи APW Inventory

3.1 Реалізація серверної частини системи

Серверна частина системи APW Inventory реалізована як окремий backend-застосунок на основі Java та Spring Boot. Вона відповідає за обробку запитів від клієнтської частини, виконання бізнес-логіки складського обліку, взаємодію з базою даних PostgreSQL, перевірку прав доступу користувачів і

формування відповідей для frontend. Same backend є основним рівнем, на якому забезпечується коректність роботи системи, оскільки всі критичні операції із залишками виконуються не на клієнті, а на сервері.

Серверна частина побудована за багаторівневою структурою, типовою для Spring Boot-застосунків. У проєкті виділено рівень контролерів, рівень сервісів, рівень репозиторіїв, рівень сутностей бази даних, DTO-об'єкти для передавання даних і окремі компоненти конфігурації безпеки. Такий поділ дозволяє розмежувати відповідальність між частинами програми: контролери приймають HTTP-запити, сервіси виконують бізнес-логіку, репозиторії працюють із базою даних, а сутності описують структуру основних об'єктів предметної області.

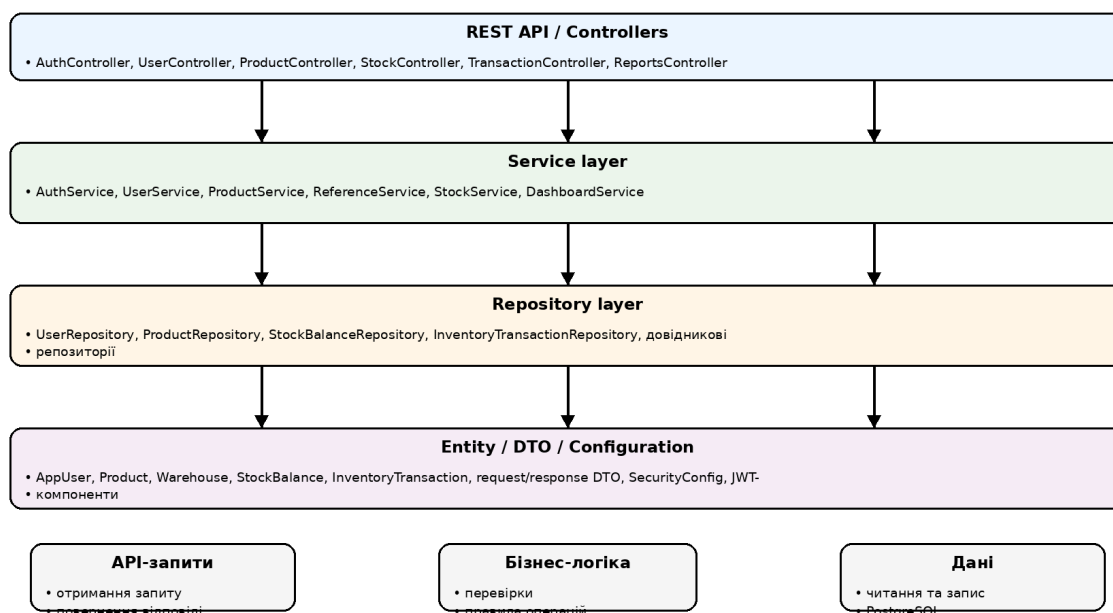


Рис. 3.2 Структура backend-частини системи APW Inventory

У backend-частині реалізовано декілька основних функціональних модулів: модуль авторизації, модуль користувачів, модуль товарів, модуль довідників, модуль залишків, модуль складських операцій, модуль журналу транзакцій, модуль dashboard і модуль звітності. Кожен із цих модулів відповідає за окрему частину функціональності системи, але всі вони працюють із єдиною базою даних і взаємодіють через сервісний шар.

Модуль авторизації забезпечує вхід користувача до системи. Він приймає логін і пароль, перевіряє їх, визначає роль користувача та формує JWT-токен для подальшої роботи. Модуль користувачів відповідає за створення, редагування, активацію та деактивацію облікових записів. Ці можливості необхідні для адміністратора системи, який керує доступом працівників до складських даних.

Модуль товарів реалізує роботу з каталогом автозапчастин. У ньому передбачено створення, перегляд, редагування, пошук і деактивацію товарних позицій. Для кожного товару зберігаються SKU, назва, опис, категорія, бренд, постачальник, сумісність із автомобілями, одиниця виміру, закупівельна й роздрібна ціна, мінімальний залишок та ознака активності. Такий набір полів відповідає специфіці обліку автозапчастин, де одного лише найменування недостатньо для точної ідентифікації товару.

Модуль довідників відповідає за роботу з категоріями, брендами, постачальниками та складами. Винесення цих даних в окремі сутності дозволяє уникнути дублювання інформації та спростити фільтрацію товарів. Наприклад, під час створення автозапчастини користувач не вводить бренд або категорію у довільному текстовому вигляді, а обирає їх із відповідного довідника.

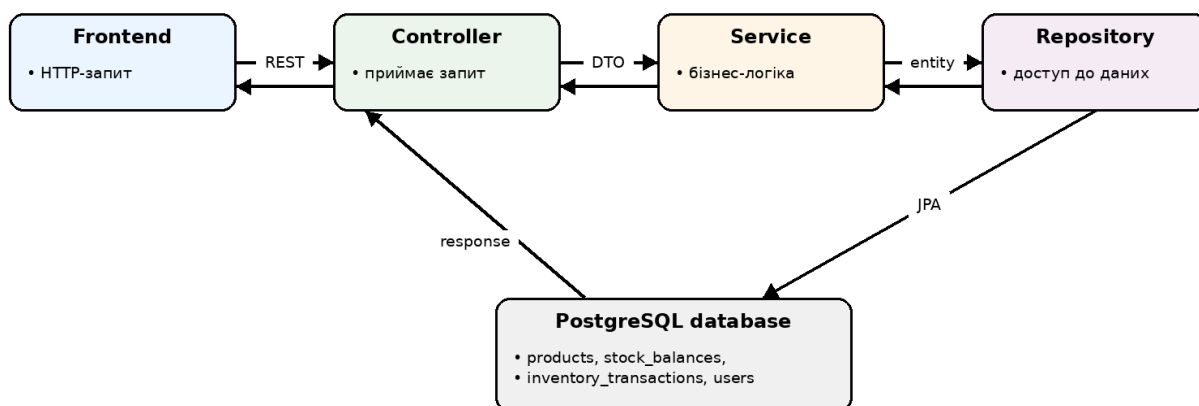
Найважливішим для предметної галузі є модуль залишків і складських операцій. Він реалізує облік кількості товарів на складах, надходження, списання, переміщення, повернення та коригування. У серверній частині ці операції виконуються через сервісний клас, який перевіряє коректність введених даних, наявність товару та складу, доступну кількість і права користувача. Після успішної операції змінюється запис залишку та створюється транзакція в журналі операцій.

Основні модулі серверної частини системи APW Inventory

Модуль backend-частини	Основні класи	Призначення
Авторизація	AuthController, AuthService	Вхід користувача до системи, перевірка облікових даних, формування JWT-токена
Користувачі	UserController, UserService, AppUser	Керування користувачами, ролями та активністю облікових записів
Товари	ProductController, ProductService, Product	Робота з каталогом автозапчастин
Довідники	ReferenceController, ReferenceService	Керування категоріями, брендами, постачальниками та складами
Залишки	StockController, StockService, StockBalance	Перегляд і зміна залишків товарів по складах
Транзакції	TransactionController, InventoryTransaction	Збереження та перегляд історії складських операцій
Dashboard і звіти	DashboardController, ReportsController, DashboardService	Формування статистики, звітів і даних для аналітичних сторінок
Безпека	SecurityConfig, JWT-компоненти	Захист API, перевірка токенів і розмежування доступу

Для реалізації REST API у системі використовуються контролери. Кожен контролер відповідає за окрему групу endpoint-ів. Наприклад, ProductController обробляє запити, пов'язані з товарами, StockController - із залишками та складськими операціями, TransactionController - із журналом операцій, а ReportsController - зі звітністю. Такий підхід робить API зрозумілим і логічно розділеним за функціональними напрямками.

Контролери не містять складної бізнес-логіки. Їхнє завдання полягає у прийнятті запиту, отриманні параметрів або тіла запиту, виклику відповідного сервісу та поверненні результату клієнтській частині. Основна логіка винесена до сервісів. Це важливо, оскільки правила предметної області повинні бути зосереджені в одному місці, а не дублюватися в різних контролерах.



Критичні перевірки виконуються на сервері: роль користувача, коректність даних, доступний залишок і створення транзакції.

Рис. 3.3 Взаємодія контролера, сервісу, репозиторію та бази даних

3.2 Реалізація авторизації та розмежування прав доступу

Безпека доступу є важливою частиною системи APW Inventory, оскільки програмний продукт працює з обліковими даними складу: товарами, залишками, складськими операціями, користувачами та звітами. Такі дані не повинні бути доступні неавторизованим користувачам, а можливість змінювати залишки або керувати користувачами має надаватися лише тим працівникам, які мають відповідні повноваження. Тому в системі реалізовано механізм автентифікації, авторизації та розмежування доступу за ролями.

У системі використано підхід, за якого користувач спочатку проходить автентифікацію за логіном і паролем. Якщо введені дані правильні, сервер формує JWT-токен, який надалі використовується для доступу до захищених API. Клієнтська частина зберігає отриманий токен і передає його в заголовок кожного запиту до серверної частини. Backend перевіряє токен, визначає користувача та його роль, після чого дозволяє або блокує виконання запиту.

У системі APW Inventory передбачено чотири ролі користувачів: ADMIN, MANAGER, WAREHOUSE_WORKER та VIEWER. Роль ADMIN має найширші права доступу, зокрема керування користувачами, довідниками, товарами та

перегляд звітів. Роль MANAGER орієнтована на контроль товарів, залишків, звітності та складських процесів. Роль WAREHOUSE_WORKER призначена для працівника складу, який виконує складські операції. Роль VIEWER надає доступ лише до перегляду даних без права їх зміни.

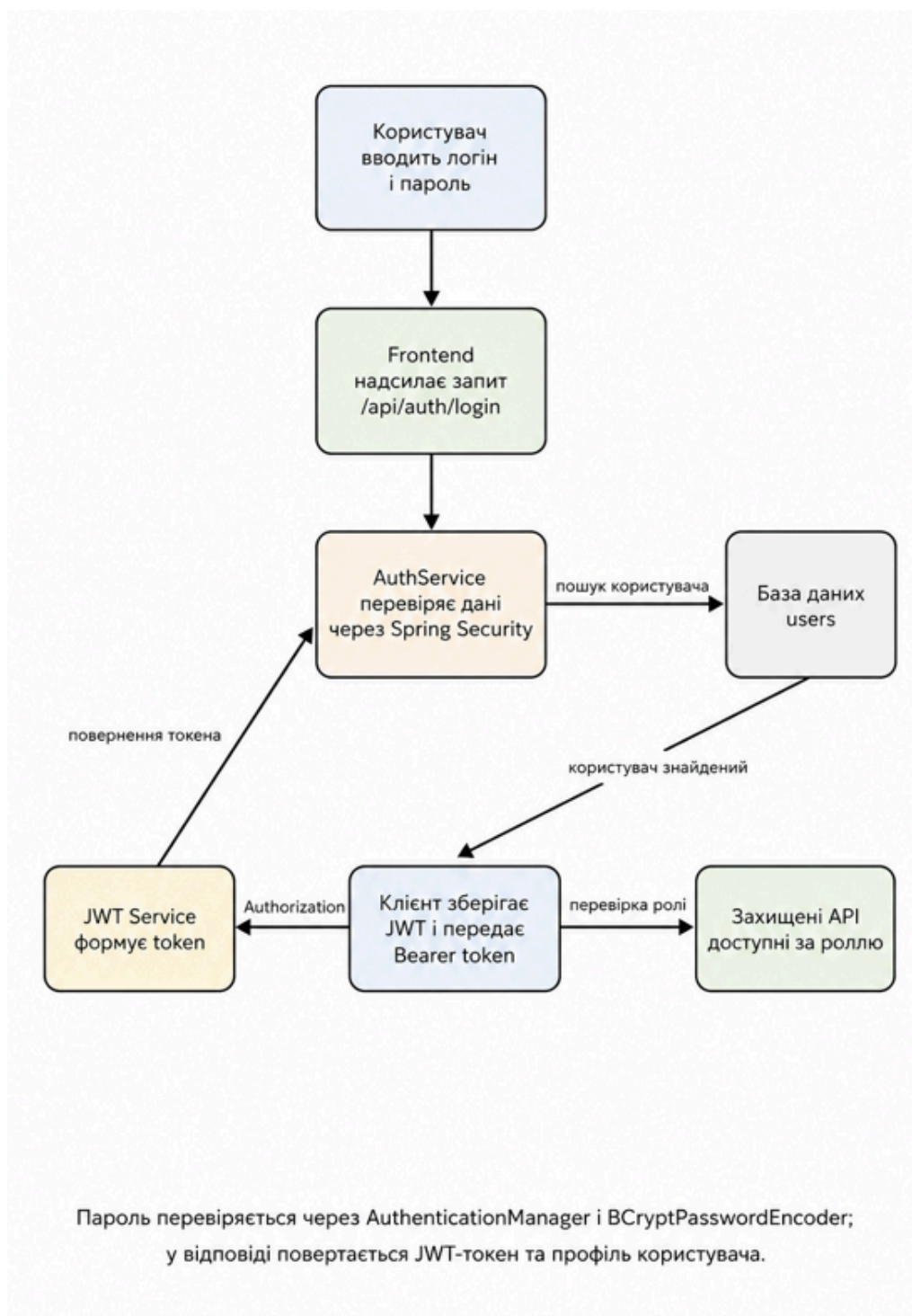


Рис. 3.4 Схема авторизації користувача в системі APW Inventory

Процес авторизації починається з того, що користувач вводить логін і пароль у формі входу. Клієнтська частина надсилає ці дані на серверний endpoint авторизації. Сервер перевіряє облікові дані через механізм Spring Security, знаходить користувача в базі даних, перевіряє пароль та формує JWT-токен. У відповідь клієнт отримує токен і профіль користувача, після чого може виконувати запити до захищених ресурсів.

