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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Підвищення ефективності управління складським обліком на базі CRM+ERP системи Odoo»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

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

_____ Валерія КУНДИК
(підпис)

Виконала: здобувачка вищої освіти групи ПДМ-64
Валерія КУНДИК

Керівник: Оксана ЗОЛОТУХІНА
д.т.н., доцент

Рецензент: _____
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кундик Валерії Олексіївні

1. Тема кваліфікаційної роботи: «Підвищення ефективності управління складським обліком на базі CRM+ERP системи Odoo »

керівник кваліфікаційної роботи Оксана ЗОЛОТУХІНА д.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023 р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, документація системи Odoo.

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

1. Збір вимог у ключових користувачів майбутньої системи;
2. Аналіз бізнес-процесу клієнта;
3. Розробка методики впровадження та її реалізація;
4. Внутрішнє тестування системи;
5. Аналіз ефективності використання системи.

5. Перелік графічного матеріалу: *презентація*

1. 2. Процес надходження товару до впровадження.
3. Процес відвантаження товару до впровадження.
4. Модифікований алгоритм обліку процесу надходження товару.
Порівняльний аналіз ефективності використання системи
5. Модифікований алгоритм обліку процесу відбору товару.
6. Розрахунок страхового запасу залишків на складі.
7. Формальна модель правил для резервування товарів.
8. Приклад роботи з терміналом збору даних.
9. Порівняльний аналіз ефективності використання системи.

6. Дата видачі завдання «19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз бізнес-вимог клієнта та проведення інтерв'ю з ключовими працівниками компанії	19.10-05.11.23	
2	Опис вимог після проведення інтерв'ю	06.11-12.11.23	
3	Аналіз особливостей бізнес-процесу клієнта	13.11-19.11.23	
4	Модифікація бізнес-процесу клієнта	20.11-26.11.23	
5	Визначення критеріїв ефективності	27.11-01.12.23	
6	Розробка моделі оптимізації управління складським обліком	02.12-04.12.23	
7	Розробка алгоритму оптимізації управління складським обліком	0.5.12-08.12.23	
8	Дослідження ефективності методики	09.12-15.12.23	
9	Оформлення роботи: вступ, висновки, реферат	13.12-20.12.23	
10	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувачка вищої освіти _____ Валерія КУНДИК
(підпис)

Керівник
кваліфікаційної роботи _____ Оксана ЗОЛОТУХІНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 75 стор., 2 табл., 18 рис., 10 джерел.

Мета роботи: зменшення часу виконання операцій та кількості помилок в процесі управління складським обліком за рахунок використання CRM+ERP системи Odoo.

Об'єкт дослідження: управління складським обліком.

Предмет дослідження: методи та засоби управління складським обліком на базі CRM+ERP системи Odoo.

Короткий зміст роботи: У роботі провдено аналіз бізнес-процесу клієнта, за допомогою залучення до інтерв'ю основних користувачів майбутньої системи були проаналізовані процеси прийому та відвантаження товарів зі складу, а також процеси переміщення, визначені побажання та цілі впровадження.

Також розроблений модифікований алгоритм надходжень та відвантажень товару зі складу, для підвищення ефективності управління складським обліком.

Для визначення кращої системи управління складським обілком для замовника був проведений порівняльний аналіз найпопулярніших WMS-систем.

В ході порівняння різних систем було прийнято рішення використовувати Odoo для впровадження складського обліку в компанії, адже система має найбільш широкий функціонал, високу гнучкість та відносно швидкий процес впровадження.

Для автоматизації процесу відбору товару зі складу створена формальна модель правил резервації товару на складі. Впроваджений прорахунок страхового запасу, для зменшення ризиків затримок поставок від постачальників.

Розроблений інтерфейс системи для роботи з ТЗД для більш зручної роботи комірників складу. Проаналізовано ефективність використання системи складського обліку за основними критеріями: час прийняття замовлення менеджером, час передачі замовлення на склад, час відбору товару, відсоток оброблених замовлень, кількість механічних помилок та відсоток задоволених клієнтів.

КЛЮЧОВІ СЛОВА: Odoo, WMS, ТЗД, СИСТЕМА УПРАВЛІННЯ СКЛАДСЬКИМ ОБЛІКОМ, КОМІРНИК, ШТРИХ-КОД.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 75 pages, 2 tables, 18 figures, 10 sources.

The goal of the work: reducing the time of operations and the number of errors in the warehouse accounting management process due to the use of Odoo's CRM+ERP system.

Research object: management of warehouse accounting.

The subject of the study: methods and means of managing warehouse accounting based on the Odoo CRM+ERP system.

Summary of the work: An analysis of the client's business process was carried out in the work, with the help of interviews with the main users of the future system, the processes of receiving and shipping goods from the warehouse were analyzed, as well as the processes of movement, the wishes and implementation goals were determined.

The analysis of the key processes of setting up warehouse accounting was carried out. Familiarized with warehouse accounting management processes, nuances and privileges of using automated warehouse management systems.

A modified algorithm for receipts and shipments of goods from the warehouse has also been developed to improve the efficiency of warehouse accounting management.

To determine the best warehouse management system for the customer, a comparative analysis of the most popular WMS systems was conducted.

During the comparison of different systems, it was decided to use Odoo for the implementation of warehouse accounting in the company, because the system has the widest functionality, high flexibility and a relatively fast implementation process.

To automate the process of selecting goods from the warehouse, a formal model of the rules for reserving goods in the warehouse has been created. The miscalculation of the insurance stock has been implemented to reduce the risks of delays in deliveries from suppliers.

The system interface for working with TZD has been developed for more convenient work of warehouse storekeepers. The effectiveness of using the warehouse accounting system was analyzed according to the main criteria: the time of order acceptance by the manager, the time of transfer of the order to the warehouse, the time of product selection, the percentage of processed orders, the number of mechanical errors and the percentage of satisfied customers.

KEYWORDS: Odoo, WMS, TZD, WAREHOUSE ACCOUNTING MANAGEMENT SYSTEM, STOREKEEPER, BAR CODE.

ЗМІСТ

ВСТУП 11

РОЗДІЛ 1 АКТУАЛЬНІСТЬ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ 13

1.1 Основні поняття 14

1.1.1 Історія виникнення ERP 14

1.1.2 Основні поняття про WMS-системи 18

1.1.3 Основні поняття про ТЗД 21

1.2 Вибір Odoo як WMS системи 23

1.2.1 Аналіз досліджень по покращенню ефективності роботи за допомогою Odoo 27

1.3 Опис існуючих програмних рішень з автоматизації бізнес-процесів підприємств. 28

1.3.1 SAP ERP 28

1.3.2 Oracle Warehouse Management 31

1.3.3 Microsoft Dynamics 365 Supply Chain Management 32

1.3.3 Порівняння існуючих рішень 35

РОЗДІЛ 2 АНАЛІЗ ТА ЗБІР ВИМОГ ДЛЯ ОПТИМІЗАЦІЇ СИСТЕМИ УПРАВЛІННЯ СКЛАДСЬКИМ ОБЛІКОМ 38

2.1 Визначення критеріїв ефективності використання оптимизіції складського управління Odoo 39

2.2 Розробка моделі та методів оптимізації складського управління Odoo 42

2.2.1 Опис складських зон 42

2.2.2 Прийом товару на склад 43

2.2.3 Розміщення товару 45

2.2.4 Відбір товару 47

2.2.5 Переміщення товару 49

2.2.6 Страховий запас та резервація товарів 52

РОЗДІЛ 3 РОЗРОБКА ТА ПРОЕКТУВАННЯ СИСТЕМИ 55
3.1 Ініціалізація та налаштування проекту 55
3.2 Розробка основної частини системи 56
3.3 Розробка front-end частини 56
3.4 Огляд системи 57
3.4.1 Огляд інтерфейсу системи 57
3.4.2 Огляд інтерфейсу ТЗД 62
3.5 Тестування системи 65
3.6 Оцінка ефективності використання системи 66
ВИСНОВОК 68
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 72
ДОДАТОК А. Приклад програмного коду 80

ВСТУП

У контексті сучасного бізнес-середовища надзвичайно важливим є ефективне управління складськими процесами, оскільки високий обсяг товарів, постійні зміни вимог споживачів і зростання конкуренції створюють значні виклики для підприємств у сфері складського господарства. Одним з ключових факторів, спрямованих на оптимізацію та підвищення ефективності складських процесів є автоматизація складського обліку.

Мета роботи - зменшення часу виконання операцій та кількості помилок в процесі управління складським обліком за рахунок використання CRM+ERP системи Odoo.

Об'єкт дослідження - управління складським обліком.

Предмет дослідження - методи та засоби управління складським обліком на базі CRM+ERP системи Odoo.

Практична значущість результатів полягає в використанні розробленої системи на підприємстві для зменшення часу виконання операцій, зменшення механічних помилок та підвищення ефективності роботи.

Розробка системи автоматизації складського обліку на основі платформи Odoo, що є інтегрованою ERP-платформою, надає широкий функціонал для керування бізнес-процесами, включаючи управління складським обліком. Вибір Odoo як базової платформи для розробки системи дозволяє поєднати функціональність з гнучкістю налаштування та розширення, що відповідає потребам сучасних підприємств.

У процесі роботи планується вирішити ряд завдань: огляд та аналіз існуючих систем управління складом, аналіз існуючих алгоритмів автоматизації складського обліку, вибір найкращих рішень для розробки системи на основі Odoo, збір вимог у замовника, аналіз існуючих вимог та модифікація процесу надходження та відвантаження товарів зі складу, детальне вивчення технології та інструментів для розробки, розробка модифікованого алгоритму процесу прийому та відвантаження товару зі складу, планування страхового запасу залишків, аналіз та впровадження

існуючих правил резервацій товару в системі, запровадження та налаштування системи на реальному підприємстві.

Основним результатом дипломної роботи буде розроблена система автоматизації складського обліку на базі Odoo, спрямована на оптимізацію та підвищення ефективності управління складськими запасами.

1 АКТУАЛЬНІСТЬ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ

Управління складським обліком є критичним елементом для багатьох підприємств, щоб максимально використовувати свої ресурси, зменшуючи затрати та збільшуючи ефективність виробничих та логістичних процесів. ERP-система дозволяє автоматизувати управління бізнес-процесами, що прискорює виконання завдань та зменшує ймовірність помилок. Це допомагає компаніям підвищити ефективність роботи та знизити витрати на виробництво, складування, управління персоналом, фінанси та інші операції. [25]

Впровадження ERP-системи дозволяє компанії оперативно реагувати на зміни в бізнес-середовищі та покращити ефективність роботи, що в свою чергу може сприяти підвищенню її конкурентоспроможності та призвести до збільшення частки ринку.

Модуль керування складом відповідає за управління запасами, складськими операціями та логістичними процесами, включаючи облік надходження та відвантаження товарів, оптимізацію логістичних процесів та ін. [2]

Власники компаній вивчають багато методів покращення ефективності роботи підприємства. В статті «Як оптимізація запасів Odoo заробила 200 тисяч доларів для американської компанії» [3] представник однієї компанії розповів про проблеми та рішення, які допомогли значно оптимізувати процес складського обліку.

Правильне управління складським обліком є ключовим аспектом для стабільного функціонування компаній, Odoo дозволяє підтримувати продуктивність управління, а також дозволяє зменшити витрати компаній. [26]

1.1 Основні поняття

1.1.1 Історія виникнення ERP

ERP - це абревіатура від Enterprise Resource Planning, що перекладається як «планування ресурсів підприємства». ERP-система - це комплексна інформаційна система, яка об'єднує в єдину базу даних всю інформацію про діяльність підприємства. ERP-системи дозволяють підприємствам автоматизувати рутинні процеси, підвищити точність та надійність обліку, зменшити витрати та покращити ефективність управління. [5]

Історія ERP-систем починається в 1960-х роках, коли компанії почали використовувати комп'ютери для автоматизації окремих бізнес-процесів, таких як бухгалтерський облік, управління запасами та виробництво. Однак ці системи були відокремленими, що ускладнювало управління підприємством в цілому.

У 1980-х роках з'явилися перші ERP-системи, які об'єднували в єдину систему кілька бізнес-процесів. Ці системи були досить складними та дорогими, тому вони використовувалися лише великими підприємствами.

У 1990-х роках ERP-системи стали більш доступними та простими у використанні, що дозволило їх використовувати підприємствам різного розміру.

Сьогодні ERP-системи є одними з найпоширеніших інформаційних систем в корпоративному секторі. Вони використовуються підприємствами в різних галузях діяльності, від виробництва до торгівлі та послуг.

ERP-системи є важливим елементом в сучасному управлінні підприємством. Ці системи призначені для інтеграції та оптимізації всіх основних бізнес-процесів в одну централізовану платформу, спрощуючи взаємодію між різними відділами та функціональними областями організації.

ERP-системи включають в себе широкий спектр функцій, від фінансового обліку та управління замовленнями до кадрового обліку та логістики. Однією з ключових особливостей ERP-систем є їхній здатність автоматизувати рутинні завдання, уніфікувати дані та надавати зручний інтерфейс для аналізу даних.

ERP-системи можуть бути налаштовані під конкретні потреби підприємства, враховуючи його розмір, галузь, стратегію та внутрішню структуру. Ці системи забезпечують єдиний погляд на весь бізнес, дозволяючи керівникам приймати обгрунтовані рішення, основані на консолідованих та актуальних даних.

Ще однією важливою характеристикою ERP-систем є їхній вплив на оптимізацію внутрішніх процесів та підвищення ефективності в управлінні ресурсами. Завдяки єдиною платформою для обробки даних, компанії можуть уникати дублювання робіт, покращувати координацію та співпрацю між відділами.

ERP-системи також дозволяють підприємствам більш гнучко реагувати на зміни в економічному середовищі та швидше впроваджувати нові стратегії. Інтеграція великої кількості функціональностей в одну систему сприяє адаптабельності підприємства до нових викликів та можливостей.

Необхідно також відзначити те, що зростання цифровизації та використання новітніх технологій, таких як штучний інтелект та аналіз даних, посилює можливості ERP-систем. Вони стають не лише інструментом для обліку, але і стратегічним партнером в управлінні бізнес-процесами та прийнятті стратегічних рішень.

Наразі ERP-системи визначають новий стандарт для ефективного управління підприємством, забезпечуючи єдиною точкою доступу до всієї необхідної інформації та сприяючи зростанню конкурентоспроможності та стійкості бізнесу в умовах сучасного ринкового середовища.

ERP-системи мають такі переваги:

- підвищення ефективності: ERP-системи можуть допомогти підприємствам підвищити ефективність своїх процесів, що може привести до зниження витрат та підвищення продуктивності;
- покращення контролю: ERP-системи можуть допомогти підприємствам покращити контроль своїх операцій, що може привести до підвищення якості продукції та послуг;

- збільшення конкурентоспроможності: ERP-системи можуть допомогти підприємствам підвищити свою конкурентоспроможність, що може привести до зростання продажів та прибутку.

Однак ERP-системи мають і деякі недоліки:

- висока вартість: впровадження ERP-системи може бути дорогим;
- складність: впровадження ERP-системи може бути складним і вимагати значних ресурсів;
- необхідність змін: впровадження ERP-системи може вимагати змін у бізнес-процесах.

ERP-системи можуть виконувати широкий спектр функцій, включаючи:

- автоматизацію рутинного обліку: ERP-системи можуть автоматизувати такі процеси, як обробка замовлень, управління запасами, облік витрат та ін. Це дозволяє звільнити час співробітникам для більш важливих завдань, таких як стратегічне планування та аналіз;
- підвищення точності та надійності обліку: ERP-системи забезпечують централізоване зберігання інформації, що підвищує точність та надійність обліку;
- зменшення витрат: ERP-системи можуть допомогти підприємствам зменшити витрати за рахунок підвищення ефективності процесів, зниження помилок та оптимізації використання ресурсів;
- покращення ефективності управління: ERP-системи надають менеджерам точну та актуальну інформацію, необхідну для прийняття ефективних рішень.

ERP-системи зазвичай включають такі модулі:

- модуль виробництва: дозволяє планувати виробництво, контролювати витрати та якість продукції;
- модуль продажів: дозволяє обробляти замовлення, формувати рахунки-фактури та відстежувати виконання замовлень.
- модуль закупівель: дозволяє планувати закупівлі, управляти запасами та контролювати витрати;

- модуль складського обліку: дозволяє контролювати рух товарів на складі;
- модуль фінансів: дозволяє вести бухгалтерський облік, фінансовий аналіз та планування.

ERP-системи можна класифікувати за такими типами:

- на основі клієнт-сервера: цей тип ERP-систем працює на локальних серверах підприємства;
- в основі хмари: цей тип ERP-систем працює на віддалених серверах, до яких підприємства можуть отримувати доступ через Інтернет;
- в основі веб-браузера: цей тип ERP-систем працює через веб-браузер, що дозволяє користувачам отримувати доступ до системи з будь-якого пристрою, підключеного до Інтернету.

При виборі ERP-системи важливо враховувати такі фактори:

- необхідність та доцільність впровадження: підприємствам слід ретельно оцінити свої потреби та визначити, чи є впровадження ERP-системи доцільним для них;
- оцінка вартості: впровадження ERP-системи може бути дорогим, тому підприємствам слід ретельно оцінити витрати, пов'язані з впровадженням;
- планування: впровадження ERP-системи вимагає ретельного планування, включаючи визначення цілей, розробку плану впровадження та підготовку персоналу;
- впровадження: впровадження ERP-системи включає в себе такі етапи, як установка системи, навчання персоналу та впровадження змін у бізнес-процеси;
- підтримка та обслуговування: після впровадження ERP-системи підприємствам необхідно забезпечити її підтримку та обслуговування.

Сучасні ERP-системи постійно розвиваються, щоб відповідати потребам підприємств. Ось деякі з основних тенденцій у розвитку ERP-систем:

- розумне виробництво (Smart Manufacturing): ERP-системи все більше інтегруються з системами автоматизації виробництва, щоб допомогти підприємствам підвищити ефективність виробництва;

- інтелектуальний аналіз (BI): ERP-системи все більше використовують технології інтелектуального аналізу для аналізу даних і прийняття рішень.
- хмарні технології: хмарні ERP-системи стають все більш популярними, оскільки вони пропонують підприємствам більшу гнучкість і доступність;
- мобільні технології: ERP-системи все більше адаптуються до мобільних пристроїв, щоб дозволити співробітникам отримувати доступ до інформації з будь-якого місця.

При виборі ERP-системи важливо враховувати всі фактори, які були описані вище.

1.1.2 Основні поняття про WMS-системи

WMS-система - це система управління складом (Warehouse Management System), яка забезпечує комплексну автоматизацію складських операцій. WMS-системи дозволяють підприємствам підвищити ефективність управління складом, зменшити витрати та покращити обслуговування клієнтів. [1]

WMS-системи можуть виконувати широкий спектр функцій, включаючи:

- автоматизацію рутинного обліку: WMS-системи можуть автоматизувати такі процеси, як прийом товарів на склад, видача товарів зі складу, облік запасів та ін. Це дозволяє звільнити час співробітникам для більш важливих завдань, таких як управління запасами та складськими операціями;
- підвищення точності та надійності обліку: WMS-системи забезпечують централізоване зберігання інформації, що підвищує точність та надійність обліку;
- зменшення витрат: WMS-системи можуть допомогти підприємствам зменшити витрати за рахунок підвищення ефективності процесів, зниження помилок та оптимізації використання ресурсів;
- покращення обслуговування клієнтів: WMS-системи можуть допомогти підприємствам покращити обслуговування клієнтів, забезпечуючи швидку та точну обробку замовлень.

WMS-системи зазвичай включають такі модулі:

- модуль прийому товарів: дозволяє реєструвати прийом товарів на склад, включаючи перевірку якості та кількості товарів;
- модуль видачі товарів: дозволяє реєструвати видачу товарів зі складу, включаючи відстеження руху товарів;
- модуль обліку запасів: дозволяє вести облік наявності товарів на складі, включаючи прогнозування потреби в товарах та управління запасами;
- модуль складських операцій: дозволяє планувати та контролювати складські операції, включаючи вантажопереміщення, сортування та пакування товарів;
- модуль управління виконанням замовлень: дозволяє відстежувати виконання замовлень та інформувати клієнтів про статус їхніх замовлень.

WMS-системи можна класифікувати за такими типами:

- на основі клієнт-сервера: цей тип WMS-систем працює на локальних серверах підприємства;
- в основі хмари: цей тип WMS-систем працює на віддалених серверах, до яких підприємства можуть отримувати доступ через Інтернет.

При виборі WMS-системи важливо враховувати такі фактори:

- необхідність та доцільність впровадження: підприємствам слід ретельно оцінити свої потреби та визначити, чи є впровадження WMS-системи доцільним для них;
- оцінка вартості: впровадження WMS-системи може бути дорогим, тому підприємствам слід ретельно оцінити витрати, пов'язані з впровадженням;
- планування: впровадження WMS-системи вимагає ретельного планування, включаючи визначення цілей, розробку плану впровадження та підготовку персоналу;
- впровадження: впровадження WMS-системи включає в себе такі етапи, як установка системи, навчання персоналу та впровадження змін у бізнес-процеси;
- підтримка та обслуговування: після впровадження WMS-системи підприємствам необхідно забезпечити її підтримку та обслуговування.

Сучасні WMS-системи постійно розвиваються, щоб відповідати потребам підприємств. Ось деякі з основних тенденцій у розвитку WMS-систем:

- інтеграція з іншими системами: WMS-системи все більше інтегруються з іншими системами, такими як ERP-системи, CRM-системи та WMS-системи, щоб забезпечити комплексний підхід до управління складом;
- використання штучного інтелекту: WMS-системи все більше використовують технології штучного інтелекту для автоматизації завдань і прийняття рішень;
- мобільні технології: WMS-системи все більше адаптуються до мобільних пристроїв, щоб дозволити співробітникам отримувати доступ до інформації з будь-якого місця.

В Україні WMS-системи є все більш популярними. На ринку представлено широкий спектр WMS систем від вітчизняних та іноземних виробників. Вартість WMS систем в Україні залежить від функціональних.

Ефективне використання систем управління складом (WMS) є ключовим чинником для оптимізації логістичних та складських процесів у сучасному бізнесі. Впровадження WMS відкриває перед компаніями низку переваг, які визначають їхню конкурентоспроможність та здатність ефективно відповідати на зростаючі вимоги ринку.

