Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА МОДУЛЯ ПРИЙМАННЯ ТОВАРІВ У WMS-СИСТЕМІ З ВИКОРИСТАННЯМ JAVA ТА PL/SQL
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-2
Костюк М.К.
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)
Керівник доц. Кобилін О. А.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Костюку Микиті Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка модуля приймання товарів у WMS-системі з використанням Java та PL/SQL

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 31 травня 2025р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, результати аналізу ринку WMS-систем.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області у галузі WMS-систем.
2. Розробка модуля приймання товарів WMS-системи.
3. Реалізація модуля приймання товарів WMS-системи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність WMS-систем, огляд існуючих WMS-систем, постановка задачі, проектування та розробка модуля приймання товарів, зображення інтерфейсу.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-08.07.25	
12	Занесення роботи в електронний архів	02.06.25-09.07.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-10.07.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Кобилін О.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 62 с., 6 табл., 22 рис., 31 джерело.

WMS, ОБЛІК ТОВАРІВ, ПРИЙОМ ТОВАРІВ, ОПТИМІЗАЦІЯ, БАЗА ДАНИХ, JAVA, PL/SQL, ORACLE DATABASE,VISUAL STUDIO CODE, ECLIPSE IDE, PL/SQL DEVELOPER.

Об’єктом роботи є дані, що включають в себе вміст накладних та поставок на складі.

Метою роботи є розробка частини програмного забезпечення WMS-системи для фіксації факту надходження товару до складу та подальшого збереження цієї інформації в базі даних.

У кваліфікаційній роботі проведено аналіз існуючих WMS-систем, та розроблено архітектуру модуля приймання товарів, яка включає в себе 3 рівня функціювання системи: клієнтську частину, серверну частину, та базу даних. Система забезпечує прийом, обробку та первинне розміщення на складі для підприємства.

У кваліфікаційній роботі здійснена програмна реалізація модуля приймання товарів, яка дозволяє підприємству швидко та надійно приймати товари, зберігати історію їх переміщень в межах складу, та вести облік продукції для прискорення логістичних процесів.

WMS, GOODS ACCOUNTING, GOODS RECIEVING, OPTIMIZATION, DATABASE, JAVA, PL/SQL, ORACLE DATABASE,VISUAL STUDIO CODE, ECLIPSE IDE, PL/SQL DEVELOPER.

The object of this work is the data that includes the contents of invoices and deliveries in the warehouse.

The aim of the work is to develop a part of the WMS system software for recording the receipt of goods in the warehouse and storing this information in a database.

The qualification work includes an analysis of existing WMS systems and the development of the architecture of the goods receiving module, which consists of three levels of system operation: the client side, the server side, and the database. The system ensures the reception, processing, and initial placement of goods in the warehouse for an enterprise.

As a result of this work, a software implementation of the goods receiving module has been created, enabling the enterprise to quickly and reliably receive goods, maintain a history of their movements within the warehouse, and keep track of inventory to accelerate logistics processes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

WMS – Warehouse Management System (система управління складом)

MDT – Mobile Data Terminal (мобільний термінал для обміну даними)

ERP – Enterprise Resource Planning (система планування ресурсів підприємства)

FIFO – First In First Out (перші надійшли – першими використовуються)

LIFO – Last In First Out (останні надійшли – першими використовуються)

PL/SQL – Procedure Language / Structured Query Language

JDBC – Java Database Connection

ВСТУП

У наш час підприємства, які працюють з великими обсягами товарів, потребують автоматизації логістичних процесів для успішної роботи. До того ж, системи управління складом є дуже важливими у забезпеченні точності, швидкості та надійності операцій з обліку та переміщення продукції. Саме тому зростає потреба у створенні гнучких програмних рішень, які здатні забезпечити ефективне управління товарами на складах.

Зростаючий попит на електронну комерцію, необхідність обробки великих масивів даних у реальному часі та інтеграція з іншими інформаційними системами (наприклад, ERP або CRM) вимагають побудови складських систем, здатних адаптуватися до змін у бізнес-процесах. Водночас високий рівень конкуренції на ринку змушує компанії шукати рішення, які не тільки автоматизують складські операції, але й забезпечують аналітику, контроль та прогнозування логістичних показників.

Основною метою кваліфікаційної роботи є розробка модуля прийому товарів на автоматизованому складі з використанням мови програмування Java для створення клієнтської частини та PL/SQL – для реалізації логіки роботи з базою даних. Такий підхід забезпечує чіткий поділ відповідальностей у системі, підвищує її масштабованість, а також надає високий рівень надійності при збереженні й обробці великих обсягів даних.

Актуальність роботи полягає у тому, що все більше підприємств потребують автоматизації бізнес-процесів задля оптимізації ресурсів, зниження втрат, прискорення логістичних процесів і точності при обробці великих обсягів даних.

Саме тому створення системи обліку товарів є актуальним і важливим завданням, що дозволяє забезпечити підприємствам ефективну та надійну організацію логістичних процесів, що у свою чергу позитивно впливає на розвиток підприємства та його здатність конкурувати на ринку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ У ГАЛУЗІ WMS-СИСТЕМ

1.1 Характеристика систем управління складом

Сучасні підприємства, які працюють з великою кількістю товарів, все частіше використовують автоматизовані цифрові системи для своїх складських приміщень (Warehouse Management System, WMS). Такі системи допомагають оптимізувати управління продукцією на складі, підвищити точність обліку та зберігання товарів, прискорити логістичні процеси та мінімізувати людський фактор при прийнятті рішень на місці. Це в свою чергу спрощує працю робітників та полегшує найм і навчання нових кадрів, завдяки чому підприємство виграє в коштах у довгостроковій перспективі [1].

WMS – це спеціальне програмне забезпечення, яке використовується на сучасних складах. Воно призначене для автоматизації всіх ключових процесів: прийом товарів, їх розміщення, зберігання, переміщення, облік, інвентаризація, підготовка до відправки клієнту тощо. Від простої облікової системи WMS відрізняє можливість керувати не тільки наявністю товарів, але й змінювати пріоритет обробки продукції, визначати оптимальні місця для її розташування та контролювати маршрути переміщення в межах складу та навіть вести статистику процесів [2].

Однією з критично важливих функцій WMS є модуль приймання товарів (емблеми), який виконує такі завдання:

– отримання накладної (отримання інформації про заплановане надходження товарів на склад від ERP-системи підприємства з даними про кількість, найменування товарів та запланований час прибуття);

– порівняння отриманих товарів з даними накладної (співробітник перевіряє, чи збігається кількість та найменування товарів в системі з реальною поставкою, якщо не збігається – він приймає рішення про частинне прийняття товару);

– реєстрація нових товарів у базу даних (система записує нові одиниці у відповідні таблиці в базі даних);

– автоматичне визначення місця розміщення (система визначає оптимальне місце на складі для розташування новоприбулої продукції. Враховується присутність таких самих артикулів на складі та кількість потрібного простору для зберігання. Завдяки цьому досягається оптимальне використання простору на складі та зменшується потрібний час для пошуку вільного місця).

Саме реалізація цієї частини WMS-системи потребує розробки. Модуль приймання товарів повинен мати:

– зручний інтерфейс для реєстрації поставки продукції;

– backend-логіку для підтримування роботи інтерфейсу та проведення операцій з базою даних [3];

– логіку для пошуку місця для зберігання товарів;

– можливість друкувати етикетки з QR-кодом артикулу та даними про місце розміщення.

З цього можна зробити висновок, що WMS-системи є дуже важливою частиною сучасної логістики, завдяки якій досягається можливість швидких та точних поставок в умовах росту торгівлі. А модуль прийому товару є невід’ємною частиною цієї системи.

1.2 Аналіз існуючих рішень у сфері управління складом

Сьогодні на ринку існує велика кількість постачальників готових рішень для управління логістикою. Вони відрізняються складністю, масштабом, гнучкістю, доступними функціями та ціною. Загалом існує три категорії: комерційні промислові продукти, open-source проєкти та внутрішні розробки компаній [4].

Комерційні промислові продукти – це програмне забезпечення від великих IT-компаній, яке поставляється як платний продукт. Як правило, воно має повний набір функцій і частіше за все має інтеграцію з іншими продуктами таких компаній.

Розглянемо рішення, яке допомагає підприємствам ефективно керувати та контролювати складські операції – SAP Extended Warehouse Management (SAP EWM). Це потужна WMS-система з великою кількістю функцій, яка підтримує розбиття складу на зони, на вибір принципи FIFO та LIFO, переміщення товарів в межах складу, використання мобільних пристроїв, кросс-докінг для передачі товару одразу на відвантаження, інтеграцію з іншими продуктами компанії SAP, статистику процесів на складі, тощо. Далі наведені переваги SAP EWM:

– потужний перелік функцій (система підтримує всі можливі логістичні процеси);

– інтеграція з іншими продуктами SAP;

– підтримка автоматизованих складів та роботів (задля більшої оптимізації та прискорення процесів деякі підприємства використовують конвеєрні склади з повною автоматизацією переміщення товарів (рис. 1.1), також для цих цілей використовують роботів (рис. 1.2);

– можливість масштабування для великих підприємств (додавання нових функцій та процесів до програмного забезпечення за потреби клієнта).

До недоліків можна віднести:

– високу вартість ліцензії та впровадження системи на склад (маленький та середній бізнес навряд чи зможе дозволити собі таку інвестицію);

– тривалий час для впровадження (від декількох місяців до року, відсутність об’єкту на такий тривалий час може стати критичним для підприємства);

– надлишковість функцій для маленьких та середніх клієнтів (середній бізнес може потребувати оптимізацію своїх логістичних процесів, але це не

буде вигідною інвестицією через велику кількість невикористовуваних та непотрібних функцій);

– складну архітектуру системи, яка потребує консультації з кваліфікованими спеціалістами SAP (перш за все спеціалісти мають оцінити можливість інтеграції системи до потрібного об’єкту, що також займає додатковий час).

Рисунок 1.1 – Приклад конвеєрного складу

Рисунок 1.2 – Приклад роботів на складі

Oracle Warehouse Management – продукт компанії Oracle що, як і попередньо розглянуте програмне забезпечення, є потужною системою для управління складом, яка має широкий функціонал та підтримує автоматизацію процесів, розбиття складу на зони, принципи FIFO та LIFO, транспортування товарів в межах складу, інтеграцію з мобільними пристроями, кросс-докінг для швидкої відсилки товарів, статистику і аналітику процесів та тісну взаємодію з іншими модулями компанії Oracle.

Переваги Oracle Warehouse Management:

– інтеграція з продуктами Oracle (підходить для підприємств, які вже мають інші продукти компанії Oracle, або бажають зробити повне програмне забезпечення свого бізнесу від цього постачальнику услуг);

– потужні алгоритми розміщення продукції (можливість гнучкого налаштування правил розміщення, наприклад, за терміном придатності, партіям та іншими побажаннями підприємства, що оптимізує логістику за потребами клієнта);

– можливість відстежування термінів придатності та цілих партій (це дуже важливо у підприємствах, працюючих з фармацевтикою, продуктами харчування та іншими чутливими товарами);

– безпека та контроль доступу (логірування всіх дій системи, що дозволяє контролювати всі складські процеси та легко відстежувати помилки, а також рольова модель, яка обмежує можливості звичайних працівників та їх небажаний доступ до важливих даних).

Недоліки цієї системи:

– складність впровадження (це потужна та велика система, яка потребує втручання та консультацію спеціалістів Oracle, а також багато часу на впровадження програмного забезпечення до об’єкту);

– вартість (як і всі продукти Oracle, ліцензія на цю систему буде дорогою, а зокрема ліцензії потребується оплатити також і впровадження системи й кастомізацію, якщо вона потрібна. Тож можливо підприємству не вистачить коштів на таку потужну рішення);

– застарілий інтерфейс (на сьогоднішній день інтерфейс виглядає застарілим та є не user-friendly).

Open-source проєкти – це програмні рішення з відкритим кодом, що поширюються безкоштовно. Зазвичай використовується серед стартапів або компаній з власними командами розробників, які можуть адаптувати програмне забезпечення під власні потреби та побажання.

Серед таких розглянемо OpenWMS.org. Це open-source система для управління складом, яка орієнтована на модульну архітектуру, тобто складається з окремих сервісів та дає можливість їх вмикати та вимикати за потребами підприємства. Більшою мірою вона направлена на індивідуальне налаштування для бізнесу, та не є готовим продуктом. Розглянемо переваги цієї системи:

– open-source система (вона є безкоштовною та має відкритий код, що може бути використано як базу для розробки своєї системи для потреб бізнесу);

– мікросервісна архітектура (це дозволяє легко масштабувати програмне забезпечення та додавати нові функції, а також видаляти непотрібні) [5];

– гнучкість (система підтримує як ручне управління, так і автоматизовані склади).