Паролі користувачів не зберігаються у відкритому вигляді. Для їх захисту використовується BCryptPasswordEncoder. Під час створення користувача або зміни пароля в базу даних записується не сам пароль, а його хеш. Під час входу до системи введений пароль перевіряється засобами Spring Security через AuthenticationManager і відповідний AuthenticationProvider.

У наведеному фрагменті метод login приймає дані користувача, нормалізує логін і передає його разом із паролем до AuthenticationManager. Якщо автентифікація проходить успішно, об'єкт користувача отримується з результату перевірки, після чого за допомогою JwtService формується токен. У відповідь клієнтська частина отримує токен і дані профілю користувача.

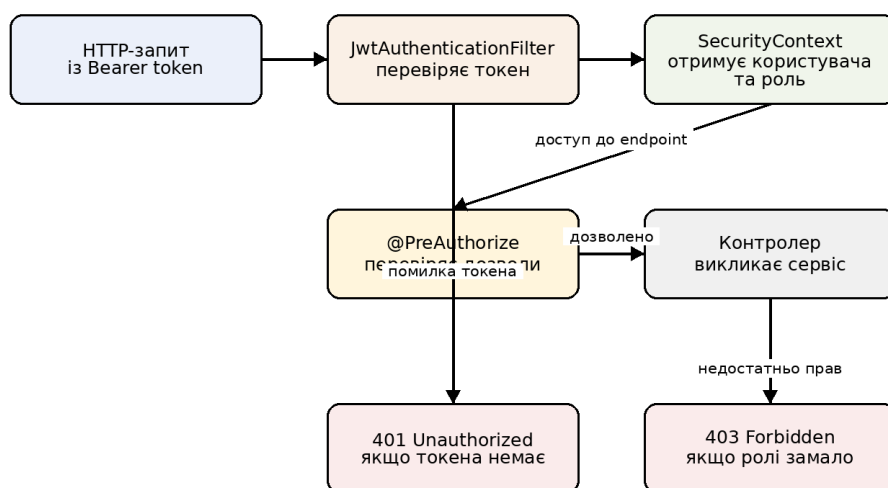
Після успішної авторизації клієнтська частина надсилає JWT-токен у заголовок запиту. Заголовок має формат Authorization: Bearer <token>. На серверній частині спеціальний фільтр перевіряє наявність токена, його коректність і строк дії. Якщо токен дійсний, у контекст безпеки Spring Security записується інформація про авторизованого користувача. Якщо токен відсутній або некоректний, захищений endpoint не виконується.

Фільтр JwtAuthenticationFilter виконується до обробки запиту контролером. Він отримує заголовок Authorization, перевіряє наявність префікса Bearer, виділяє токен і за допомогою JwtService визначає користувача. Якщо токен коректний, у SecurityContextHolder записується об'єкт автентифікації з правами користувача. Завдяки цьому інші компоненти Spring Security можуть перевіряти доступ до endpoint-ів за ролями.

Таблиця 3.2

Реалізація прав доступу в системі APW Inventory

Функція системи	ADMIN	MANAGER	WAREHOUSE_WORKER	VIEWER
Вхід до системи	+	+	+	+
Перегляд товарів і залишків	+	+	+	+
Створення та редагування товарів	+	+	-	-
Керування довідниками	+	+	-	-
Виконання складських операцій	+	+	+	-
Перегляд журналу операцій	+	+	частково	-
Формування звітів	+	+	-	-
Керування користувачами	+	-	-	-
Видалення критичних записів	+	-	-	-



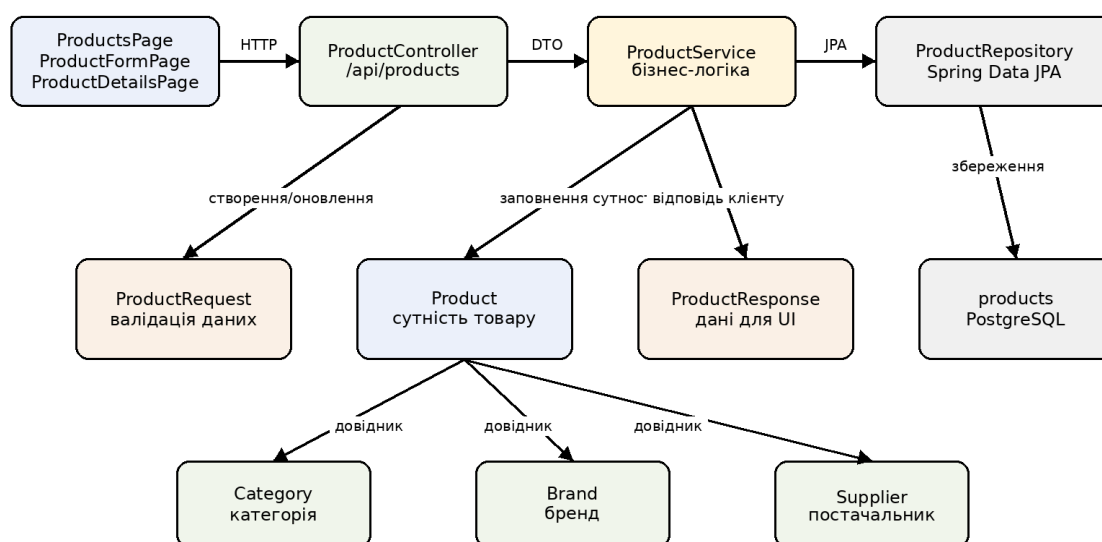
Клієнтська частина лише приховує недоступні кнопки, але остаточна перевірка прав виконується на backend-рівні.

Рис. 3.5 Схема перевірки прав доступу під час виконання запиту

3.3 Реалізація модуля каталогу автозапчастин

Модуль каталогу автозапчастин є одним із базових модулів системи APW Inventory, оскільки саме він забезпечує збереження та обробку інформації про товарні позиції. У складському обліку автозапчастин товар не можна описувати лише назвою та кількістю, тому в системі передбачено розширену структуру картки товару. Для кожної автозапчастини зберігається SKU, назва, опис, категорія, бренд, постачальник, сумісність із транспортними засобами, одиниця виміру, закупівельна ціна, роздрібна ціна, мінімальний залишок і статус активності.

Каталог товарів тісно пов'язаний з іншими частинами системи. Дані про автозапчастини використовуються під час перегляду залишків, створення складських операцій, формування журналу транзакцій, побудови звітів і визначення товарів із критичним залишком. Саме тому модуль каталогу реалізовано не як простий список записів, а як повноцінний компонент системи, який взаємодіє з довідниками, залишками та складськими операціями



Модуль каталогу поєднує frontend-сторінки, REST API, сервісний шар, репозиторій, сутність Product та довідники категорій, брендів і постачальників.

Рис. 3.6 Структура реалізації модуля каталогу автозапчастин

У backend-частині основною сутністю модуля є клас Product. Він відповідає таблиці products у базі даних і описує структуру товарної позиції. У цій сутності визначено зв'язки з категорією, брендом і постачальником, а також поля, які потрібні для ідентифікації та обліку автозапчастини.

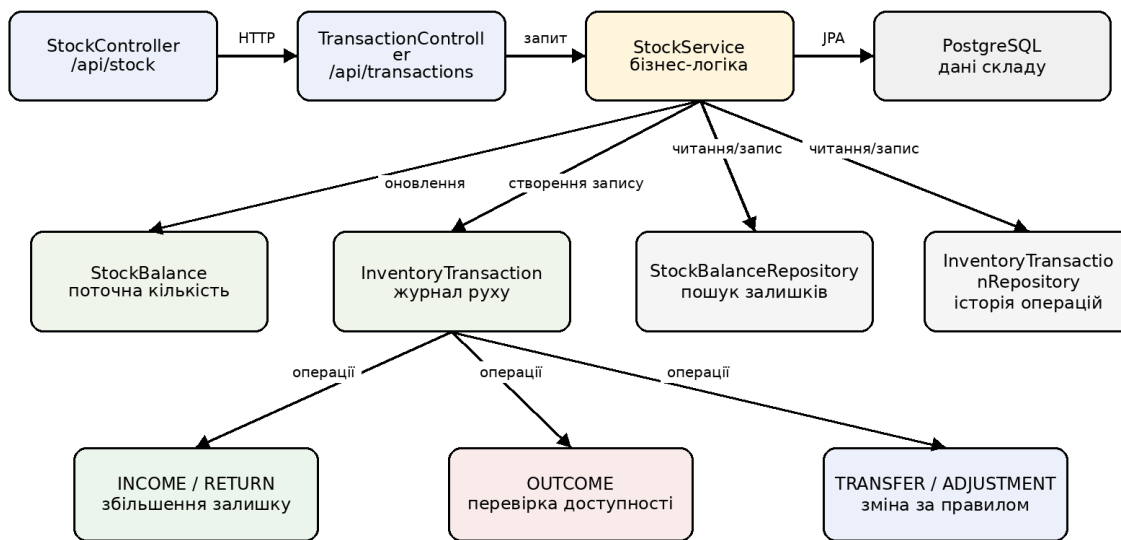
```
@Entity
@Table(name = "products")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

3.4 Реалізація модуля обліку залишків та складських операцій

Модуль обліку залишків та складських операцій є центральною частиною системи APW Inventory, оскільки саме він забезпечує практичне виконання основної мети програмного продукту - контроль наявності автозапчастин на складах і фіксацію руху товарів.

У реалізованій системі залишок не зберігається безпосередньо в сутності товару. Це зроблено тому, що одна й та сама автозапчастина може одночасно зберігатися на різних складах. Наприклад, певний фільтр або комплект гальмівних колодок може бути на центральному складі, у магазині та на складі сервісної зони. Тому кількість товару зберігається в окремій сутності StockBalance, яка пов'язує конкретний товар із конкретним складом.

```
@Entity
@Table(name = "stock_balances")
public class StockBalance {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```



Модуль забезпечує перегляд поточних залишків, контроль низького запасу, виконання складських операцій і збереження історії руху товарів.

Рис. 3.7 Структура модуля обліку залишків та складських операцій

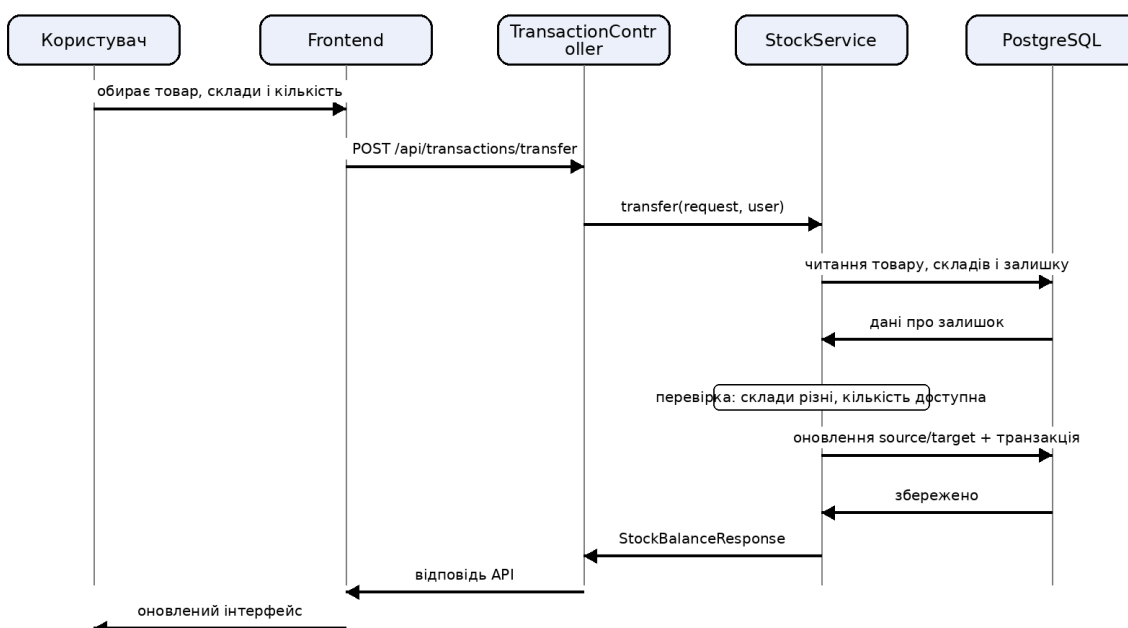


Рис. 3.8 Послідовність виконання операції переміщення товару між складами

3.5 Реалізація модуля звітності та експорту даних

Модуль звітності в системі APW Inventory призначений для узагальнення складських даних і надання користувачу інформації, необхідної для контролю

стану запасів. Якщо модулі товарів, залишків і складських операцій забезпечують щоденне внесення та зміну даних, то модуль звітності дозволяє отримати загальну картину: скільки товарів є в системі, яка кількість одиниць зберігається на складах, які позиції потребують поповнення, якою є загальна вартість залишків і які операції виконувалися за певний період.

У системі звітність реалізована через два пов'язані напрями: dashboard і звіти. Dashboard використовується для швидкого перегляду ключових показників, а сторінка звітів — для детальнішого аналізу залишків і руху товарів. Такий поділ є зручним для користувача, оскільки на головній сторінці він бачить коротку аналітичну інформацію, а в окремому розділі може працювати з фільтрами, таблицями й експортом даних.

У backend-частині за формування звітних даних відповідають DashboardController, ReportsController і DashboardService. Контролери приймають запити від клієнтської частини, а сервіс збирає потрібну інформацію з репозиторіїв товарів, довідників, складів, залишків і транзакцій. Такий підхід дозволяє зосередити логіку формування звітів в одному сервісному шарі та не дублювати розрахунки в різних контролерах.

Загальна структура модуля звітності наведена на рисунку 3.9.