Однією з ключових переваг є ефективне управління запасами, яке дозволяє детально відслідковувати рух товарів на складі, визначати їх точні розташування та уникати перенадлишків або дефіцитів. Точність обліку запасів впливає на фінансові показники компанії та її здатність ефективно планувати постачання.

Крім того, WMS допомагає підвищити продуктивність трудових ресурсів за рахунок автоматизованих процесів, що зменшує час на виконання завдань, таких як прийомка та відвантаження товарів. Технології сканування штрих-кодів та системи автоматизації відіграють важливу роль у цьому процесі. [1]

Важливим аспектом є інтеграція WMS з іншими елементами ланцюга постачання та екосистемою підприємства. Система повинна взаємодіяти з іншими

ERP системами, транспортними рішеннями, системами управління виробництвом та іншими компонентами, щоб забезпечити єдиноцільну інформаційну базу.

Застосування WMS дозволяє підприємствам мінімізувати помилки та підвищити точність обліку, що має прямий вплив на якість обслуговування клієнтів та репутацію компанії. Інформація про стан запасів, рух товарів та інші параметри надає підприємствам можливість приймати обгрунтовані стратегічні рішення та вчасно реагувати на зміни в ринкових умовах.

Усе це сприяє підвищенню видимості та контролю над логістичними процесами. Використання технологій сканування, IoT рішень та аналітичних інструментів дозволяє компаніям отримувати більш детальний огляд своєї діяльності та шляхів оптимізації.

Оптимізація використання простору складу є ще однією вагомою перевагою. WMS допомагає розташовувати товари так, щоб використовувати простір ефективно та максимізувати його потенціал.

Важливим аспектом є також адаптабельність WMS до змін в запитах клієнтів та бізнес-процесах. Здатність швидко реагувати на зміни у попиті або стратегії компанії надає підприємствам конкурентну перевагу.

Не менш важливою перевагою є зниження витрат та підвищення вибірковості в управлінні складом. Оптимізація процесів та ефективне використання ресурсів дозволяє компаніям знижувати витрати та забезпечувати більш ефективне використання робочих ресурсів.

Загалом, використання WMS системи виявляється надзвичайно вигідним стратегічним кроком для підприємств, що дозволяє їм не тільки оптимізувати свої внутрішні процеси, але й підвищувати конкурентоспроможність на ринку завдяки швидкій реакції на зміни та підвищенню загальної ефективності бізнесу.

1.1.3 Основні поняття про ТЗД

ТЗД - це скорочення від «термінал збору даних» (англ. - data collection terminal). ТЗД - це мобільний електронний пристрій, призначений для збору та обробки

інформації в процесі роботи. ТЗД використовуються в різних сферах діяльності, зокрема в торгівлі, логістиці, виробництві та інших.

Сучасні ТЗД є потужними мобільними комп'ютерами з вбудованим сканером штрихкодів. Вони можуть виконувати широкий спектр функцій, включаючи:

- сканування штрихкодів;
- введення тексту;
- відображення інформації;
- підключення до інших пристроїв.

ТЗД використовуються в різних сферах діяльності, зокрема в торгівлі, логістиці, виробництві та інших. Вони є важливим інструментом для автоматизації бізнес-процесів і підвищення ефективності роботи підприємств. [14]

Ефективне управління складом – це складний процес, який вимагає детального контролю та точності в кожному етапі. В цьому контексті, використання Терміналів збору даних (ТЗД) визначається як важливий стратегічний елемент для досягнення оптимальної ефективності та надійності управління складськими процесами.

Однією з ключових переваг використання ТЗД є підвищення точності та унікальності даних. Сканування штрих-кодів дозволяє уникнути помилок та неправильного внесення інформації в систему, забезпечуючи надійність обліку товарів на складі.

Застосування ТЗД також сприяє прискоренню рутинних операцій та зменшенню часу на їх виконання. Автоматизація прийому товарів, інвентаризації та відвантаження робить процеси більш ефективними, збільшуючи продуктивність та раціоналізуючи трудові ресурси.

Використання ТЗД також вдосконалює логістику та маршрутизацію товарів на складі. За допомогою точних даних, отриманих з ТЗД, оператори можуть отримувати оптимальні маршрути та точні вказівки щодо розташування товарів, що сприяє зниженню часу переміщення та підвищенню ефективності складських операцій.

Підвищення точності інвентаризації – ще одна вагома перевага використання ТЗД. Систематичне сканування штрих-кодів під час інвентаризації дозволяє оперативно виявляти розходження між фактичними та обліковими залишками, забезпечуючи точність обліку запасів.

Окрім того, ТЗД сприяє підвищенню зручності та мобільності на складі. Оператори можуть взаємодіяти з системою, перебуваючи на різних ділянках складу, що забезпечує гнучкість та зручність у виконанні робочих завдань.
Важливо також відзначити, що ТЗД легко інтегрується з іншими складськими та управлінськими системами, створюючи єдину інформаційну екосистему. Інформація, отримана з ТЗД, автоматично оновлюється в центральній базі даних, що забезпечує єдність даних та уникнення дублювання інформації.

Загалом, використання ТЗД в сфері управління складом значно полегшує рутинні операції, підвищує точність та ефективність роботи, дозволяє оперативно реагувати на зміни та створює підґрунтя для модернізації та оптимізації логістичних процесів на складі.

1.2 Вибір Odoo як WMS системи

Odoo - це відкрите програмне забезпечення для управління бізнес-процесами (ERP), яке пропонує комплексний підхід до управління бізнесом. Odoo містить набір модулів, які можна використовувати для автоматизації різних бізнес-процесів, таких як управління складом, продажі, закупівлі, фінанси, проектний менеджмент, виробництво та інші.

Однією з ключових переваг Odoo є те, що він є повністю відкритим програмним забезпеченням, що дозволяє користувачам вільно налаштовувати його функціонал відповідно до власних потреб. Це означає, що Odoo можна використовувати для управління будь-яким типом бізнесу, незалежно від його розміру та галузі діяльності.

Ще однією перевагою Odoo є те, що він має велику та активну спільноту користувачів та розробників, яка постійно вдосконалює та доповнює його

функціонал. Це забезпечує підтримку та оновлення програмного забезпечення відповідно до останніх технологічних тенденцій та вимог користувачів. [10]

На рисунку 1.1. зображений приклад інтерфейсу системи Odoo:

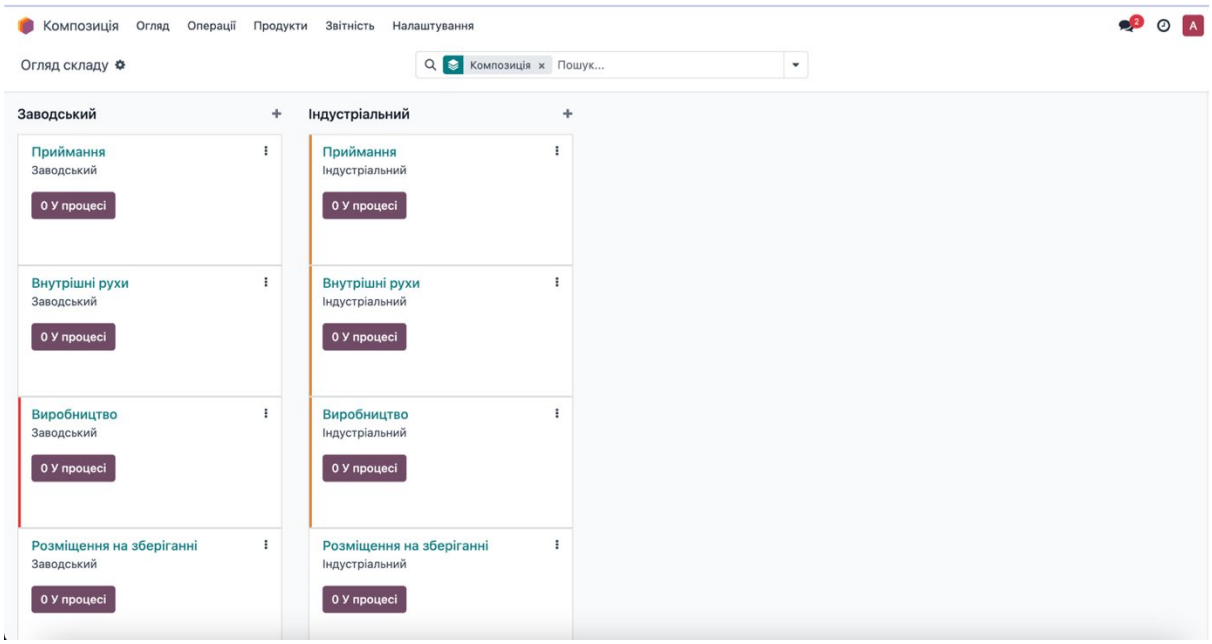


Рис. 1.1. Приклад інтерфейсу системи Odoo

Odoo може бути налаштований для виконання різних завдань, включаючи:

- управління складом;
- виробництво;
- продажі;
- закупівлі;
- фінанси.

Управління складом - це один з основних модулів Odoo. Він дозволяє підприємствам автоматизувати такі завдання, як прийом товарів, видача товарів, відстеження запасів та управління складськими операціями.

Виробництво - це ще один важливий модуль Odoo. Він дозволяє підприємствам автоматизувати такі завдання, як планування виробництва, управління запасами матеріалів, відстеження якості та управління виробничими процесами.

Продажі - це модуль Odoo, який дозволяє підприємствам автоматизувати такі завдання, як створення рахунків-фактур, управління замовленнями та відстеження продажів.

Закупівлі - це модуль Odoo, який дозволяє підприємствам автоматизувати такі завдання, як створення замовлень на закупівлі, управління постачальниками та відстеження поставок.

Фінанси - це модуль Odoo, який дозволяє підприємствам вести бухгалтерський облік, управління фінансами та оподаткування.

Однак, як і у будь-якого програмного забезпечення, Odoo також має свої недоліки. Наприклад, при налаштуванні системи можуть виникати складнощі та проблеми з безпекою, особливо якщо користувачі не мають досвіду в роботі з програмним забезпеченням.

Крім того, якщо бізнес вимагає великої кількості користувачів, то можуть виникнути проблеми з продуктивністю та швидкістю роботи системи.

У цілому, Odoo є потужним та гнучким програмним забезпеченням для управління бізнесом, яке може бути ефективним рішенням для управління різними аспектами діяльності, включаючи складський облік.

Ось деякі конкретні приклади того, як Odoo може бути використаний для управління складським обліком:

- для автоматизації прийому товарів на склад;
- для автоматизації видачі товарів зі складу;
- для відстеження запасів товарів на складі;
- для управління складськими операціями, такими як переміщення товарів та замовлення поповнення запасів.

Обираючи Odoo як систему управління складом (WMS), компанії отримують широкий спектр переваг, які відображаються на ефективності, точності та інтегрованості їхніх логістичних та складських процесів.

Odoo пропонує повний набір бізнес-додатків, що включає WMS, але також охоплює області від управління клієнтами та продажами до обліку та виробництва. Інтеграція всіх цих функцій на одній платформі дозволяє створити єдиний, злагоджений бізнес-екосистему.

Система також надає можливість гнучкої настройки, що дозволяє адаптувати систему під конкретні потреби компанії. Зокрема, модуль WMS може бути

налаштований для відповіді на унікальні бізнес-процеси та вимоги складського управління.

Інтуїтивний інтерфейс користувача ускладнює навчання персоналу та полегшує використання системи WMS. Завдяки інтегрованому та дружньому інтерфейсу, персонал може ефективно взаємодіяти з системою та виконувати завдання без зайвих труднощів.

Odoo WMS дозволяє автоматизувати багато рутинних операцій, таких як відстеження запасів, управління рухом товарів та оптимізація складських процесів. Це призводить до зменшення помилок та підвищення продуктивності.

Вбудовані засоби аналітики та звітності в Odoo дозволяють здійснювати моніторинг та аналіз різних аспектів логістичних та складських операцій, що сприяє прийняттю обґрунтованих стратегічних рішень.

Однією з переваг вибору Odoo є активна спільнота користувачів та широкий спектр ресурсів, включаючи документацію та форуми підтримки, що полегшує впровадження та використання системи.

Також є інтеграція з системою керування замовленнями. Це дозволяє автоматизувати та координувати виконання замовлень, від початкового прийняття до фінальної доставки, що спрощує обробку замовлень та підвищує задоволеність клієнтів.

Odoo WMS надає можливість взаємодії з системою через мобільні пристрої. Це дозволяє персоналу працювати на складі безпосередньо з місця події, здійснювати сканування товарів та виконувати інші завдання, підвищуючи мобільність та ефективність операцій.

Одною з суттєвих переваг Odoo WMS є його здатність легко інтегруватися з різноманітним обладнанням для сканування штрих-кодів, етикетування та автоматизації складських операцій, що підвищує точність та швидкість виконання завдань.

Враховуючи важливість безпеки даних, Odoo WMS забезпечує надійний рівень захисту конфіденційної інформації, зокрема, враховуючи доступ до різних рівнів користувачів та шифрування даних.

Odoo має глобальну спільноту користувачів та забезпечує локалізацію для різних регіонів та країн, що полегшує впровадження та адаптацію системи до конкретних вимог ринку.

Odoo WMS є масштабованим рішенням, що може легко розширюватися разом з ростом бізнесу. Це робить його ідеальним вибором для компаній будь-якого розміру та галузі.

З урахуванням цих факторів Odoo стає відмінним рішенням для компаній, які прагнуть ефективно оптимізувати свої складські процеси та впроваджувати інновації у логістичному управлінні.

Odoo - це потужний інструмент, який може допомогти підприємствам автоматизувати складський облік та підвищити ефективність роботи.

1.2.1 Аналіз досліджень по покращенню ефективності роботи за допомогою Odoo

Власники компаній вивчають багато методів покращення ефективності роботи підприємства. В статті «Як оптимізація запасів Odoo заробила 200 тисяч доларів для американської компанії» [3] представник однієї компанії розповів про проблеми та рішення, які допомогли значно оптимізувати процес складського обліку.

В статті «Реалізація Odoo CRM/ERP/WMS для магазину електронної комерції» [15] Метою проекту було повністю оцифрувати компанію та дизайн-студію та оптимізувати всі процеси в компанії для забезпечення швидкого розвитку студії. Проект передбачав впровадження системи Odoo в широкому діапазоні, що дозволить комплексно управляти всією компанією.

Така модель вимагає дуже ефективного наскрізного управління компанією від процесу набору співробітників і постачальників до продажів, побудови CRM, управління проектами, фінансами, співробітниками, витратами, календарем ділових зустрічей і каталогами з файлами, водночас надаючи клієнтам і постачальникам доступ до зовнішнього порталу для спілкування. Здатність інтегрувати ERP-систему з іншими зовнішніми системами також була

вирішальною у функціонуванні компанії. Мета була досягнута під час 3-місячного проекту впровадження для команди з 6 співробітників.

В той же час, при аналізі статті «Випадок використання оцінки відкритого вихідного коду: TETRA виявила значні покращення якості системи Odoo ERP» [11] та під час перевірки зручності використання не було виявлено жодних критичних проблем із блокуванням, користуванням системою та зручністю інтерфейсу. Додаток Odoo має хорошу якість щодо простоти використання, а на сайті є повні та якісні відповіді на поширені запитання та документація. Є лише кілька незначних проблем (пов'язаних із контрольним списком і списком знайдених дефектів), які практично не впливають на роботу користувача.

В статтях [16-20] також розглядається застосовуваність Odoo в різних галузях. Найголовнішою перевагою для бізнесу стало інтегроване керування даними: компанії прагнули оптимізувати керування даними шляхом інтеграції процесів між відділами та модулями, забезпечуючи узгодженість та ефективність. За допомогою Odoo компанії змогли автоматизувати ручні процеси та підвищити продуктивність.

1.3 Опис існуючих програмних рішень з автоматизації бізнес-процесів підприємств.

1.3.1 SAP ERP

SAP ERP (Enterprise Resource Planning) - це інтегрована система управління підприємством, розроблена компанією SAP SE. Ця платформа надає рішення для управління різними бізнес-процесами в організації, включаючи фінанси, логістику, виробництво, ресурси людей, збут та інші аспекти діяльності.

SAP ERP забезпечує інтеграцію різноманітних бізнес-процесів в єдину систему. Це дозволяє компаніям ефективно взаємодіяти між різними підрозділами та функціональними областями.

Дана система надає інструменти для ефективного фінансового планування та обліку, включаючи бюджетування, облік активів, управління витратами та

фінансовий аналіз. Система включає в себе модулі для управління запасами, замовленнями, виробництвом та постачанням, сприяючи ефективному управлінню ланцюгом постачання.

SAP ERP дозволяє оптимізувати виробничі процеси, враховуючи планування виробництва, контроль якості та виробничий облік.

Модуль управління ресурсами людей (HR) включає функції управління кадровими процесами, включаючи оплату праці, навчання та розвиток. SAP ERP допомагає в управлінні збутовими процесами, від обслуговування клієнтів до аналізу ринкової діяльності.

Платформа надає розширені засоби аналітики та звітності для здійснення обґрунтованих стратегічних рішень на основі даних.

Вона є гнучкою системою, яка може бути налаштована під індивідуальні потреби підприємства. Спільнота користувачів SAP та наявність ресурсів для навчання та підтримки дозволяють підприємствам ефективно впроваджувати та використовувати систему.

SAP визнана як одна з провідних ERP-систем, що пропонує рішення для обліку запасів. Вона забезпечує функціональні можливості управління запасами, прогнозування попиту, операцій, логістики та фінансів.

Система дозволяє організувати діяльність усього підприємства. Має широкий функціонал і забезпечує ефективність роботи організації. Працівники різних напрямків можуть вести свою діяльність в програмі. [4]

Модулі SAP ERP:

- управління фінансами. Дозволяє вести облік основних засобів, контролювати фінансові операції, автоматизувати роботу бухгалтерії, вести необхідну звітність;
- виробництво. Управління всіма аспектами виробництва. Здійснюється планування, контроль і аналіз робіт;
- замовлення. Регуляція взаємовідносин з контрагентами. Підтримка операцій продажу, обслуговування, сервісу клієнтів і т.д .;

- управління якістю. Забезпечується планування та контроль якості продукції під час виробництва і закупівель;
- ремонт і обслуговування обладнання. Планування коштів на проведення ремонтних робіт і облік витрат;
- інформаційні потоки. З'єднання модулів з інструментами, технологіями та сервісами, які загальні для додатків. Можна задати правила, згідно з якими дана частина системи автоматизує господарські операції;
- управління персоналом. Планування зайнятості співробітників, розрахунок заробітної плати, відряджень, пільг і т.д. Модуль дозволяє наймати персонал, проводити підвищення кваліфікації.

На рисунку 1.2. зображений приклад інтерфейсу системи:

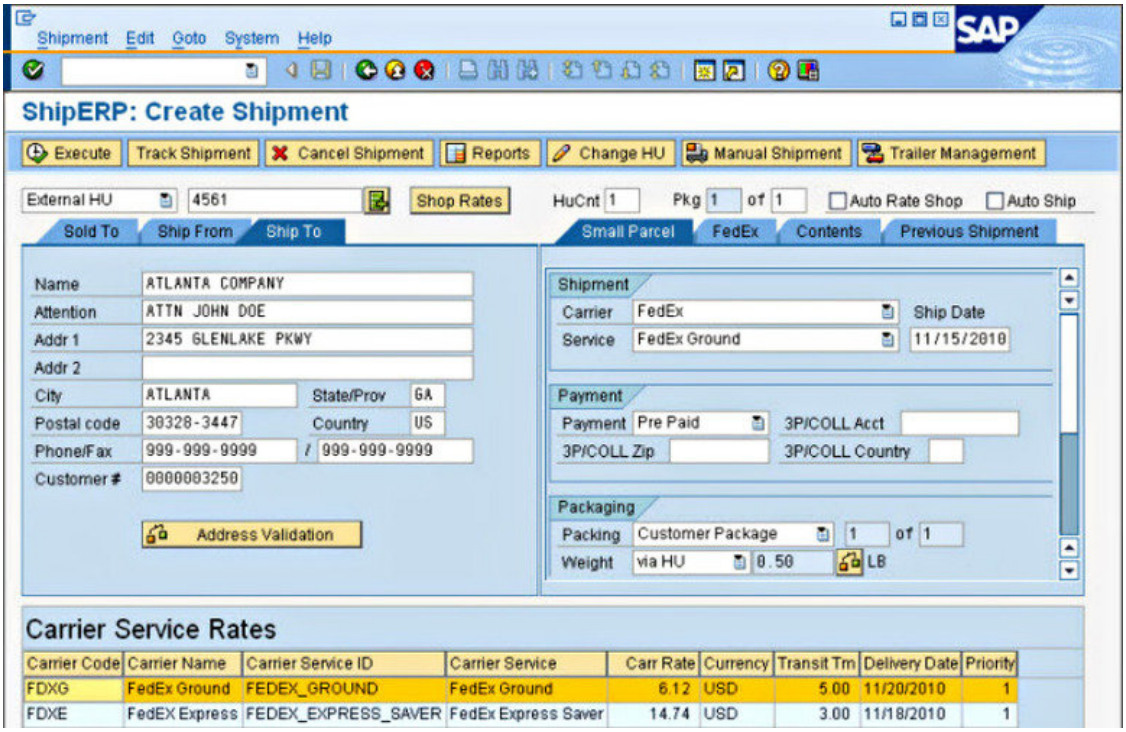


Рис. 1.2. Приклад інтерфейсу системи SAP ERP

Крім того, SAP ERP підтримує різні мови та валюти, що робить його корисним для міжнародних компаній.

Незважаючи на це, використання SAP є витратним та складним. Дану ERP-систему використовують компанії всіх розмірів та сфер діяльності, надаючи їм

інструменти для ефективного управління бізнесом та досягнення стратегічних цілей.

1.3.2 Oracle Warehouse Management

Oracle Warehouse Management (OWM) - це система управління складом від компанії Oracle, яка призначена для оптимізації та автоматизації операцій у складському обліку. Вона дозволяє компаніям ефективно вирішувати завдання, пов'язані зі зберіганням, переміщенням і відвантаженням товарів на складах, веде точний облік запасів та допомагає у зниженні надлишкових запасів, оптимізації замовлень та забезпеченні належного рівня запасів. Система також дозволяє ефективно використовувати простір на складі, враховуючи розмір, форму та характеристики товарів.