Недоліки OpenWMS.org:

– відсутність технічної підтримки (відсутні спеціалісти, які будуть допомагати з проблемами у роботі системи);

– відсутність готового рішення (хоча присутній стартовий інтерфейс, але це тільки приклад, а не повноцінна система, тому в будь якому випадку доведеться розробляти додатковий функціонал, з цього виходить, що таке рішення може позволити тільки бізнес, який має своє IT-відділення);

– недостатня документація (доведеться розбиратися в роботі програмного забезпечення шляхом читання коду чи його запуском, що потребує більше часу, ніж готова документація).

Також існують внутрішні розробки бізнесів, для специфічних задач або вимог. В таких випадках підприємства розробляють власні системи , які будуть задовольняти потреби бізнесу та коштувати дешевше за готові продукти від техногігантів. Зазвичай таке програмне забезпечення не публікується в інтернеті, тому присутні труднощі з пошуком таких систем для їх подальшого аналізу.

Таким чином після аналізу ми з'ясували, що хоча готові рішення від крупних компаній мають потужний функціонал та можливості інтеграції з іншими модулями цих компаній, вони є дуже дорогими, та потребують багато часу на консультацію зі спеціалістами та подальше впровадження. Також система Oracle Warehouse Management має застарілий інтерфейс, а рішення SAP EWM має велику кількість надлишкових функцій, за які не всі бізнеси хочуть додатково доплачувати. Open-source проєкти навпаки же є безкоштовними та можуть гнучко налаштовувати потрібний функціонал, але вони не мають технічної підтримки та мають проблеми з доступною документацією, що робить додаткову розробку та адаптацію під потреби бізнесу тривалим та складним завданням [6].

Саме тому в кваліфікаційній роботі розробляється додатковий модуль прийому товарів, а саме емблем, що вирішує проблеми коштів, додаткового негнучкого функціоналу, а також відсутність готового рішення зі сторони проєктів з відкритим кодом, для підприємств, які потребують подібний процес на своєму складі. Обрана тема дозволяє адаптувати програмне забезпечення до роботи конкретного бізнес процесу [7].

1.3 Вибір технологій

Для реалізації власного модуля приймання та обліку емблем важливо обрати технології, які забезпечують швидкість та надійність при роботі з базами даних, бо це основна логіка такого програмного забезпечення. Також

варто звернути увагу на масштабованість та зміни бізнес логіки у майбутньому, бо вимоги підприємств завжди змінюються з часом. З оглядом на ці потреби було обрано такі технології як Java для розробки клієнтської частини та PL/SQL для реалізації логіки роботи з базою даних [8].

Перед вибором технологій було сформовано основні вимоги для програмного забезпечення:

– зручний інтерфейс для працівника (введення або сканування даних, можливість огляду даних та інтуїтивно зрозумілий інтерфейс);

– автоматичне розміщення товарів (при обробці новоприбулої продукції важливо швидке та точне розміщення продукції на складі, що буде уникати втручання людини у прийняття рішення);

– збереження у базу даних (товар має бути збережений в базі даних у відповідні таблиці);

– логування та формування звітності;

– гнучкість при зміні потреб підприємства без переробки всієї системи;

– підтримка мобільних терміналів (рис. 1.3).

Рисунок 1.3 – Приклад MDT (Mobile Data Terminal)

Java є актуальним вибором для створення цього додатку, оскільки має такі переваги:

– кросплатформеність (має можливість запуску програмних застосунків на різних операційних системах, що допомагає з втіленням вимог на використання MDT);

– широка підтримка бібліотек та фреймворків (для будування клієнтської частини буде використовуватися Java Swing);

– інтеграція з обладнанням (можливість використання сканерів для вводення даних та спеціальних принтерів для друкування етикеток);

– масштабованість (бізнеси мають властивість зростати та з часом потребують новий функціонал, тому розроблюване програмне забезпечення має бути готовим для оновлення та росту функцйі) [9].

PL/SQL також є оптимальним вибором для створення додатку через такі переваги:

– швидкість при обробці запитів (виконання функцій та запитів у базу даних має бути швидким для оптимальної роботи програмного забезпечення через велику кількість даних);

– зменшення навантаження на клієнтську частину (Java має багато переваг, але задля цих переваг вона може програвати у швидкості, саме тому цей недолік нівелюється);

– підтримка перевірки цілісності даних (точність та цілісність даних дуже важлива для коректного функціювання підриємства) [10].

На основі даних переваг можна зробити висновок, за використання зв’язки Java + PL/SQL дає нам надійність, масштабованість, швидкість та кроссплатформеність, що є головними вимогами для системи управління складом. Ці технології забезпечать зручний графічний інтерфейс та повний контроль над інформацією, яка зрештою зберігається у базі даних [11].

1.4 Огляд архітектури WMS-систем

Розглянемо типову архітектуру WMS-систем. Зазвичай система управління складом складається з трьох рівнів, де кожен рівень відповідає за свій набір функцій.

Презентаційний рівень відповідає за інтерфейс та взаємодію з користувачем. Це може бути десктопний додаток або мобільний застосунок, який пов’язує працівника та програмне забезпечення. Презентаційний рівень виводить інформацію про поставки, результат дії, помилки, тощо для працівника. Також він відповідає за введення та редагування даних [12].

Рівень логіки реалізує бізнес логіку системи та відповідає за обробку даних. Тут знаходяться правила прийому товарів та визначення місця їх розміщення, а також перевірка унікальності товарів та перевірка даних введених працівником.

Останнім є рівень зберігання даних. Він відповідає за перевірку цілісності інформації, отриманої в накладній, та її зберігання у базу даних. Також цей рівень відповідає за генерацію статистики та логування операцій.

Ці рівні поєднані між собою за допомогою внутрішніх інтерфейсів, через які вони надсилають інформацію. Невеликий приклад роботи при прийманні товару:

Крок 1. Працівник натискає кнопку бронювання товару за допомогою Java-інтерфейсу.

Крок 2. Система передає інформацію до бізнес-логіки, зробленої за допомогою PL/SQL процедур, яка перевіряє дані, визначає місце зберігання та записує інформацію до бази даних.

Крок 3. У базі даних створюються потрібні записи та активується логування.

Також варто розібрати принцип модульності сучасних WMS-систем. Системи управління складом складаються з окремих частин, які відповідають за свій власний процес та можуть незалежно один від одного розгортатися,

розроблятися та оновлюватися, що є дуже важливим для адаптації під змінні умови, а також забезпечує безпеку системи, бо помилки окремих модулів не впливають на роботу інших [13]. Система складається з таких модулів:

– модуль приймання товарів (первинна реєстрація новоприбулої продукції та розподілення на оптимальні місця для зберігання);

– модуль переміщення (зміна місця зберігання товарів на складі);

– модуль інвентаризації (контроль та перевірка товарів, які знаходяться на складі);

– модуль комісіонування (підготовка товарів до відправки кінцевому клієнту, розподілення товарів по окремих коробках та підготовлення заказів);

– модуль відправки (контроль готових заказів, друкування чеків та розподілення заказів до служби доставки);

– модуль статистики та аналітики (статистика роботи складу для керівництва підприємства).

Можна зробити висновок, що розробка WMS-системи є складним завданням, тому потребує розділення на модулі для окремої розробки потрібного функціоналу. Це дозволяє створювати масштабовані й легко підтримуванні рішення та гнучко поділяти завдання між командою розробників. Саме через те, що модулі можуть розроблятися незалежно один від одного, завданням кваліфікаційної роботи є розробка модуля прийому товарів (емблем) [14].

1.5 Постановка задачі

На основі попереднього аналізу існуючих систем управління складом, а також архітектури таких WMS-рішень, сформульовано основні задання, які мають бути вирішені в рамках розробки програмного забезпечення.

Об’єктом роботи є дані, що включають в себе вміст накладних та поставок на складі.

Метою роботи є розробка модуля приймання товарів, який забезпечує зручний графічний інтерфейс, первинний облік новоприбулої продукції, автоматизований пошук оптимального місця та зберігання інформації у базі даних [15].

Основними завданнями, які потрібно вирішити, є наступні:

– розробка зручного інтерфейсу: створення форми демонстрації існуючих поставок, можливість пошуку поставки за номером, можливість редагування кількості товару в разі відмінностей між реальністю та системою, інформування користувача про успішний або помилковий результат роботи;

– розробка бізнес-логіки: перевірка правильності введених працівником даних, створення нових записів у базі даних при успішній операції, автоматичний пошук оптимального місця розміщення, друк етикетки з даними про товар та обране місце;

– розробка зв’язку з базою даних: актуалізація таблиць та створення потрібних View для спрощення реалізації користувацького інтерфейсу, розробка процедур для перевірки цілісності інформації та її збереження у базі даних.

Далі необхідно оцінити якість розробленого рішення на основі тестування функціювання системи [16].

2 РОЗРОБКА МОДУЛЯ ПРИЙМАННЯ ТОВАРІВ WMS-СИСТЕМИ

2.1 Визначення вимог

Чітке формулювання вимог будує основу для подальшого проєктування логіки, інтерфейсу та бази даних модуля. Вимоги поділяються на функціональні, які забезпечують бізнес логіку, конкретні можливості системи, та функції, які потребує замовник, та нефункціональні, які визначають критерії безпеки, швидкості, масштабованості та надійності програмного забезпечення, що потребує система для коректного функціонування та виконання своїх завдань [17].

Успішне проєктування системи потребує чуткого формулювання функціональних та нефункціональних вимог. Завдяки цьому етапу забезпечується відповідність розробки очікуванням замовника, а також гарантується технічна обґрунтованість обраних технологій та бізнес-логіки системи [18].

Функціональні вимоги:

– отримання накладних від програмного забезпечення клієнта (накладні відправляються ERP-системою замовника, а накладні для емблем мають оброблятися за особливими правилами);

– демонструвати перелік поставок та додати можливість пошуку за номером заказу (у начальній формі має бути виведений перелік поставок , у формі має бути місце для введення номеру заказу, та обмеження виводу за номером заказу);

– можливість зміни кількості (в разі відмін між системою та реальністю користувач має мати можливість записати актуальну кількість товарів та додати до бази даних в такому вигляді);

– автоматичне збереження у базі даних (дані мають бути автоматично актуалізовані, та створені нові записи у базі даних у всіх потрібних таблицях);

– визначення місця збереження (система приймає рішення про місце розташування товару на складі);

– логування процесів (усі дії мають бути збережені у базі даних з часом, ідентифікаційним номером користувача та результатом дії);

– друкування етикеток (мають буди надруковані етикетки, які містять QR-код для ідентифікації товару, місце розміщення та короткий опис продукту);

– підтримка MDT-пристроїв (для завершення процесу прийому товару та його розміщення потрібно програмне забезпечення, яке буде працювати на MDT-пристроях);

– обробка помилок (при введені некоректних даних система має надавити відповідні повідомлення з помилками).

Нефункціональні вимоги:

– зручність інтерфейсу (графічний інтерфейс має бути інтуїтивно зрозумілим);

– надійність (система має зберігати цілісність даних у разі збоїв та помилок вводу);

– швидкодія (швидкість обробки однієї операції не має перевищувати 1-2 секунди);

– масштабованість (модуль має бути гнучким для змін або адаптування процесу та його підтримки);

– сумісність (програмне забезпечення має коректно працювати з обраними технологіями для розробки).

Технології для розробки:

– база даних: Oracle із використанням PL/SQL;

– мови розробки: Java та PL/SQL;

– доступ до бази даних: Java Database Connectivity;

– інтерфейс користувача: Java Swing;

– середовища розробки: Eclipse IDE(Java), Oracle PL/SQL Developer.

2.2 Проєктування системи

Проєктування модуля приймання товарів для WMS-системи потребує архітектуру, яка забезпечить ефективну обробку та реєстрацію товарних одиниць, визначення місця зберігання, перевірку та збереження інформації у базі даних. Найважливішими критеріями під час проєктування були надійність, масштабованість, відповідність архітектурі WMS-системи та простота використання [19].

У проєкті використовується трьохрівнева архітектура «презентаційний рівень – рівень логіки – рівень зберігання даних (клієнт – сервер – база даних)», де кожен рівень має свою сферу відповідальності [20].

Презентаційний рівень розроблений за допомогою Java та відповідає за взаємодію з користувачем. Клієнтська частина відповідальна за графічний інтерфейс(розробляється за допомогою Java Swing), введення та валідацію даних, повідомлення про помилки та успішне виконання процесів [21].