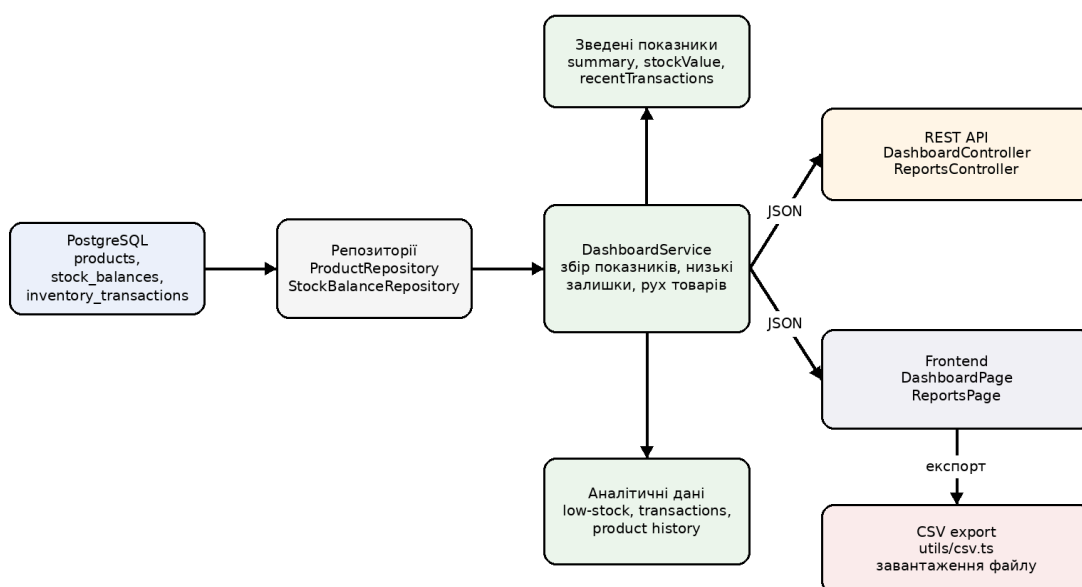


Рис. 3.9 Структура модуля звітності та експорту даних

Основним об'єктом для відображення даних на dashboard є DashboardSummaryResponse. Він містить узагальнені показники системи: загальну кількість товарів, кількість активних товарів, кількість категорій, брендів, постачальників і складів, загальну кількість одиниць на складі, зарезервовану кількість, кількість товарів із низьким залишком, вартість запасів, список позицій для поповнення та останні складські операції.

```
public record DashboardSummaryResponse(
```

```
    long totalProducts,
```

```
    long activeProducts,
```

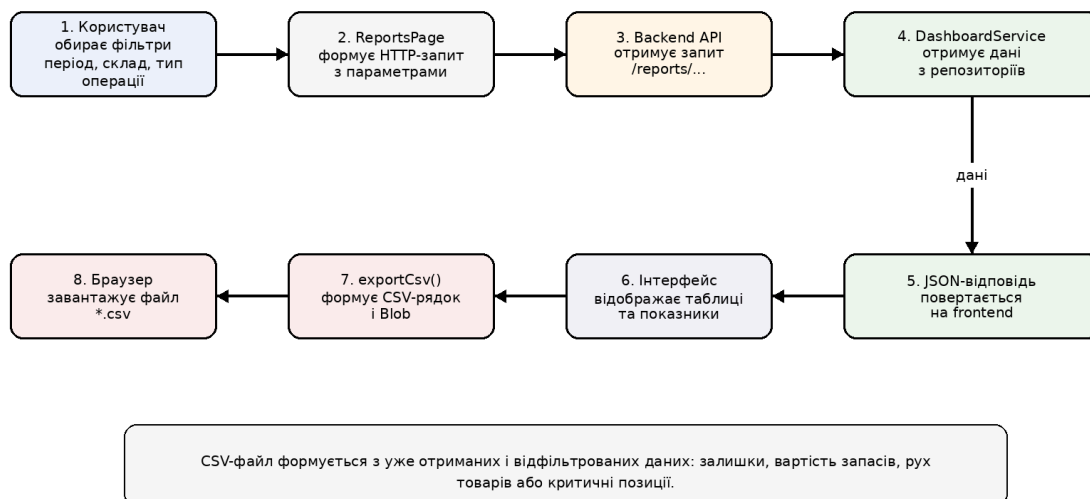


Рис. 3.10 Схема формування звітів та експорту CSV

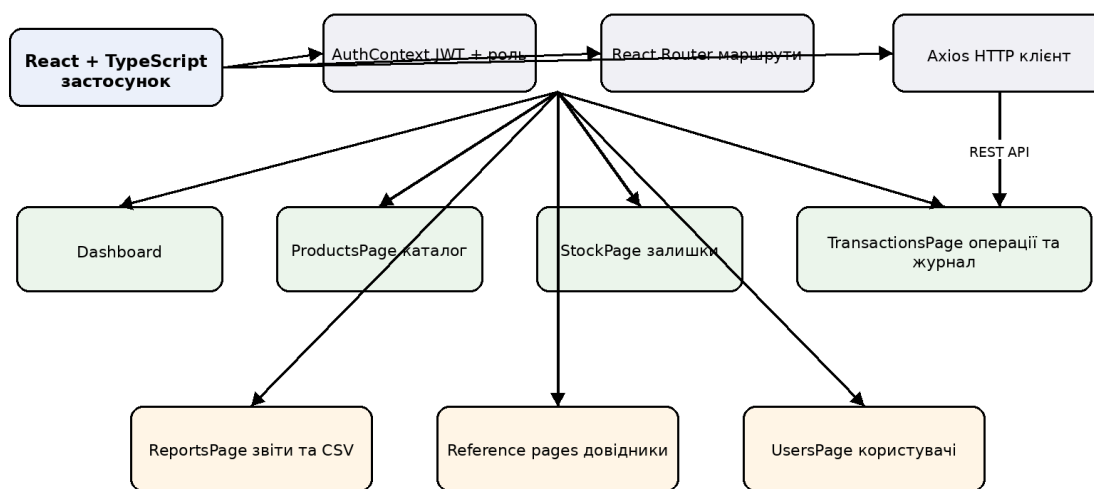
3.6 Реалізація клієнтської частини системи

Клієнтська частина системи APW Inventory реалізована як вебзастосунок, який забезпечує взаємодію користувача з основними функціями складського обліку. Через інтерфейс користувач може авторизуватися, переглядати панель статистики, працювати з каталогом автозапчастин, контролювати залишки по

складах, створювати складські операції, переглядати журнал транзакцій, формувати звіти та виконувати дії відповідно до своєї ролі.

Для реалізації frontend-частини використано React, TypeScript і Vite. React застосовується для побудови інтерфейсу на основі компонентів, TypeScript - для типізації даних і зменшення кількості помилок під час розробки, а Vite - для швидкого запуску та збирання клієнтського застосунку. Для взаємодії з backend використовується Axios, який надсилає HTTP-запити до REST API серверної частини.

Клієнтська частина побудована як набір сторінок і повторно використовуваних компонентів. До основних сторінок належать сторінка авторизації, dashboard, сторінка товарів, форма створення та редагування товару, сторінка залишків, сторінка складських операцій, журнал транзакцій, звіти, довідники та сторінка керування користувачами. Такий поділ відповідає функціональним модулям backend-частини та робить інтерфейс логічно зрозумілим для користувача.



Клієнтська частина відображає сторінки, працює з ролями користувачів і надсилає запити до backend через REST API.

Рис. 3.11 Структура клієнтської частини системи APW Inventory

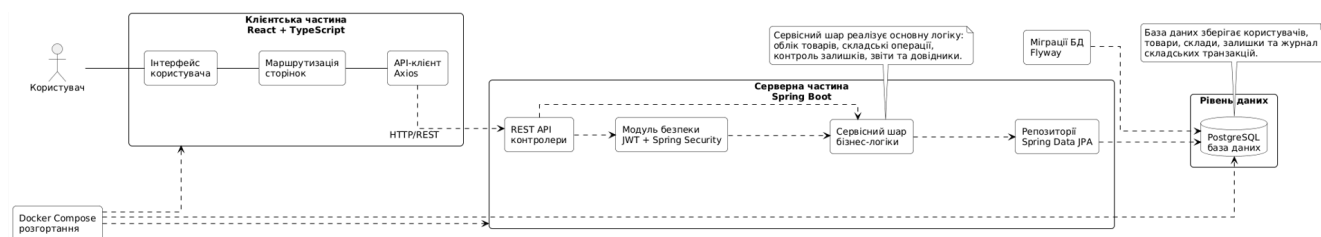


Рис. 3.12 Діаграма компонентів системи

Одним із важливих елементів frontend-частини є налаштування HTTP-клієнта. У системі використовується окремий екземпляр Axios, у якому визначено базову адресу API та механізм автоматичного додавання JWT-токена до заголовків запитів. Це дозволяє не дублювати логіку авторизації в кожному компоненті, а централізовано передавати токен під час звернення до захищених endpoint-ів.

У наведеному фрагменті видно, що перед кожним запитом frontend перевіряє наявність токена в локальному сховищі браузера. Якщо токен існує, він додається до заголовка Authorization у форматі Bearer. Завдяки цьому backend може ідентифікувати користувача та перевірити його роль.

Для збереження інформації про поточного користувача в клієнтській частині використовується контекст авторизації. Він дозволяє різним компонентам отримувати доступ до даних користувача, його ролі та функцій входу й виходу із системи.

Функція `signIn` надсилає логін і пароль на сервер, отримує JWT-токен та інформацію про користувача, після чого зберігає їх у стані застосунку та локальному сховищі браузера. Функція `signOut` очищує ці дані та повертає користувача до неавторизованого стану. Функція `hasAnyRole` використовується для перевірки ролей у компонентах інтерфейсу.

Маршрутизація в клієнтській частині реалізована за допомогою React Router. Вона дозволяє переходити між сторінками без повного перезавантаження браузера. Для захисту внутрішніх сторінок використовується компонент захищеного маршруту. Якщо користувач не авторизований, система не відкриває внутрішню сторінку, а перенаправляє його на сторінку входу.

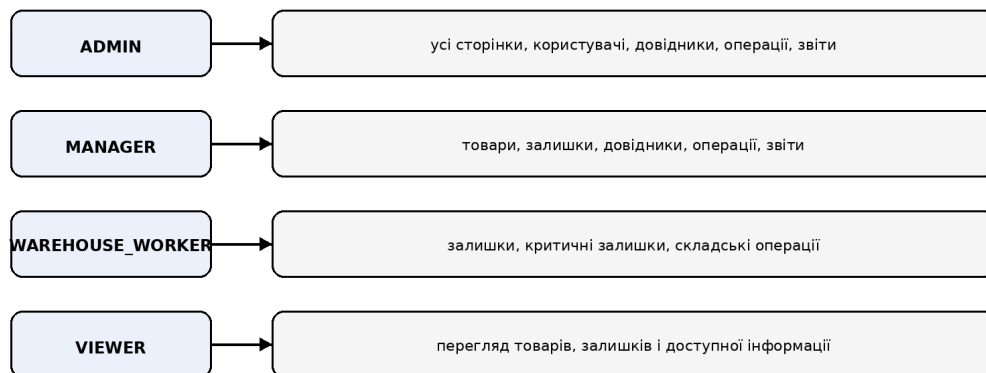


Рис. 3.13 Логіка рольової навігації у клієнтській частині системи



Клієнтська частина спрощує введення даних, але остаточну перевірку кількості, ролі та залишку виконує сервер.

Рис. 3.14 Сценарій виконання складської операції у frontend

Таблиця 3.3

Основні сторінки клієнтської частини APW Inventory

Сторінка	Призначення
LoginPage	Авторизація користувача за логіном і паролем
DashboardPage	Відображення ключових показників складу
ProductsPage	Перегляд, пошук і фільтрація автозапчастин
ProductFormPage	Створення та редагування товарної позиції
StockPage	Перегляд залишків по складах і критичних запасів
TransactionsPage	Виконання складських операцій і перегляд журналу транзакцій
ReportsPage	Формування звітів і експорт CSV
Reference pages	Робота з категоріями, брендами, постачальниками та складами
UsersPage	Керування користувачами та ролями

3.7 Опис основних екранних форм вебсистеми

У системі APW Inventory клієнтська частина реалізована у вигляді набору екранних форм, кожна з яких відповідає окремому функціональному процесу складського обліку. Основними екранними формами є сторінка авторизації, головна панель, каталог автозапчастин, форма створення та редагування товару, сторінка залишків, сторінка складських операцій, журнал транзакцій, сторінка звітів, довідники та сторінка керування користувачами. Такий поділ дозволяє користувачу швидко переходити до потрібного розділу та виконувати дії відповідно до своєї ролі.

Початковою екранною формою системи є сторінка авторизації. Вона використовується для входу користувача до системи за логіном і паролем. На цій сторінці розміщено поля введення облікових даних і кнопку підтвердження входу. Після введення логіна й пароля frontend надсилає запит до backend, а у разі успішної перевірки користувач отримує JWT-токен і переходить до внутрішньої частини системи. Якщо облікові дані неправильні або користувач деактивований, система відображає повідомлення про помилку.

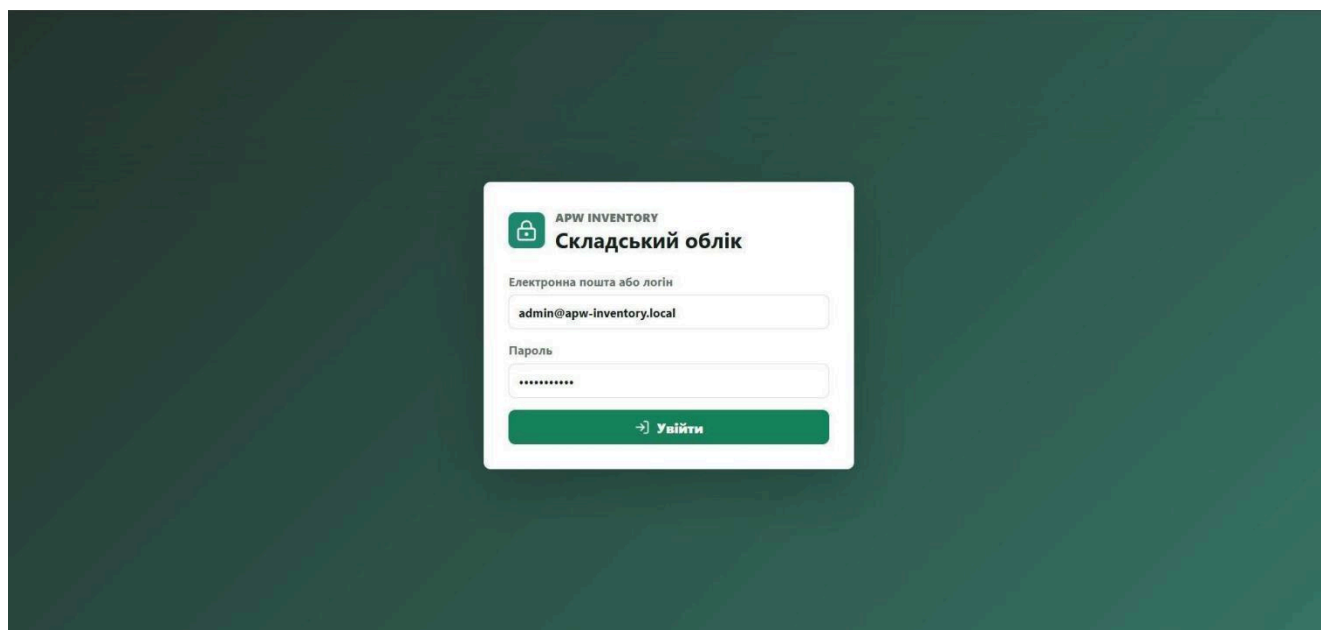


Рис. 3.15 Екранна форма авторизації користувача

Після входу до системи користувач потрапляє на головну панель, яка виконує роль інформаційного центру системи. На цій сторінці відображаються основні показники складського обліку: кількість автозапчастин, категорій, брендів, постачальників, складів, загальна кількість одиниць товару, кількість товарів із критичним залишком і вартість запасів. Також на dashboard можуть відображатися останні складські операції та перелік позицій, які потребують поповнення.