Oracle Warehouse Management автоматизує обробку замовлень, розподілення товарів та виконання вимог замовників, що допомагає забезпечити швидку та точну обробку замовлень, а також дозволяє в режимі реального часу відстежувати рух товарів на складі, що сприяє зниженню втрат і покращенню контролю за запасами.

Oracle Warehouse Management також підтримує управління запасами на різних складах, включаючи географічно розподілені підприємства. Система легко інтегрується з іншими ентерпрайз-системами, такими як ERP (Enterprise Resource Planning), що дозволяє обмін даними та підтримує однорідність інформації.

Також вона надає різноманітні засоби аналітики та звітності, які допомагають компаніям зрозуміти ефективність їхніх складських операцій та приймати обґрунтовані рішення та дозволяє підприємствам управляти своїм складом більш ефективно, зменшуючи витрати та покращуючи обслуговування клієнтів. [13]
На рисунку 1.3. зображений приклад інтерфейсу системи:

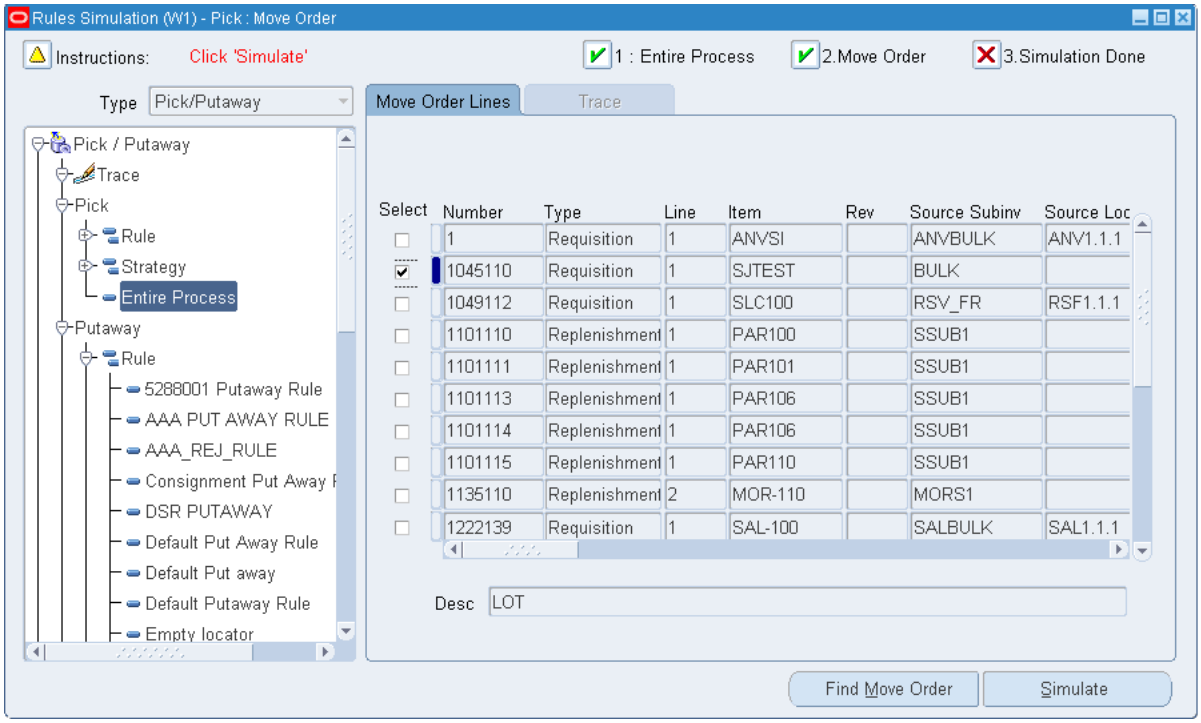


Рис. 1.3. приклад інтерфейсу Oracle Warehouse Management

Однією з найголовніших переваг Oracle Warehouse Management є можливість інтеграції з іншими системами управління запасами, такими як системи радіочастотної ідентифікації (RFID). Проте, використання Oracle Warehouse Management є витратним та складним. Впровадження цієї системи може виявитися трудомістким та витратним за часом процесом.

Для успішної реалізації необхідно провести ретельний аналіз бізнес-процесів, налаштувати систему з урахуванням потреб підприємства, інтегрувати її з іншими системами та навчити персонал користуватися новою системою.

Навчання основних користувачів може займати багато часу, так як інтерфейс системи може викликати певні запитання через його складність.

1.3.3 Microsoft Dynamics 365 Supply Chain Management

Microsoft Dynamics 365 Supply Chain Management - це програмне рішення, яке розроблене для оптимізації управління логістикою, виробництвом та операціями з управління ланцюжком постачання в організаціях. Ця платформа дозволяє

компаніям покращувати ефективність та контроль у всіх аспектах ланцюжка постачання, від управління запасами до оптимізації виробництва.

Система надає інструменти для точного управління запасами, прогнозування попиту, оптимізації рівня запасів та уникнення втрат, планувати та оптимізувати виробничі процеси з урахуванням попиту на товари та наявних ресурсів. Це допомогає оптимізувати процеси управління замовленнями від постачальників до клієнтів, забезпечуючи точність та швидкість виконання замовлень.

Microsoft Dynamics 365 Supply Chain Management вважається інтегрованою системою управління ланцюгом постачання, розроблена корпорацією Microsoft.

Це рішення спрямоване на оптимізацію та автоматизацію ключових процесів в ланцюзі постачання підприємств, що сприяє підвищенню ефективності, зниженню витрат та покращенню обслуговування клієнтів.

Основні характеристики та переваги Microsoft Dynamics 365 Supply Chain Management включають логістику та управління запасами, ефективне управління замовленнями, моніторинг ланцюга постачання, прогнозування та планування попиту, співпрацю з постачальниками, мобільний доступ, аналітику та звітність, інтеграцію з іншими додатками Microsoft.

Система дозволяє підприємствам ефективно керувати виробничими та постачальницькими процесами, забезпечуючи високий рівень видимості та контролю. Заснована на принципах гнучкості та інтеграції, Microsoft Dynamics 365 Supply Chain Management допомагає вирішувати виклики управління запасами, оптимізувати логістичні потоки та підтримувати стратегічні бізнес-цілі компаній.

Microsoft Dynamics 365 Supply Chain Management надає засоби для відстеження руху товарів у ланцюжку постачання та надає аналітичні звіти для прийняття обгрунтованих рішень та легко інтегрується з іншими системами, такими як ERP та CRM, щоб забезпечити взаємодію між різними функціональними областями підприємства.

Новою функцією системи є використання AI для оптимізації прогнозування попиту, планування запасів та прийняття стратегічних рішень.

Також система надає можливість доступу до даних та функцій системи через мобільні пристрої, що дозволяє працювати з системою у будь-який час та в будь-якому місці.

Microsoft Dynamics 365 Supply Chain Management допомагає підприємствам підвищити продуктивність, оптимізувати витрати та підвищити якість обслуговування клієнтів шляхом оптимізації всього ланцюжка постачання.
На рисунку 1.4. зображений приклад інтерфейсу системи:

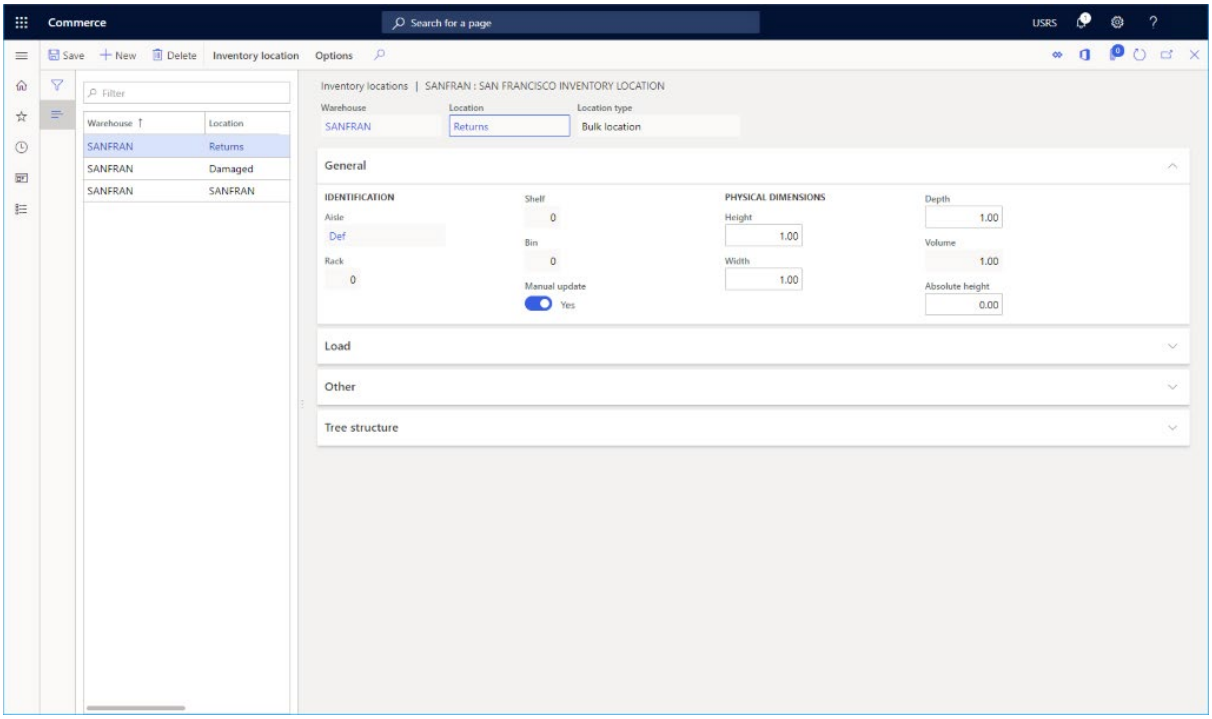


Рис. 1.4. Приклад інтерфейсу Microsoft Dynamics 365 Supply Chain Management

Однією з переваг Microsoft Dynamics 365 Supply Chain Management є можливість інтеграції з іншими продуктами Microsoft, такими як Microsoft Office та Power BI. В порівнянні з SAP та Oracle, Microsoft Dynamics 365 Supply Chain Management володіє більшою доступністю та меншою складністю використання.

Серед недоліків системи слід відзначити обмеження в можливостях налаштування, оскільки гнучкість налаштувань та модифікацій у Microsoft Dynamics 365 Supply Chain Management обмежена, що може становити обмеження для системи та ускладнювати впровадження, особливо для компаній з унікальними бізнес-процесами.

1.3.3 Порівняння існуючих рішень

Зважаючи на комплексність та різноманітність потреб підприємств у сучасному управлінні ланцюгом постачання, вибір системи управління складом та ланцюгом постачання є стратегічно важливим рішенням. Microsoft Dynamics 365 Supply Chain Management, SAP, Oracle Warehouse Management та Odoo представляють собою різні підходи та можливості для підтримки різних бізнес-сценаріїв.

SAP, який визначається своєю великою функціональністю та високою ступенем комплексності, може бути оптимальним вибором для великих корпорацій, що мають потребу в розширених можливостях, налаштувань та глибокій інтеграції.

Microsoft Dynamics 365 Supply Chain Management виграє за рахунок інтеграції з іншими продуктами Microsoft, забезпечуючи єдину екосистему та мобільність для працівників на різних рівнях.

Oracle Warehouse Management зосереджений на оптимізації управління складом та може бути сприятливим вибором для тих, хто шукає рішення, інтегроване з іншими продуктами Oracle.

З іншого боку, Odoo, який є відкритою та гнучкою системою, може вирізнятися в невеликих та середніх підприємствах, які прагнуть ефективно управляти складськими процесами за доступною ціною та з можливістю швидкої адаптації до змінних вимог.

Важливим аспектом є також спільнота користувачів та підтримка. Odoo має активну спільноту, що може бути важливим фактором у вирішенні питань та обміні досвідом.

Вибір найбільш підходящої системи варто здійснювати на основі конкретних бізнес-потреб, фінансових можливостей та стратегічних цілей компанії. Система повинна дозволяти підприємствам ефективно керувати виробничими та постачальними процесами, забезпечуючи високий рівень видимості та контролю, бути заснована на принципах гнучкості та інтеграції,

Правильне управління складським обліком є ключовим аспектом для стабільного функціонування компаній, Odoo дозволяє підтримувати продуктивність управління, а також дозволяє зменшити витрати компаній.

Таблиця 1.1.

Порівняльний аналіз існуючих систем

Особливості	SAP	Microsoft Dynamics 365 Supply Chain Management	Oracle Warehouse Management	Odoo
Ключові функції системи	- Управління матеріальними потоками - Планування потреби в матеріалах і послугах - Управління збутом та коригування залишків	- Управління складами та виконання замовлень - Закупівлі - Управління замовленнями та ціни - Обслуговування активів та управління	- Підтримка виконання замовлень - Управління поверненнями - Координація руху товарів - Інвентаризація - Оптимізація складські операції	- Відстеження руху товарів - Адресне місцезберігання - Коригування залишків - Відстеження серійних номерів - Стратегії вилучення - Прогнозований запас залишків
Впровадження	Складний та довгий процес впровадження	Складний та довгий процес впровадження	Складний та довгий процес впровадження	Швидкий процес впровадження
Гнучкість	Висока	Обмеження в налаштуваннях	Обмеження в налаштуваннях	Висока

В даній таблиці наведено ключові порівняльні характеристики, такі як: особливості систем, ключові функції, процес впровадження та гнучкість систем. Ключовими функціями SAP є управління матеріальними потоками, планування потреби в матеріалах і послугах та можливість керування збутом та коригування залишків.

В Microsoft Dynamics 365 Supply Chain Management основними функціями системи є управління складами та замовленнями, контроль закупівель, контроль замовлень та цін, а також лбслуговування активів та управління.

Для Oracle Warehouse Management ключовими функціями виступають підтримка виконання замовлень, управління поверненнями, координація руху товарів, інвентаризація та оптимізація складських операцій.

Odoo має найбільш розширений функціонал, такий як відстеження руху товарів, адресне місцезберігання, коригування залишками, відстеження товарів по серійним номерам, створення стратегій вилучення товарів, перегляд прогнозованого запасу залишків.

Впровадження всіх існуючих систем, окрім Odoo, є складним та довгим процесом.

Гнучкість систем є високою у SAP та Odoo, в той час як Microsoft Dynamics 365 Supply Chain Management та Oracle Warehouse Management мають досі великі обмеження в налаштуваннях.

В ході порівняння різних систем було прийнято рішення використовувати Odoo для впровадження складського обліку в компанії, адже система має найбільш широкий функціонал, високу гнучкість та відносно швидкий процес впровадження.

2 АНАЛІЗ ТА ЗБІР ВИМОГ ДЛЯ ОПТИМІЗАЦІЇ СИСТЕМИ УПРАВЛІННЯ СКЛАДСЬКИМ ОБЛІКОМ

На момент проведення інтерв'ю у замовника функціонують два склади, що фізично перебувають на різних адресах.

Перший склад є основним, з цього складу відбувається відвантаження товару для всіх клієнтів, в системі цей склад називатиметься «Заводський».

Другий склад знаходиться на стадії завершення ремонту та введення в експлуатацію. Надалі саме цей склад планується надходження всіх поставок. В системі цей склад називатиметься «Індустріальний».

Впровадження системи управління складським обліком буде поділено на 2 етапи. На першому будуть автоматизовані процеси на складі «Індустріальний», на другому на складі «Заводський».

На складі встановлені стелажні системи двох видів, фронтальні стелажі та консольні стелажі. У разі потреби користувача системи буде можливість створювати нові місця зберігання згідно з прописаною логікою в системі.
На складі передбачено дві брами, перші призначені для відвантаження продукції, другі для приймання.

На складі планується використання навантажувальної техніки, такої як: поводкові штабелери, вилові навантажувачі, річ траки, гідравлічні візки (рокли).

Система, що розробляється, націлена на вирішення завдань складської логістики та обліку замовника. У рамках розробки системи буде створено такі основні компоненти:

- інтерфейс оператора складу. Цей модуль є повною версією веб-додатку і містить повну логіку складського обліку;
- інтерфейс мобільного працівника (для ТЗД). Даний модуль є мобільною, урізаною версією веб-додатку та відповідає за взаємодію системи з ТЗД під час роботи мобільним співробітником із завданнями.

2.1 Визначення критеріїв ефективності використання оптимізіції складського управління Odoo

В ході проведення інтерв'ю був визначений процес надходження товару на склад до впровадження системи.

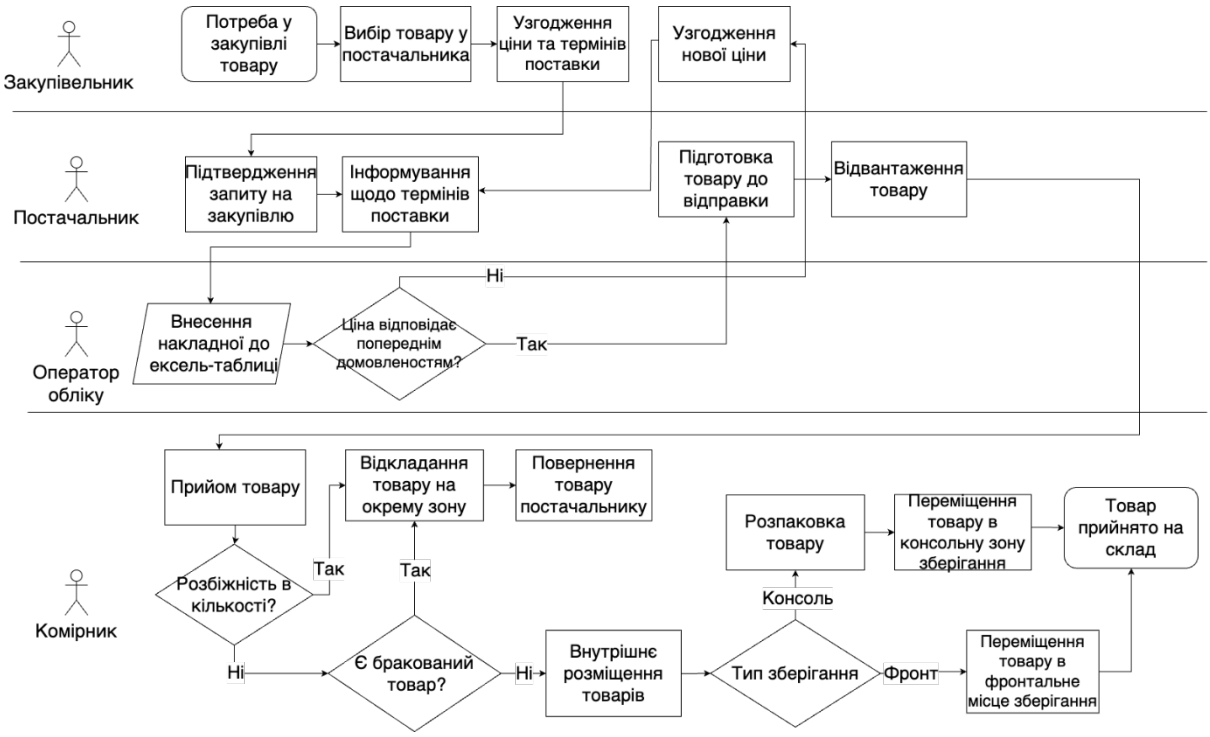


Рис. 2.1. Процес надходження товару до впровадження

На рисунку 2.1. зображена модель прийому товару на склад до впровадження системи. Коли в замовника виникає потреба закупівлі товару - відбувається вибір товару закупівельником у постачальника та узгодження ціни та термінів поставки з ним. Після узгодження постачальник підтверджує запиту на закупівлю та інформує щодо термінів поставки. Оператор обліку вносить накладну до ексель-таблиці та перевіряє, чи відповідає ціна попереднім домовленостям. Якщо ціна не відповідає домовленостям - закупівельник узгоджує нову ціну з постачальником.

У випадку, якщо ціна відповідає попереднім домовленостям - постачальник займається підготовкою та відправкою товару замовнику.

При прийомі товару комірник має перевірити розбіжності в кількостях. Якщо надійшла більша кількість товару - комірник відкладає її на окрему зону зберігання для подальшого повернення постачальнику.

Наступним кроком є перевірка на бракований товар. Якщо було виявлено бракований товар - його так само відкладають на окрему зону зберігання для подальшого повернення постачальнику. Якщо браку немає, то відбувається внутрішнє розміщення товару за типами зберігання (фронт або консоль), після переміщень товарів на відповідні зони - товар прийнято на склад.

Також був визначений процес відвантаження товару зі складу.

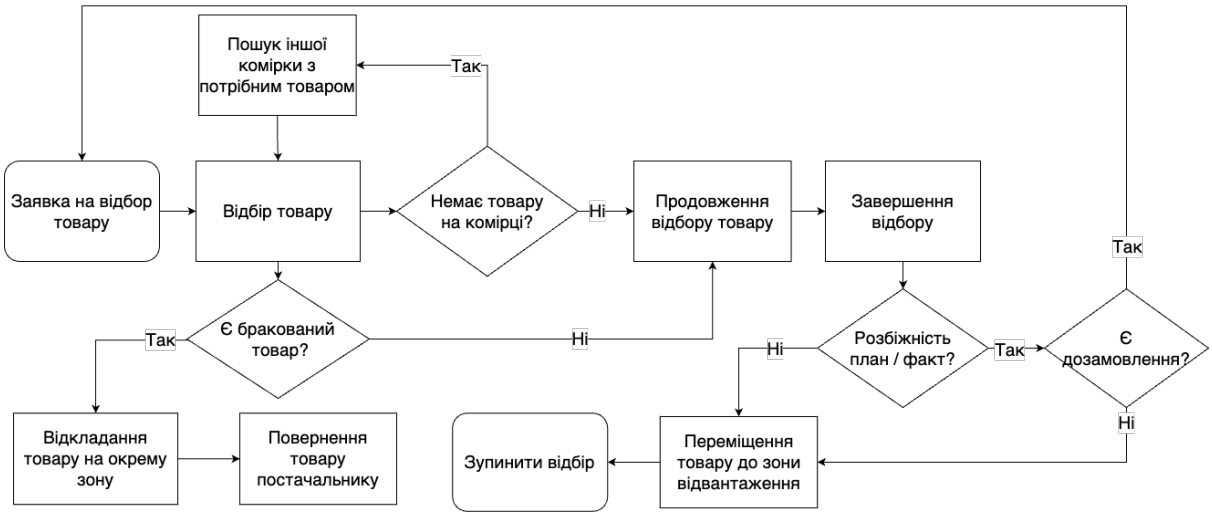


Рис. 2.2. Процес відвантаження товару до впровадження

На рисунку 2.2. зображений процес відвантаження товару зі складу. При створенні заявки починається відбір товару по коміркам. У випадку, якщо комірник не бачить товару на потрібній комірці - відбувається пошук іншої комірки з потрібним товаром.