Рівень логіки сформований за допомогою PL/SQL процедур, які реалізують основну бізнес-логіку системи, шукають оптимальне місце для товару, обробляють вхідну інформацію , перевіряють її цілісність та додають у базу даних, та Java-класів, які викликають ці PL/SQL процедури [22].

Рівень зберігання даних – база даних Oracle, яка зберігає всю інформацію щодо товарів, користувачів, які їх обробляли, місця розташування продукції, історію операцій тощо. Основна логіка обробки транзакцій в базі даних реалізується за допомогою PL/SQL [23].

Послідовність процесу виглядає приблизно таким чином:

– користувач відкриває форму вибору поставки;

– користувач обирає поставку;

– відкривається форма приймання товару з даними з поставки;

– користувач перевіряє дані на правильність;

– якщо дані правильні, користувач натискає на кнопку, яка запускає серверну логіку;

– валідація даних у Java та виклик PL/SQL процедур;

– процедура ще раз перевіряє дані на правильність;

– процедура створює запис у таблиці одиниць зберігання;

– процедура шукає місце для розташування та створює запис у таблиці переміщень зі статусом «00» (відкритий);

– процедура друкує етикетку з даними про товар та обране місце розміщення;

– у разі успіху користувач отримує в інтерфейсі повідомлення про успіх, або повідомлення про помилку у випадку некоректної роботи;

– також для завершення переміщення користувач має використовувати MDT-пристрій та відсканувати QR-код, роздрукований на етикетці та QR-код, роздрукований на місці розташування.

Тобто ключовими компонентами є:

– модуль введення даних;

– модуль валідації даних;

– модуль зв’язку Java з базою даних та PL/SQL;

– модуль бізнес логіки (PL/SQL);

– MDT-процес;

– база даних.

Проєктування системи в багаторівневій архітектурі дозволяє розділити відповідальність між рівнями, спростити масштабування в майбутньому і найголовніше – інтегрувати модуль в повноцінну WMS-систему [24].

2.3 Опис основних масок та компонентів системи

У контексті кваліфікаційної роботи маска – це блок програмного забезпечення, який служить окремим інтерфейсом для взаємодії з користувачем. Вона містить потрібний функціонал та контент, й змінюється системою в залежності від дій користувача. Кожна окрема маска може

включати в себе елементи виводу, такі як текст, елементи взаємодії, наприклад, кнопки, та форми вводу даних [25].

В кваліфікаційній роботі інтерфейс реалізовано за допомогою Java та бібліотеки Java Swing. Вікно модуля приймання товарів включає в себе такі маски:

– маска вибору поставки (огляд таблиці поставок та таблиці позицій поставок, які мають залежність «одна поставка – багато позицій», форма вводу тексту для користувача, для пошуку за номером заказу, кнопка вибору поставки, яка змінює маску);

– маска огляду та запису поставки (огляд позицій поставки та перевірка коректності даних користувачем. Інтерфейс показує позиції по одній, з всіма даними, також присутні 4 кнопки: «Назад» – для переходу до маски вибору поставки, «Пропустити» – для переходу до наступної позиції, «Помилка кількості» – для переходу до маски редагування, «Записати» – для виклику бізнес-логіки);

– маска помилкової кількості (огляд позиції поставки, яка була відкрита перед натисканням кнопки «Помилкова кількість», та додатково два елемента вводу для заповнення актуальних даних користувачем (кількість та коментар), також дві кнопки «Назад» для повернення на попередню маску та «ОК» для виклику бізнес-логіки);

– маска для MDT-пристрою (має два стани, в першому виводить повідомлення про потрібність відсканувати товар та має форму вводу, друга повідомляє місце, яке має бути відсканово, та також форма вводу).

Кожна з цих масок обладнана спеціальними можливостями та окремими функціями, маски будуть змінюватися автоматично системою за потребою бізнес-логіки та надавати потрібну інформацію користувачу. У сукупності ці маски створюють повноцінний модуль, який повністю виконує визначені у попередньому розділі вимоги.

2.4 Огляд та проєктування бази даних

Бази даних – це організовані колекції даних, які дозволяють зберігати, керувати та отримувати інформацію. У WMS-системах бази даних є ключовим компонентом, оскільки основна логіка такого програмного забезпечення пов'язана з зберіганням та маніпуляцією даних й найважливіше у цих процесах – надійність та коректність даних, а також швидкий доступ до них за допомогою запитів, індексів та View. Бази даних поділяються на реляційні (SQL) та нереляційні (NoSQL) [26].

Реляційна база даних – база даних, де інформація організована у вигляді таблиць з чітко визначеною структурою та зв'язками між ними. У таких базах таблиці складаються з рядків, де кожен рядок є окремим записом, та стовпців, де кожний стовпець є окремим полем з визначеним типом даних. Кожна таблиця має дані, які заздалегідь відповідають визначеному формату [27].

Нереляційна база даних – база даних, що зберігає інформацію без строго визначених структур і зв'язків між елементами. На відміну від традиційних таблиць, такі бази містять різноманітні типи даних, наприклад документи, відео чи зображення, тощо. У NoSQL відсутня підтримка стандартних SQL-запитів, що використовуються у реляційних базах даних [28].

Для розробки модуля прийому товарів WMS-системи було обрано Oracle Database, що дозволяє надійно зберігати всю необхідну інформацію та забезпечувати цілісність даних і коректну роботу програмного забезпечення.

Oracle Database – це потужна об'єктно-реляційна система управління базами даних (СУБД) з вбудованою мовою PL/SQL. Вона підтримує транзакційність. При цьому процес поділяється на дії, що разом являють собою одну транзакцію. Транзакція буде успішною тільки за умови виконання всіх дій, якщо виконається тільки половина дій, то дані будуть повернені у стан до змін. Це забезпечує цілісність даних і підтримує велику кількість одночасних підключень, що є дуже важливим для WMS-системи, через велику кількість процесів та робочих місць, які функціонують одночасно. Також

Oracle Database забезпечує швидке збереження даних, та швидкий доступ до них завдяки кешуванню SQL-запитів, які часто викликаються. Вона ідеально підходить для систем з великими обсягами структурованих даних. Нижче наведено деякі важливі переваги, що виділяють Oracle Database:

– надійність і стабільність, СУБД працює навіть при високих навантаженнях, має механізми захисту від збоїв та підтримує резервне копіювання та відновлення, що є критичною перевагою для підприємства, де кожен запис в базі даних є реальним товаром;

– СУБД сертифікована для використання у високозахищених середовищах, а також підтримує рольову модель доступу, що є обов'язковим для запобігання зміни або видалення даних працівниками підприємства;

– Oracle Database має оптимізований механізм обробки запитів та підтримує індекси таблиць, паралельну обробку та кешування даних, а також Materialized View для швидкого доступу до інформації в базі даних;

– СУБД підтримує не тільки зберігання у таблицях, а ще й конвертування у такі типи даних як JSON та XML, що допомагає пересилати інформацію між системами, наприклад для обміну даних з ERP-системою замовника;

– підтримка Oracle, оскільки СУБД розвивається вже протягом 40 років і досі отримує технічну підтримку від Oracle, саме тому вона має обширну документацію та регулярні оновлення.

Після ознайомлення з Oracle Database стає зрозуміло, що це ідеальний варіант для розробки WMS-системи. Модуль прийняття товарів загалом має п'ять таблиць – «avise» (поставки), «avispos» (позиції поставок), «plaetze» (місця), «quanten» (одиниці зберігання), «artikel»(артикули), «bewegungen» (дії товарів). Кожна з них представляє різні типи даних та використовується для своїх спеціалізованих цілей.

Таблиця «avise» відповідає за збереження інформації про поставки (накладні). Вона містить дані про кожну поставку на склад, такі як ідентифікатор замовника (id_klient), номер поставки (nr_avis), номер

розділеного заказу (nr_avis_ta), статус обробки (stat), пріоритет (prio), вид поставки (art_avis), дата прибуття (datum_anlief_soll), ідентифікатор поставника (id_lieferant). Ідентифікатор клієнта, номер поставки та номер розділеного заказу формують собою первинний ключ. Додатково кожна поставка має поля статусу для відстеження прогресу обробки, пріоритету для правильного порядку обробки, дати прибуття та ідентифікатору поставника для кращої ідентифікації поставки повноти даних для підприємства. Ця інформація використовується в масці обзору поставок, зберігання та аналізу даних, а також для подальшої обробки. Детальну схему наведено у таблиці 2.1.

Таблиця 2.1 – Схема таблиці «avise» (поставки)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор замовника	id_klient	VARCHAR2(12)	Існують системи, які використовуються більше ніж одним складом
Номер поставки	nr_avis	VARCHAR2(12)	
Номер розділеного заказу	nr_avis_ta	NUMBER2(4)	Закази можуть бути розділені на декілька частин
Статус обробки	stat	VARCHAR2(2)	Число від 00 до 90, яке демонструє ступінь обробки заказу
Пріоритет обробки	prio	NUMBER2(2)	
Вид поставки	art_avis	VARCHAR2(12)	Повідомляє про вид поставки, для емблем це «EMBLEMWE»
Дата прибуття	datum_anlief_soll	DATE	Дата запланованого прибуття
Ідентифікатор поставника	id_lieferant	VARCHAR2(12)	

Таблиця «avispos» відповідає за збереження інформації про вміст поставок, тобто найменування та кількість товарів, які мають прийти на склад. Вона містить дані про кожну товарну позицію поставок, такі як ідентифікатор замовника (id_klient), номер поставки (nr_avis), номер розділеного заказу (nr_avis_ta), взяті з попередньої таблиці, а також номер позиції (nr_avis_pos), статус обробки (stat), ідентифікатор власнику товару (id_klient_artikel), ідентифікатор товару (id_artikel), кількість запланованих товарів (mng_soll), кількість оброблених товарів (mng_ist), номер заказу (nr_bestell). Ідентифікатор клієнта, номер поставки та номер розділеного заказу взяті з таблиці поставок, а також номер позиції формують первинний ключ. Статус, кількість запланованих товарів та кількість оброблених товарів дозволяють відстежувати прогрес, а також записувати актуальні дані у випадку відмін між системою та реальністю. Ідентифікатор товару є номером відповідної продукції. Ця інформація використовується для ідентифікації товарів, які включає в себе поставка, а також для обробки продукції з подальшим зберіганням на складі. Детальну схему наведено у таблиці 2.2.

Таблиця «plaetze» відповідає за збереження інформації про місця на складі. Вона містить дані про кожне окреме місце, такі як склад (lager), розділ складу (ort), область (bereich), місце (platz), статус зайнятості (stat_belegt), тип місця (typ_platz), номер комісіювання (nr_kom), ідентифікатор власнику товару (id_artikel), ідентифікатор товару (id_artikel). Список місць допомагає відстежувати наявні місця на складі. Ідентифікатор товару та ідентифікатор власнику зв’язують з конкретним товаром, статус зайнятості показує його поточний стан, вільно воно чи ні. Номер комісіювання являє собою номер, який означає порядковий номер обробки цього місця й визначається за бізнес-логікою замовника, бо є місця, доступ до яких складніший й саме там будуть розташовані товари, які використовуються не так часто. Також місце зберігає товар, який там розташований. Детальну схему наведено у таблиці 2.3.

Таблиця 2.2 – Схема таблиці «avispos» (позиції поставки)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор замовника	id_klient	VARCHAR2(12)	Існують системи, які використовуються більше ніж одним складом
Номер поставки	nr_avis	VARCHAR2(12)	
Номер розділеного заказу	nr_avis_ta	NUMBER2(4)	Закази можуть бути розділені на декілька частин
Номер позиції	nr_avis_pos	NUMBER(6)	
Ідентифікатор власнику артикулу	id_klient_artikel	VARCHAR2(12)	
Ідентифікатор артикулу	id_artikel	VARCHAR2(40)	
Статус обробки	stat	VARCHAR2(2)	
Кількість запланованих товарів	mng_soll	NUMBER(18,9)	Запланована кількість товарів за системою
Кількість оброблених товарів	mng_ist	NUMBER(18,9)	Кількість оброблених товарів, які вже пройшли прийняття товарів
Номер заказу	nr_bestell	NUMBER(10)	Номер заказу з ERP-системи замовника

Таблиця «quanten» відповідає за збереження одиниць зберігання. У випадку емблем це всі товари, які знаходяться в межах одного місця, є одним артикулом та мають однакову дату обробки. Таблиця містить дані про всі одиниці зберігання на складі, такі як ідентифікатор кванту (id_quant), ідентифікатор власника (id_klient), ідентифікатор товара (id_artikel), склад (lager), розділ складу (ort), область (bereich), місце (platz), кількість доступних товарів (mng_frei), кількість товарів у русі (mng_zulag), критерій для розділення (trenn), номер останньої дії (nr_bew).