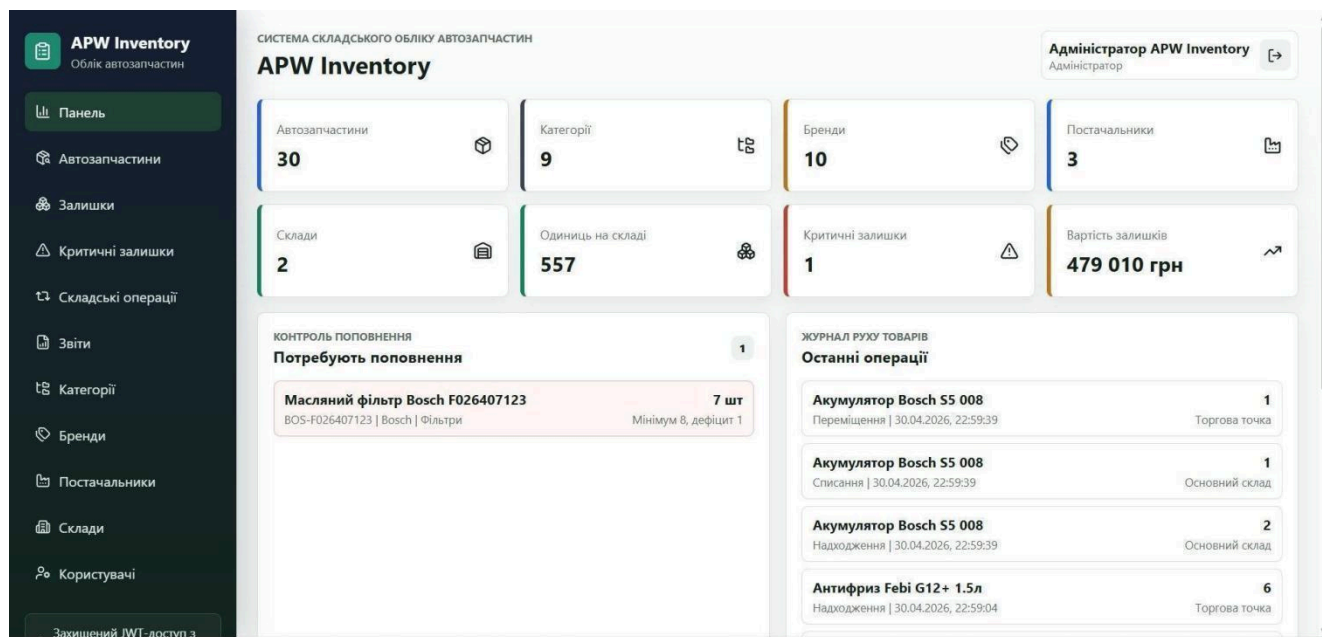


Рис. 3.16 Головна панель системи APW Inventory

Однією з основних екранних форм є сторінка каталогу автозапчастин. Вона призначена для перегляду, пошуку, фільтрації та керування товарними позиціями. У каталозі кожен товар відображається з основними характеристиками: назвою, SKU, брендом, категорією, сумісністю з автомобілями, закупівельною та роздрібною ціною, одиницею виміру, загальним залишком і мінімальним залишком. Для зручності користувача передбачено пошук за назвою, SKU, брендом або сумісністю, а також фільтрацію за категорією та брендом.

На сторінці каталогу також враховується роль користувача. Користувачі з ролями ADMIN і MANAGER можуть створювати та редагувати товарні позиції, тоді як працівник складу або користувач із правами перегляду переважно використовують каталог для пошуку та перегляду інформації. Такий підхід

дозволяє не перевантажувати інтерфейс зайвими кнопками для тих користувачів, які не мають права змінювати каталог.

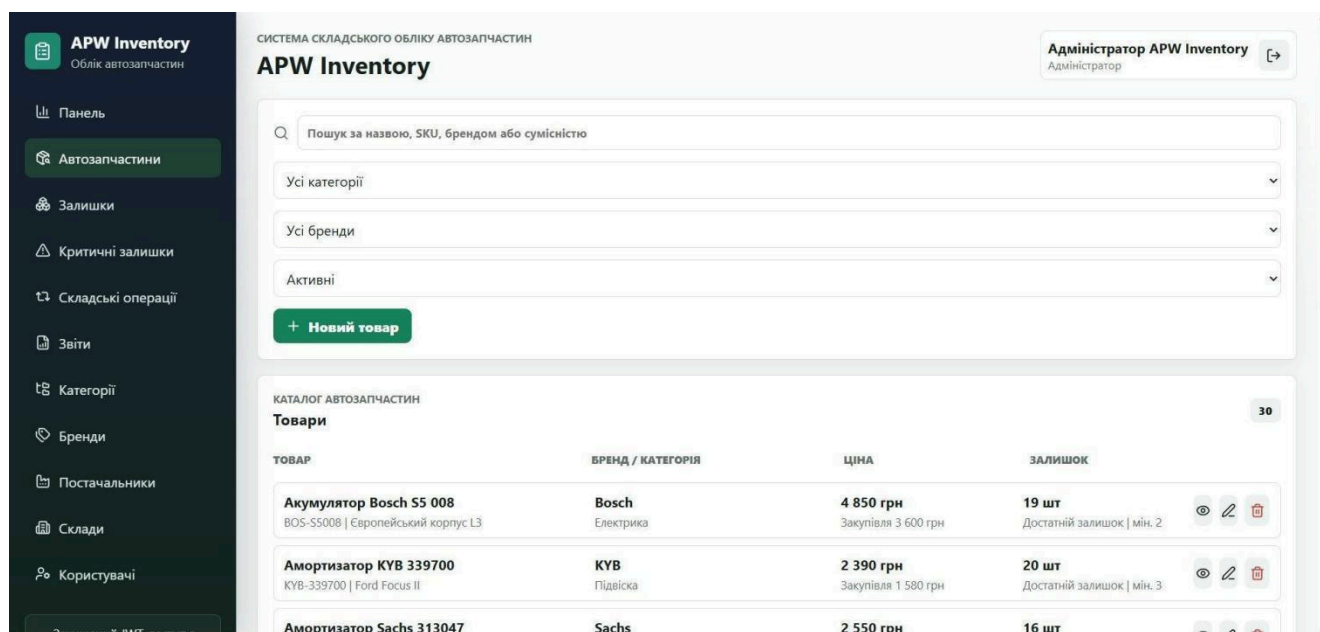


Рис. 3.17 Екранна форма каталогу автозапчастин

Для створення або редагування товарної позиції використовується окрема екранна форма. Вона містить поля для введення SKU, назви, опису, категорії, бренду, постачальника, сумісності з автомобілями, одиниці виміру, закупівельної ціни, роздрібною ціни, мінімального залишку та статусу активності. Частина полів заповнюється шляхом вибору значення з довідника, що зменшує ймовірність помилок і дублювання даних. Наприклад, користувач не вводить бренд вручну, а обирає його зі списку вже створених брендів.

Форма товару має важливе значення для якості даних у системі. Якщо товар буде створено з неправильним SKU, некоректною ціною або без категорії, це ускладнить подальший пошук, облік залишків і формування звітів. Тому під час збереження товару система перевіряє обов'язкові поля, унікальність SKU, коректність числових значень і зв'язок із довідниками.

СИСТЕМА СКЛАДСЬКОГО ОБЛІКУ АВТОЗАПЧАСТИН

APW Inventory

Адміністратор APW Inventory
Адміністратор

УПРАВЛІННЯ КАТАЛОГОМ

Створення товару

SKU: Назва: Категорія: Бренд:

Постачальник: Одиниця: Закупівельна ціна: Роздрібна ціна:

Мінімальний залишок: ☒ Активний товар

Сумісність:

Опис:

Зберегти товар

Рис. 3.18 Екранна форма створення та редагування автозапчастини

Сторінка залишків призначена для перегляду кількості автозапчастин у розрізі складів. На цій сторінці користувач бачить, на якому складі зберігається товар, яка його загальна кількість, зарезервована кількість, доступний залишок і мінімальний залишок. Якщо кількість товару дорівнює нулю або є меншою чи рівною мінімальному залишку, система відображає відповідний статус. Це дає змогу швидко визначити позиції, які потребують поповнення.

Особливість сторінки залишків полягає в тому, що вона показує не просто загальну кількість товару в системі, а конкретне розміщення товару по складах. Для складу автозапчастин це важливо, оскільки одна й та сама позиція може бути наявна на одному складі й відсутня на іншому. Завдяки такому поданню користувач може швидше приймати рішення щодо продажу, видачі, переміщення або замовлення товару.

СИСТЕМА СКЛАДСЬКОГО ОБЛІКУ АВТОЗАПЧАСТИН

APW Inventory

Адміністратор APW Inventory
Адміністратор

Склад: Усі склади ☐ Тільки низькі або нульові залишки Оновити

Залишки по складах 60

ТОВАР	СКЛАД	КІЛЬКІСТЬ	ВАРТІСТЬ	СТАТУС
Акумулятор Bosch S5 008 BOS-S5008 Bosch Електрика	Основний склад 04.05.2026, 12:31:55	14 шт 13 доступно, 1 зарезервовано	50 400 грн Закупівля 3 600 грн	Достатній залишок
Акумулятор Bosch S5 008 BOS-S5008 Bosch Електрика	Торгова точка 04.05.2026, 12:31:55	4 шт 4 доступно, 0 зарезервовано	14 400 грн Закупівля 3 600 грн	Достатній залишок
Амортизатор KYB 339700 KYB-339700 KYB Підвіска	Основний склад 04.05.2026, 12:31:55	14 шт 13 доступно, 1 зарезервовано	22 120 грн Закупівля 1 580 грн	Достатній залишок
Амортизатор KYB 339700 KYB-339700 KYB Підвіска	Торгова точка 04.05.2026, 12:31:55	6 шт 6 доступно, 0 зарезервовано	9 480 грн Закупівля 1 580 грн	Достатній залишок
Амортизатор Sachs 313047	Основний склад	16 шт	26 880 грн	

Захищений JWT-доступ з

Рис. 3.19 Екранна форма перегляду залишків по складах

Екранна форма складських операцій використовується для виконання основних дій із товарами. Користувач обирає тип операції: надходження, списання, переміщення, повернення або коригування. Далі він вибирає товар, склад, кількість і за потреби вводить коментар. Для операції переміщення інтерфейс додатково відображає поля складу-відправника та складу-одержувача. Це дозволяє чітко розмежувати логіку звичайної операції та операції між складського руху.

Під час роботи з цією формою frontend виконує попередню перевірку введених даних. Наприклад, користувач не повинен залишати порожні обов'язкові поля або вводити нульову кількість. Проте остаточна перевірка виконується на сервері. Якщо backend виявляє, що товару недостатньо для списання або переміщення, операція блокується, а користувач отримує повідомлення про причину відмови. Такий підхід забезпечує зручність введення даних і водночас зберігає контроль над цілісністю залишків.

СИСТЕМА СКЛАДСЬКОГО ОБЛІКУ АВТОЗАПЧАСТИН

APW Inventory

Адміністратор APW Inventory
Адміністратор

СКЛАДСЬКА ОПЕРАЦІЯ
Надходження, списання, повернення або коригування

Операція: Товар:

Склад: Постачальник:

Кількість: Ціна операції:

Коментар:

+ Створити

ПЕРЕМІЩЕННЯ
Перемістити залишок між складами

Товар: Зі складу:

На склад: Кількість:

Коментар:

➡ Перемістити

ЖУРНАЛ АУДИТУ
Складські операції

Усі типи

ДАТА	ТИП	ТОВАР	НАПРЯМ	КІЛЬКІСТЬ
------	-----	-------	--------	-----------

Рис. 3.20 Екранна форма виконання складської операції

Журнал транзакцій призначений для перегляду історії складських операцій. На цій сторінці відображаються записи про надходження, списання, переміщення, повернення та коригування. Для кожної транзакції показується тип операції, товар, склад або склади, кількість, дата виконання, користувач і коментар. Журнал потрібний для контролю руху автозапчастин, перевірки дій працівників і пошуку причин розбіжностей між фактичними та обліковими залишками.

Для зручності користувача на сторінці журналу передбачено фільтрацію. Менеджер або адміністратор може переглянути операції за певний період, за типом операції, товаром або постачальником. Це особливо корисно під час інвентаризації або перевірки руху конкретної товарної позиції. Наприклад, якщо залишок певної автозапчастини не збігається з фактичним, за допомогою журналу можна переглянути всі операції з нею та визначити, коли саме відбулася зміна.