Також, якщо при відборі товару на зоні зберігання був знайдений бракований товар - його переміщують на окрему зону зберігання для подальшого повернення постачальнику.

При завершенні відбору, якщо є розбіжність і плановій та фактичній кількості відібраного товару - комірник вказує чи буде додаткове відвантаження товару.

Якщо так - цикл повторюється заново, якщо ні - переміщується товар до зони відвантаження та зупиняється відбір.

Провівши аналіз бізнес-процесу клієнту було запропоновано модифікувати процес прийому та відвантаження товару зі складу для збільшення ефективності низки показників.

Також були визначені основні показники ефективності в компанії:

- час передачі замовлення на склад:

На даному етапі менеджер з продажів самостійно інформує комірників про нову заявку на відвантаження товару. Враховуючи завантаженість іншими задачами - це може займати до 15 хвилин. В свою чергу комірник має зафіксувати заявку самостійно, для того, щоб не упустити деталі замовлення.

- час відбору товару:

Комірнику потрібен час для того, щоб знайти потрібний товар на складі, так як облік зон та місць зберігання на момент впровадження не ведеться. Комірники приблизно пам’ятають, на які місця зберігання були переміщені товари для зберігання.

- відсоток оброблених замовлень:

У разі великої кількості замовлень (наприклад, в акційні дні чи напередодні свят) комірники не завжди своєчасно та якісно можуть виконувати свою роботу, враховуючи велику завантаженість менеджерів та комірників.

- кількість механічних помилок:

Так як немає обліку в єдиній системі, можуть виникати помилки при передачі заявки на відбір менеджером на склад. Комірник також може робити механічні помилики, такі як переплутати кількість товару, місце зберігання, плановий час відвантаження товару.

- відсоток задоволених клієнтів:

Через низку факторів (час передачі замовлень на склад, час відбору товару та ін.) в роботі компанії можуть відбуватись затримки у видачі замовленні клієнту. Ці фактори впливають на задоволеність клієнтів.

Враховуючи всі вище зазначені фактори було прийнято рішення оптимізувати складський облік на базі єдиної системи управління складом.

2.2 Розробка моделі та методів оптимізації складського управління Odoo

2.2.1 Опис складських зон

На складі замовника на момент впровадження буде реалізовано поділ складу на складські зони, виходячи з їхнього функціонального призначення.

- зона прийому:
- зона зберігання (палетні стелажі);
- зона зберігання (консольні стелажі);
- зона комплектації;
- зона бракованого товару;
- зона втраченого товару (віртуальна);
- документ для приймання товару;
- документ для відвантаження товару.

Також планується оперувати кількома типами місць зберігання залежно від типу стелажного обладнання, що використовується, або без використання такого. Основна інформація, записана в системі про фізичні властивості місця - це:

• габарити місця;
• вантажопідйомність місця;
• тип (зберігання або комплектація) ;
• основні логічні властивості місця, записані в системі: чи є місце активним осередком відбору;
• чи є місцем зберігання:

Кожне фізичне місце складу, на якому буде складуватися товар, має бути внесене до системи WMS. На складі компанії планується використовувати стандартне маркування складських місць:

FF.XX.YY.Z.PP, де

FF - назва складської зони,

XX- ряд (стелаж),

YY - секція,

Z - рівень,

PP - місце.

Приклад: I5-F-06-04-03

Кожне місце має бути промарковане етикеткою відповідного місця.

2.2.2 Прийом товару на склад

Призначенням цього процесу є ідентифікація товару, внесення всіх необхідних логістичних характеристик у WMS, прийом, перевірка (за кількістю та якістю) та оприбуткування товару на склад.

Вхідною умовою процесу «Приймання» є прибуття поставки на склад та відповідний документ в системі.

Результатом цього процесу є:

- товар, що пройшов вхідний контроль, прийнято на склад;
- відбракований товар прийнято складу в зону шлюбу;
- нестачі відображені відповідним актом;
- факт приймання товару зареєстрований у WMS та вивантажений в ERP (автоматично за налаштуваннями інтеграції);
- формування та оформлення документу приймання (у тому числі акти розбіжностей та фото).

Запропонований процес прийому товару на склад зображений на рисунку 2.3.

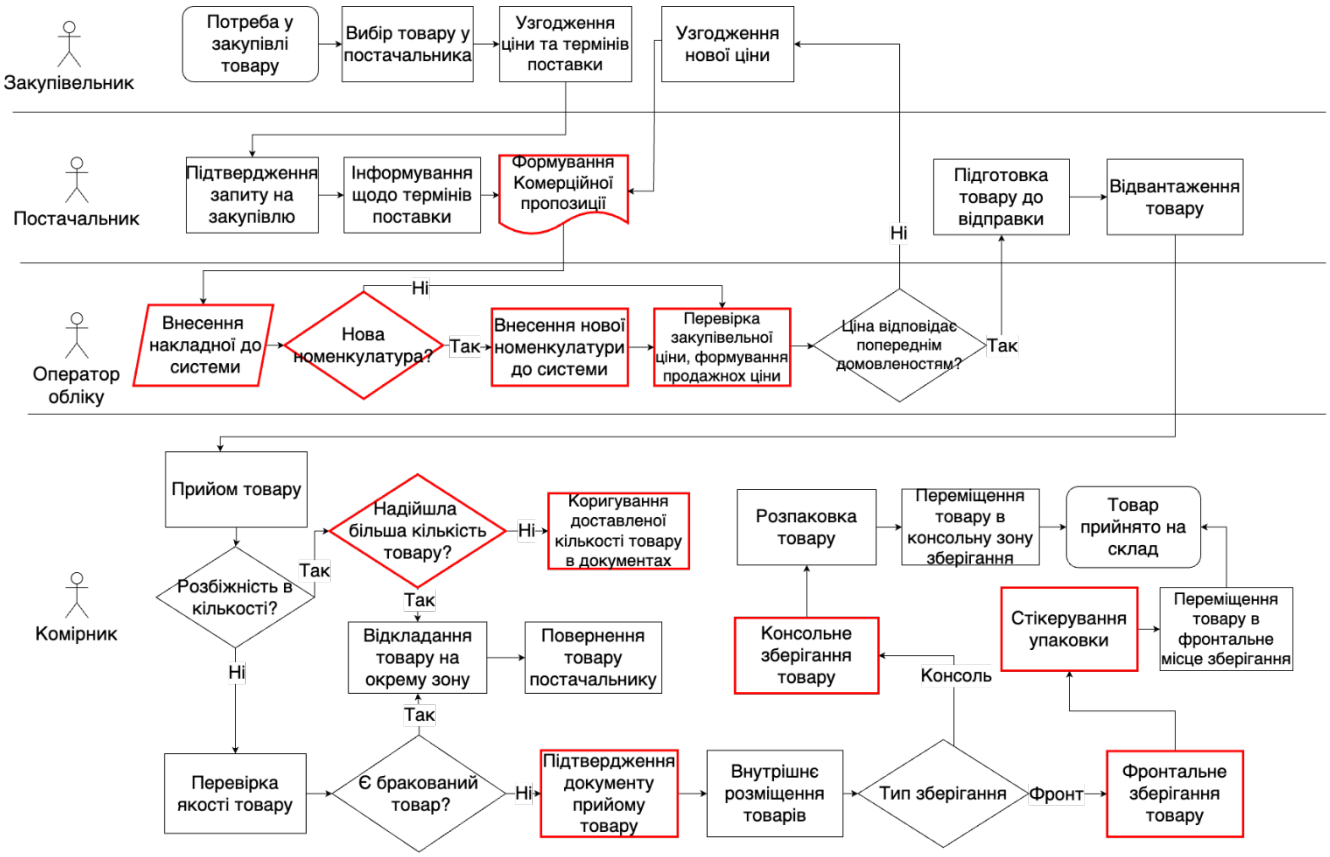


Рис. 2.3. Модифікований алгоритм обліку процесу надходження товару на склад

Коли в замовника виникає потреба закупівлі товару - відбувається вибір товару закупівельником у постачальника та узгодження ціни та термінів поставки з ним. Після узгодження постачальник підтверджує запиту на закупівлю та інформує щодо термінів поставки, після чого створюється документ комерційної пропозиції в системі. Оператор обліку вносить накладну до системи та перевіряє, чи відповідає ціна попереднім домовленостям. Якщо ціна не відповідає домовленостям - закупівельник узгоджує нову ціну з постачальником. Також оператор обліку перевіряє чи є дана номенкулатура в системі та при потребі створює її.

У випадку, якщо ціна відповідає попереднім домовленостям - постачальник займається підготовкою та відправкою товару замовнику.

При прийомі товару комірник має перевірити розбіжності в кількостях. Якщо надійшла більша кількість товару - комірник відкладає її на окрему зону зберігання

для подальшого повернення постачальнику та коригує кількість товару в документі прийому.

Наступним кроком є перевірка на бракований товар. Якщо було виявлено бракований товар - його так само відкладають на окрему зону зберігання для подальшого повернення постачальнику. Якщо браку немає, то комірнику потрібно підтвердити документ прийому товару, після чого відбувається внутрішнє розміщення товару за типами зберігання (фронт або консоль). Якщо товар з консольним типом зберігання - потрібно простікерувати пакування. Після переміщень товарів на відповідні зони - товар прийнято на склад.

2.2.3 Розміщення товару

Призначенням цього процесу є виділення місць зберігання для сформованих на етапі приймання носіїв (піддонів) з товаром, розміщення на підібрані місця зберігання, фіксація результату у системі.

Вхідною умовою процесу «Розміщення» є запакований на етапі «Приймання» складський носій. Весь носій розміщується повністю, перевірок на кількість найменувань на палеті немає.

Результатом цього процесу є прийнятий товар розміщено на підібрану системою комірку зберігання згідно з правилами та стратегіями розміщення.

Алгоритм виконання процесу зі сторони системи:

1. За запитом користувача сформувати список вільних осередків;
2. Відсортувати комірки за зменшенням обсягу, а потім за зростанням коду сортування;
3. Розмістити товар у комірку з найменшою різницею між обсягом комірки та обсягом товару, що розміщується, першу зі списку вільних осередків, відсортованого за індексами за зростанням;
4. Якщо товару розміщення більше, ніж міститься у комірку, то нерозміщеного залишку алгоритм пошуку повторюється;
5. Якщо один з лінійних розмірів прийнятого товару перевищує 2000 мм. - підбирати комірку для розміщення в групі осередків «консольні стелажі» ;

6. За фактом підтвердження розміщення завершити завдання.

Алгоритм виконання процесу на стороні користувача:

1. За фактом завершення приймання першого піддону з продукцією користувач може розпочати розміщення;
2. Користувач на ТЗД заходить у режим розміщення та сканує штрих-код першого піддону;
3. За фактом сканування відбувається запит у систему управління складським обліком і за вищеописаним алгоритмом система намагається знайти відповідну для розміщення комірку;
4. Якщо з якоїсь причини підібрати комірку не вдалося, на екрані ТЗД буде виведено інформаційне повідомлення про неможливість підібрати комірку для поточного складського носія;
5. Якщо підібрати комірку не вдалося, користувачеві необхідно звернутися до оператора складу. Оператор складу за фактом звернення має виконати перевірку наявності вільних місць за системою, а також виконати перевірку ваги та габаритів продукції, у разі виявлення помилок, виправити;
6. Якщо використання механізму автоматичного підбору осередків не доцільно (наприклад, у разі розміщення продуктових візків та/або великогабаритних вітринних холодильників) у користувача є можливість виконати розміщення вручну, підібравши комірку на свій розсуд. Для цього користувач повинен натиснути кнопку «Розмістити вручну» ;
7. Попередньо, в системі будуть створені осередки в проїздах між стелажами, наприклад, осередок в проїзді між стелажом А і В, в топології складу буде заведено як осередок АВ. Такий клас осередків буде умовно безрозмірним і не матиме обмеження за кількістю носіїв, які в таку комірку можна розмістити. Даний клас осередків буде призначений для розміщення негабаритних товарів і, якщо основні місця зберігання будуть зайняті, заблоковані та розміщення на них буде неможливим;

8. Якщо підбір комірки пройшов без зауважень, користувачеві на ТЗД відкриється наступна екранна форма із зазначенням комірки одержувача для поточного носія;
9. Користувач переміщає складський носій до зазначеної комірки та виконує сканування штрих-код комірки одержувача, тим самим завершуючи транспортне доручення та фіксуючи в системі, що всі товари на даному носії розміщені в даному осередку;
10. Якщо замість комірки одержувача відсканована штрих-код іншого комірки, система видасть повідомлення на екран ТЗД з текстом про те, що відсканована комірка не є коміркою одержувачем для поточного носія, що розміщується. Користувачеві необхідно перевірити ще раз правильність введення штрих-код;
11. У разі якщо при розміщенні в осередку одержувача виявлено інший складський носій з товаром, користувачеві необхідно витягти цей носій з осередку, а на його місце розмітити свій носій;
12. Вилучений носій необхідно перемістити в зону зберігання носіїв до з'ясування, з більшою часткою ймовірності, витягнутий носій був розміщений в комірку помилково і по цьому товару є недостача, необхідно виконати точкову інвентаризацію;
13. Факт розміщення палети автоматично завершує завдання розміщення.

Місця на складі не будуть поділятися на місця зберігання та відбору, відповідно розміщення може відбуватися в будь-яке вільне місце.

2.2.4 Відбір товару

Призначенням цього процесу є відбір товару згідно з заявкою від клієнта або між складським переміщенням. Вхідною умовою є отримання замовлень на відвантаження та/або переміщень між складами з ERP. Результатом процесу є зібраний та упакований товар на замовлення готовий до відвантаження.

На рисунку 2.4. зображений модифікований алгоритм відбору товару:

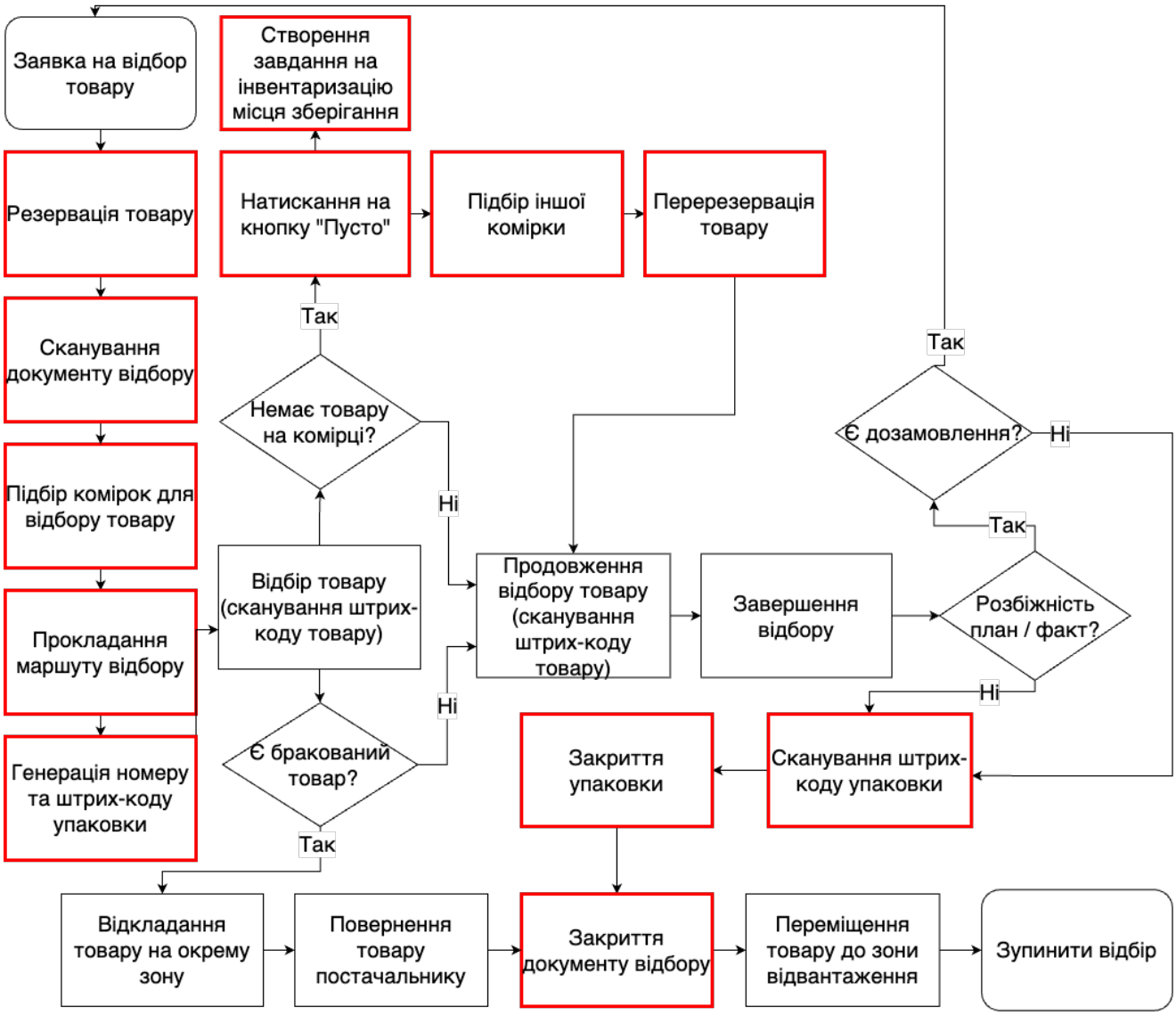


Рис. 2.4. Модифікований процес алгоритму відбору товару

При створенні заявки система повинна зарезервувати необхідну кількість товару на відповідних місцях зберігання. Відсканувавши документ відбору, система автоматично повинна підібрати потрібні комірки та прокласти маршрут відбору, по якому буде виконувати завдання комірник. Йому також потрібно згенерувати назву та штрих-код пакування для подальшого контролю переміщень товару в системі.

Якщо при відборі товару на зоні зберігання був знайдений бракований товар - його переміщують на окрему зону зберігання для подальшого повернення постачальнику.

При відборі товару, якщо комірник не знайшов на запропонованому системою місці даного товару - він натискає на кнопку «Пусто» в ТЗД для того, щоб система створила завдання на інвентаризацію даного місця та запропонувала інше місцезнаходження товару. В той же час система має перерезервувати товару у відповідній комірці.

При завершенні відбору, якщо є розбіжність і плановій та фактичній кількості відібраного товару - комірник вказує чи буде додаткове відвантаження товару. Якщо так - цикл повторюється заново, якщо ні, то сканує штрих-код пакування для її закриття, а також закриває документ відбору. Таким чином переміщується товар до зони відвантаження та зупиняється відбір.

2.2.5 Переміщення товару

Призначенням цього процесу є управління переміщенням товарів на складських носіях між місцями зберігання, фіксація результату у системі. Вхідною умовою процесу є розміщений на місцях зберігання товар, виникнення потреби у переміщенні. Результатом цього процесу є:

Товар переміщений між осередками, результат відображено у системі. Далі будуть розглянуті 2 варіанти створення переміщень, з ініціативи оператора, на запит з ТЗД співробітника складу.

Алгоритм виконання процесу з ініціативи оператора

1. Оператор складу створює звіт (Звіт повинен передбачати сортування даних по будь-якій колонці звіту) щодо утилізації обсягу осередків, в якому знаходиться інформація;
2. Адреса осередку (у звіті зазначаються всі осередки зберігання та комплектації, у тому числі порожні), товар, який зберігається в осередку, кількість товару, що зберігається в осередку;
3. Обсяг товару в осередку, обсяг осередку, к-т утилізації складського обсягу;
4. Оператор аналізує звіт та планує переміщення товару за логікою;
5. Якщо виявлено той самий товар у різних осередках, то логічно його поєднати в одну осередок за умови, що це не заборонено налаштуваннями;

6. Товар з великих осередків з к-том утилізації складського обсягу <50% - переміщається в осередки меншого розміру;
7. Оператор в інтерфейсі на ПК створює переміщення та вказує:
    - осередок джерело, з якого потрібно забрати товар;
    - товар, який необхідно перемістити та його кількість;
    - осередок приймач, до якого необхідно перемістити товар.

    Якщо місці зберігання числиться більше одного артикула, і користувач вибере лише одне артикул, система видасть попередження у тому, що у вибраному місці зберігання є ще товари і запропонує перемістити й інші товари.

    Такі завдання автоматично випадають на ТЗД працівникам складу.