Таблиця 2.3 – Схема таблиці «plaetze» (місць)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Склад	lager	VARCHAR2(3)	
Розділ складу	ort	VARCHAR2(10)	
Область	bereich	VARCHAR2(10)	
Місце	platz	VARCHAR2(40)	
Статус зайнятості	stat_belegt	VARCHAR2(2)	«00» якщо вільне, «90» в іншому випадку
Вид місця	typ_platz	VARCHAR2(6)	Різні види зберігання, для емблем це конверти
Номер комісіювання	nr_kom	VARCHAR2(12)	
Ідентифікатор власнику	id_klient	VARCHAR2(12)	
Ідентифікатор артикулу	id_artikel	VARCHAR2(40)	

Склад, розділ складу, область та місце зв’язують одиницю зберігання з конкретним місцем, ідентифікатор товару та ідентифікатор власника зв’язують з конкретним товаром, кількість доступних товарів показує кількість товарів, готових для наступних процесів, кількість товарів у русі показує кількість товарів, які знаходяться у процесі приймання й очікують підтвердження зберігання, номер останньої дії зв’язує з конкретною дією. Ця інформація використовується для огляду всієї продукції на складі та управління товарами. Детальну схему показано у таблиці 2.4.

Таблиця «artikel» відповідає за зберігання даних про існуючі артикули в асортименті замовника програмного забезпечення. Вона містить дані про кожен артикул, який може зберігатися на складі, такі як ідентифікатор власника (id_klient) та ідентифікатор артикулу (id_artikel), що являє собою первинний ключ, а також короткий опис (bez_kurz) і позначка про власне ім’я (knz_eigenname). Ця інформація використовується для огляду та коректування асортименту товарів, які продає замовник системи. Детальну схему наведено у таблиці 2.5.

Таблиця 2.4 – Схема таблиці «quanten» (Одиниць зберігання)

Зміст поля	Назва поля	Тип даних	Нотатки
Ідентифікатор одиниці збереження	id_quant	NUMBER(9)	
Ідентифікатор власнику	id_klient	VARCHAR2(12)	
Ідентифікатор артикулу	id_artikel	VARCHAR2(40)	
Склад	lager	VARCHAR2(3)	
Розділ складу	ort	VARCHAR2(10)	
Область	bereich	VARCHAR2(10)	
Місце	platz	VARCHAR2(40)	
Кількість доступних товарів	mng_frei	NUMBER(18,9)	Кількість товарів доступних для інших операцій
Кількість товарів у русі	mng_zugang	NUMBER(18,9)	Кількість товарів, які знаходяться у процесі приймання та очікують розміщення на складі
Номер дії	nr_bew	NUMBER(9)	Номер останньої відкритої дії для одиниці розміщення

Таблиця «bewegungen» відповідає за збереження даних про дії з квантами. Вона містить дані про кожну окрему операцію зміни місця квантами, такі як номер переміщення (nr_bew), що є первинним ключом таблиці, вид переміщення (art_bew), ідентифікатор власнику (id_klient) та ідентифікатор артикулу (id_artikel), які вказують на конкретний артикул, статус (stat), кількість (mng), ідентифікатор кванту з (quant_von), який указує на конкретний квант, ідентифікатор кванту на(quant_an),який указує на конкретний квант, з якого складу (lager_von), з якого розділу складу (ort_von), з якої області

(bereich_von), з якого місця (platz_von), що вказує на конкретне місце, з якого почався рух, на який склад (lager_an), на який розділ складу (ort_an), на яку область (bereich_an), на яке місце (platz_an), вказує на конкретне місце, на якому має закінчитися або закінчився рух. Ця інформація використовується для огляду та контролю переміщень продукції в межах складу. Детальну схему наведено у таблиці 2.6.

Отже, було описано структуру шести різних таблиць: «avise», «avispos», «quanten», «plaetze», «artikel», «bewegungen», кожна з яких невід'ємною для функціювання модуля прийому товарів.

Кожна таблиця використовується для різних частин бізнес-логіки, для коректного зберігання різних видів даних, які отримуються під час процесу приймання, та для забезпечення зв'язку між даними та функціоналом застосунку. Також первинні ключі таблиць, які також є зовнішніми ключами для інших таблиць, забезпечують зв'язок між записами у межах бази даних, що дозволяє ефективно зберігати, читати, аналізувати та відображати потрібну інформацію [28].

Таблиця 2.5 – Схема таблиці «artikel» (артикулів)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор власнику	id_klient	VARCHAR2(12)	
Ідентифікатор артикулу	id_artikel	VARCHAR2(40)	
Короткий опис	bez_kurz	VARCHAR2(40)	
Позначка про власне ім'я	knz_eigenname	VARCHAR2(1)	Демонструє чи має емблема власне ім'я «1», якщо так, «0», якщо ні

Таблиця 2.6 – Схема таблиці «bewegungen» (переміщень)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Номер переміщення	nr_bew	NUMBER(9)	
Вид переміщення	art_bew	VARCHAR2(4)	Двоетапне переміщення чи одноетапне
Статус	stat	VARCHAR(2)	Відкрите (00) чи закрите (90) переміщення
Ідентифікатор власнику	id_klient	VARCHAR2(12)	
Ідентифікатор артикулу	id_artikel	VARCHAR2(40)	
Кількість	mng	NUMBER(18,9)	Кількість товарів у русі
Квант з якого переміщують	quant_von	NUMBER(9)	
Квант в який переміщують	quant_an	NUMBER(9)	
З якого складу	lager_von	VARCHAR2(3)	
З якого розділу	Ort_von	VARCHAR2(10)	
З якої області	Bereich_von	VARCHAR2(10)	
З якого місця	Platz_von	VARCHAR2(40)	Місце, з якого переміщують товар
На який склад	Lager_an	VARCHAR2(3)	
На який розділ	Ort_an	VARCHAR2(10)	
На яку область	Bereich_an	VARCHAR2(10)	
На яке місце	Platz_an	VARCHAR2(40)	Місце, на яке переміщують товар

Діаграма класів для бази даних представлена на рисунку 2.1.

Ця структуризація даних допомагає не тільки ефективно зберігати інформацію, але й підвищує зручність обробки даних під час роботи програмного забезпечення, а також зручний доступ до потрібних даних у випадку помилок, оскільки кожна таблиця відповідає за свою унікальну область, забезпечуючи відповідність даних потребам замовника [29].

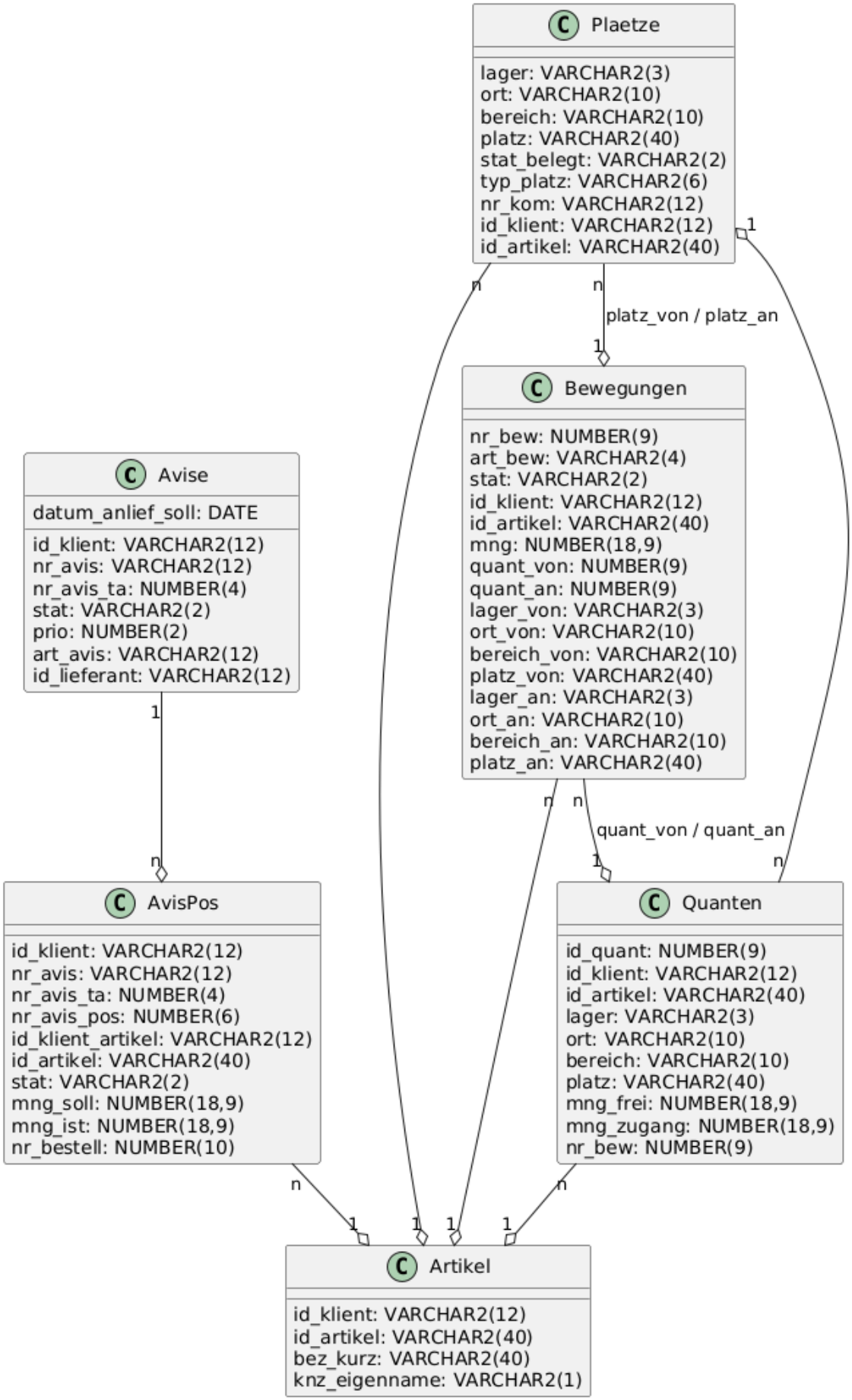

Рисунок 2.1 – Діаграма класів для бази даних

2.5 Обґрунтування вибору середовища розробки

Для реалізації модуля приймання товарів було обрано три середовища розробки, кожне з яких відповідає своїй частині проєкту: Eclipse IDE для роботи з Java, PL/SQL Developer для роботи з базою даних, SQL-запитів, Visual Studio Code для роботи з PL/SQL. Далі наводяться основні переваги кожного середовища, які роблять їх ідеальним для розробки програмного забезпечення.

Перевагою Eclipse IDE є безкоштовність та відкритий код. Eclipse – відкрите програмне забезпечення з великою спільнотою розробників та великою кількістю безкоштовних плагінів, а також воно не потребує додаткових затрат на покупку ліцензії. Це також зручне середовище для командної роботи: WMS-система складається з багатьох модулів, та зазвичай розробляється в команді, тому контроль версій (SVN чи Git) є дуже важливим для такого проєкту. Середовище розробки має зручні інструменти для налагодження, що є дуже важливим для розробки та пошуку помилок.

PL/SQL Developer має інтуїтивно зрозумілий інтерфейс, можливість роботи з декількома вікнами, форматування, підсвітка синтаксису та автодоповнення коду та назв таблиць, або їх рядків. Забезпечується повна сумісність з Oracle Database та глибока підтримка PL/SQL. Висока продуктивність у вигляді швидкого виконання запитів, скриптів та коду є критично важливими для роботи з великими обсягами коду. Крім того, PL/SQL Developer має потужний функціонал у формі можливості конвертування результатів запитів у XML чи JSON файли, дозволяє перегляд структури об’єктів бази даних (таблиць чи процедур, тощо), є можливість генерації SQL-скриптів для експорту чи імпорту таблиць, а також підтримка роботи з ролями, сесіями та правами користувачів.

Visual Studio Code безкоштовна, є велика кількість розширень, створених спільнотою для підсвітки синтаксису, зміни цвіту вікон для різних проєктів, підтримки різних мов програмування (також PL/SQL), можлива інтеграція з системами контролю версій та базами даних. VS Code є дуже швидким

середовищем розробки та споживає менше ресурсів за конкурентів. Її можна налаштувати повністю під свої потреби, можна змінити майже все – теми оформлення, комбінації клавіш, цвіт вікон, стиль синтаксису та інше.