ДАТА	ТИП	ТОВАР	НАПРЯМ	КІЛЬКІСТЬ
04.05.2026, 12:31:55	Находження	Антифриз Febi G12+ 1.5л FEB-32945 Febi	АвтоЛайн Дистрибуція → Торгова точка	6 210 грн
04.05.2026, 12:31:55	Находження	Антифриз Febi G12+ 1.5л FEB-32945 Febi	АвтоЛайн Дистрибуція → Основний склад	23 210 грн
04.05.2026, 12:31:55	Находження	Гальмівна рідина Bosch DOT4 1л BOS-DOT4-1L Bosch	ТехІмпорт Запчастини → Торгова точка	5 180 грн
04.05.2026, 12:31:55	Находження	Гальмівна рідина Bosch DOT4 1л BOS-DOT4-1L Bosch	ТехІмпорт Запчастини → Основний склад	11 180 грн
04.05.2026, 12:31:55	Находження	Фара Valeo 088358 VAL-088358 Valeo	ЄвроПартс Україна → Торгова точка	4 2 250 грн
04.05.2026, 12:31:55	Находження	Фара Valeo 088358 VAL-088358 Valeo	ЄвроПартс Україна → Основний склад	13 2 250 грн
04.05.2026, 12:31:55	Находження	Салонний фільтр Mann CU2939 MANN-CU2939 Mann Filter	АвтоЛайн Дистрибуція → Торгова точка	3 210 грн
04.05.2026, 12:31:55	Находження	Салонний фільтр Mann CU2939 MANN-CU2939 Mann Filter	АвтоЛайн Дистрибуція → Основний склад	20 210 грн

Рис. 3.21 Екранна форма журналу складських операцій

Сторінка звітів призначена для отримання узагальненої інформації про стан складу. У цьому розділі користувач може переглядати вартість залишків, список товарів із низьким запасом, рух товарів за період і загальну інформацію про склади. Також реалізовано можливість експорту даних у CSV. Це дозволяє користувачу завантажити звіт і надалі використовувати його для аналізу, передачі керівництву або обробки в електронних таблицях.

Експорт CSV є важливою функцією для практичного використання системи. У невеликому або середньому складі часто виникає потреба швидко отримати перелік залишків або операцій у вигляді файлу. Завдяки експорту користувач не копіює дані вручну, а отримує готовий файл із потрібними полями. Це зменшує час підготовки звітності та ризик помилок під час перенесення даних.

ТОВАР	СКЛАД	КІЛЬКІСТЬ	ЗАКУПІВЛЯ	ВАРТІСТЬ
Акумулятор Bosch S5 008 BOS-S5008	Основний склад	14 шт	3 600 грн	50 400 грн
Акумулятор Bosch S5 008 BOS-S5008	Торгова точка	4 шт	3 600 грн	14 400 грн
Амортизатор KYB 339700 KYB-339700	Основний склад	14 шт	1 580 грн	22 120 грн
Амортизатор KYB 339700 KYB-339700	Торгова точка	6 шт	1 580 грн	9 480 грн

Рис. 3.22 Екранна форма звітів та експорту CSV

Окрема екранна форма призначена для роботи з довідниками. До довідників у системі належать категорії, бренди, постачальники та склади. Вони використовуються під час створення товарів, фільтрації каталогу та обліку залишків. Наявність довідників дозволяє підтримувати впорядковану структуру даних і не допускати довільного дублювання назв. Наприклад, якщо бренд або категорія вводилися б вручну в кожному товарі, користувачі могли б створити кілька варіантів одного й того самого значення.

Сторінка керування користувачами призначена для адміністратора. На ній можна переглядати облікові записи, створювати нових користувачів, змінювати їх ролі, активувати або деактивувати доступ. Така сторінка необхідна для підтримки розмежування прав у системі. Наприклад, працівник складу не повинен отримувати доступ до керування користувачами, а користувач із правами перегляду не повинен мати можливості змінювати товари чи виконувати складські операції.

Під час створення користувача адміністратор задає логін, ім'я, пароль і роль. Надалі роль визначає, які сторінки та дії будуть доступні користувачу. Якщо працівник більше не повинен мати доступ до системи, його обліковий запис можна деактивувати без видалення історії операцій, які він виконував раніше.

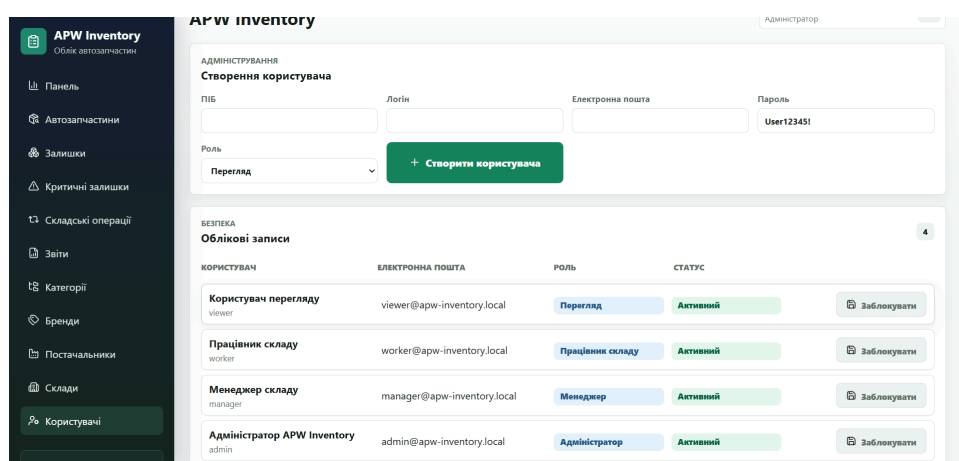


Рис. 3.23 Екранна форма керування користувачами

Загалом інтерфейс системи побудований так, щоб основні дії користувача відповідали реальним процесам складського обліку. Для перегляду великих наборів даних використовуються таблиці, для швидкого пошуку — пошукові поля та фільтри, для створення записів — форми з перевіркою обов'язкових полів, для контролю проблемних позицій — статуси залишків і візуальні повідомлення. Завдяки цьому користувач може виконувати основні дії без зайвих переходів і складних налаштувань.

Особливу увагу приділено повідомленням про результат виконання дій. Якщо операцію виконано успішно, користувач отримує підтвердження. Якщо сталася помилка, інтерфейс повинен показати зрозумілу причину. Наприклад, під час спроби списання більшої кількості, ніж доступно на складі, система не просто блокує операцію, а повідомляє про недостатній доступний залишок. Це робить роботу з системою більш передбачуваною.

Рольова модель також відображається на рівні інтерфейсу. Користувач бачить лише ті сторінки та кнопки, які відповідають його ролі. Це зменшує кількість зайвих елементів і допомагає уникати помилкових дій. Водночас приховування кнопок на frontend не є основним механізмом безпеки, оскільки остаточне обмеження доступу виконується на backend. Такий підхід поєднує зручність інтерфейсу та захищеність системи.

Таким чином, основні екранні форми APW Inventory забезпечують повний цикл роботи користувача із системою складського обліку: вхід до системи, перегляд статистики, роботу з каталогом товарів, контроль залишків, виконання складських операцій, перегляд журналу транзакцій, формування звітів, керування довідниками та користувачами. Реалізований інтерфейс відповідає функціональним модулям системи й дає змогу ефективно використовувати програмний продукт для відстеження залишків на складі автозапчастин.

4 ТЕСТУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ЗАЛИШКІВ НА СКЛАДІ АВТОЗАПЧАСТИН

4.1 Обґрунтування вибору видів тестування

Для перевірки APW Inventory застосовано функціональне, інтеграційне, негативне, безпекове, нефункціональне та регресійне тестування. Такий набір дозволяє перевірити як користувацькі сценарії, так і внутрішню бізнес-логіку складських операцій.

Функціональне тестування підтверджує роботу авторизації, каталогу, залишків, операцій, журналу, звітів і CSV-експорту. Інтеграційне тестування перевіряє взаємодію контролерів, сервісів, репозиторіїв і бази даних.

Негативні та безпекові перевірки потрібні для сценаріїв неправильного пароля, дублювання SKU, списання понад доступний залишок, запитів без JWT-токена та дій користувача без потрібної ролі.

Найбільшу увагу приділено забороні від'ємних залишків, журналюванню транзакцій, рольовому доступу та збереженню даних після перезапуску системи.

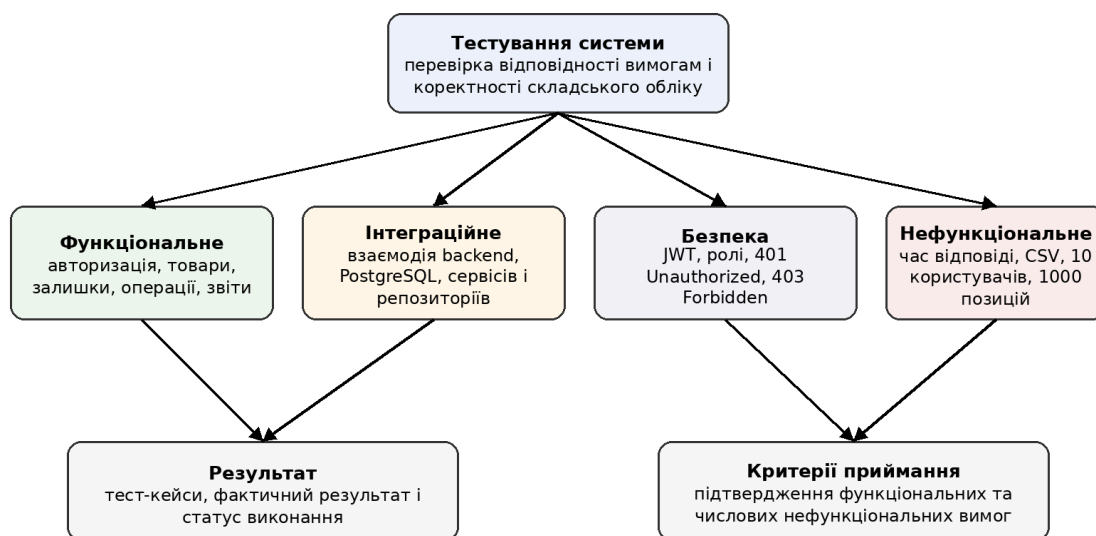


Рис. 4.1 Структура тестування системи APW Inventory

4.2 Розробка тест-кейсів для перевірки функціональних можливостей системи

Для перевірки функціональних можливостей системи APW Inventory було підготовлено набір тест-кейсів, які охоплюють основні сценарії роботи користувача. Тест-кейси сформовано відповідно до функціональних вимог, визначених у підрозділі 2.1, і з урахуванням реалізованих модулів системи: авторизації, каталогу автозапчастин, довідників, залишків, складських операцій, журналу транзакцій, звітності та керування користувачами.

Основна увага під час розробки тест-кейсів приділялася перевірці тих функцій, які безпосередньо впливають на достовірність складського обліку. До таких функцій належать створення товару з унікальним SKU, перегляд залишків по складах, виконання операцій надходження, списання, переміщення, повернення та коригування, а також заборона дій, які можуть призвести до некоректного стану системи. Окремо перевіряється авторизація користувачів і розмежування доступу за ролями.

Перед виконанням тестування в системі мають бути створені тестові дані: користувачі з ролями ADMIN, MANAGER, WAREHOUSE_WORKER та VIEWER, декілька категорій автозапчастин, брендів, постачальників, складів і товарних позицій. Наявність таких даних дозволяє перевірити як позитивні сценарії, коли система повинна успішно виконати дію, так і негативні сценарії, коли дія має бути заблокована.

Для тестування використано такі основні умови: система запущена через Docker Compose; backend доступний за портом 8080; frontend відкривається у браузері; база даних PostgreSQL містить початкові демонстраційні записи; користувачі мають різні ролі доступу; у каталозі є товари з унікальними SKU; у системі створено щонайменше два склади для перевірки переміщення товарів.

Основні тест-кейси для перевірки функціональних можливостей системи наведено в таблиці 4.1.

Тест-кейси для перевірки функціональних можливостей системи APW

Inventory

ID	Назва тест-кейсу	Передумова	Кроки виконання	Очікуваний результат
TC-01	Успішна авторизація адміністратора	У системі існує активний користувач з роллю ADMIN	Відкрити сторінку входу, ввести логін і пароль адміністратора, натиснути кнопку входу	Користувач успішно входить до системи, відкривається dashboard
TC-02	Вхід із неправильним паролем	У системі існує активний користувач	Ввести правильний логін і неправильний пароль, натиснути кнопку входу	Система не авторизує користувача та показує повідомлення про помилку
TC-05	Створення нового товару	Користувач має роль ADMIN або MANAGER	Відкрити форму створення товару, заповнити обов'язкові поля, зберегти запис	Новий товар створюється в каталозі та доступний для перегляду
TC-06	Створення товару з дубльованим SKU	У системі вже існує товар із таким SKU	Спробувати створити новий товар із тим самим SKU	Система блокує створення товару та показує повідомлення про дублювання SKU
TC-08	Перегляд залишків по складах	У системі є товари та склади	Перейти до розділу «Залишки»	Відображаються залишки товарів із прив'язкою до складів
TC-09	Операція надходження товару	Користувач має роль ADMIN, MANAGER або WAREHOUSE_WORKER	Обрати товар, склад, тип операції INCOME, ввести кількість і підтвердити	Залишок товару на вибраному складі збільшується, у журналі створюється транзакція
TC-10	Операція списання товару	На складі є достатня кількість товару	Обрати товар, склад, тип операції OUTCOME, ввести кількість, меншу або рівну доступній, підтвердити	Залишок товару зменшується, у журналі створюється транзакція списання
TC-11	Заборона списання понад доступний залишок	На складі кількість товару менша за кількість для списання	Спробувати списати більшу кількість товару, ніж доступно	Система блокує операцію, залишок не змінюється, користувач бачить повідомлення про недостатній залишок
TC-12	Переміщення товару між складами	У системі є два активні склади, на складі-відправнику є товар	Обрати операцію TRANSFER, товар, склад-відправник, склад-одержувач і кількість	Залишок зменшується на складі-відправнику та збільшується на складі-одержувачі
TC-16	Перегляд журналу транзакцій	У системі вже виконані складські операції	Перейти до розділу журналу операцій	Відображаються записи з типом операції, товаром, складом, кількістю, датою і користувачем
TC-20	Експорт звіту у CSV	На сторінці звітів відображені дані	Натиснути кнопку експорту CSV	Система формує та завантажує CSV-файл із даними звіту
TC-22	Заборона доступу до керування користувачами для неадміністратора	Користувач має роль MANAGER, WAREHOUSE_WORKER або VIEWER	Спробувати відкрити сторінку керування користувачами	Система не надає доступ до адміністративної функції

4.3 Перевірка нефункціональних вимог до системи

Перевірка нефункціональних вимог виконується для оцінювання якості роботи системи APW Inventory. Якщо функціональне тестування підтверджує наявність основних можливостей, то нефункціональне тестування показує, наскільки система є швидкою, захищеною, стабільною та придатною до практичного використання на складі автозапчастин.