8. Співробітник складу заходить у відповідне меню на ТЗД, тисне кнопку «отримати завдання». Після чого, якщо є доступні завдання, користувачу відкриється екранна форма з першим осередком джерелом;

9. Користувач переміщається до вказаної комірки джерела, сканує її штрих-код. Якщо з якоїсь причини сканування штрих-коду осередку неможливе (відсутня бирка, бірка пошкоджена), користувач може ввести адресу осередку джерела вручну. Після завершення завдання користувач повідомляє оператора про проблемну комірку, оператор виконує друк нової бирки на комірку, передає її відповідальному для перестикерування місця;

10. Далі користувачу стає доступна форма для введення штрих-коду товару, а також на екрані ТЗД буде відображено найменування товару, який необхідно відібрати з поточного осередку;

11. Після сканування штрих-коду товару відкривається поле для введення кількості;

12. Якщо ШК товару введено неправильно або введено не штрих-код товару, на екрані ТЗД з'явиться відповідне повідомлення, користувачеві необхідно перевірити ще раз правильність введення штрих-коду товару. Якщо штрих-код відсутній або пошкоджений, користувач має можливість ввести код артикула руками (код відображатиметься на екрані ТЗД);

13. Якщо на етапі створення переміщень оператор вибрав переміщення цілого піддону з усім його вмістом, користувач на ТЗД може відсканувати не штрих-код товару, а штрих-код складського носія. У цьому етап введення кіл-ва буде пропущений, і система перемістить весь носій разом із усім кіл-вом всіх найменувань. У разі якщо на піддоні відсутня штрих-код носія, тоді користувачеві необхідно відсканувати всі штрих-код товару, що переміщається і по кожному з них ввести кількість;

14. Після введення кількості товару або штрих-коду носія, на екрані ТЗД буде відображено осередок одержувач;

15. Користувач переміщає піддон до зазначеного осередку одержувача, сканує її штрих-коду. Якщо з якоїсь причини сканування штрих-коду осередку неможливе (відсутня бирка, бірка пошкоджена),користувач може ввести адресу осередку одержувача вручну. Якщо не весь товар можна розмістити на комірку одержувач, у користувача буде можливість скасувати поточне переміщення повністю, натиснувши кнопку скасувати переміщення, тоді на екрані з'явиться інформація про початкову комірку, на яку необхідно повернути товар, користувач повертає товар на первісну комірку і сканує її штрих-код.

Або розмістити частину товару, у такому разі користувачеві необхідно буде натиснути кнопку «розмістити частково», відсканувати штрих-код товару, який розміщується, підтвердити кількість, а залишок повернути на комірку джерело;

16. Після розміщення останнього піддона завдання завершується автоматично, товар значиться на новому осередку;

17. Для отримання наступного завдання користувач знову натискає кнопку "отримати завдання". якщо завдань більше немає, на екрані з'явиться повідомлення про те, що завдань більше немає.

Алгоритм виконання процесу зі сторони працівника складу на ТЗД

1. Співробітник складу заходить у відповідне меню на ТЗД;
2. Сканує осередок джерело, з якого потрібно перемістити товар. Якщо з якоїсь причини сканування штрих-коду осередку неможливе (відсутня бирка, бірка пошкоджена),користувач може ввести адресу осередку джерела вручну;
3. Сканує товар, який слід перемістити;
4. Вводить необхідну кількість товару для переміщення та підтверджує взяття зазначеної кількості товару;
5. Далі система пропонує вказати кінцевий осередок для переміщення;
6. Співробітник сканує кінцевий осередок, при цьому система робиться такі перевірки:
    - чи є в осередку інший товар;
    - якщо є, робить перевірку, чи можна поєднувати товари;
    - якщо немає забороняючих факторів, то товар переміщається до зазначеного осередку;
    - при цьому перевірка розмірів та ваги товару на місткість осередку не проводиться.
7. Після виконаних дій товар переміщається з початкового осередку до кінцевої. При необхідності можна переміщати товар у комірку з таким же товаром. В іншому випадку на ТЗД має відображатися помилка із забороною на цю дію.

2.2.6 Страховий запас та резервація товарів

Прорахунок страхового запасу товарів на складі необхідний для забезпечення стійкості постачання та мінімізації ризиків у випадках затримок або змін на ринку. Цей запас допомагає захистити компанію від несподіваних подій і забезпечити гнучкість в управлінні запасами.

Також він визначає оптимальний баланс між витратами та готовністю до змін у ланцюжку постачання, сприяючи уникненню втрат від відсутності товарів і

оптимізації витрат на запаси. Такий підхід допомагає компаніям забезпечити ефективність та стабільність у їхній діяльності.

Формула прорахунку страхового запасу товарів:

Safety stock = (Maximum Daily Demand x Maximum Lead Time in days) – (Minimum Daily Demand x Average Lead Time in days) (2.1)

Формула розрахунку кількості товарів на складі, при якому потрібно робити наступне замовлення на закупівлю:

ROP = (Average Daily Demand x Average Lead Time in days) + Safety Stock (2.2)

Де,

Maximum Daily Demand - максимальна кількість продажів в день,

Average Daily Demand – середня кількість продажів в день,

Minimum Daily Demand - мінімальна кількість продажів в день,

Maximum Lead Time in days – максимальний час поставки товару,

Average Lead Time in days – середній час поставки товару,

Minimum Lead Time in days – мінімальний час поставки товару.

Налаштування автоматичної резервації товарів в системі Odoo є важливим етапом у забезпеченні ефективного управління складськими процесами. Ця функціональність вирішує ключові завдання, спрямовані на оптимізацію та забезпечення точності управління запасами на складі.

Перш за все, автоматична резервація гарантує наявність товарів для обробки замовлень, сприяючи швидкому та ефективному виконанню клієнтських замовлень. Це також допомагає уникнути ризику перепродажу, забезпечуючи, що один і той самий товар не буде резервований для декількох замовлень.

Додатково, налаштування автоматичної резервації сприяє оптимізації управління запасами, дозволяючи системі ефективно використовувати наявні ресурси та уникати ситуацій, коли товари є на складі, але не враховуються у замовленнях.

Ще однією важливою перевагою є можливість системи реагувати на зміни в попиті швидше та ефективніше. Завдяки автоматичній резервації, система Odoo може швидко адаптувати резервування товарів до нових замовлень або змін у вже існуючих замовленнях.

Додатково, ця функція сприяє підвищенню точності інвентаризації, оскільки система автоматично відстежує зміни в кількості резервованих товарів. Це полегшує процес інвентаризації та забезпечує більш точний облік наявності товарів на складі.

Загалом, налаштування автоматичної резервації товарів в системі Odoo є стратегічно важливим елементом для підвищення ефективності управління запасами та оптимізації процесів виконання замовлень на підприємстві.

Тому було запропоновано ввести формальну модель правил резервації товарів на складі:

1) Мінімальна дельта ([<тип пакування>] [(резервація <обсяг>)] <операція> замовлена кільк.);
2) Замовлена кільк. <операція> кільк. на зберіганні ([<тип пакування>] [(резервація <обсяг>)] <операція> замовлена кільк.)).

Тобто, резервація товарів може відбуватись по двом моделям.

Перша - це різниця між замовленою кількістю та кількістю товарів в пакуванні (коробка, палета, мішок) резервується кількість, яка більша / більша або дорівнює / менша / менша або дорівнює / дорівнює замовленій кількості.

Друга – це якщо замовлена кількість більша / більша або дорівнює / менша / менша або дорівнює / дорівнює кількості на зберіганні у відповідному типі пакуванні (коробка, палета, мішок), то резервується обсяг, який більший / більший або дорівнює / менший / менший або дорівнює / дорівнює замовленій кількості.

3 РОЗРОБКА ТА ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Ініціалізація та налаштування проекту

На етапі початку розробки важливо провести процес установки всіх необхідних компонентів, щоб забезпечити ефективний робочий процес. Першим кроком є завантаження та встановлення інтегрованого середовища розробки (IDE) PyCharm, розробленого компанією JetBrains. Це інструментарій, який надає зручне та інтуїтивно зрозуміле середовище для написання коду на мові програмування Python. Для цього варто відвідати офіційний веб-сайт розробника та завантажити безкоштовну версію Community Edition.

Далі, для створення надійної основи даних для системи Odoo, необхідно встановити базу даних Postgresql SQL. Цей крок є ключовим, оскільки база даних виступає основним сховищем інформації для Odoo. Відповідну версію бази даних можна завантажити з офіційного веб-сайту Postgresql, враховуючи операційну систему користувача.

Щоб розпочати роботу з Odoo, важливо скопіювати проект з офіційного репозиторію GitHub Odoo. Рекомендовано використовувати найновішу версію 17.0 Enterprise Edition, що можна здійснити через клонування репозиторію за допомогою Git або завантаження архівного файлу з GitHub та подальше розпакування його.

Крім цього, важливим етапом є встановлення всіх необхідних залежностей, використовуючи команду `pip install -r requirements.txt`. Для запуску локального сервера у вікні конфігурації проекту потрібно вказати параметр `-c` та вказати шлях до файлу конфігурації.

Усі ці кроки взаємодії створюють оптимальні умови для ефективного розроблення та впровадження проекту на основі Odoo, забезпечуючи гнучкість та готовність до подальших змін і вдосконалень.

3.2 Розробка основної частини системи

Важливим етапом є старт власного модуля, який розширює можливості базового модулю Склад в системі Odoo. Для цього потрібно створити директорію з відповідним ім'ям модулю, у цьому випадку вона отримає назву "diploma_wms". У цій директорії слід створити основні файли та додаткові папки.

Основні файли, які необхідно створити, включають:

1. __init__.py. Цей файл залишається порожнім і використовується для подальшої ініціалізації модуля.

2. __manifest__.py. Головний файл, що описує модуль. Тут потрібно вказати основну інформацію, таку як назва, залежності від інших модулів, версія, автор, опис та ін. Можна також вказати необхідні ресурси, такі як: CSS-стилі, зображення, файли перекладу, тощо.

3. models. Ця папка призначена для визначення моделей даних вашого модуля. За допомогою цієї папки є змога створити нові моделі або розширити існуючі моделі базового модулю Склад, визначаючи різні поля, методи, спостерігачі та інші атрибути, які потрібні для функціональності.

4. views. Тут зберігаються файли опису вигляду модуля у форматі XML. Вони визначають, звіти, списки, форми та інші елементи інтерфейсу користувача, пов'язані з відповідним модулем.

При необхідності можна створювати додаткові каталоги для зберігання файлів системної безпеки, JS, CSS та файлів перекладу. Важливо враховувати, що в Odoo файли безпеки є ключовим елементом, оскільки вони містять конфіденційну, особисту або важливу інформацію, яку необхідно захищати від несанкціонованого доступу та інших потенційних загроз.

3.3 Розробка front-end частини

Розробка нових відображень у системі Odoo через XML виявляється вельми потужним інструментом, що відкриває широкі можливості для творчості та налаштування функціоналу. Використання XML дозволяє не лише створювати

кастомні інтерфейси, але й вносити зміни у вигляд і поведінку системи, адаптуючи її під конкретні вимоги бізнес-процесів.

У контексті кастомізації вигляду, використання вбудованих CSS-стилів в XML-файлах надає можливість зручно задавати параметри вигляду створених елементів. Це включає в себе кольори, розміри, шрифти та інші стилізаційні аспекти, що роблять інтерфейс більш привабливим та відповідним конкретним дизайнерським вимогам.

Також, використання Python-коду в XML-файлах дає можливість вбудовувати бізнес-логіку та реалізовувати динамічні зміни у відображенні. Це означає, що в подальшому можна виконувати розрахунки, проводити валідацію даних, взаємодіяти з базою даних та впроваджувати інші функціональності, що робить відображення більш гнучкими та пристосованими до конкретних умов.

Для ефективного використання ваших нових відображень в системі Odoo, слід встановити їх у вигляді модуля та активувати їх у системі. Після цього є змога користуватися розробленими елементами інтерфейсу в різних модулях та взаємодіяти з ними відповідно до потрібної бізнес-логіки.

Створення нових відображень в системі Odoo через XML є гнучким і потужним інструментом для розробки індивідуальних інтерфейсів та налаштування системи під конкретні потреби. Це відкриває можливість змінювати та розширювати функціональність системи, забезпечуючи зручне та ефективне використання Odoo відповідно до потрібного бізнесу.

3.4 Огляд системи

3.4.1 Огляд інтерфейсу системи

Документ переміщення в Odoo є важливим інструментом для відстеження та контролю переміщення товарів між різними локаціями у системі, дозволяє точно відстежувати, коли та які товари переміщаються між різними локаціями в межах організації. Це стає важливим для забезпечення правильності обліку залишків та контролю над рухом товарів.

Також, документи переміщення сприяють організації процесу переміщення товарів та допомагають у плануванні оптимального використання ресурсів.

Вони дозволяють ефективно керувати запасами, оновлюючи інформацію про залишки товарів в реальному часі. Це важливо для уникнення дублювання або втрати товарів.

Документи переміщення взаємодіють з іншими аспектами системи, такими як замовлення на постачання, продажі, виробництво і т. д. Вони допомагають в реалізації складських та логістичних бізнес-процесів, створюють історію руху товарів, що може бути використана для аналізу та аудиту. Це дозволяє виявляти і виправляти помилки, вирішувати проблеми та вдосконалювати логістичні процеси. Він також взаємодіє з іншими модулями Odoo, такими як складський облік, управління закупівлями, продажами та іншими, що сприяє інтегрованому управлінню ресурсами.

Загалом, використання документів переміщення в Odoo є ефективним засобом оптимізації та контролю логістичних та складських операцій у бізнес-системі. На рисунку 3.1. зображений приклад складського документу в системі:

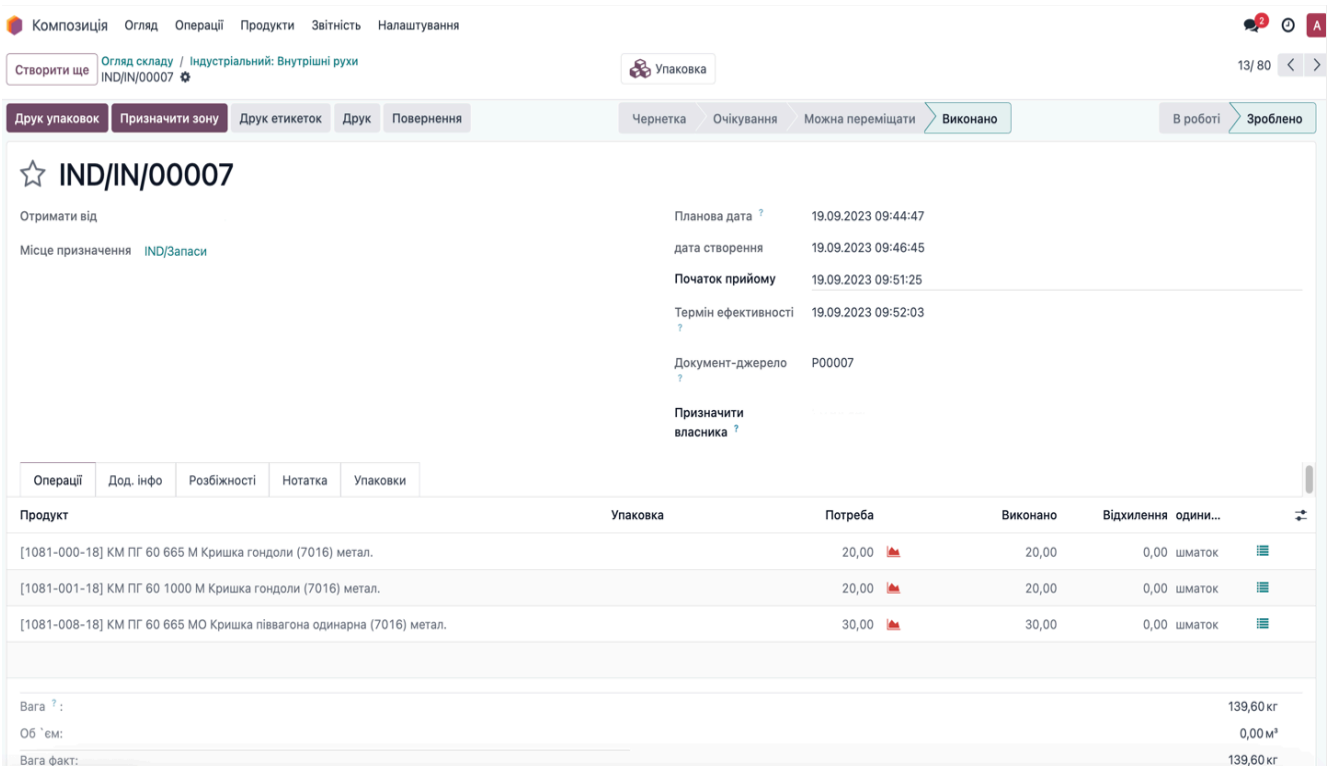


Рис. 3.1. Приклад складського документу в системі

У системі Odoo, картка товару є необхідною складовою для ефективного управління та контролю за товарами. Вона містить детальну інформацію про кожен конкретний товар, включаючи його назву, опис, артикул, одиницю виміру та інші характеристики. Це спрощує процес введення та зберігання даних про товари.

Картка товару допомагає в ефективному управлінні запасами, оскільки вона містить інформацію про кількість товару на складі, мінімальний та максимальний рівень запасів, що допомагає у плануванні закупівель та уникненні дефіциту чи перевищення запасів. На картці товару можна вказати ціну товару, знижки, податки та інші фінансові параметри. Це допомагає в обліку вартості товарів, а також у визначенні прибутковості. Вона зберігає історію руху товару, таку як приймання на склад, відвантаження, переміщення між складами тощо. Це надає можливість відстежувати всі етапи життєвого циклу товару, а також може включати аналітичні дані, такі як звіти про продажі, залишки на складі, рейтинги продуктів тощо, що допомагає приймати управлінські рішення на підставі зібраної інформації.

Загалом, картка товару в Odoo є центральним інструментом для комплексного управління товарами та забезпечення ефективності логістичних та складських процесів.

На рисунку 3.2. зображений вигляд картки товару в системі:

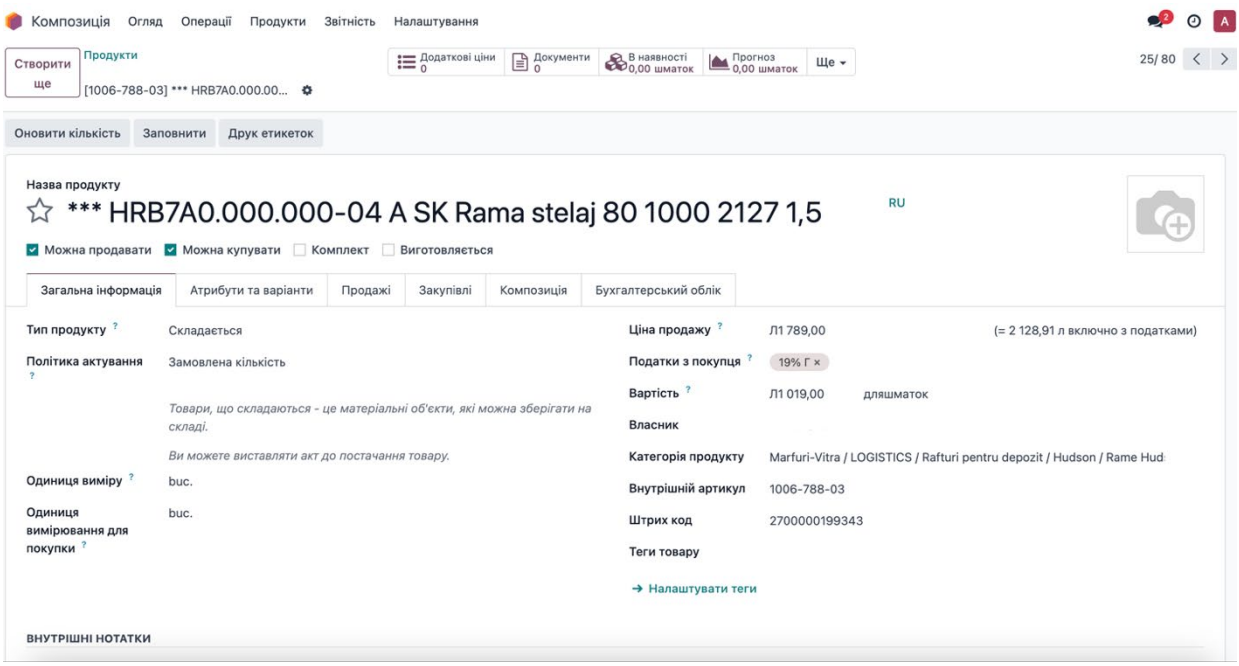


Рис. 3.2. Приклад інтерфейсу картки товару

Також, ключовим фактором роботи в системі є коректне введення місць зберігання товару та їх зонування для правильної побудови маршутів розміщення та відбору товарів. На рисунку 3.3. зображений приклад відображення місць зберігання в системі:

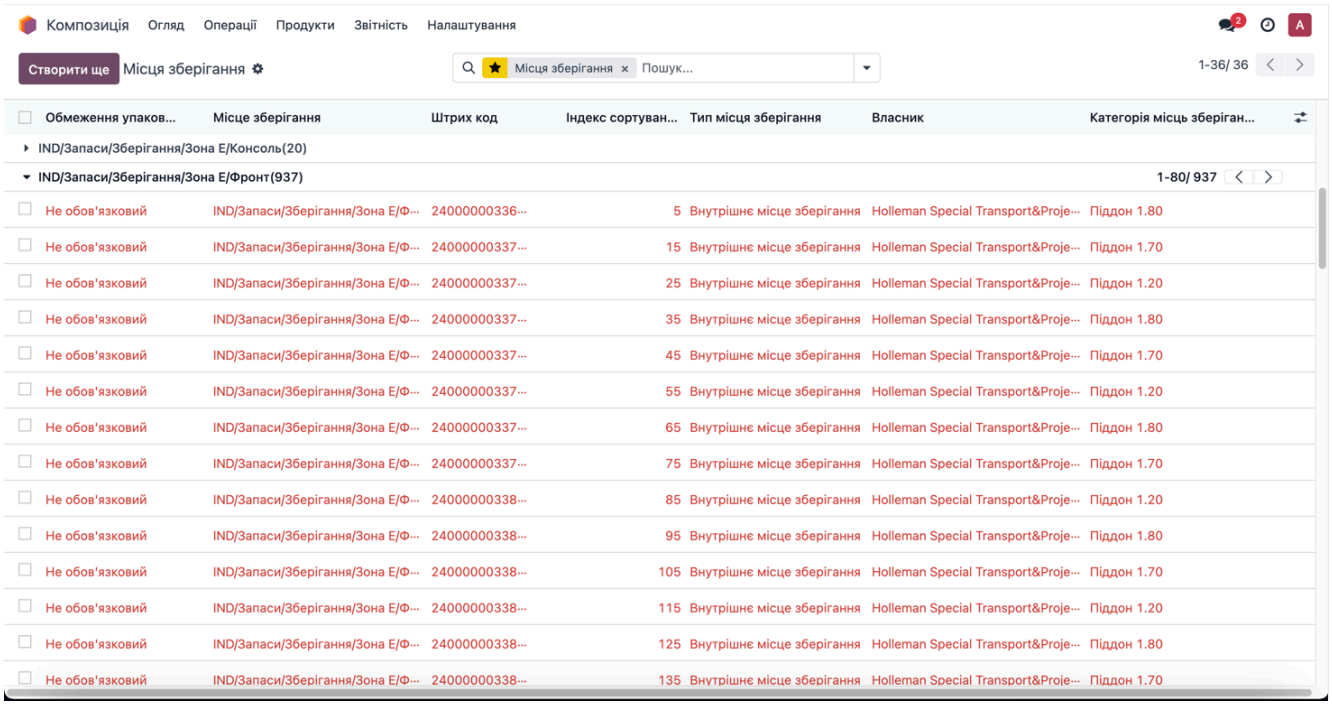


Рис. 3.3. Загальний список місць зберігання в системі

Всі місця зберігання поділені на зони фронтального та консольного зберігання. Для кожної комірки також прописаний штрих-код, індекс сортування та категорія місць зберігання. Обов’язковим для заповнення також є тип місця зберігання для кожної комірки.