Таким чином вибір Eclipse IDE, PL/SQL Developer, Visual Studio Code як середовищ розробки є обґрунтованим рішенням через збіг переваг програмного забезпеченнями зі запланованими вимогами та методами їх втілення. Комбіноване використання середовищ, які є найкращим вибором для своїх задач, дозволяє підвищити продуктивність та ефективність на всіх етапах розробки системи [30].

3 РЕАЛІЗАЦІЯ МОДУЛЯ ПРИЙМАННЯ ТОВАРІВ WMS-СИСТЕМИ

3.1 Розробка клієнтської частини системи

Основною метою програмного забезпечення є оптимізація процесу приймання товарів для працівників складу, для ефективної роботи з системою вона потребує інтуїтивно зрозумілий та простий інтерфейс, саме тому розробка має починатися з клієнтської частини. Для створення клієнтської частини використовується мова Java та бібліотека Java Swing, яка дозволяє будувати віконний інтерфейс [31].

Клієнтська частина модуля приймання товарів включає в себе такі екрани:

– форма вибору поставки: екран, де оператор обирає поставку для прийняття на склад;

– форма приймання товарів: екран, який показує інформацію про позицію поставки, оператор перевіряє дані записані в системі з реальністю, та підтверджує прийняття у разі збігу даних;

– форма неправильної кількості: екран, в якому оператор заповнює правильну кількість одиниць товару, які надійшли на склад, та коментар;

– MDT-діалог: для завершення процесу використовується мобільний пристрій, для якого потрібен окремий інтерфейс.

Інтерфейс було реалізовано за допомогою бібліотеки Java Swing, яка надає зручні засоби для побудови графічного інтерфейсу а також входить до складу стандартної бібліотеки Java і дозволяє створювати кроссплатформені програмні застосунки без додаткових залежностей. Зовнішній вид був побудований у вигляді вікна (JFrame) та декількох панелей (JPanel), кожна з яких відповідає за свій функціонал, та навігацію між цих панелей за допомогою кнопок (JButton).

Кожна форма є свої індивідуальні задачі, та має показувати інформацію згідно з цими задачами. Таким чином потрібно визначити потрібні дані, графічні компоненти та їх розташування. Розберемо форму вибору поставки(рис. 3.1):

– виведення таблиці поставок (у WMS-системі присутні інші поставки окрім емблем, тому для обмеження даних поставок емблем було створено View, що шукає тільки активні поставки емблем);

– виведення таблиці позицій поставок (позиції поставок мають виводитися в залежності від обраної поставки для того щоб огляд даних був зрозумілий оператору);

– поле вводу (оператор має можливість пошуку поставок за номером заказу, який він вводить в це поле);

– повідомлення для оператора (нагорі повідомлення релевантне для роботи з формою, знизу повідомлення про останній виконаний процес або остання помилка);

– кнопка вибору поставки (відкриває наступну форму, в яку передає дані обраної курсором поставки).

Рисунок 3.1 – Схема інтерфейсу форми вибору поставки

Детальніше про форму приймання товарів (рис. 3.2):

– виведення таблиці позицій поставок по одному запису;

– повідомлення для оператора (підказка щодо наступного кроку для подальшої роботи з системою);

– кнопка «Abbrechen» для виходу на основну форму (система повертається на екран вибору поставки);

– кнопка «Skip» для переходу до наступного запису (після натискання кнопки показується наступна позиція, якщо наступної не існує, то показується знову перший запис);

– кнопка «Fehlmenge» для переходу до форми неправильної кількості;

– кнопка «Buchen» для запису поставки на склад (кнопка запускає бізнесу логіку програмного забезпечення, що обирає місце розташування ,записує товари у базу даних та друкує етикетку з інформацією та QR-кодом).

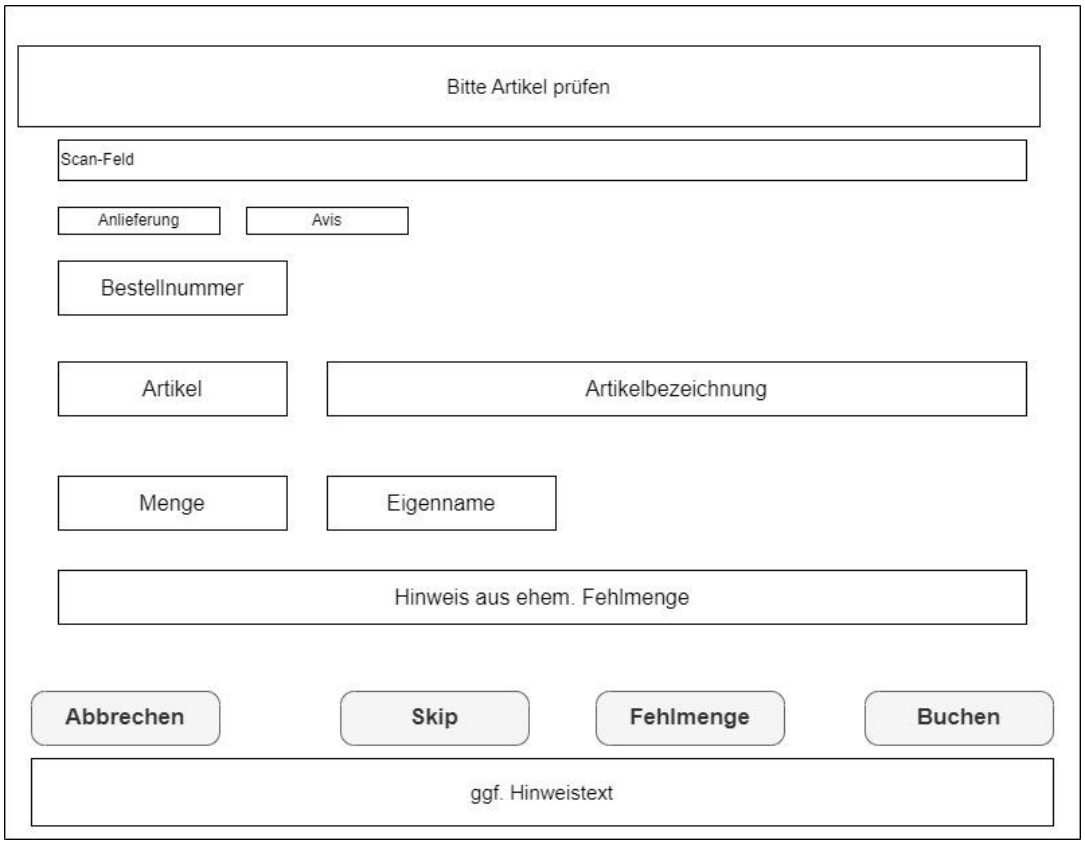

Рисунок 3.2 – Схема інтерфейсу форми вибору поставки

Детальніше про форму неправильної кількості (рис. 3.3):

– виведення одної конкретної позиції поставки;

– повідомлення для оператора (підказка щодо роботи з програмним забезпеченням);

– поля вводу (ператор має ввести актуальну кількість товарів та коментар, якщо такий є);

– кнопка «Abbrechen» для виходу до форми приймання товарів (у випадку, якщо оператор знайде емблеми, яких не вистачає, або вирішить зробити це пізніше має бути кнопка для повернення);

– кнопка «OK» для запису поставки з введеною кількість на склад (виконується бізнес логіка системи, як і у випадку з кнопкою «Buchen»).

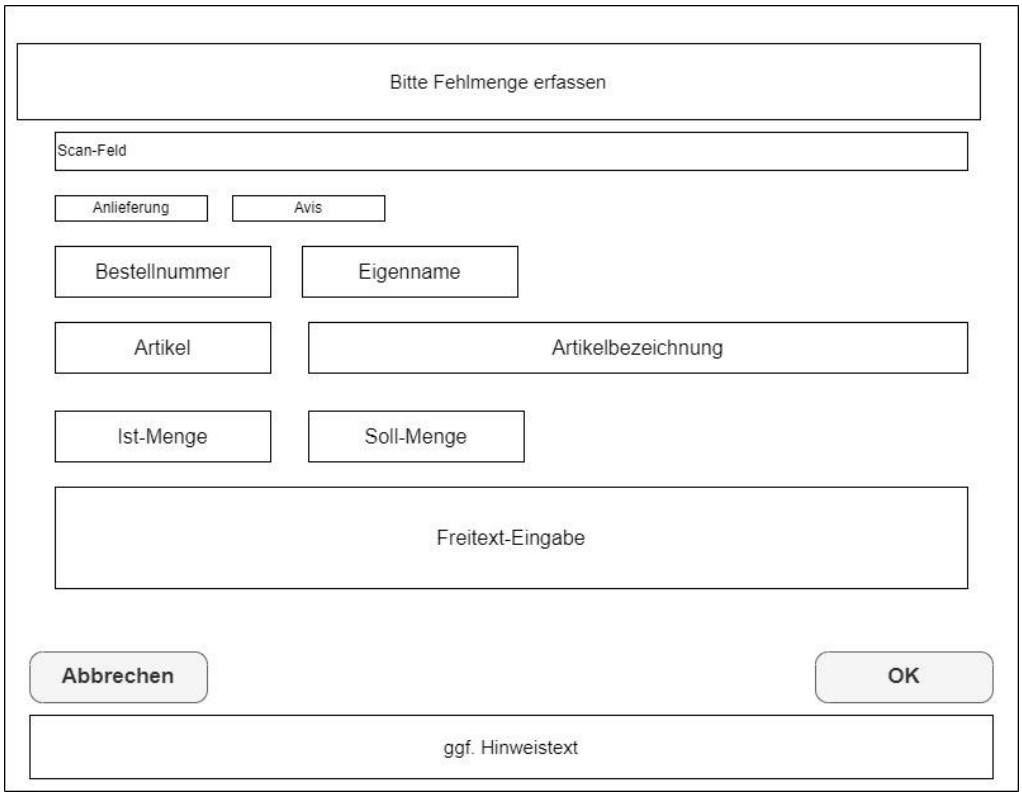

Рисунок 3.3 – Схема інтерфейсу форми неправильної кількості

Детальніше про MDT-діалог (рис. 3.4 та рис. 3.5):

– повідомлення для оператора (підказка щодо потрібної, для роботи програмного забезпечення, дії);

– повідомлення про успішне завершення попередньої дії (оператору потрібно підтвердження виконання процесу);

– кнопка «OK» (для підтвердження введення даних та запуск бізнес логіки);

– кнопка «Zurück» (для повернення до попереднього екрану або головного меню);

– поля вводу даних (обов’язкова валідація вводу, бо кількість має бути числом);

– кнопка «Information» для перегляду документації;

– кнопка «Hilfe» для перегляду можливого рішення;

– інформація про місце та артикул.

Рисунок 3.4 – Інтерфейс MDT-діалогу (екран сканування товару)

Рисунок 3.5 – Інтерфейс MDT-діалогу (екран сканування місця)

Так як всі форми (окрім MDT-діалогу) мають знаходитися в межах одного вікна, було використано менеджер компоновки CardLayout, який зберігає в собі декілька панелей та показує лише одну з них, завдяки чому можна налаштувати вивід потрібної для системи інформації в залежності від умов. У кваліфікаційній роботі умовами зміни виведеної панелі є натискання навігаційних кнопок.

Для коректного виведення даних с таблиць було створено Views один для таблиці «avise» та два для таблиці «avispos» (один для огляду позицій поставок у головній формі, а другий, який групує однакові артикули разом, для форми збереження на склад), що забезпечує виведення тільки потрібних стовпців, а

також тільки потрібних записів (поставки емблем) за допомогою обмежень за видом поставки.

Таким чином, під час розробки клієнтської частини було сплановано та реалізовано зручний та функціональний інтерфейс, який забезпечує інтуїтивну взаємодію з користувачем та ефективну роботу з даними. Також обрана технологія дозволяє забезпечити масштабованість клієнтської частини, що робить систему дуже гнучкою до оновлень та змін в майбутньому. Отриманий результат повністю відповідає вимогам до модуля.

3.2 Розробка серверної частини системи

У кваліфікаційній роботі використовується реляційна база даних Oracle Database, що забезпечує надійне, швидке та ефективне управління великими обсягами структурованих даних, а також підтримку PL/SQL. Для створення серверної частини програмного забезпечення використовується PL/SQL – процедурне розширення мови SQL, що дозволяє виконувати бізнес-логіку безпосередньо у базі даних, що використовується для тісної та повністю контрольованої роботи з даними. Також використовується бібліотека JDBC(Java Database Connectivity) для зв’язку клієнтської частини із базою даних та серверною частиною шляхом виклику PL/SQL-процедур.