Для системи складського обліку важливо, щоб основні сторінки відкривалися без значної затримки, пошук товарів виконувався швидко, складські операції зберігалися коректно, а дані не втрачалися після перезапуску. Окреме значення мають вимоги безпеки: доступ до API повинен надаватися лише авторизованим користувачам, а дії мають обмежуватися відповідно до ролі користувача.

Основні результати перевірки нефункціональних вимог наведено в таблиці 4.2.

Під час перевірки швидкодії основна увага приділялася сторінкам, які найчастіше використовуються в роботі складу: каталогу товарів, залишкам, операціям і звітам. При тестовому наборі даних система відкривала ці сторінки в межах установленого критерію до 2 секунд. Пошук товару за SKU, назвою, брендом або категорією також виконувався у межах критерію до 1 секунди.

Операції надходження, списання, повернення, коригування та переміщення перевірялися через інтерфейс і REST API. Успішні операції зберігалися в межах 1,5 секунди, після чого оновлювався залишок і створювався запис у журналі транзакцій. Окремо перевірено спроби списання та переміщення кількості, більшої за доступний залишок. У таких випадках система блокувала операцію, залишок не змінювався, а користувач отримував повідомлення про недостатню кількість товару.

Перевірка нефункціональних вимог системи APW Inventory

Код	Вимога	Критерій приймання	Результат
NF-01	Відкриття основних сторінок	«Товари», «Залишки», «Операції», «Звіти» відкриваються за ≤ 2 с при 1000 товарах	Виконано
NF-02	Пошук товару	Пошук за SKU, назвою, брендом або категорією виконується за ≤ 1 с	Виконано
NF-03	Збереження операції	INCOME, OUTCOME, TRANSFER, RETURN, ADJUSTMENT зберігаються за $\leq 1,5$ с	Виконано
NF-04	Одночасна робота	10 користувачів працюють без помилок 5хх	Виконано
NF-05	Заборона від'ємного залишку	У 100% випадків списання понад доступний залишок блокується	Виконано
NF-06	Захист API	Запити без токена або з недійсним токеном повертають 401 Unauthorized	Виконано
NF-07	Ролі доступу	Заборонені дії для ролей повертають 403 Forbidden	Виконано
NF-08	Захист паролів	Паролі зберігаються як BCrypt-хеші, не відкритим текстом	Виконано
NF-09	Журналювання операцій	Після 100% успішних операцій створюється запис у журналі	Виконано
NF-10	Експорт CSV	CSV-файл до 1000 рядків формується за ≤ 5 с	Виконано
NF-11	Збереження даних	Після перезапуску Docker-контейнерів дані залишаються доступними	Виконано

4.4 Аналіз результатів тестування

Після розробки тест-кейсів і перевірки нефункціональних вимог було виконано узагальнення результатів тестування системи APW Inventory. Основна мета аналізу полягала в тому, щоб визначити, чи відповідає реалізована система поставленим вимогам, чи коректно працюють основні модулі.

Тестування проводилося для основних функціональних частин системи: авторизації, розмежування прав доступу, каталогу автозапчастин, довідників, залишків, складських операцій, журналу транзакцій, звітності та експорту CSV. Окремо перевірялися негативні сценарії, зокрема неправильний пароль, дублювання SKU, спроба списання більшої кількості товару, ніж доступно на складі, звернення до API без JWT-токена та виконання заборонених дій користувачем із недостатньою роллю.

Узагальнені результати тестування наведено в таблиці 4.3.

Таблиця 4.3

Узагальнені результати тестування системи APW Inventory

Напрямок тестування	Що перевірялося	Результат
Авторизація користувачів	Вхід із правильними та неправильними обліковими даними	Виконано
Розмежування доступу	Доступ до функцій відповідно до ролей ADMIN, MANAGER, WAREHOUSE WORKER, VIEWER	Виконано
Каталог автозапчастин	Створення, перегляд, редагування, пошук і фільтрація товарів	Виконано
Унікальність SKU	Заборона створення товару з уже наявним артикулом	Виконано
Довідники	Робота з категоріями, брендами, постачальниками та складами	Виконано
Облік залишків	Відображення кількості товарів у розрізі складів	Виконано
Складські операції	Надходження, списання, переміщення, повернення та коригування	Виконано
Заборона від'ємних залишків	Блокування списання або переміщення понад доступну кількість	Виконано
Журнал транзакцій	Створення запису після кожної успішної складської операції	Виконано
Звітність	Перегляд залишків, критичних запасів і руху товарів	Виконано
Експорт CSV	Формування файлу звіту для подальшої обробки	Виконано
Нефункціональні вимоги	Швидкодія, захист API, ролі, збереження даних після перезапуску	Виконано

Результати тестування показали, що авторизація користувачів працює коректно. Користувач із правильними обліковими даними отримує доступ до системи, а при введенні неправильного пароля система блокує вхід і показує повідомлення про помилку. Захищені API недоступні без JWT-токена, що підтверджує правильність реалізації механізму автентифікації.

Перевірка рольової моделі показала, що доступ до функцій системи обмежується відповідно до ролі користувача. Адміністратор має доступ до керування користувачами, менеджер може працювати з товарами, залишками та звітами, працівник складу може виконувати складські операції, а користувач із роллю VIEWER не має права змінювати дані. При спробі виконати заборонену дію система повертає відмову доступу.

Тестування каталогу автозапчастин підтвердило, що система дозволяє створювати, редагувати, переглядати та шукати товарні позиції. Пошук за SKU, назвою, брендом або категорією працює коректно. Також перевірено заборону дублювання SKU: якщо користувач намагається створити товар із уже наявним артикулом, система блокує таку дію. Це важливо для підтримки впорядкованого каталогу автозапчастин.

Найважливішою частиною тестування була перевірка складських операцій. Операція надходження збільшує залишок товару на вибраному складі, операція списання зменшує залишок, переміщення змінює кількість одразу на двох складах, повернення додає товар назад на склад, а коригування встановлює нове фактичне значення залишку. Після кожної успішної операції в журналі створюється відповідний запис.

ВИСНОВКИ

У процесі виконання бакалаврської роботи було розроблено вебсистему відстеження залишків на складі автозапчастин APW Inventory, призначену для автоматизації основних процесів складського обліку малого або середнього підприємства. Розроблена система дає змогу вести каталог автозапчастин, контролювати залишки товарів по складах, виконувати складські операції, фіксувати історію руху товарів, розмежовувати права доступу користувачів і формувати звітну інформацію.

У першому розділі було досліджено предметну галузь складського обліку автозапчастин. Визначено, що облік таких товарів має специфічні особливості, оскільки кожна автозапчастина може характеризуватися SKU, брендом, категорією, постачальником, сумісністю з автомобілями, закупівельною та роздрібною ціною, мінімальним залишком і місцем зберігання. Було встановлено, що ручний або частково автоматизований облік за допомогою паперових журналів чи електронних таблиць не забезпечує достатньої точності, оскільки може призводити до дублювання товарів, несвоєчасного оновлення залишків, відсутності журналу операцій і складності контролю критичних запасів.

Також у першому розділі було розглянуто існуючі програмні рішення для складського обліку, зокрема Odoo Inventory, Zoho Inventory, Sortly та inFlow Inventory. Аналіз показав, що такі системи мають корисні функції для управління запасами, однак не завжди повністю відповідають потребам малого або середнього складу автозапчастин без додаткового налаштування. Одні рішення є занадто масштабними й орієнтованими на комплексну ERP-автоматизацію, інші залежать від хмарної моделі або більше підходять для простого інвентарного обліку. Це підтвердило актуальність розробки спеціалізованої вебсистеми, орієнтованої саме на контроль залишків автозапчастин.

У другому розділі було виконано проєктування системи APW Inventory. Сформульовано функціональні та нефункціональні вимоги до програмного

забезпечення, визначено ролі користувачів, змодельовано варіанти використання системи, спроектовано клієнт-серверну архітектуру, структуру бази даних, логіку складських операцій та інтерфейс користувача. У базі даних було передбачено таблиці для користувачів, категорій, брендів, постачальників, складів, товарів, залишків і складських транзакцій. Окрему увагу було приділено тому, що залишок товару повинен зберігатися не в картці товару, а як окремий зв'язок між товаром і складом, що дає змогу правильно обліковувати одну й ту саму автозапчастину на різних складах.

Під час проектування логіки складських операцій було визначено правила виконання операцій надходження, списання, переміщення, повернення та коригування. Було передбачено, що система не повинна дозволяти списання або переміщення товару в кількості, більшій за доступний залишок. Також було визначено, що після кожної успішної складської операції має створюватися запис у журналі транзакцій. Це дозволяє забезпечити прозорість руху автозапчастин і можливість подальшого аналізу історії змін.

У третьому розділі було описано реалізацію системи. Серверну частину розроблено мовою Java з використанням Spring Boot, Spring Web, Spring Data JPA, Spring Security, JWT, Flyway і PostgreSQL. Backend-частина реалізує REST API, бізнес-логіку, роботу з базою даних, автентифікацію, авторизацію, перевірку ролей користувачів і контроль складських операцій. Для захисту доступу використано JWT-токени, BCrypt-хешування паролів і рольову модель доступу з ролями ADMIN, MANAGER, WAREHOUSE_WORKER та VIEWER.

Клієнтську частину системи реалізовано з використанням React, TypeScript, Vite та Axios. Вебінтерфейс містить сторінки авторизації, головної панелі, каталогу автозапчастин, створення та редагування товарів, перегляду залишків, виконання складських операцій, журналу транзакцій, звітів, довідників і керування користувачами. Інтерфейс враховує роль користувача: користувачі бачать лише ті дії та сторінки, які відповідають їхнім повноваженням, а остаточна перевірка доступу виконується на серверній частині.

У четвертому розділі було проведено тестування системи APW Inventory. Для перевірки програмного продукту було використано функціональне, інтеграційне, негативне, безпекове, нефункціональне та регресійне тестування. Було розроблено тест-кейси для перевірки авторизації, ролей користувачів, каталогу товарів, унікальності SKU, залишків по складах, складських операцій, журналу транзакцій, звітності та експорту CSV. Окремо перевірено критичні сценарії: заборону списання понад доступний залишок, блокування доступу без JWT-токена, повернення помилки 403 для заборонених дій і збереження даних після перезапуску Docker-контейнерів.

Результати тестування показали, що система коректно виконує основні функції складського обліку. Операції надходження, списання, переміщення, повернення та коригування змінюють залишки відповідно до запроєктованої логіки. Система не допускає появи від'ємних залишків, створює записи в журналі транзакцій після успішних операцій, забезпечує рольове розмежування доступу та дозволяє формувати звіти з можливістю експорту у CSV. Нефункціональні вимоги щодо швидкодії, захисту API, журналювання, збереження даних і формування звітів також були виконані.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Дзядевич Б.О., Корецька В.О. Розробка системи відстеження залишків на складі автозапчастин. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.129 -130.

2. Дзядевич Б.О., Корецька В.О. Забезпечення безпеки інформації у web-системі відстеження залишків на складі автозапчастин. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація». 1-3 травня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій Збірник тез. *(подано до друку)*.

ПЕРЕЛІК ПОСИЛАНЬ

1. IBM. What is Inventory Management? // IBM Think. URL: <https://www.ibm.com/think/topics/inventory-management>.
2. Odoo S.A.. Inventory // Odoo 19.0 Documentation. URL: https://www.odoo.com/documentation/19.0/applications/inventory_and_mrp/inventory.html.
3. Zoho Corporation. Online Inventory Management Software // Zoho Inventory. URL: <https://www.zoho.com/inventory/>.
4. Sortly. Inventory Simplified // Sortly. URL: <https://www.sortly.com/>.
5. inFlow Inventory. Inventory Management Software // inFlow. URL: <https://www.inflowinventory.com/>.
6. Oracle. JDK 21 Documentation // Oracle Java Documentation. URL: <https://docs.oracle.com/en/java/javase/21/>.
7. Spring. Spring Boot Reference Documentation // Spring Documentation. URL: <https://docs.spring.io/spring-boot/reference/>.
8. Spring. Spring Security Reference // Spring Documentation. URL: <https://docs.spring.io/spring-security/reference/index.html>.
9. Spring. Spring Data JPA Reference Documentation // Spring Documentation. URL: <https://docs.spring.io/spring-data/jpa/reference/index.html>.
10. The PostgreSQL Global Development Group. PostgreSQL Documentation // PostgreSQL. URL: <https://www.postgresql.org/docs/>.
11. Redgate. Redgate Flyway Documentation // Redgate Flyway. URL: <https://documentation.red-gate.com/fd>.
12. Meta Open Source. React Documentation // React. URL: <https://react.dev/>.
13. Microsoft. TypeScript Documentation // TypeScript. URL: <https://www.typescriptlang.org/docs/>.
14. Vite. Getting Started // Vite Documentation. URL: <https://vite.dev/guide/>.

15. Axios. Getting Started // Axios Documentation. URL: <https://axios-http.com/uk/docs/intro>.
16. Docker Inc.. Docker Compose // Docker Documentation. URL: <https://docs.docker.com/compose/>.
17. IETF. RFC 7519: JSON Web Token (JWT) // IETF Datatracker. URL: <https://datatracker.ietf.org/doc/html/rfc7519>.
18. OWASP Foundation. OWASP API Security Top 10 – 2023 // OWASP API Security Project. URL: <https://owasp.org/API-Security/editions/2023/en/0x00-header/> (дата звернення: 04.05.2026).
19. OWASP Foundation. JSON Web Token for Java Cheat Sheet // OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html.
20. Mozilla. HTTP Response Status Codes // MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status>.
21. Spring. Testing // Spring Boot Reference Documentation. URL: <https://docs.spring.io/spring-boot/reference/testing/index.html>.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Розробка системи відстеження залишків на складі автозапчастин

Виконав студент 4 курсу
групи ПД-42

Дзядевич Богдан Олександрович
Керівник роботи

канд. пед. наук, доцент Корецька Вікторія Олександрівна
Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення оперативності контролю складських залишків автозапчастин засобами веб-системи, що забезпечує ведення каталогу товарів, облік складських операцій, розмежування ролей доступу, контроль критичних залишків і формування звітності.

Об'єкт дослідження: процес обліку та відстеження залишків автозапчастин на складі.