Також були прописані правила резервації товарів, приклад яких зображено на рисунку 3.4.:

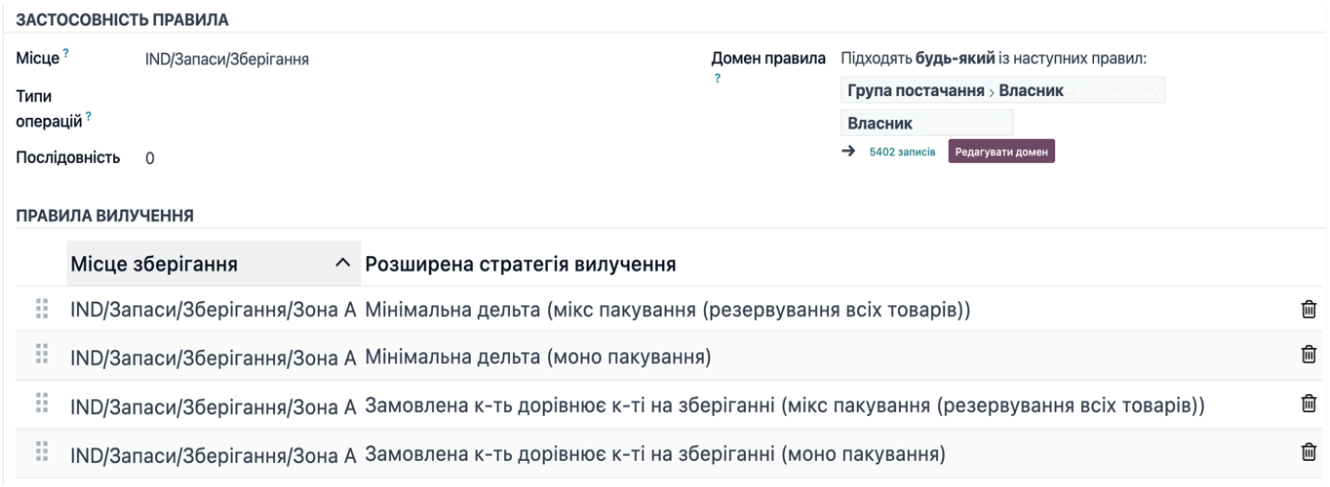


Рис. 3.4. Правила резервації товарів

Для прорахунку страхового запасу на момент впровадження був наданий доступ внесення певних показників вручну користувачем. Після користування системою більше ніж півроку - всі дані будуть прораховуватись автоматично. На рисунку 3.5 зображено інтерфейс прорахунку страхового запасу:

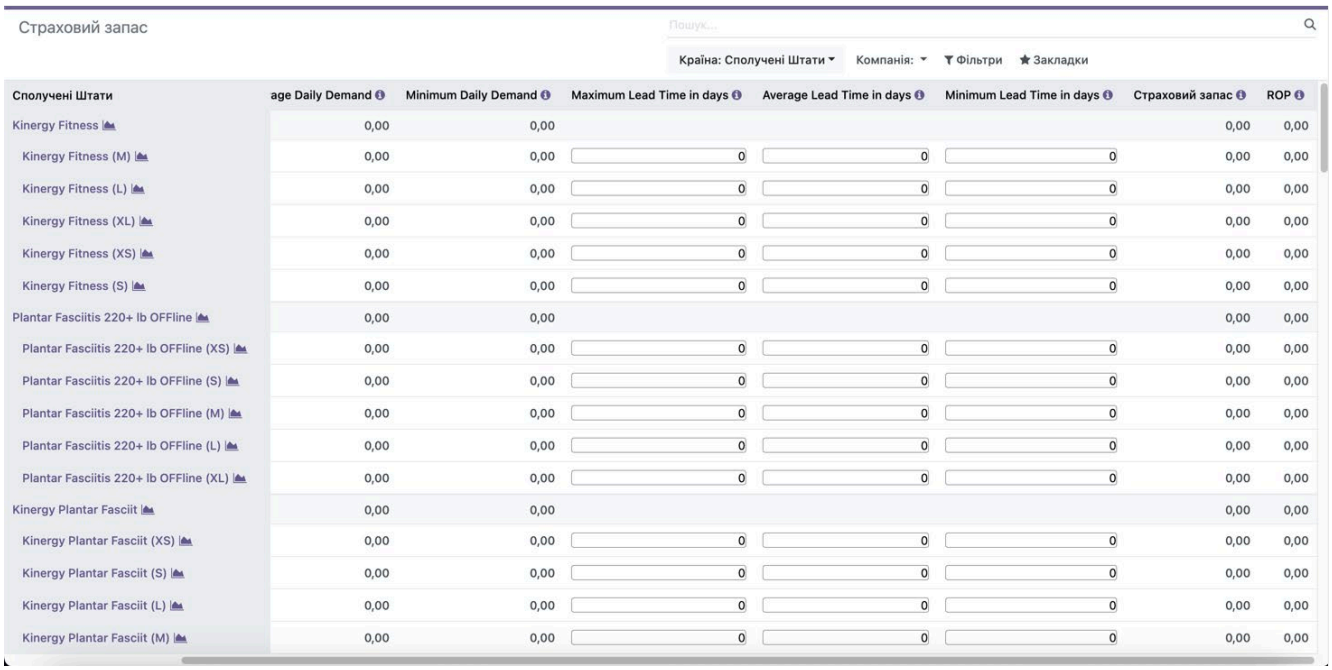


Рис. 3.5. Приклад інтерфейсу прорахунку страхового запасу

3.4.2 Огляд інтерфейсу ТЗД

Інтерфейс ТЗД включає в себе меню, яке складається з основних пунктів, таких як:

- прийом товару,
- відвантаження,
- розміщення,
- переміщення,
- відбір
- пакування
- сканування та ін.

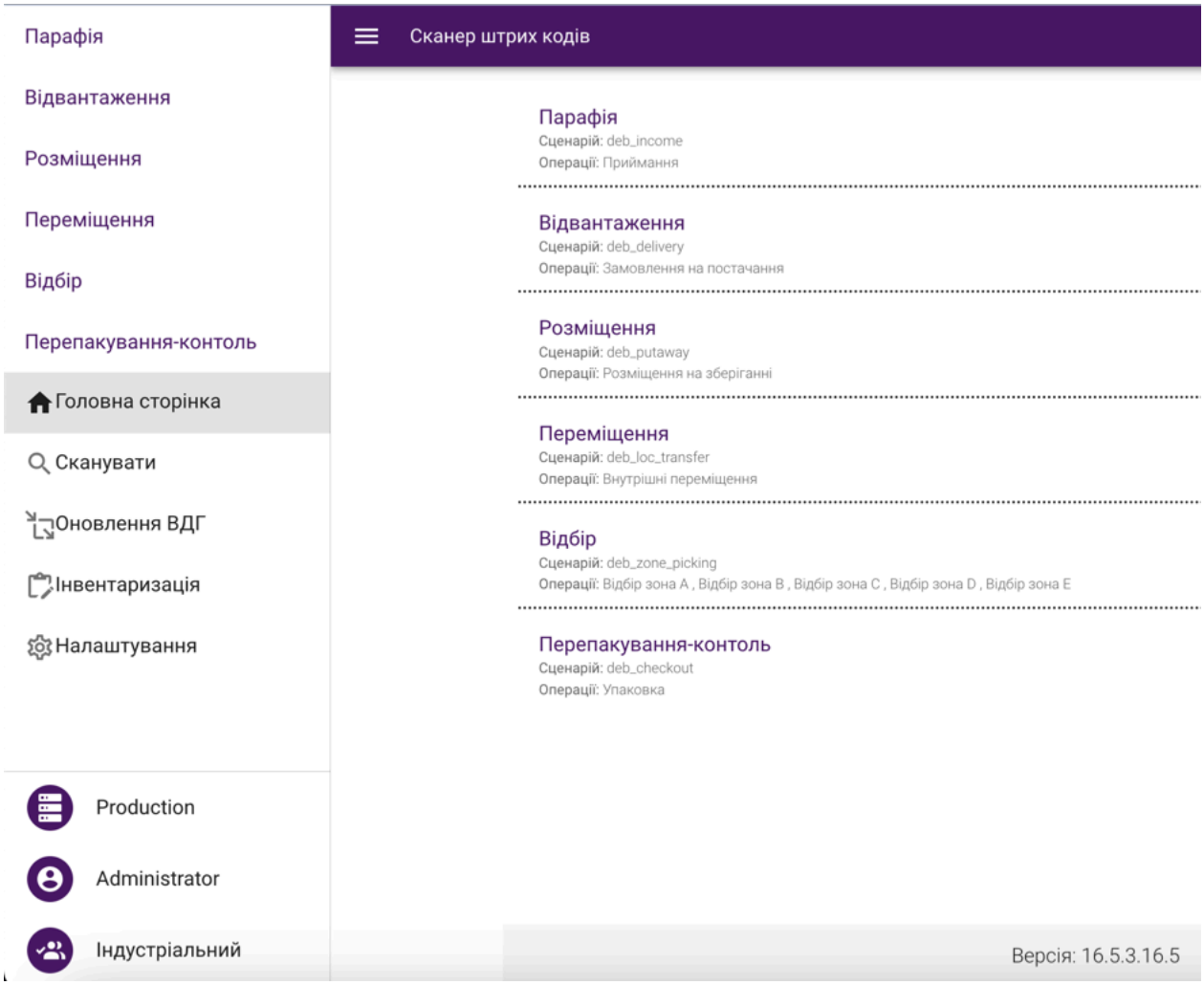


Рис. 3.6. Приклад інтерфейсу головного меню ТЗД

Після вибору операції потрібно сканувати відповідний документ, товар або пакунок, залежно від типу операції та процесу, який планується виконати.

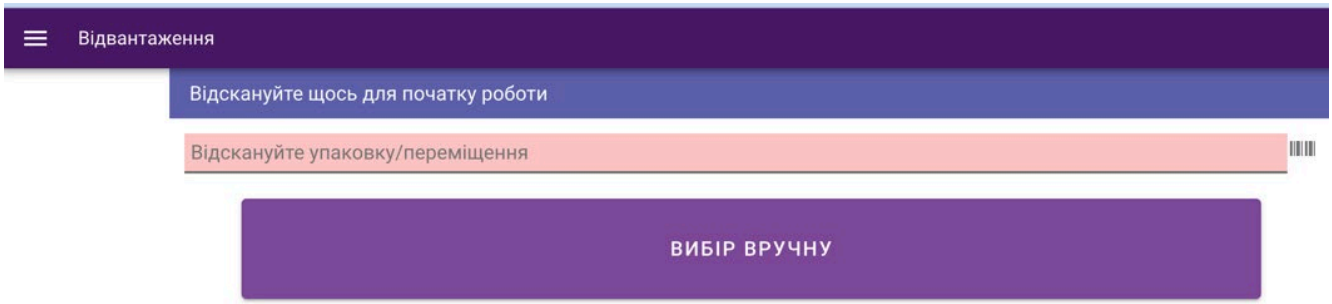


Рис. 3.7 Інтерфейс сканування

Після сканування відповідного штрих-коду, наприклад, штрих-коду документа для операції відвантаження товару – відкривається нове вікно, в якому є можливість виконати всі можливі дії в межах обраної операції.

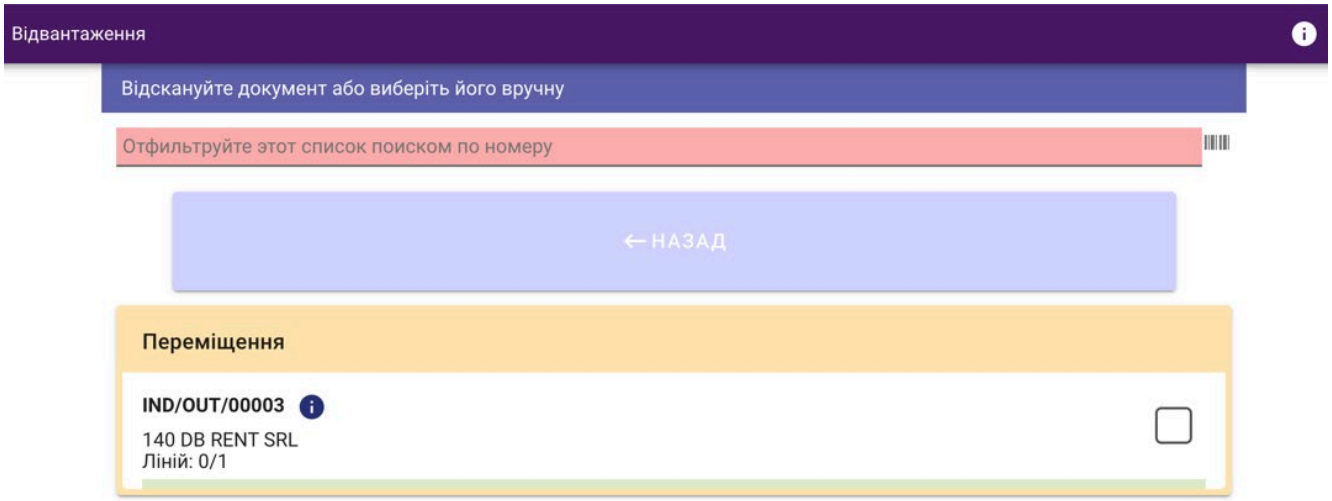


Рис. 3.8. Інтерфейс вибору документу відвантаження

В даному вікні можна переглянути інформацію про товар, який буде відвантажено, обробляти кількості та підтверджувати документ відвантаження.

Рис. 3.9. Приклад процесу відвантаження товару

За допомогою інтерфейсу ТЗД також є можливість сканувати товар або пакування для перегляду додаткової інформації, такої як:

- Вміст пакування
- Місце зберігання
- Кількість в наявності
- Зарезервована кількість
- Вага
- Тип зберігання

На рисунку 3.10. зображений приклад сканування товару:

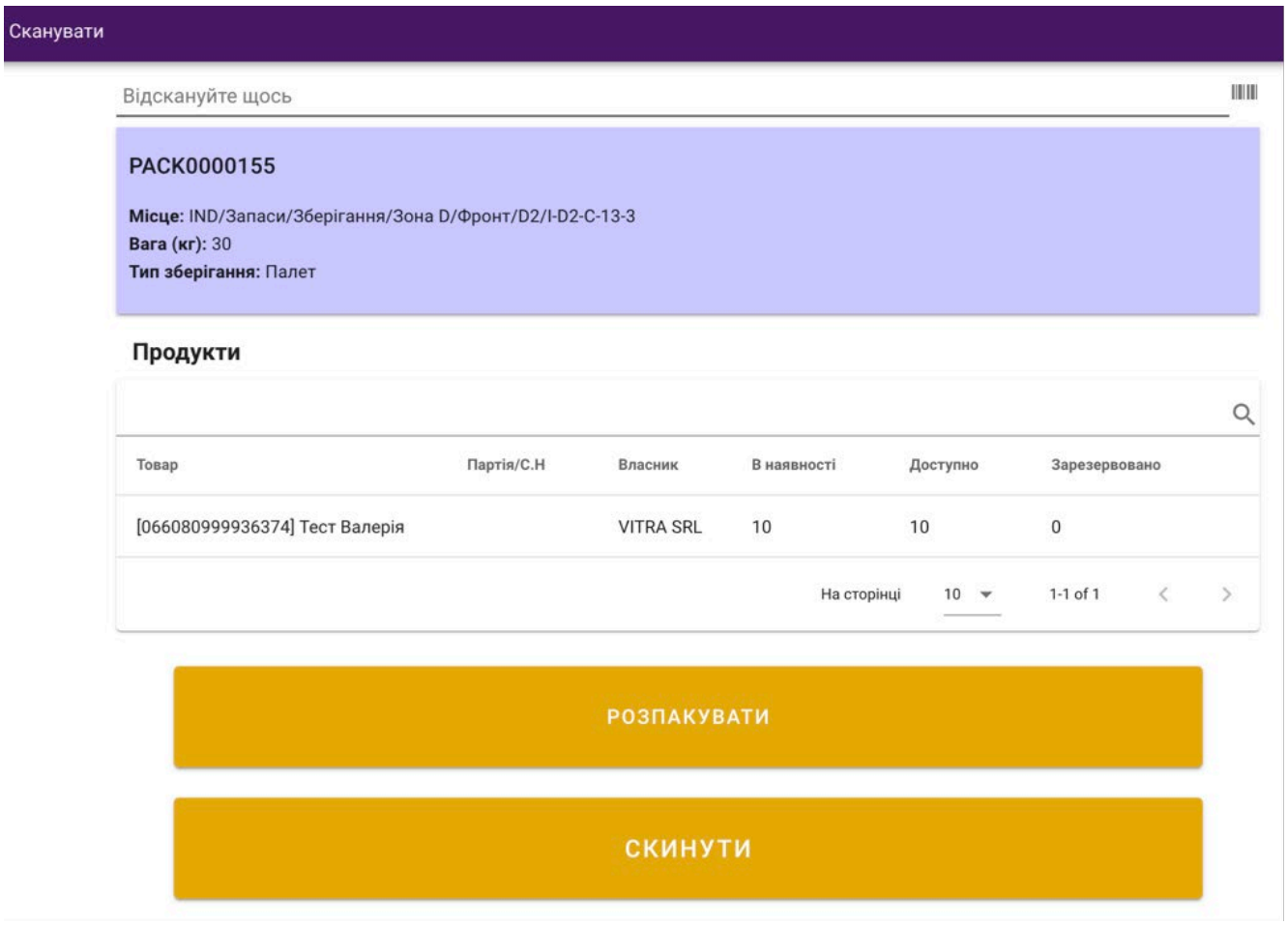


Рис. 3.10. Приклад сканування товару

3.5 Тестування системи

Odoo має свій власний інструмент для виконання тестів - Odoo Testing Framework. Це потужне засіб для автоматизованого тестування Odoo, який дозволяє розробникам створювати тести для перевірки коректності роботи модулів, включаючи функціональність, правила бізнес-логіки, інтерфейс користувача та інші аспекти системи.

Odoo Testing Framework має ряд переваг, які роблять його цінним інструментом для розробників Odoo. Він надає широкий спектр можливостей для автоматизованого тестування.

Цей інструмент може генерувати тестові дані на основі заданого набору правил. Це може допомогти розробникам значно скоротити час і зусилля, необхідні для створення тестових наборів.

Він також дозволяє запускати тести за допомогою одного натискання кнопки. Це значно економить час і може запобігти помилкам при запуску тестів. Також дана функція надає зручний інтерфейс для управління тестовими наборами.

Крім того, Odoo Testing Framework дозволяє виконувати тести повторно, що може допомогти виявити помилки, які виникають лише в певних умовах. Це може допомогти розробникам забезпечити високу якість програмного забезпечення та покращити досвід користувачів.

Odoo Testing Framework може використовуватися компаніями, які використовують Odoo, для покращення якості та стабільності системи, зменшення ризику неправильної роботи функціональності та поліпшення довіри користувачів до системи.

Розробники можуть використовувати даний інструмент для створення тестів, які перевіряють правильність роботи нового модуля, перш ніж його розгорнути в продуктивному середовищі. Це може допомогти запобігти поширенню помилок серед користувачів.

Також можна робити перевірку на те, чи не викликає оновлення системи жодних непередбачених проблем. Це може допомогти забезпечити безперебійну роботу системи та захистити дані користувачів.

Користувачі Odoo можуть використовувати Odoo Testing Framework для створення тестів, які перевіряють, чи відповідає система їхнім конкретним потребам. Це може допомогти виявити проблеми з системою та внести необхідні зміни.

Odoo Testing Framework - це потужний інструмент, який може допомогти компаніям, які використовують Odoo, забезпечити високу якість програмного забезпечення та покращити досвід користувачів.

3.6 Оцінка ефективності використання системи

Завдяки впровадженню системи були покращені такі показники, як час передачі замовлення на склад, час відбору товару, відсоток оброблених замовлень,

кількість механічних помилок та відсоток задоволених клієнтів. Це дозволяє співробітникам працювати ефективніше, в той час як витрати на оплату праці комірника не змінились. В таблиці 3.6 наведений порівняльний аналіз ефективності використання системи:

Таблиця 3.1

Порівняльний аналіз ефективності використання системи

Показник	**До впровадження системи**	**Після впровадження системи**
Час прийняття замовлення	5 хв.	5 хв.
Час передачі замовлення на склад	15 хв.	до 5 с
Час відбору товару	20 хв.	12 хв.
Відсоток оброблених замовлень	80%	100%
Кількість механічник помилок	5	1
Відсоток задоволених клієнтів	75-80%	95%

ВИСНОВОК

В ході написання дипломної роботи було виконано всі поставлені задачі.

1. Проведено аналіз бізнес-процесів клієнта, проаналізовано процеси прийому та відвантаження товарів зі складу, а також процеси переміщення. Виявлено наступні недоліки на етапах прийому та відбору товару: ведення комірником вручну всіх робочих записів, мануальний пошук місць зберігання товару, неструктуроване зберігання товару по зонам, довгий час передачі замовлення клієнту.

2. Проведено аналіз методів та засобів для складського обліку. Виявлено переваги та недоліки на прикладі SAP, Microsoft Dynamics 365 Supply Chain Management, Oracle Warehouse Management та Odoo. Виявлено, що перевагами Odoo є швидкий процес впровадження, гнучке налаштування та широкий функціонал системи, що дозволяє за відносно короткі терміни впровадити цілісну систему управління складським обліком.