Через те, що серверна частина прив’язана до кнопок з користувацького інтерфейсу, вона була розроблена у другу чергу. Кнопки «Buchen» та «OK» викликають PL/SQL процедуру, що містить в собі всі бізнес-логіку та відповідає за процеси пошуку місця, занесення товару у базу даних складу, створення та початок переміщення продукції, а також друкування емблеми.

Для реалізації зв’язку з базою даних був створений окремий клас, що представляє собою PL/SQL-процедуру та містить в собі змінні, які потрібні для виклику процедури, а також строку з викликом збереженої процедури. Приклад класу наведено у лістингу 3.1.

Лістинг 3.1 Поля класу що представляє собою збережену процедуру:

```
private String storedProcedureCall = "{call PKG_ES_WE_RUESTEN_INF.RUEST_EMBLEM (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}";
// StoredProcedure Parameter
private String pLager;
private String pIdKlient;
private String pNrAvis;
private Double pNrAvisTa;
private String pIdArtikel;
private Double pMng;
private String pOrt;
private String pBereich;
private String pPlatz;
private Double pNrBestell;
private String pHinweis;
private String pOpid;
```

При натисканні кнопки «Buchen» викликається функція, у якій створюється об’єкт класу з лістингу 3.1 та заповнюється даними, які пройшли валідацію і згодом будуть передані у PL/SQL процедуру, що виконує бізнес-логіку. Приклад функції наведено у лістингу 3.2.

Лістинг 3.2 Реалізація функції виклику PL/SQL-процедури:

```
public void buttonBuchen() {
    PkgEsWeRuestenInfRuestEmblem pkg = new PkgEsWeRuestenInfRuestEmblem();
    pkg.setPLager(currentObject.get(0).getLagerReadOnly());
    pkg.setPIdKlient(currentObject.get(0).getIdKlientReadOnly());
    pkg.setPNrAvis(currentObject.get(0).getNrAvisReadOnly());
```

```
        pkg.setPNrAvisTa(currentObject.get(0).getNrAvisTaReadOnly());
        pkg.setPOrt(currentObject.get(0).getOrt());
        pkg.setPBereich(currentObject.get(0).getBereich());
        pkg.setPPlatz(currentObject.get(0).getPlatz());
        pkg.setPIdArtikel(currentObject.get(0).getIdArtikel());
        pkg.setPMng(currentObject.get(0).getMenge());
        pkg.setPOpid(ModelGlobals.getIdAnwender());
        pkg.execute();
        data.setMeldung(NLS.translate(
WordTrans.EmblemenWeMeldungBuchErfolg));
        updateFields();
      }
```

PL/SQL процедура є ядром модуля приймання товарів, бо виконує основну бізнес логіку:

Крок 1. Програмне забезпечення обирає місце розташування для товару. У випадку, якщо такий артикул вже розташований на складі, продукція буде направлена у це місце, якщо артикул не розташований, обирається наступне вільне місце за порядком вибору місць. Приклад логіки наведений у лістингу 3.3.

Крок 2. Зробити запис у таблицю квантів, що зберігає всі товари, що знаходяться на складі. Квант це одиниця зберігання, що характеризує собою всі товари в межах одного місця, та у випадку емблем дати прибуття, для зручного зберігання та ідентифікації товарів на складі(на одному місця мають зберігатися декілька квантів, які прибули в різний час).

Крок 3. Створити запис двоетапного переміщення у таблицю переміщень для подальшого зберігання. Приклад логіки наведений у лістингу 3.4.

Крок 4. Змінити статус місця на 90 (зайнято). Приклад логіки наведений у лістингу 3.4.

Крок 5. Роздрукувати етикетку з інформацією про місце розташування та артикул.

Лістинг 3.3 Реалізація пошуку місця:

```
l_platz :=null;
        for c_quant in (select *
                        from quanten q
                        where q.id_klient = p_id_klient
                        and q.id_artikel = p_id_artikel
                        and q.lager = p_lager
                        and q.ort = 'EMB'
                        and q.bereich = 'SCHRANK')
        loop
          l_platz.lager   := c_quant.lager;
          l_platz.ort     := c_quant.ort;
          l_platz.bereich := c_quant.bereich;
          l_platz.platz   := c_quant.platz;
          exit;
        end loop;

        if not zz_plaetze.is_pk_ok(l_platz)
        then
          for c_platz in (select *
                          from plaetze p
                          where p.lager = p_lager
                          and p.ort = 'EMB'
                          and p.bereich = 'SCHRANK'
                          and p.stat_belegt = '00'
                          order by p.nr_kom)
          loop
```

```
      l_platz := c_platz;
      exit;
    end loop;
  end if;

  if not zz_plaetze.is_pk_ok(l_platz)
  then
    pkg_error.raise(uname,'WEI-10054','@1');
  end if;
```

Лістинг 3.4 Реалізація створення переміщення та оновлення місця:

```
  l_bew             := null;
  l_bew.lager       := p_lager;
  l_bew.art_bew     :='UM2Q';
  l_bew.fkt_bumo    := pkg_konst.get('GCR_BUMOFK_INIT');
  l_bew.id_quant_von := l_quant.id_quant;
  l_bew.mng         := l_mng_ein;
  l_bew.ort_an      := l_platz.ort;
  l_bew.bereich_an  := l_platz.bereich;
  l_bew.platz_an    := l_platz.platz;
  l_bew.text_lvs    := uname;
  l_bew.opid_neu    := p_opid;
  l_bew.opid_aen    := p_opid;
  pkg_bumod.buche(l_bew);
  for c_platz in (select *
          from plaetze p
          where p.lager = p_lager
          and p.ort = l_platz.ort
          and p.bereich = l_platz.bereich
```

```
        and p.platz = l_platz.platz)
loop
  c_platz.stat_belegt := pkg_konst.get('GCS_PLBELE_BEL');
  zz_plaetze.upd(c_platz);
exit;
end loop;
```

Також серверна частина включає в себе розробку програмного забезпечення для MDT-пристрою. Логіка програмного забезпечення базується на скінченному автоматі. Схему логіки наведено на рисунку 3.6.

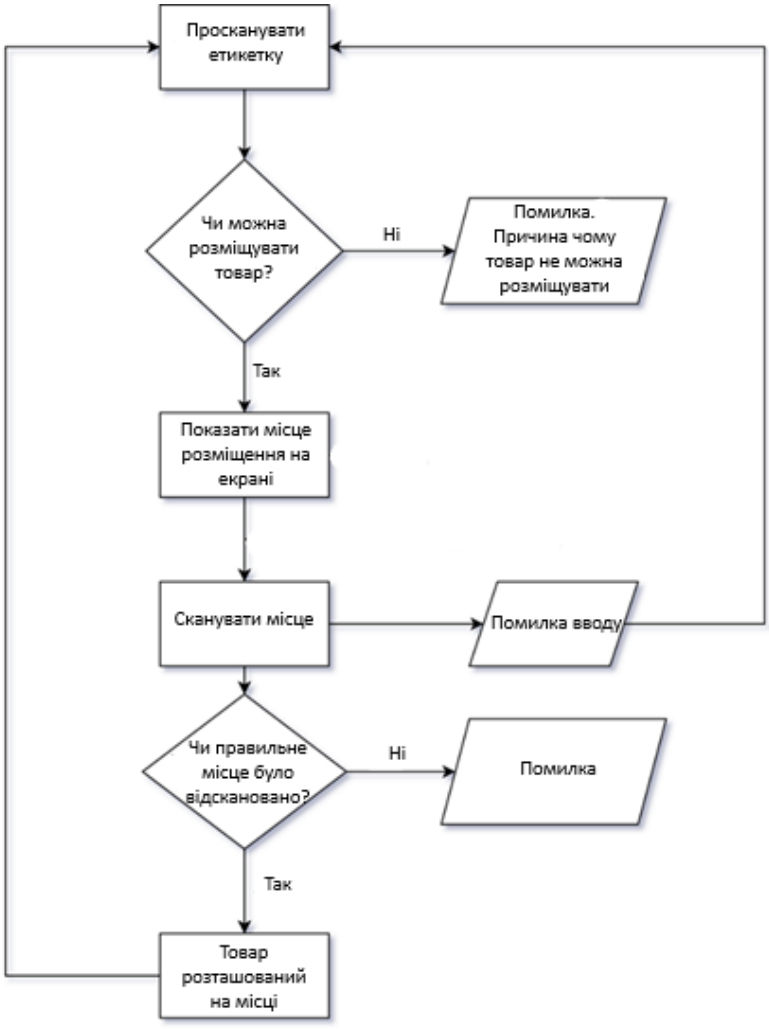

Рисунок 3.6 – Схема логіки скінченного автомату для MDT-діалогу

Програмне забезпечення для MDT-пристрою складається з двох функцій, кожна з яких характеризує один з етапів процесу: сканування етикетки та сканування місця.

Детальніше про процес сканування етикетки:

Крок 1. Читання строки з поля вводу та розбиття на номер артикулу та дату прибуття.

Крок 2. Перевірка вводу. Виклик помилки, якщо поле вводу є пустим.

Крок 3. Перевірка вводу. Виклик помилки, якщо такого артикулу не існує, або це не артикул типу емблем.

Крок 4. Пошук кванту за номером артикулу та датою прибуття.

Крок 5. Виведення місця розташування на екран.

Процес сканування місця:

Крок 1. Читання строки з поля вводу.

Крок 2. Перевірка вводу. Виклик помилки, якщо поле вводу є пустим.

Крок 3. Перевірка вводу. Виклик помилки, якщо такого місця не існує.

Крок 4. Перевірка чи існує відкрите переміщення для цього товару

Крок 5. Закриття переміщення та запис товару до конкретного місця.

Крок 6. Вивід повідомлення про успіх.

Таким чином була розроблена серверна логіка модуля приймання товарів з дотримання принципу розділення відповідальностей. Основна бізнес-логіка у вигляді збережених процедур PL/SQL, та викликається за допомогою Java-інтерфейсу. Такий підхід забезпечує централізовану перевірку та валідацію даних, безпечне зберігання інформації у базі даних, можливість масштабування та надійність системи, а також високу продуктивність при роботі програмного забезпечення. Система продумана та реалізована таким чином, гарантувати стійкість до помилок та зручність підтримки та подальшої розробки програмного коду.

3.3 Тестування функціональності системи

Після реалізації модуля приймання товарів було проведено тестування коректності його роботи, стабільності при обробці вхідних даних та стійкості до помилок. Основна мета тестування – переконатися, що система відповідає функціональним та нефункціональним вимогам, визначеним на етапі проєктування. Тестування було проведено шляхом прийняття емблеми, а також шляхом введення некоректних даних для перевірки роботи системи при помилковому вводі.

Перед початком тестування було створено декілька тестових місць складу та системно відправлено декілька поставок, а також створено тестові артикули емблем.

Після створення тестових даних було відкрито модуль прийняття товарів на його головній формі, та введено номер заказу для перевірки можливості пошуку за ним (рис. 3.7).

Рисунок 3.7 – Результат роботи головної форми та пошуку за номером заказу

Для продовження роботи потрібно натиснути кнопку «Auswahl» для вибору поставки та переходу до наступного етапу функціювання системи. Після натискання кнопки програмне забезпечення показує наступну форму, яка демонструє інформацію про позиції поставки (рис. 3.8).