Предмет дослідження: програмні засоби та технології автоматизації обліку товарів, виконання складських операцій, контролю залишків автозапчастин і формування звітності у веб-системі управління складом.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Дослідити предметну галузь складського обліку автозапчастин з метою коректної розробки.
2. Проаналізувати проблеми ручного та частково автоматизованого обліку залишків, для уникнення їх при розробці.
3. Розглянути існуючі програмні рішення для складського обліку та визначити їх переваги та недоліки, для кращого розуміння потреби користувача.
4. Сформулювати функціональні та нефункціональні вимоги до системи.
5. Спроекувати архітектуру, структуру бази даних, ролі користувачів та логіку складських операцій.
6. Реалізувати backend, frontend, авторизацію, каталог товарів, залишки, транзакції та звітність.
7. Провести тестування системи й перевірити коректність сценаріїв роботи.

3

АНАЛІЗ АНАЛОГІВ

Критерій	Odoo Inventory	Zoho Inventory	Sortly	inFlow Inventory	APW Inventory
SKU / артикул	+	+	+	+	+
Залишки по складах	+	+	-	+	+
Складські операції	+	+	-	+	+
Критичні залишки	+	+	+	-	+
Ролі доступу	+	+	+	+	+
Локальне розгортання	-	-	-	+	+
Контрольована backend-логіка для 5 типів складських операцій	-	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

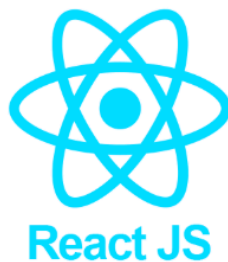
1. Авторизація користувачів і ролі доступу;
2. Каталог автозапчастин із SKU, брендом, категорією та постачальником;
3. Облік залишків по складах;
4. Операції: надходження, витрата, коригування, переміщення та журнал транзакцій
5. журнал транзакцій, звіти та експорт CSV.

Нефункціональні вимоги:

1. Відкриття основних сторінок – до 2 с при 1000 товарах;
2. Пошук товару – до 1 с;
3. Збереження операції – до 1,5 с;
4. Заборона від'ємних залишків у 100% випадків;
5. Захист API JWT-токеном і перевірка ролей.

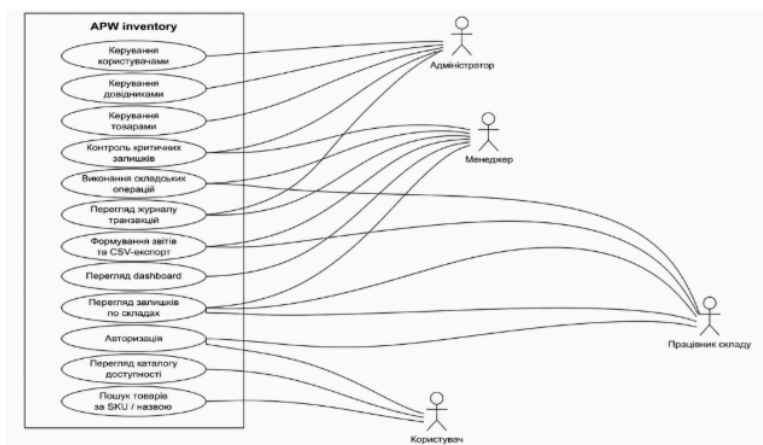
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



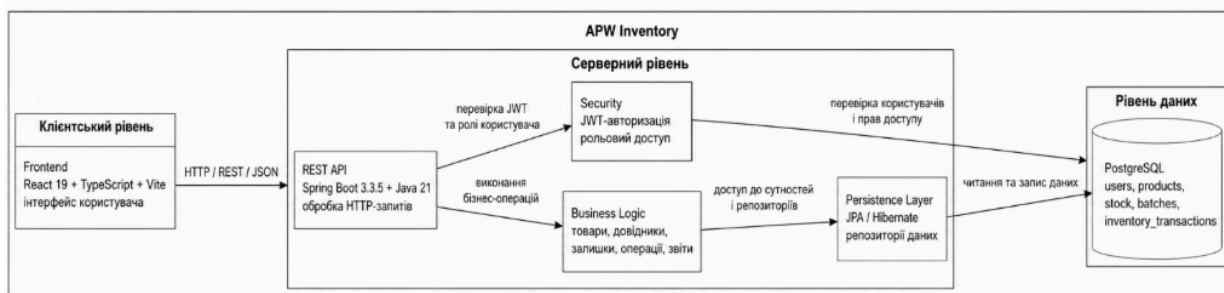
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



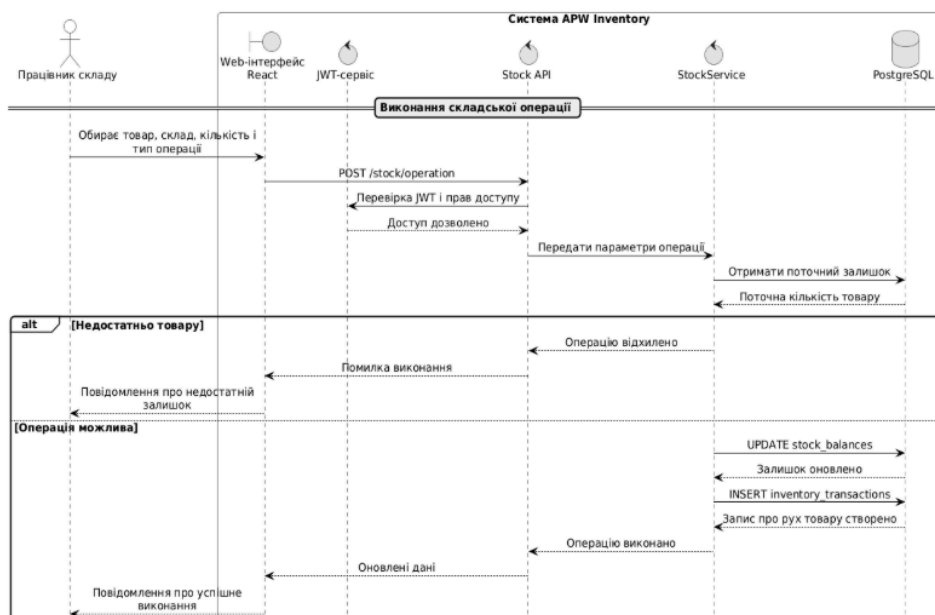
7

АРХІТЕКТУРА СИСТЕМИ



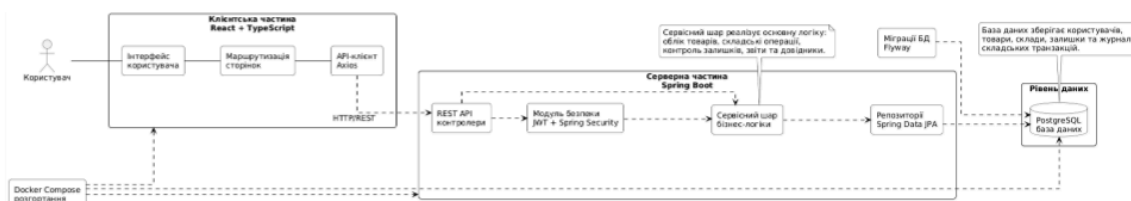
8

ДІАГРАМА ПОСЛІДОВНОСТІ ВИКОНАННЯ СКЛАДСЬКИХ ОПЕРАЦІЙ



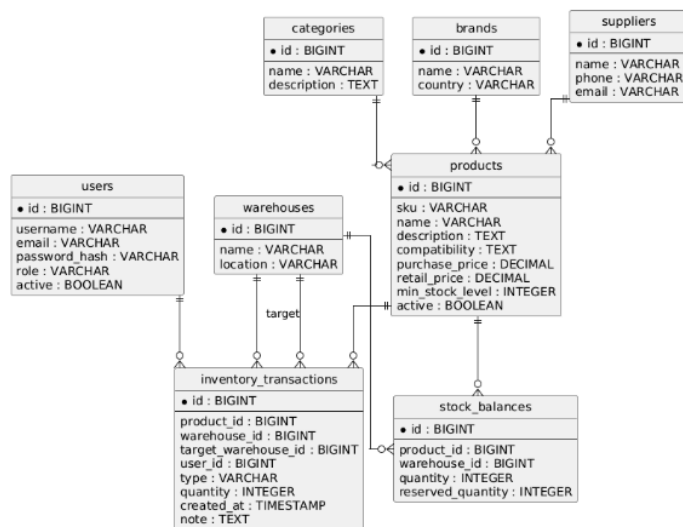
9

ДІАГРАМА КОМПОНЕНТІВ СИСТЕМИ



10

СТРУКТУРА БАЗИ ДАНИХ



11

ЕКРАННІ ФОРМИ

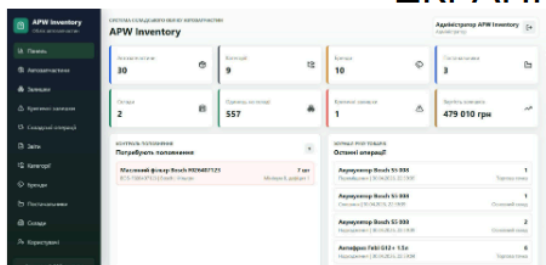


Рис 1. Головна панель системи

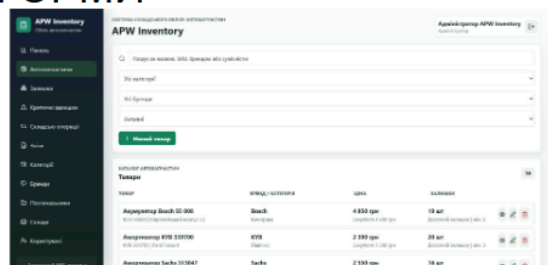


Рис 2. Каталог автозапчастин



Рис 3. Виконання складської операції

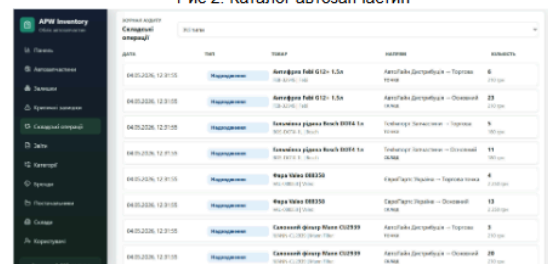


Рис 4. Журнал складських операцій

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Дзядевич Б.О., Корецька В.О. Розробка системи відстеження залишків на складі автозапчастин. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.129 -130.
2. Дзядевич Б.О., Корецька В.О. Забезпечення безпеки інформації у web-системі відстеження залишків на складі автозапчастин. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація». 1-3 травня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій Збірник тез. (подано до друку).

13

ВИСНОВКИ

1. Проаналізовано предметну галузь складського обліку автозапчастин, та на основі цього визначено основні проблеми ручного й частково автоматизованого контролю залишків, зокрема таких як: помилки під час внесення даних, несвоєчасне оновлення залишків, дублювання інформації, складність швидкого пошуку потрібних запчастин, відсутність актуальної аналітики, складений контроль руху запчастин між складом і продажами
2. Проаналізовано існуючі аналоги, на основі цього визначено функціональні та нефункціональні вимоги, обрано стек технологій для розробки, а саме Java, ReactJS, JWT, PostgreSQL.
3. На основі обраного стеку технологій, , реалізовано backend на Java/Spring Boot, frontend на React/TypeScript, реалізовано такий функціонал: авторизацію JWT, модулі товарів, складів, залишків, транзакцій, звітів і CSV-експорту.
4. Було протестовано систему на основі визначених функціональних і нефункціональних вимог до системи, а також шляхом перевірки роботи основних сценаріїв використання: обліку залишків, обмеження від'ємних значень, розмежування прав доступу, фіксації операцій у журналі та стабільності роботи системи.

14

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

Лістинг Б.1 – Фрагмент міграції структури бази даних

V1_init_schema.sql

```
create table products (
    id bigserial primary key,
    sku varchar(64) not null unique,
    name varchar(180) not null,
    description varchar(1000),
    category_id bigint not null references
categories(id),
    brand_id bigint not null references
brands(id),
    supplier_id bigint references
suppliers(id),
    compatibility varchar(500),
    unit varchar(30) not null default
'шт',
    purchase_price numeric(12, 2) not
null,
    retail_price numeric(12, 2) not null,
    minimum_stock integer not null default
1,
    active boolean not null default true,
    created_at timestamptz not null
default now(),
    updated_at timestamptz not null
default now(),
    constraint chk_products_prices check
(purchase_price >= 0 and retail_price >=
0),
    constraint chk_products_minimum_stock
check (minimum_stock >= 0)
);

create table stock_balances (
    id bigserial primary key,
    product_id bigint not null references
products(id),
    warehouse_id bigint not null
references warehouses(id),
    quantity integer not null default 0,
    reserved_quantity integer not null
default 0,
    updated_at timestamptz not null
default now(),
    constraint uk_stock_product_warehouse
unique (product_id, warehouse_id),
    constraint chk_stock_quantity check
(quantity >= 0),
    constraint
chk_stock_reserved_lte_quantity check
(reserved_quantity <= quantity)
);

create table inventory_transactions (
    id bigserial primary key,
    type varchar(40) not null,
    product_id bigint not null references
products(id),
    from_warehouse_id bigint references
warehouses(id),
```

```
to_warehouse_id bigint references
warehouses(id),
    supplier_id bigint references
suppliers(id),
    quantity integer not null,
    unit_price numeric(12, 2),
    comment varchar(700),
    created_by_id bigint references
users(id),
    operation_date timestamptz not null
default now(),
    constraint chk_transaction_quantity
check (quantity >= 0),
    constraint chk_transaction_unit_price
check (unit_price is null or unit_price >=
0)
);
```

Лістинг Б.2 – Конфігурація безпеки SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {
    private final UserRepository
userRepository;

    @Bean
    public SecurityFilterChain
securityFilterChain(
        HttpSecurity http,
        CorsConfigurationSource
corsConfigurationSource,
        JwtAuthenticationFilter
jwtAuthenticationFilter,
        ObjectMapper objectMapper
    ) throws Exception {
        http
        .csrf(AbstractHttpConfigurer::disable)
        .cors(cors ->
cors.configurationSource(corsConfiguration
Source))
        .sessionManagement(session
->
session.sessionCreationPolicy(SessionCreat
ionPolicy.STATELESS))
        .exceptionHandling(exceptions ->
exceptions
        .authenticationEntryPoint((request,
response, authException) ->
writeSecurityError(response,
HttpStatus.UNAUTHORIZED,
"Потрібна авторизація", objectMapper))
        .accessDeniedHandler((request, response,
accessDeniedException) ->
```