3. Розроблено модифікований алгоритм обліку надходжень та відвантажень товару зі складу для підвищення ефективності управління складським обліком. Виробничі процеси покращено за рахунок додавання операцій сканування, що реалізуються в системі Odoo. Розроблений інтерфейс системи для роботи з ТЗД для більш зручної роботи комірників складу.

4. Оптимізовано правила резервації товарів. Розроблена формальна модель правил резервування товару. Впроваджений прорахунок страхового запасу, для зменшення ризиків затримок поставок від постачальників.

5. Програмно реалізовано розроблені алгоритми та моделі системи автоматизації складського обліку на базі Odoo, спрямовані на оптимізацію та підвищення ефективності управління складськими запасами.

6. Проаналізовано ефективність використання системи складського обліку за основними критеріями: час прийняття замовлення менеджером, час передачі замовлення на склад, час відбору товару, відсоток оброблених замовлень, кількість механічних помилок та відсоток задоволених клієнтів. Розроблена система забезпечує зменшення часу передачі замовлень на склад в три рази, час відбору товару знизився з 20хв. до 12 хв., відсоток оброблених замовлень підвищився до 100%, а кількість механічних помилок зменшилась в 5 разів.

ПЕРЕЛІК ПОСИЛАНЬ

1. SSK.UA [Електронний ресурс] // Sklad Service – 2020. – Режим доступу до ресурсу: https://ssk.ua/ua/blog/wms-sistema-kak-eto-rabotaet-501
2. CleverForms [Електронний ресурс] // CleverForms | CleverForms. – Режим доступу до ресурсу : https://clever-forms.com/shho-take-erp-sistema-dlya-chogo-vona-potribna-biznesu/#:~:text=Збільшення%20ефективності%20роботи.,персоналом,%20фінанси%20та%20інші%20операції.
3. How an Odoo Inventory optimization generated $200K for a US company | VentorTech [Електронний ресурс] // VentorTech. – Режим доступу до ресурсу: https://ventor.tech/warehouse-management/how-an-odoo-inventory-optimization-generated-200k-for-a-us-company/
4. Що таке SAP ERP та яку програму викростовувати як альтернативу в Україні | A4 Company. [Електронний ресурс] // A4 Company. – Режим доступу до ресурсу: https://a4.com.ua/scho-take-sap-erp/
5. Лапчук А. Що таке ERP-система та як вона допоможе вашому бізнесу? [Електронний ресурс] / Анна Лапчук // Дія Бізнес. – 2021. – Режим доступу до ресурсу: https://business.diia.gov.ua/cases/sistematizacia-biznesprocesiv/so-take-erp-sistema-ta-ak-vona-dopomoze-vasomu-biznesu.
6. WMS система: як це працює? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://ssk.ua/blog/wms-sistema-kak-eto-rabotaet-501
7. О'Лири Д. ERP-системи: вибір, впровадження, експлуатація. Сучасне планування і управління ресурсами підприємства / Дэниел О'Лири., 2019. – 271 с.
8. Ландватер Д. World Class Production and Inventory Management / Дэррил Ландватер., 2017.
9. ТСД [Електронний ресурс] – Режим доступу до ресурсу: https://www.iterator.com.ua/ru/poleznye-materialy/212-tsd-terminaly-sboradannykh-v-voprosakh-i-otvetakh.

10. Odoo docs [Електронний ресурс] – Режим доступу до ресурсу: https://www.odoo.com/documentation/17.0/
11. Open-Source Assessment Use Case: TETRA™ Detected Significant Improvements in the Quality of the Odoo ERP System. Intetics. [Електронний ресурс] – Режим доступу до ресурсу: https://intetics.com/blog/open-source-assessment-use-case-tetra-detected-significant-improvements-in-the-quality-of-the-odoo-erp-system/
12. Python documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/doc/
13. Warehouse Management | SCM Oracle СНГ. Oracle | Cloud Applications and Cloud Platform. [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/cis/scm/logistics/warehouse-management/
14. Що таке термінал збору даних // Vostok.dp.ua. [Електронний ресурс] – Режим доступу до ресурсу: https://www.vostok.dp.ua/ukr/infa1/tsd/terminal_sbora_dannih/
15. [Case Study] Odoo CRM/ERP/WMS Implementation For E-Commerce Store - MOVO.TRAINING - Cosmonauts.dev | Software House. cosmonauts.dev | Software House. [Електронний ресурс] – Режим доступу до ресурсу: https://cosmonauts.dev/blog/case-study-odoo-crm-erp-wms-implementation-for-e-commerce-store-movo-training/
16. [Case Study] Odoo CRM/ERP/WMS Implementation For E-Commerce Store - MOVO.TRAINING - Cosmonauts.dev | Software House [Electronic resource] // cosmonauts.dev | Software House. – Mode of access: https://cosmonauts.dev/blog/case-study-odoo-crm-erp-wms-implementation-for-e-commerce-store-movo-training/.
17. Business Cases | Odoo ERP Case Studies [Electronic resource] // Port Cities International Limited. – Mode of access: https://portcities.net/blog/du-an-odoo-erp-dien-hinh-3.
18. Odoo Case Study: Veolia transformed PET Recycling with Odoo [Electronic resource] // Port Cities International Limited. – Mode of

access: https://portcities.net/blog/du-an-odoo-erp-dien-hinh-3/odoo-case-study-veolia-pet-recycling-85.

19. How PSE provides better care for underprivileged Cambodian children with Odoo [Electronic resource] // Port Cities International Limited. – Mode of access: https://portcities.net/blog/du-an-odoo-erp-dien-hinh-3/how-pse-provides-better-care-for-underprivileged-cambodian-children-with-odoo-erp-44.
20. Customizing Odoo to Improve Customer Experience in Retail - The 3Sach Food Success Story [Electronic resource] // Port Cities International Limited. – Mode of access: https://portcities.net/blog/du-an-odoo-erp-dien-hinh-3/customizing-odoo-to-improve-customer-experience-in-retail-3sach-food-success-story-50.
21. O'Donnell J. warehouse management system (WMS) [Електронний ресурс] / Jim O'Donnell – Режим доступу до ресурсу: https://www.techtarget.com/searcherp/definition/warehouse-managementsystem-WMS
22. .XML introduction [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction
23. Ричардс Г. Warehouse Management : A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse / Гвинн Ричардс., 2016.
24. Рейс Д. Odoo Development Essentials: Fast Track Your Development Skills to Build Powerful Odoo Business Applications / Деніел Рейс., 2019.
25. МАРЧЕНКО В. М., ШУТЮК В. В. // ЛОГІСТИКА. / Київ, 2019. 314 с.
26. Автоматизація за допомогою ERP Odoo [Електронний ресурс] // dou.ua – 2023. – Режим доступу до ресурсу: https://dou.ua/forums/topic/42444/

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ УПРАВЛІННЯ СКЛАДСЬКИМ ОБЛІКОМ НА БАЗІ CRM+ERP СИСТЕМИ ODOO»

Виконав: студентка групи ПДМ-64 Кундик Валерія Олексіївна

Керівник: к.т.н., доц., доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ - 2024

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: зменшення часу виконання операцій та кількості помилок в процесі управління складським обліком за рахунок використання CRM+ERP системи Odoo.

Об'єкт дослідження: управління складським обліком.

Предмет дослідження: методи та засоби управління складським обліком на базі CRM+ERP системи Odoo.

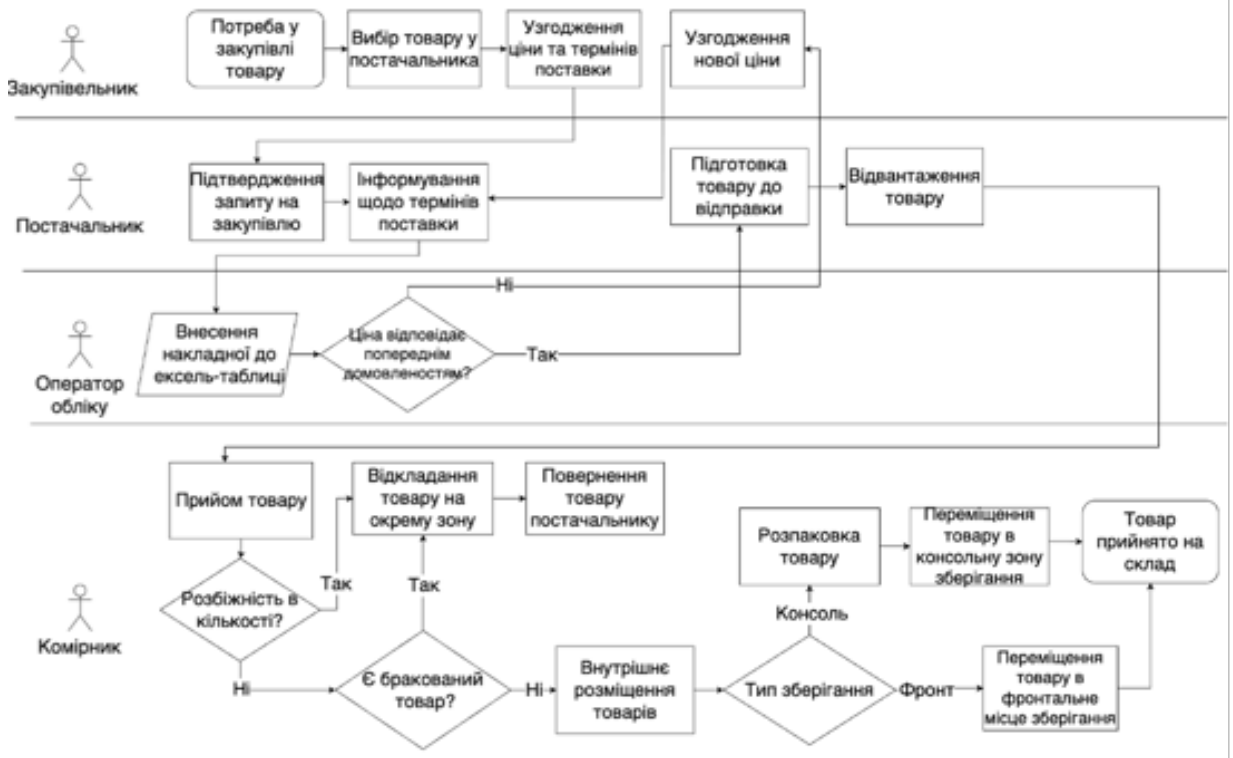
2

ПОРІВНЯННЯ ІСНУЮЧИХ СИСТЕМ

Особливості	SAP	Microsoft Dynamics 365 Supply Chain Management	Oracle Warehouse Management	Odoo
Ключові функції системи	- Управління матеріальними потоками - Планування потреби в матеріалах і послугах - Управління збутом та кригування залишків	- Управління складами та виконання замовлень - Закупівлі - Управління замовленнями та ціни - Обслуговування активів та управління	- Підтримка виконання замовлень - Управління поверненнями - Координація руху товарів - Інвентаризація - Оптимізація складські операції	**- Відстеження руху товарів - Адресне місцезберігання - Коригування залишків - Відстеження серійних номерів - Стратегії вилучення - Прогнозований запас залишків**
Впровадження	Складний та довгий процес впровадження	Складний та довгий процес впровадження	Складний та довгий процес впровадження	**Швидкий процес впровадження**
Гнучкість	Висока	Обмеження в налаштуваннях	Обмеження в налаштуваннях	**Висока**

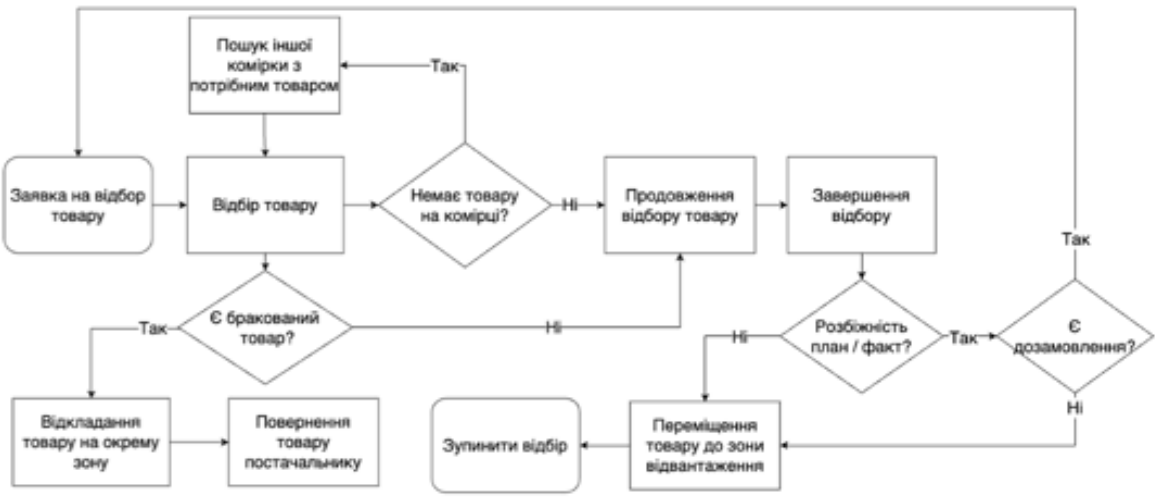
3

ПРОЦЕС НАДХОДЖЕННЯ ТОВАРУ ДО ВПРОВАДЖЕННЯ

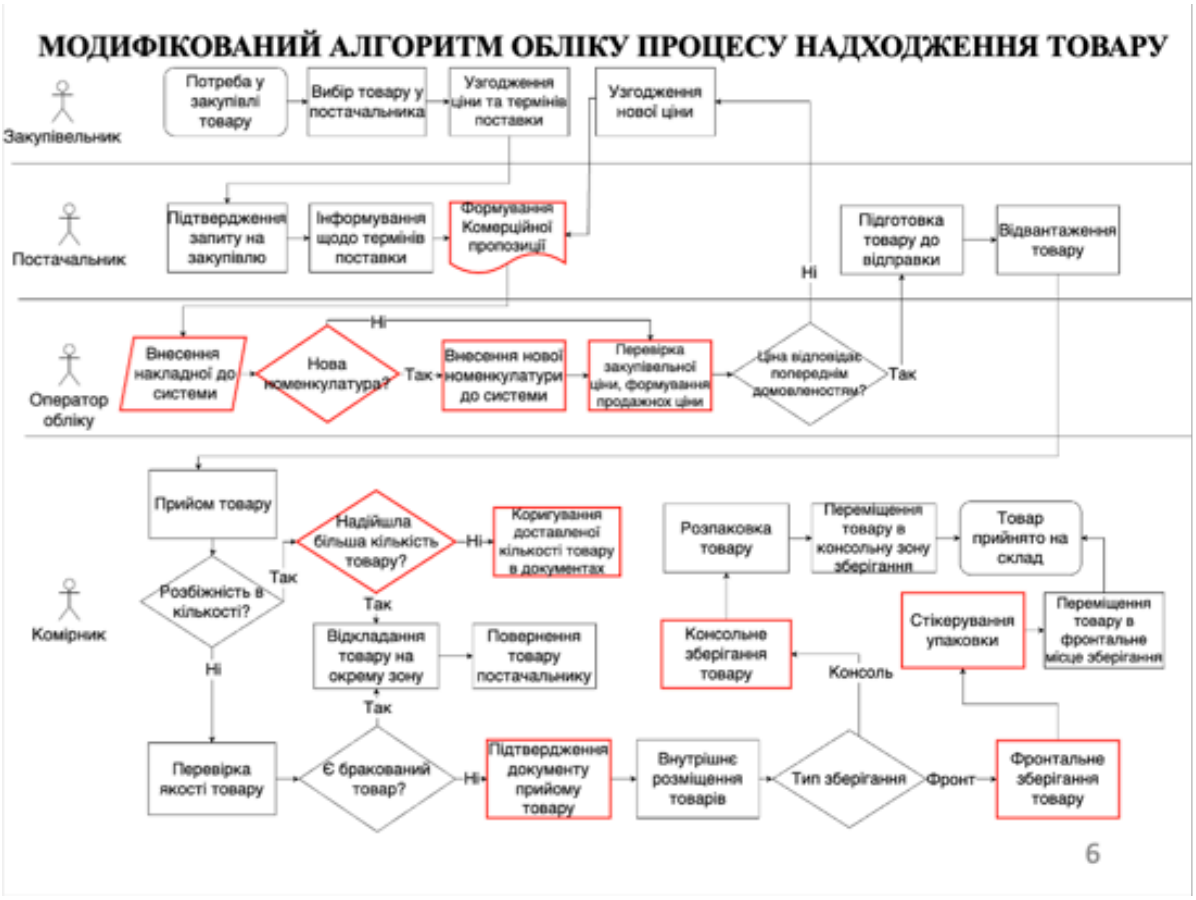


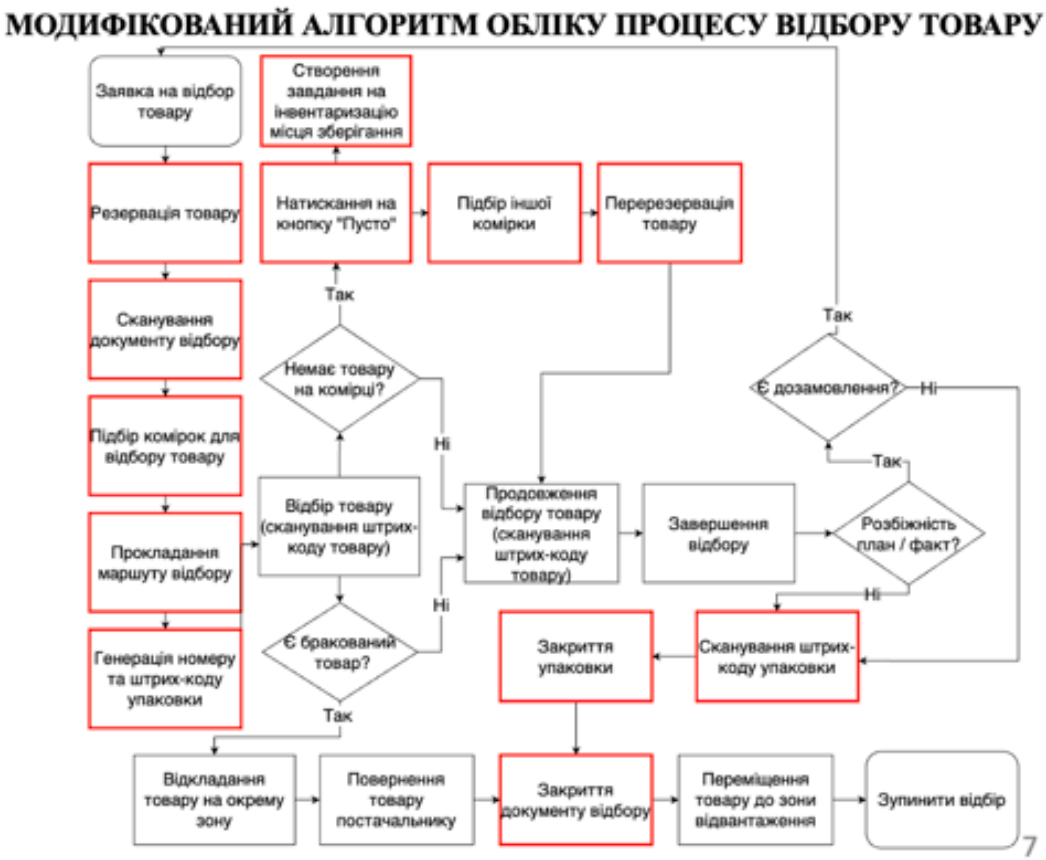
4

ПРОЦЕС ВІДВАНТАЖЕННЯ ТОВАРУ ДО ВПРОВАДЖЕННЯ



5


МОДИФІКОВАНИЙ АЛГОРИТМ ОБЛІКУ ПРОЦЕСУ НАДХОДЖЕННЯ ТОВАРУ
Закупівельник
Потреба у закупівлі товару
Вибір товару у постачальника
Узгодження ціни та термінів поставки
Узгодження нової ціни
Постачальник
Підтвердження запиту на закупівлю
Інформування щодо термінів поставки
Формування Комерційної пропозиції
Підготовка товару до відправки
Відвантаження товару
Ні
Оператор обліку
Внесення накладної до системи
Нова номенклатура?
Так
Ні
Внесення нової номенклатури до системи
Перевірка закупівельної ціни, формування продажних цін
Ціна відповідає попереднім домовленостям?
Так
Комірник
Прийом товару
Надійшла більша кількість товару?
Ні
Так
Коригування доставленої кількості товару в документах
Розбіжність в кількості?
Так
Ні
Відкладання товару на окрему зону
Так
Повернення товару постачальнику
Розпаковка товару
Переміщення товару в консольну зону зберігання
Товар прийнято на склад
Консольне зберігання товару
Консоль
Стікерування упаковки
Переміщення товару в фронтальне місце зберігання
Перевірка якості товару
Є бракований товар?
Ні
Підтвердження документу прийому товару
Внутрішнє розміщення товарів
Тип зберігання
Фронт
Фронтальне зберігання товару
6


МОДИФІКОВАНИЙ АЛГОРИТМ ОБЛІКУ ПРОЦЕСУ ВІДБОРУ ТОВАРУ
Заявка на відбор товару
Створення завдання на інвентаризацію місця зберігання
Резервація товару
Натискання на кнопку "Пусто"
Підбір іншої комірки
Перерезервація товару
Так
Сканування документу відбору
Немає товару на комірці?
Так
Ні
Є дозамовлення?
Ні
Так
Підбір комірок для відбору товару
Ні
Відбір товару (сканування штрих-коду товару)
Продовження відбору товару (сканування штрих-коду товару)
Завершення відбору
Розбіжність план / факт?
Прокладання маршуту відбору
Ні
Є бракований товар?
Ні
Закриття упаковки
Сканування штрих-коду упаковки
Генерація номеру та штрих-коду упаковки
Так
Відкладання товару на окрему зону
Повернення товару постачальнику
Закриття документу відбору
Переміщення товару до зони відвантаження
Зупинити відбір
7