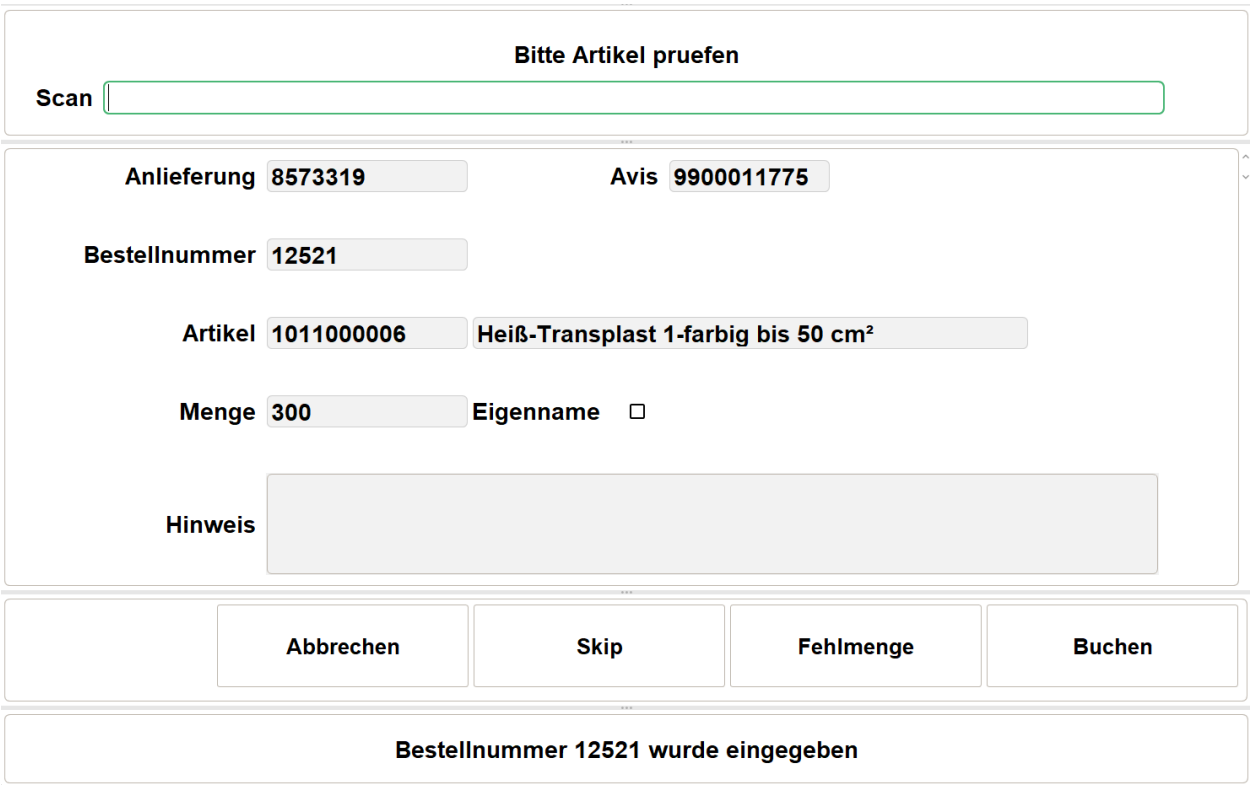

Рисунок 3.8 – Приклад роботи форми прийняття товарів

Далі для наступного кроку роботи програмного забезпечення потрібно натиснути кнопку «Buchen», що запускає бізнес логіку. Система знаходить оптимальне місце розташування, створює квант (рис. 3.9), який в цьому випадку складається з 300 емблем з артикулом 1011000006, створює переміщення прийому товару на склад, переміщення розблокування кванту для операцій на складі, і двохетапне переміщення для зміни місця кванту на місце розташування (рис. 3.10), а також створює етикетку та друкує її (під час розробки використовується сервер, на який відсилається код етикетки).

```
select q.id_quant ,q.id_artikel,q.lager,q.ort,q.bereich,q.platz, q.mng_frei, q.mng_zugang, q.nr_bew
from quanten q
order by q.time_neu desc
```

	ID_QUANT	ID_ARTIKEL	LAGER	ORT	BEREICH	PLATZ	MNG_FREI	MNG_ZUGANG	NR_BEW
1	800006404	1011000006	LSG	EMB	SCHRANK	01-01-A-04	0,000000000	300,000000000	15009

Рисунок 3.9 – Квант створений у базі даних

```
select b.nr_bew
     ,b.art_bew
     ,b.stat
     ,b.id_klient
     ,b.id_artikel
     ,b.mng
     ,b.lager_platz_von
     ,b.ort_von
     ,b.bereich_von
     ,b.platz_von
     ,b.lager_platz_an
     ,b.ort_an
     ,b.bereich_an
     ,b.platz_an
from bewegungen b
order by b.nr_bew desc
```

	NR_BEW	ART_BEW	STAT	ID_KLIENT	ID_ARTIKEL	MNG	LAGER_PLATZ_VON	ORT_VON	BEREICH_VON	PLATZ_VON	LAGER_PLATZ_AN	ORT_AN	BEREICH_AN	PLATZ_AN
1	15009	UM2Q	10	ES	1011000006	300,000000000	LSG	EMB	WE	AP-01-00	LSG	EMB	SCHRANK	01-01-A-04
2	15008	WEES	90	ES	1011000006	300,000000000	LSG	EMB	WE	AP-01-00	LSG	EMB	WE	AP-01-00
3	15007	WEIN	90	ES	1011000006	300,000000000	LSG	STD	QE	001	LSG	EMB	WE	AP-01-00

Рисунок 3.10 – Переміщення, створене у базі даних

Після натискання кнопки було створено записи у базі даних, а також відправлено код етикетки, який після реалізації системи буде замінено на принтер (рис. 3.11). Приклад коду етикетки наведено у лістингу 3.5

Лістинг 3.5 Код етикетки написаний на Zebra Programming Language:

```
^XA^MMT,Y^XZ
^XA
^FT575,720
^BY4,,10^BQN,,9,,^FDMA,1011000009-13.05.2025^FS
^FT25,540
^A0N,40,40 ^FDArtikel:^FS
^FT175,540
^A0N,40,40 ^FD1011000009^FS
^FT25,590
^A0N,25,25 ^FDSiebdruck farbige Shirts 1-farbig Brust^FS
^FT25,640
```

^A0N,40,40 ^FDWE-Datum:^FS

^FT245,640

^A0N,40,40 ^FD13.05.2025^FS

^FT25,690

^A0N,40,40 ^FDPlatz:^FS

^FT145,690

^A0N,40,40 ^FD01-01-A-02^FS

^FT464,634

^A0N,150,150 ^FD^FS

^XZ

^XA

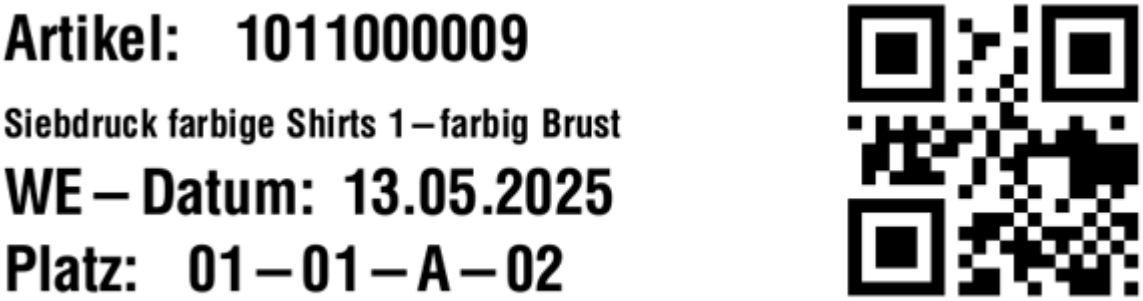

Рисунок 3.11 – Результат виконання коду з лістингу 3.5, вигляд етикетки

Для завершення процесу прийняття товарів потрібно підтвердити розміщення товару на відповідне місце за допомогою MDT-пристрою. Замість пристрою у тесті було використано сайт, який симулює роботу цього пристрою.

Для початку роботи процесу розміщення було відкрито головну форму (рис. 3.12), що відповідає за сканування етикетки, а саме QR-коду. На стадії проектування було розроблено шаблон для вертикального формату, проте вимоги змінилися й на складі буде використовуватися горизонтальний формат.

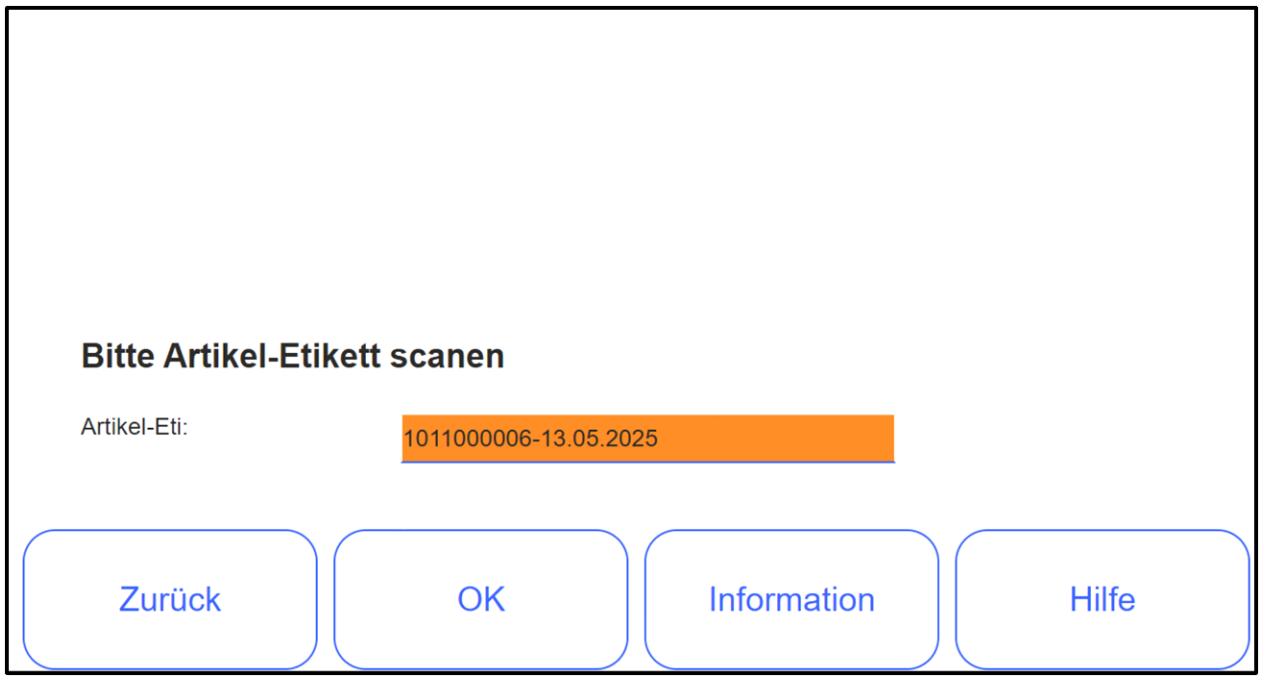

Рисунок 3.12 – Результат роботи першої форми MDT-процесу та сканування етикетки

Після натискання кнопки «OK» було відкрито другу маску процесу, що відповідає за сканування місця. На екран виведено місце розташування а також номер та опис артикулу. Для завершення процесу потребується відсканувати QR-код місця розташування (рис. 3.13).

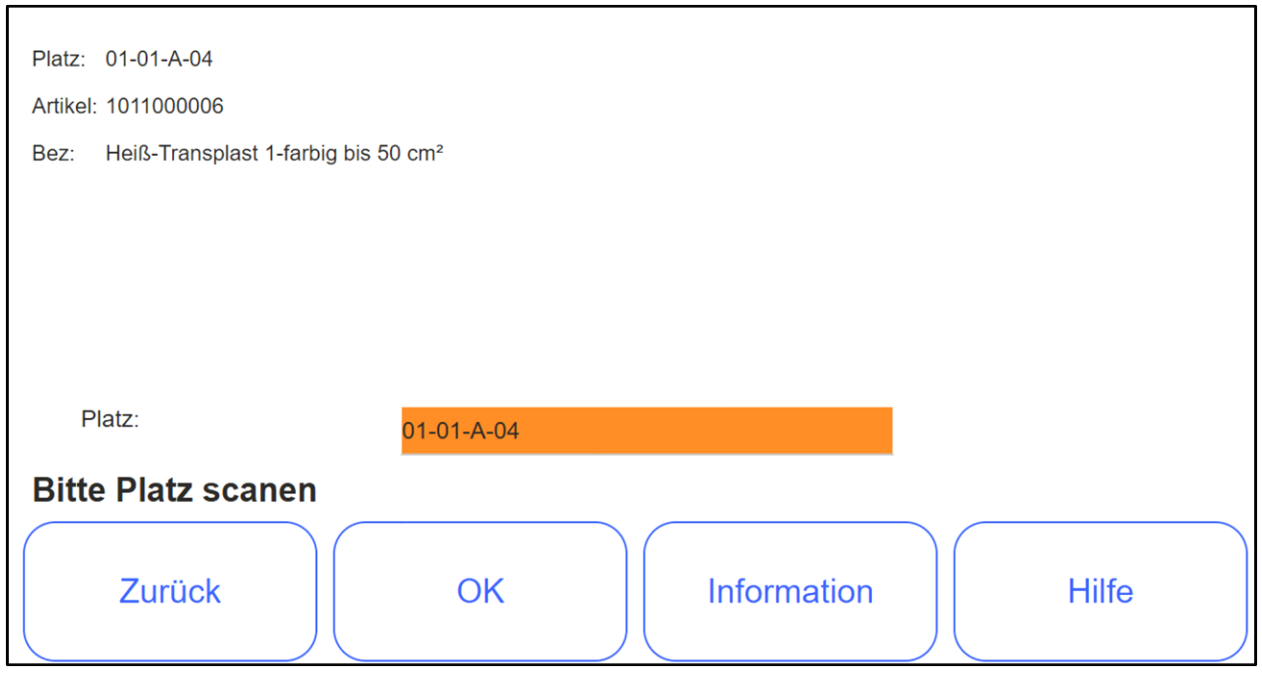

Рисунок 3.13 – Результат роботи другої форми MDT-процесу

Після успішного завершення процесу на екран було виведено повідомлення (рис. 3.14), а також переміщення було закрито, що відображається у базі даних як статус 90 (рис. 3.15).