```

writeSecurityError(response,
HttpStatus.FORBIDDEN,

"Недостатньо прав для виконання операції",
objectMapper))

.authorizeHttpRequests(auth -> auth

.requestMatchers(HttpMethod.OPTIONS,
"/**").permitAll()

.requestMatchers("/api/auth/login",
"/actuator/health").permitAll()

.requestMatchers(HttpMethod.DELETE,
"/api/**").hasRole("ADMIN")

.anyRequest().authenticated())

.authenticationProvider(authenticationProv
ider())

.addFilterBefore(jwtAuthenticationFilter,
UsernamePasswordAuthenticationFilter.class
);

return http.build();
}

@Bean
public PasswordEncoder
passwordEncoder() {
return new
BCryptPasswordEncoder();
}
}

```

Лістинг Б.3 – Генерація та перевірка JWT-токена

JwtService.java

```

@Service
public class JwtService {
private final SecretKey key;
private final long expirationMinutes;

public JwtService(
@Value("${app.jwt.secret}")
String secret,

@Value("${app.jwt.expiration-minutes}")
long expirationMinutes
) {
this.key =
Keys.hmacShaKeyFor(secret.getBytes(Standar
dCharsets.UTF_8));
this.expirationMinutes =
expirationMinutes;
}

public String generateToken(AppUser
user) {
Instant now = Instant.now();
return Jwts.builder()
.subject(user.getEmail())
.claims(Map.of(
"userId",
user.getId(),
"username",
user.getLoginUsername(),
"email",
user.getEmail(),

```

```

"role",
user.getRole().name(),
"fullName",
user.getFullName()))
.issuedAt(Date.from(now))

.expiration(Date.from(now.plus(expirationM
inutes, ChronoUnit.MINUTES)))
.signWith(key)
.compact();
}

public boolean isValidToken(String
token, UserDetails userDetails) {
String username =
extractUsername(token);
return
username.equals(userDetails.getUsername())
&& !isTokenExpired(token);
}

private <T> T extractClaim(String
token, Function<Claims, T> resolver) {
Claims claims = Jwts.parser()
.verifyWith(key)
.build()
.parseSignedClaims(token)
.getPayload();
return resolver.apply(claims);
}
}

```

Лістинг Б.4 – Сервіс авторизації AuthService.java

```

@Service
public class AuthService {
private final AuthenticationManager
authenticationManager;
private final JwtService jwtService;
private final UserRepository
userRepository;
private final PasswordEncoder
passwordEncoder;

public AuthResponse login(LoginRequest
request) {
if (request == null) {
throw new
BadRequestException("Логін і пароль є
обов'язковими");
}
String login =
request.login().trim().toLowerCase();
Authentication authentication =
authenticationManager.authenticate(
new
UsernamePasswordAuthenticationToken(login,
request.password())
);
AppUser user = (AppUser)
authentication.getPrincipal();
return new
AuthResponse(jwtService.generateToken(user
), UserProfileResponse.from(user));
}

@Transactional
public UserProfileResponse
register(RegisterRequest request) {
String email =
request.email().trim().toLowerCase();

```

```

        String username =
request.username().trim().toLowerCase();
        if
(userRepository.existsByEmailIgnoreCase(email)) {
            throw new
BadRequestException("Користувач із таким
email уже існує");
        }
        if
(userRepository.existsByUsernameIgnoreCase(username)) {
            throw new
BadRequestException("Користувач із таким
username уже існує");
        }
        AppUser user = new AppUser();
        user.setEmail(email);
        user.setUsername(username);

user.setFullName(request.fullName().trim()
);
        user.setRole(request.role());

user.setPasswordHash(passwordEncoder.encode(request.password()));
        user.setActive(true);
        return
UserProfileResponse.from(userRepository.save(user));
    }
}

```

Лістинг Б.5 – Основні операції з каталогом

ProductService.java

```

@Service
public class ProductService {
    @Transactional(readOnly = true)
    public PageResponse<ProductResponse>
search(String query, Long categoryId, Long
brandId,

Boolean active, Pageable pageable) {
    return
PageResponse.from(productRepository.search(query, categoryId, brandId,
active,
pageable).map(this::toResponse));
}

    @Transactional
    public ProductResponse
create(ProductRequest request) {
        String sku =
normalizeSku(request.sku());
        if
(productRepository.existsBySkuIgnoreCase(sku)) {
            throw new
BadRequestException("Товар із таким SKU
уже існує");
        }
        Product product = new Product();
        apply(product, request, sku);
        return
toResponse(productRepository.save(product));
    }

    @Transactional

```

```

    public ProductResponse update(Long id,
ProductRequest request) {
        Product product =
findDetailed(id);
        String sku =
normalizeSku(request.sku());

productRepository.findBySkuIgnoreCase(sku)
        .filter(existing ->
!existing.getId().equals(id))
        .ifPresent(existing -> {
            throw new
BadRequestException("Товар із таким SKU
уже існує");
        });
        apply(product, request, sku);
        return toResponse(product);
    }

    private void apply(Product product,
ProductRequest request, String sku) {
        Category category =
categoryRepository.findById(request.categoryId())
        .orElseThrow(() -> new
ResourceNotFoundException("Категорію не
знайдено"));
        Brand brand =
brandRepository.findById(request.brandId())
        .orElseThrow(() -> new
ResourceNotFoundException("Бренд не
знайдено"));
        product.setSku(sku);

product.setName(request.name().trim());
        product.setCategory(category);
        product.setBrand(brand);

product.setSupplier(findSupplier(request.s
upplierId()));

product.setCompatibility(clean(request.com
patibility()));

product.setUnit(request.unit().trim());

product.setPurchasePrice(request.purchaseP
rice());

product.setRetailPrice(request.retailPrice
());

product.setMinimumStock(request.minimumSto
ck());
        product.setActive(request.active()
== null || request.active());
    }
}

```

Лістинг Б.6 – Бізнес-логіка складських операцій

StockService.java

```

@Service
public class StockService {
    @Transactional
    public StockBalanceResponse
income(StockOperationRequest request,
AppUser user) {

        requirePositive(request.quantity(),

```

```

"Кількість надходження має бути більшою за
нуль");
    Product product =
activeProduct(request.productId());
    Warehouse warehouse =
activeWarehouse(requiredWarehouseId(request.
t));
    Supplier supplier =
findSupplier(request.supplierId());
    StockBalance balance =
getOrCreateBalance(product, warehouse);

balance.setQuantity(balance.getQuantity()
+ request.quantity());

transactionRepository.save(transaction(Transac
tionType.INCOME, product, null,
warehouse, supplier,
request.quantity(), request.unitPrice(),
request.comment(), user));
    return
StockBalanceResponse.from(balance);
}

@Transactional
public StockBalanceResponse
outcome(StockOperationRequest request,
AppUser user) {

    requirePositive(request.quantity(),
"Кількість списання має бути більшою за
нуль");
    Product product =
activeProduct(request.productId());
    Warehouse warehouse =
activeWarehouse(requiredWarehouseId(request.
t));
    StockBalance balance =
existingBalance(product.getId(),
warehouse.getId());
    ensureAvailable(balance,
request.quantity());

balance.setQuantity(balance.getQuantity()
- request.quantity());

transactionRepository.save(transaction(Transac
tionType.OUTCOME, product, warehouse,
null, null,
request.quantity(), request.unitPrice(),
request.comment(), user));
    return
StockBalanceResponse.from(balance);
}

@Transactional
public List<StockBalanceResponse>
transfer(TransferRequest request, AppUser
user) {
    if
(request.fromWarehouseId().equals(request.
toWarehouseId())) {
        throw new
BadRequestException("Склади для
переміщення мають бути різними");
    }
    Product product =
activeProduct(request.productId());
    Warehouse from =
activeWarehouse(request.fromWarehouseId())
;

```

```

Warehouse to =
activeWarehouse(request.toWarehouseId());
    StockBalance source =
existingBalance(product.getId(),
from.getId());
    ensureAvailable(source,
request.quantity());
    StockBalance target =
getOrCreateBalance(product, to);

source.setQuantity(source.getQuantity() -
request.quantity());

target.setQuantity(target.getQuantity() +
request.quantity());

transactionRepository.save(transaction(Transac
tionType.TRANSFER, product, from,
to, null,
request.quantity(), null,
request.comment(), user));
    return
List.of(StockBalanceResponse.from(source),
StockBalanceResponse.from(target));
}

private void
ensureAvailable(StockBalance balance, int
quantity) {
    int available =
balance.getQuantity() -
balance.getReservedQuantity();
    if (available < quantity) {
        throw new
BadRequestException("Недостатньо товару на
складі для виконання операції");
    }
}
}

```

Лістинг Б.7 – Модуль звітності ReportsController.java та DashboardService.java

```

@RestController
@RequestMapping("/api/reports")
public class ReportsController {
    private final DashboardService
dashboardService;

    @GetMapping("/dashboard")
    public DashboardSummaryResponse
dashboard() {
        return dashboardService.summary();
    }

    @GetMapping("/low-stock")
    public List<LowStockResponse>
lowStock() {
        return
dashboardService.lowStock();
    }

    @GetMapping("/stock-value")
    public StockValueResponse
stockValue(@RequestParam(required = false)
Long warehouseId) {
        return new
StockValueResponse(warehouseId,
dashboardService.stockValue(warehouseId));
    }

    @GetMapping("/transactions")

```

```

    public List<TransactionResponse>
transactions(
    @RequestParam(required =
false) Instant from,
    @RequestParam(required =
false) Instant to,
    @RequestParam(required =
false) TransactionType type,
    @RequestParam(required =
false) Long productId,
    @RequestParam(required =
false) Long supplierId) {
    return
dashboardService.transactions(from, to,
type, productId, supplierId);
}

@Service
public class DashboardService {
    @Transactional(readOnly = true)
    public DashboardSummaryResponse
summary() {
        List<LowStockResponse> lowStock =
lowStock();
        return new
DashboardSummaryResponse(
            productRepository.count(),
productRepository.countByActiveTrue(),

categoryRepository.count(),
brandRepository.count(),
supplierRepository.count(),

warehouseRepository.count(),
zero(stockBalanceRepository.totalQuantity(
)),

zero(stockBalanceRepository.totalReservedQ
uantity()),

zero(productRepository.countLowStockProduc
ts()),

zero(stockBalanceRepository.stockValue(nul
l)),

lowStock.stream().limit(8).toList(),
stockService.recentTransactions(10));
    }
}

```

Лістинг Б.8 – Маршрутизація клієнтської частини App.tsx

```

export default function App() {
    return (
        <Routes>
            <Route path="/login"
element={<LoginPage />} />
            <Route element={<ProtectedRoute />}>
                <Route element={<AppLayout />}>
                    <Route index
element={<DashboardPage />} />
                    <Route path="products"
element={<ProductsPage />} />
                    <Route path="products/new"
element={<ProductFormPage />} />
                    <Route path="products/:id"
element={<ProductDetailsPage />} />
                    <Route path="products/:id/edit"
element={<ProductFormPage />} />
                </Route>
            </Route>
        </Routes>
    );
}

```

```

        <Route path="categories"
element={<CategoriesPage />} />
        <Route path="brands"
element={<BrandsPage />} />
        <Route path="suppliers"
element={<SuppliersPage />} />
        <Route path="warehouses"
element={<WarehousesPage />} />
        <Route path="stock"
element={<StockPage />} />
        <Route path="low-stock"
element={<LowStockPage />} />
        <Route path="transactions"
element={<TransactionsPage />} />
        <Route path="reports"
element={<ReportsPage />} />
    </Route>
</Route>
    <Route element={<ProtectedRoute
roles={['ADMIN']} />}>
        <Route element={<AppLayout />}>
            <Route path="users"
element={<UsersPage />} />
        </Route>
    </Route>
    <Route path="*" element={<Navigate
to="/" replace />} />
</Routes>
);
}

```

Лістинг Б.9 – Контекст авторизації та захищені маршрути AuthContext.tsx, ProtectedRoute.tsx

```

export function AuthProvider({ children }:
{ children: ReactNode }) {
    const [user, setUser] =
useState<UserProfile | null>(() =>
getStoredUser());

    async function signIn(loginValue:
string, password: string) {
        const session = await
login(loginValue, password);
        saveSession(session);
        setUser(session.user);
    }

    function signOut() {
        clearSession();
        setUser(null);
    }

    const value =
useMemo<AuthContextValue>(() => ({
        user,
        isAuthenticated: Boolean(user),
        signIn,
        signOut,
        hasAnyRole: (...roles) => Boolean(user
&& roles.includes(user.role)),
    })), [user]);

    return <AuthContext.Provider
value={value}>{children}</AuthContext.Prov
ider>;
}

export function ProtectedRoute({ roles }:
{ roles?: Role[] }) {
    const { isAuthenticated, user } =
useAuth();
}

```

```

    if (!isAuthenticated) {
      return <Navigate to="/login" replace
/>;
    }
    if (roles?.length && user &&
!roles.includes(user.role)) {
      return <Navigate to="/" replace />;
    }
    return <Outlet />;
  }
}

```

Лістинг Б.10 – HTTP-клієнт для взаємодії з REST API
http.ts

```

const API_BASE_URL =
import.meta.env.VITE_API_BASE_URL ??
'http://localhost:8080/api';
const TOKEN_KEY = 'apw-inventory.jwt';
const USER_KEY = 'apw-inventory.user';

export const http = axios.create({
  baseURL: API_BASE_URL,
  headers: { 'Content-Type':
'application/json' },
});

http.interceptors.request.use((config) =>
{
  const token = getToken();
  if (token) {
    config.headers.Authorization = `Bearer
${token}`;
  }
  return config;
});

http.interceptors.response.use(
(response) => response,
(error) => {
  if (error.response?.status === 401) {
    clearSession();
  }
  const message =
error.response?.data?.message ??
error.message ?? 'Помилка запиту';
  return Promise.reject(new
Error(message));
},
);

export async function login(loginValue:
string, password: string) {
  const { data } = await
http.post<AuthResponse>('/auth/login', {
    login: loginValue,
    password,
  });
  return data;
}

```