Рисунок 3.14 – Основна маска з повідомленням про успіх

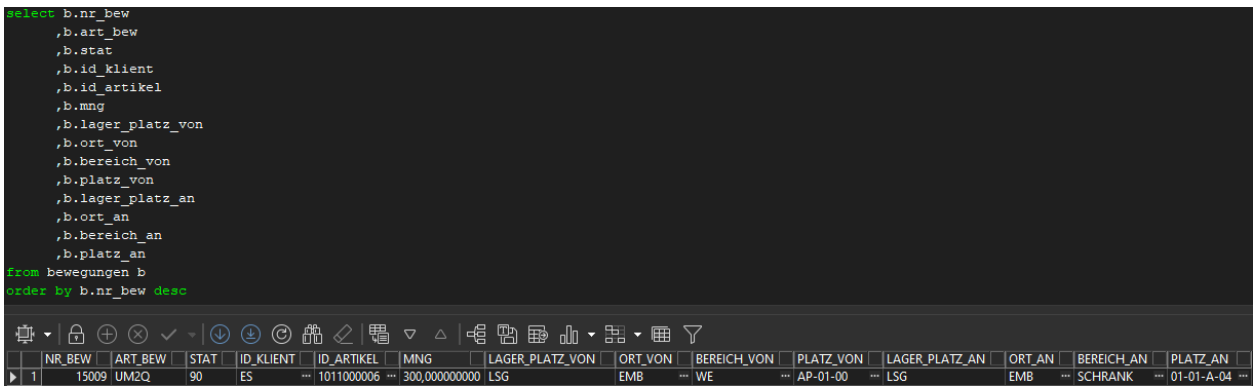

Рисунок 3.15 – Завершене переміщення, збережене у базі даних

Також було проведено тестування з помилковими даними. Через те що артикули можуть включати в себе також букви, валідація номеру артикулу має мало сенсу, тому система перевіряє чи існує введений артикул у базі даних,

якщо ні, то система викликає помилку про те, що був відсканований не артикул емблеми (рис. 3.16).

Рисунок 3.16 – Помилка вводу, артикул с введеним ідентифікатором(asd) відсутній

При скануванні артикулу емблем, який вже був прийнятий на склад система викликає помилку про відсутність відкритих переміщень (рис. 3.17).

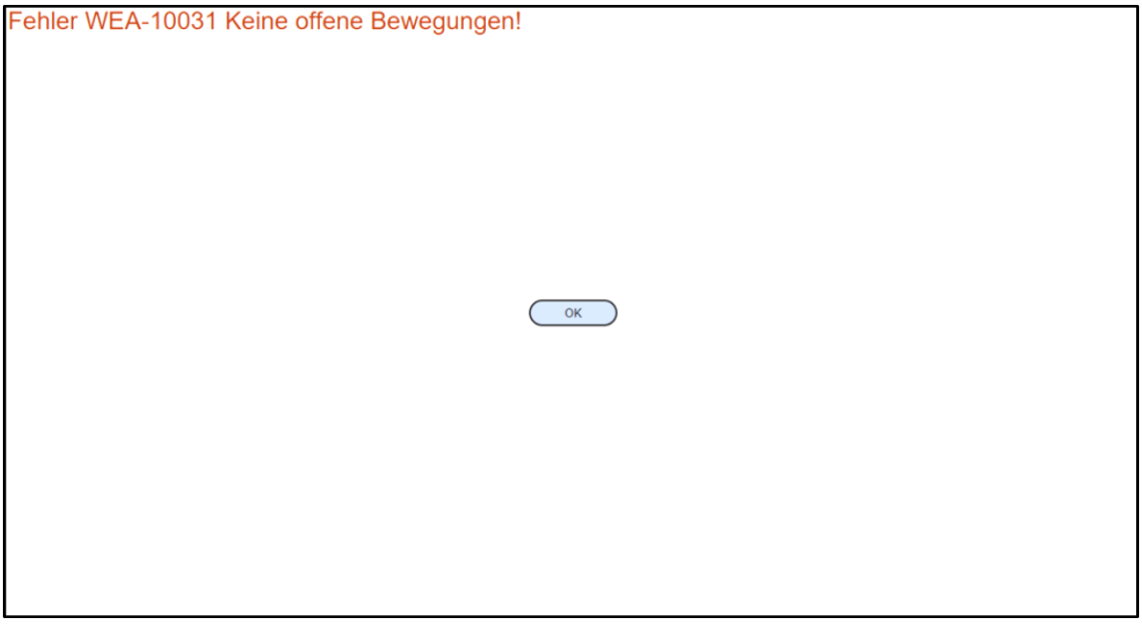

Рисунок 3.17 – Помилка вводу, відсутнє відкрите переміщення

Так само при скануванні неправильного місця система викликає помилку (рис. 3.18), що запобігає помилковому розміщенню товару та порушенню бізнес-логіки.

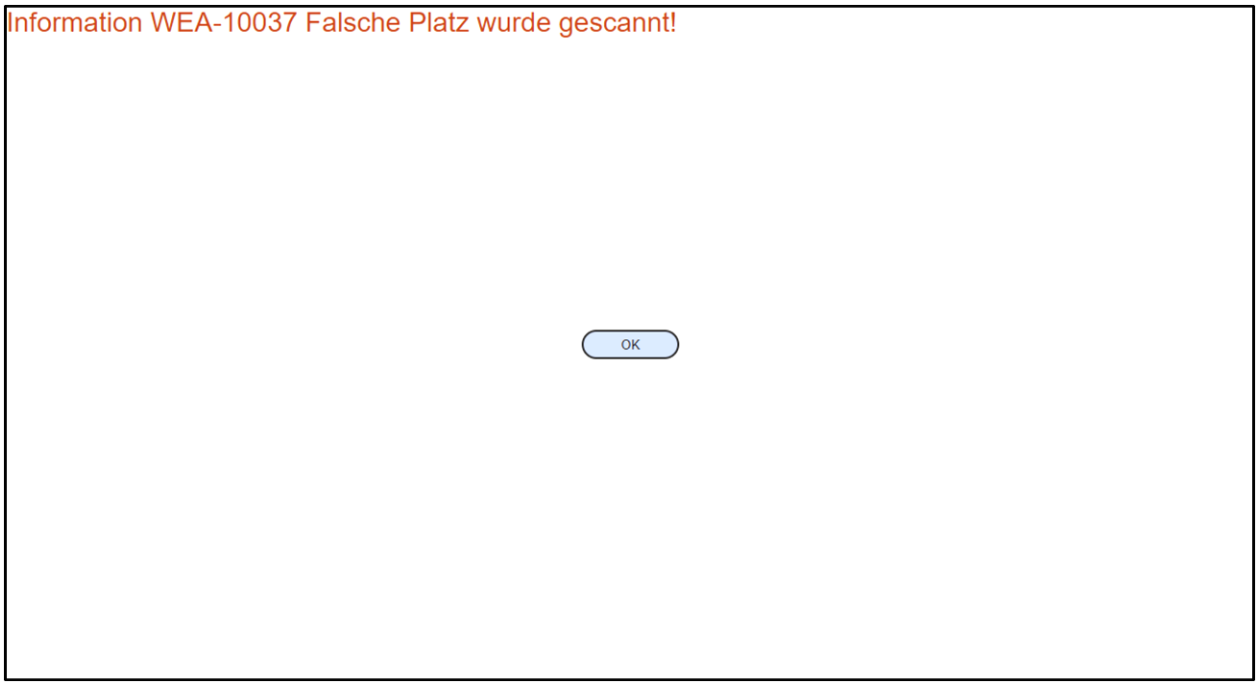

Рисунок 3.18 Помилка, відсканово неправильне місце

ВИСНОВКИ

У кваліфікаційній роботі було розроблено модуль прийняття товарів (емблем) — один із ключових компонентів системи управління складом. Цей модуль є частиною більш широкого комплексу автоматизації логістичних процесів, зосередженого на підвищенні ефективності операцій приймання продукції.

Основною метою роботи було створення зручного та ефективного інструменту для користувачів, який дозволяє оптимізувати роботу складу, зокрема процес приймання товарів, за допомогою автоматизованої системи, що виконує облік та ухвалення рішень замість реального працівника. Такий підхід дозволяє зменшити кількість помилок, пришвидшити обробку вантажів і знизити витрати на людські ресурси.

Під час реалізації було використано перевірені, надійні та ефективні технології й бібліотеки. Клієнтська частина програмного забезпечення створена з використанням мови програмування Java та бібліотеки Java Swing, що забезпечує платформонезалежний графічний інтерфейс користувача. Серверна логіка реалізована за допомогою мови PL/SQL, що дає змогу безпечно та ефективно працювати з базою даних, а також використовувався інтерфейс JDBC для взаємодії між клієнтською та серверною частинами.

База даних системи була розроблена відповідно до вимог нормалізації та з урахуванням майбутнього масштабування, використовуючи Oracle Database як надійне та продуктивне середовище зберігання. Вона включає всі необхідні таблиці, зв’язки, тригери та процедури для коректного функціонування модуля.

Робочу версію модуля було успішно розгорнуто і він працює відповідно до всіх вимог та є стійким до помилок у користуванні.

Результатом роботи є функціональний інструмент, який використовується для приймання та обліку товарів на складі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57).

2. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

3. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

4. Gorokhovatskyi, V., & Tvoroshenko, I. (2023). Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S»* (pp. 25-27).

5. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2023). Statistical data analysis models for determining the relevance of structural image descriptions. IEEE Access, 11, 126938-126949.

6. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

7. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

8. Ibrahim, D. Y., Gorokhovatskyi, V., Tvoroshenko, I., & Mujahed, A. D. (2022). Classification of Images Based on a System of Hierarchical Features.

9. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International*

Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.

10. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool to Study the Structure of Blood Components.

11. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In Eleventh International Conference on Machine Vision (ICMV 2018) (Vol. 11041, pp. 176-183). SPIE.

12. Gorokhovatskyi, V. A., Rusakova, N., & Tvoroshenko, I. S. (2020). The application of image analysis methods and predicate logic in applied problems of magnetic monitoring. Telecommunications and Radio Engineering, 79(20).

13. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

14. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Vlasenko, N. V. (2020). Using fuzzy clustering in structural methods of image classification. Telecommunications and Radio Engineering, 79(9).

15. Yakovleva, O., & Nikolaieva, K. (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. Advanced Information Systems, 4(4), 89-101.

16. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

17. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks. *International Journal of Academic Information Systems Research*, 7(7), (pp. 25-36).

18. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

19. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* (pp. 60-63). ACM.

20. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

21. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. In COLINS (3) (pp. 69-86).

22. Yakovleva, O., Slyusar, V., Kushnir, O., & Sabovchyk, A. (2021). New trends in scientific and technological revolution (STR) and transformation of science and education systems in the paradigm of sustainable development. In E3S Web of Conferences (Vol. 277, p. 06006). EDP Sciences.

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

24. Gorokhovatskyi, V., Tvoroshenko, I., & Olena, Y. (2024). Transforming image descriptions as a set of descriptors to construct classification features. *Indonesian Journal of Electrical Engineering and Computer Scienc*, 33 (1), (pp. 113-125)

25. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 210-230). Springer, Cham.

26. Кобилін, О. А., & Путятіна, О. Є. (2024). Знешумлення зображень, зіпсованих дробовим шумом, у реальному часі. Системи обробки інформації, (1 (176), 46-51.

27. Gorokhovatskyi , V., Chmutov , Y., Tvoroshenko , I., & Kobylin , O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5–12.

28. Кобилін, О., Вечірська, І., Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 63–76.

29. Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 97–107.

30. Vechirska, I., Kobylin, O., Prokopiev , S., Vechirska, A., & Kucherenko, M. (2022). Building a logical network for solving the problem of car rental by means algebra of finite predicates. Computer Systems and Information Technologies, (2), 78–87.

31. Kobylin, O.; Putiatina, O. (2025). Some aspects of real-time image denoising influenced by shot noise and compound Poisson noise. CEUR Workshop Proceedings ,volume = 3943 , 109-117.