РОЗРАХУНОК СТРАХОВОГО ЗАПАСУ ЗАЛИШКІВ НА СКЛАДІ

1. **Safety stock = (Maximum Daily Demand x Maximum Lead Time in days) – (Minimum Daily Demand x Average Lead Time in days)**

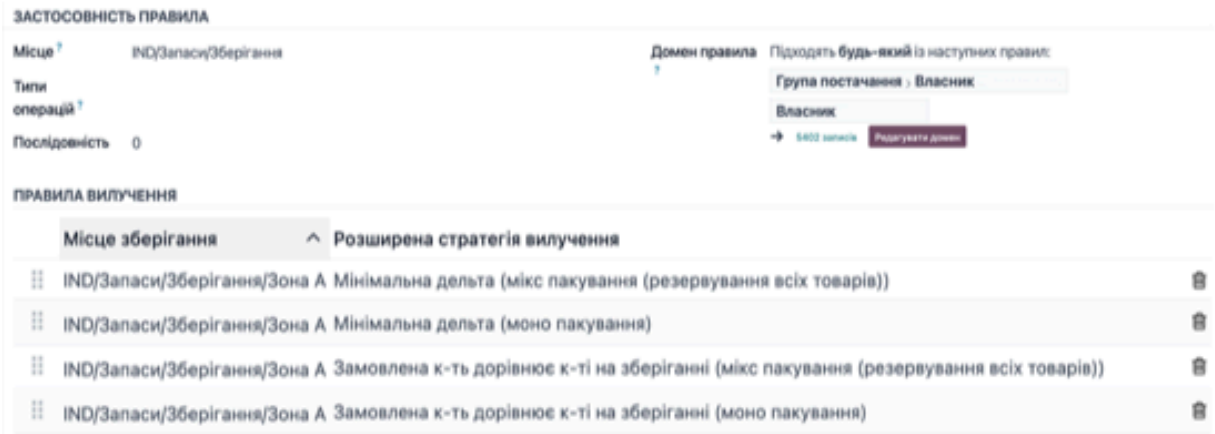
2. **ROP = (Average Daily Demand x Average Lead Time in days) + Safety Stock**

Maximum Daily Demand - максимальна кількість продажів в день
Average Daily Demand – середня кількість продажів в день
Minimum Daily Demand - мінімальна кількість продажів в день
Maximum Lead Time in days – максимальний час поставки товару
Average Lead Time in days – середній час поставки товару
Minimum Lead Time in days – мінімальний час поставки товару

8

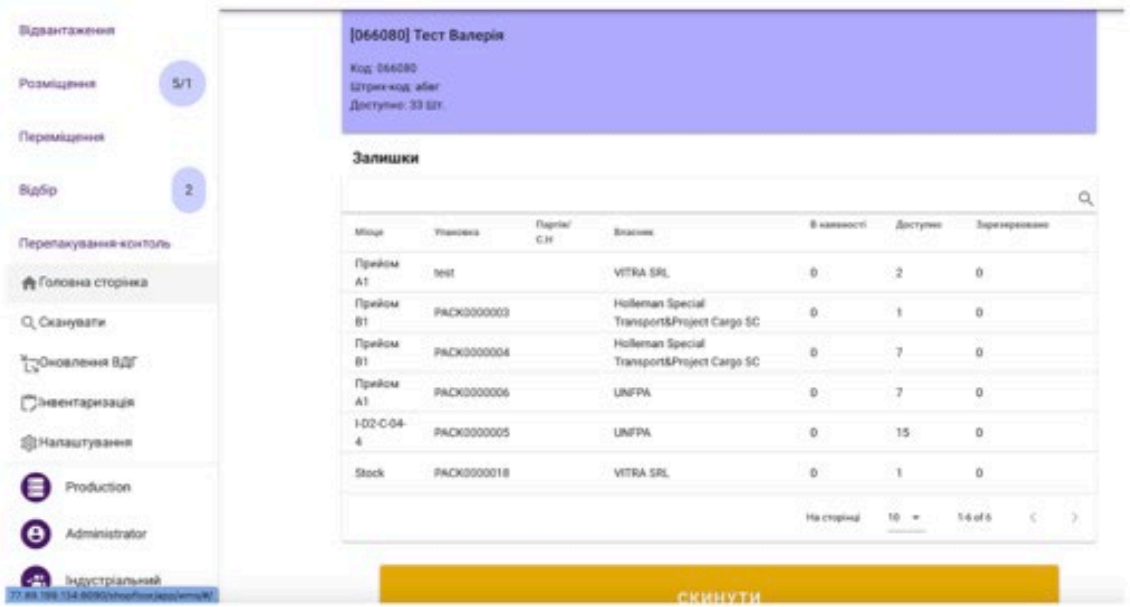
ФОРМАЛЬНА МОДЕЛЬ ПРАВИЛ ДЛЯ РЕЗЕРВУВАННЯ ТОВАРІВ

- Мінімальна дельта ([<тип пакування>] [(резервація <обсяг>)] <операція> замовлена кільк.)
- Замовлена кільк. <операція> кільк. на зберіганні ([<тип пакування>] [(резервація <обсяг>)] <операція> замовлена кільк.))



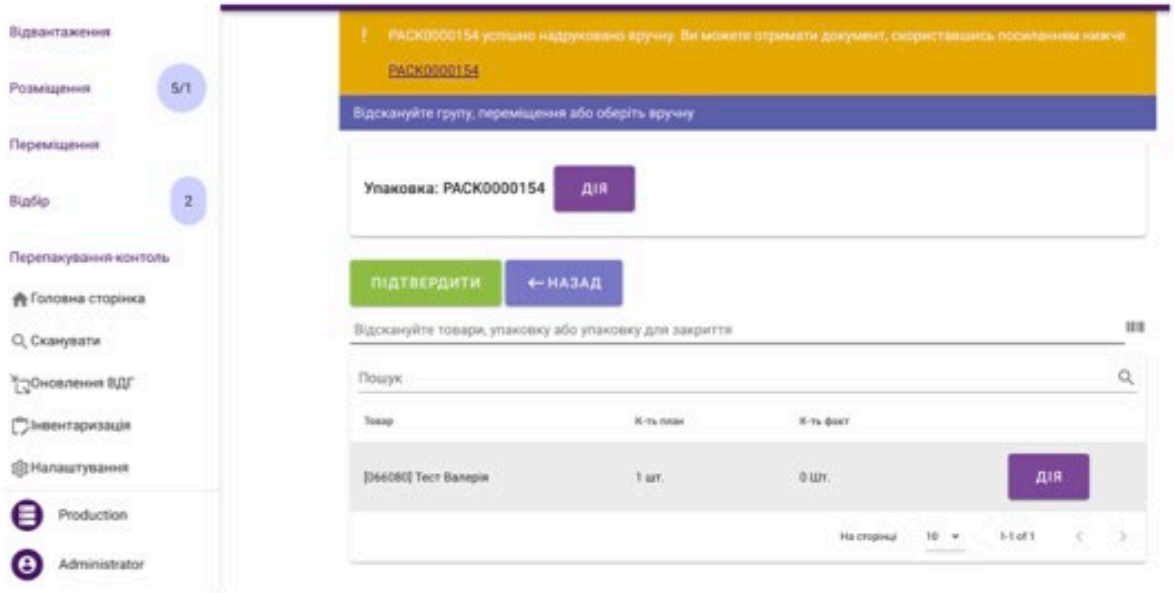
9

ПРИКЛАД РОБОТИ З ТЕРМІНАЛОМ ЗБОРУ ДАНИХ. СКАНУВАННЯ ТОВАРУ



10

ПРИКЛАД РОБОТИ З ТЕРМІНАЛОМ ЗБОРУ ДАНИХ. ПРОЦЕС ПРИЙОМУ ТОВАРУ НА СКЛАД



11

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ СИСТЕМИ

Показник	До впровадження системи	Після впровадження системи
Час прийняття замовлення	5 хв.	5 хв.
Час передачі замовлення на склад	15 хв.	до 5 с
Час відбору товару	20 хв.	12 хв.
Відсоток оброблених замовлень	80%	100%
Кількість механічних помилок	5	1
Відсоток задоволених клієнтів	75-80%	95%

12

ВИСНОВКИ

1. Проведено аналіз бізнес-процесів клієнта, проаналізовано процеси прийому та відвантаження товарів зі складу, а також процеси переміщення. Виявлено наступні недоліки на етапах прийому та відбору товару: ведення комірником вручну всіх робочих записів, мануальний пошук місць зберігання товару, неструктуроване зберігання товару по зонам, довгий час передачі замовлення клієнту.

2. Проведено аналіз методів та засобів для складського обліку. Виявлено переваги та недоліки на прикладі SAP, Microsoft Dynamics 365 Supply Chain Management, Oracle Warehouse Management та Odoo. Виявлено, що перевагами Odoo є швидкий процес впровадження, гнучке налаштування та широкий функціонал системи, що дозволяє за відносно короткі терміни впровадити цілісну систему управління складським обліком.

3. Розроблено модифікований алгоритм обліку надходжень та відвантажень товару зі складу для підвищення ефективності управління складським обліком. Виробничі процеси покращено за рахунок додавання операцій сканування, що реалізуються в системі Odoo. Розроблений інтерфейс системи для роботи з ТЗД для більш зручної роботи комірників складу.

4. Оптимізовано правила резервації товарів. Розроблена формальна модель правил резервування товару. Впроваджений прорахунок страхового запасу, для зменшення ризиків затримок поставок від постачальників.

5. Програмно реалізовано рощроблені алгоритми та моделі системи автоматизації складського обліку на базі Odoo, спрямовані на оптимізацію та підвищення ефективності управління складськими запасами.

6. Проаналізовано ефективність використання системи складського обліку за основними критеріями: час прийняття замовлення менеджером, час передачі замовлення на склад, час відбору товару, відсоток оброблених замовлень, кількість механічних помилок та відсоток задоволених клієнтів. Розроблена система забезпечує зменшення часу виконання замовлень на склад в три рази, час выдбору товару знизився с 20хв. До 12 хв., відсоток оброблених замовлень підвищився до 100%, а кількість механічних помилок зменшилась в 5 разів.

13

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

Кундик В.О., Золотухіна О.А. «Підвищення ефективності управління складським обліком на базі CRM+ERP системи Odoo» // Всеукраїнська науково-практична конференція «Telecommunication: Problems And Innovation», 10 грудня 2023 року, Державний університет інформаційно-комунікаційних технологій, Київ, Україна. Прийнято до друку.

Стаття:

Золотухіна О.А., Кундик В.О. Підвищення ефективності управління складським обліком на базі CRM+ERP системи Odoo. Наукові записки Державного університету телекомунікацій. № 1. 2024. Прийнято до друку.

ДЯКУЮ ЗА УВАГУ!

15

ДОДАТОК А
ПРИКЛАД ПРОГРАМНОГО КОДУ

```
class Uom(models.Model):
    _inherit = "uom.uom"

    remote_id = fields.Integer("Remote ID")


class Category(models.Model):
    _inherit = "product.category"

    remote_id = fields.Integer("Remote ID")


class Product(models.Model):
    _inherit = "product.product"

    remote_id = fields.Integer("Remote ID")

    is_kits = fields.Boolean(compute='_compute_is_kits', compute_sudo=False, store=True, inverse="_set_is_kits",
copy=False)

    def _set_is_kits(self):
        for product in self:
            product.product_tmpl_id.is_kits = product.is_kits

    height = fields.Float('Height', digits='WMS Size')
    width = fields.Float('Width', digits='WMS Size')
    depth = fields.Float('Depth', digits='WMS Size')

    def _default_uom(self):
        uom = self.env['product.template']._get_size_product_uom_name_from_ir_config_parameter()
        return uom

    size_uom = fields.Many2one('uom.uom', default=_default_uom)
    size_uom_name = fields.Char(related='size_uom.name', string="Size unit of measure label")

    def _update_uoms(self):
        meter = self.env.ref('uom.product_uom_meter')
        current_uom = self.size_uom
        if meter == current_uom:
            volume = self.width * self.height * self.depth
        else:
            if current_uom.uom_type == 'smaller':
                volume = (self.width * self.height * self.depth) * 0.000000001
            else:
                volume = (self.width * self.height * self.depth) / current_uom.factor
        self.volume = volume

    @api.onchange('height', 'width', 'depth')
    def _change_vals(self):
        self._update_uoms()

    @api.depends('height', 'width', 'depth')
    def _depends_vals(self):
        self._update_uoms()

    def get_quats(self, simple=False):
        domain = [('product_id', 'in', self.ids), ('location_id.usage', '=', 'internal'), ('on_hand', '=', True)]
        self = self.with_context(search_default_on_hand=True)
```

```
        # If user have rights to write on quant, we define the view as editable.
        if self.user_has_groups('stock.group_stock_manager'):
            self = self.with_context(inventory_mode=True)
            # Set default location id if multilocations is inactive
            if not self.user_has_groups('stock.group_stock_multi_locations'):
                user_company = self.env.company
                warehouse = self.env['stock.warehouse'].search(
                    [('company_id', '=', user_company.id)], limit=1
                )
                if warehouse:
                    self = self.with_context(default_location_id=warehouse.lot_stock_id.id)
        # Set default product id if quants concern only one product
        if len(self) == 1:
            self = self.with_context(
                default_product_id=self.id,
                single_product=True
            )
        else:
            self = self.with_context(product_tmpl_ids=self.product_tmpl_id.ids)

        if not simple:
            quants = self.env['stock.quant'].search(domain)
        else:
            quants = self.env['stock.quant'].search_read(domain, fields=['id'])

        return quants
from odoo import models, fields, api


class ProductTemplate(models.Model):
    _inherit = "product.template"

    is_kits = fields.Boolean(compute='_compute_is_kits', compute_sudo=False, store=True, inverse="_set_is_kits",
copy=False)
    is_manufactured = fields.Boolean(compute='_compute_is_kits', compute_sudo=False, store=True, copy=False)

    def _set_is_kits(self):
        for product in self:
            product.product_variant_ids.is_kits = product.is_kits

    @api.depends("bom_ids")
    def _compute_is_kits(self):
        super()._compute_is_kits()
        domain = [('product_tmpl_id', 'in', self.ids), ('type', '!=', 'phantom')]
        bom_mapping = self.env['mrp.bom'].search_read(domain, ['product_tmpl_id'])
        manuf_ids = set(b['product_tmpl_id'][0] for b in bom_mapping)
        for template in self:
            template.product_variant_ids.is_kits = template.is_kits

            template.is_manufactured = (template.id in manuf_ids)
            template.product_variant_ids.is_manufactured = (template.id in manuf_ids)

    # def write(self, vals):
    #     prod_vals = {}
    #     if 'is_kits' in vals:
    #         prod_vals['is_kits'] = vals.get('is_kits')
    #
    #     if 'is_manufactured' in vals:
    #         prod_vals['is_manufactured'] = vals.get('is_manufactured')
    #
    #     if prod_vals:
    #         self.mapped('product_variant_ids').write(vals)
```

```
    #     res = super().write(vals)
    #     return res

    @api.model
    def _get_size_product_uom_id_from_ir_config_parameter(self):
        product_size_param = self.env['ir.config_parameter'].sudo().get_param('deb_wms.default_product_size_uom')
        if product_size_param:
            return self.env['uom.uom'].browse(int(product_size_param))
        else:
            return self.env.ref('deb_wms.product_uom_millimeter')

    @api.model
    def _get_size_product_uom_name_from_ir_config_parameter(self):
        return self._get_size_product_uom_id_from_ir_config_parameter()
from odoo import fields, models, api


class ResConfigSettings(models.TransientModel):
    _inherit = 'res.config.settings'

    def _domain_uom_categ(self):
        uom_catergory = self.env.ref('uom.uom_categ_length')
        return "[('category_id', '=', {})]".format(uom_catergory.id)

    product_size_uom = fields.Many2one('uom.uom', string='Product Size UOM',
                                       config_parameter='deb_wms.default_product_size_uom',
                                       domain=lambda self: self._domain_uom_categ())
    product_packaging_size_uom = fields.Many2one('uom.uom', string='Product Packaging Size UOM',
                                       config_parameter='deb_wms.default_product_packaging_size_uom',
                                       domain=lambda self: self._domain_uom_categ())

    @api.model
    def get_values(self):
        res = super(ResConfigSettings, self).get_values()
        default_uom = self.env.ref('deb_wms.product_uom_millimeter')
        get_param = self.env['ir.config_parameter'].sudo().get_param
        set_param = self.env['ir.config_parameter'].sudo().set_param
        if get_param('product_size_uom'):
            res.update(
                product_size_uom=int(get_param('product_size_uom'))
            )
        else:
            res.update(product_size_uom=default_uom.id)
            set_param('product_size_uom', default_uom.id)
        if get_param('product_packaging_size_uom'):
            res.update(
                product_packaging_size_uom=int(get_param('product_packaging_size_uom')),
            )
        else:
            res.update(product_packaging_size_uom=default_uom.id)
            set_param('product_packaging_size_uom', default_uom.id)

        # self.env['product.product'].search([('size_uom', '!=', self.product_size_uom.id)]).write({
        #     'size_uom': default_uom.id
        # })
        # self.env['product.packaging'].search([('length_uom_id', '!=', self.product_packaging_size_uom.id)]).write({
        #     'length_uom_id': default_uom.id
        # })
        return res

    def set_values(self):
        super(ResConfigSettings, self).set_values()
        set_param = self.env['ir.config_parameter'].sudo().set_param
```

```
        set_param('product_size_uom', self.product_size_uom.id)
        set_param('product_packaging_size_uom', self.product_packaging_size_uom.id)
        if self.product_size_uom:
            self.env['product.product'].search([('size_uom', '!=', self.product_size_uom.id)]).write({
                'size_uom':  self.product_size_uom.id
            })
        if self.product_packaging_size_uom:
            self.env['product.packaging'].search([('length_uom_id', '!=', self.product_packaging_size_uom.id)]).write({
                'length_uom_id': self.product_packaging_size_uom.id
            })
import logging

from odoo import _, api, fields, models
from odoo.exceptions import ValidationError
from odoo.osv import expression

_logger = logging.getLogger(__name__)


class StorageCategory(models.Model):
    _inherit = "stock.storage.category"

    length = fields.Float('Length (mm)')
    width = fields.Float('Width (mm)')
    volume = fields.Float('Volume (m³)', compute='_compute_volume', digits='Volume', store=True)

    @api.onchange('length', 'width', 'max_height')
    def _compute_volume(self):
        for item in self:
            if item.length and item.width and item.max_height:
                item.volume = (item.length * item.width * item.max_height) / (10 ** 9)
            else:
                item.volume = 0

    recycle_percent = fields.Integer('Recycling Percentage', default=80)

    @api.onchange('recycle_percent')
    def _change_percent(self):
        for item in self:
            if item.recycle_percent < 0 or item.recycle_percent == 0:
                raise ValidationError(
                    _('Invalid recycling percentage. It should be bigger than 0 and does not equal 0'))


class StorageCategoryProductCapacity(models.Model):
    _inherit = "stock.storage.category.capacity"

    ignore_only_empty_loc_ids = fields.Many2many('stock.location', string="Ignore only empty locations")

    def _domain_location_storage_type(self, candidate_locations, quants, products):
        """
        Compute a domain which applies the constraint of the
        Stock Storage Category Capacities to select locations among candidate
        locations.
        """
        self.ensure_one()
        location_domain = [
            ("id", "in", candidate_locations.ids),
            ("computed_storage_category_id.capacity_ids", "in", self.ids),
        ]
        # Build the domain using the 'allow_new_product' field
        if self.allow_new_product == "empty":
            if self.ignore_only_empty_loc_ids:
```

```
            to_add_candidate_locations = candidate_locations.filtered_domain(
                [('id', 'child_of', self.ignore_only_empty_loc_ids.ids)])
            candidate_locations |= to_add_candidate_locations

            location_domain.append(("id", "in", candidate_locations.ids), )
            location_domain = expression.OR([location_domain, [("location_is_empty", "=", True)]])
        else:
            location_domain.append(("id", "in", candidate_locations.ids), )
            location_domain.append(("location_is_empty", "=", True))
    elif self.allow_new_product == "same":
        location_domain += self._get_product_location_domain(products)
    elif self.allow_new_product == "same_lot":
        lots = quants.mapped("lot_id")
        # As same lot should filter also on same product
        location_domain += self._get_product_location_domain(products)
        location_domain += [
            "|",
            # same comment as for the products
            ("location_will_contain_lot_ids", "in", lots.ids),
            ("location_will_contain_lot_ids", "=", False),
        ]
    return location_domain
import datetime

from odoo import api, fields, models, SUPERUSER_ID, _
from odoo.tools.safe_eval import safe_eval
from odoo.tools.float_utils import float_compare, float_is_zero, float_round


class StockPiking(models.Model):
    _inherit = "stock.picking"

    split_done_picking_id = fields.Many2one("stock.picking")

    kit_picking = fields.Boolean()
    bom_id = fields.Many2one('mrp.bom')
    user_id = fields.Many2many(
        'res.users', string='Responsible',
        domain=lambda self: [('groups_id', 'in', self.env.ref('stock.group_stock_user').id)],
        default=lambda self: self.env.user)

    def _get_default_wms_state(self):
        if (self._context.get('params') and self._context['params'].get('model') == 'purchase.order') or
self._context.get('quotation_only'):
            return 'in_transit'
        elif self._context.get('params') and self._context['params'].get('model') == 'sale.order':
            return 'in_work'
        else:
            if self._context.get('default_picking_type_id'):
                picking_type_id = self.env['stock.picking.type'].browse(self._context['default_picking_type_id'])
                if picking_type_id and picking_type_id.code == 'incoming':
                    return 'in_transit'
                else:
                    return 'in_work'

    wms_state = fields.Selection([('in_transit', 'In Transit'),
                                  ('arrived', 'Arrived'),
                                  ('in_work', 'In Work'),
                                  ('pick_error', 'Picking Error'),
                                  ('loading', 'Loading'),
                                  ('loaded', 'Loaded in Transport'),
                                  ('done', 'Done'),
                                  ('cancel', 'Canceled')], string="WMS State",
```

```
                    tracking=True,
                    default=_get_default_wms_state, copy=False)

@api.onchange('picking_type_id')
def _change_picking_type(self):
    for picking in self:
        if picking.picking_type_id:
            if picking.picking_type_id.code == 'incoming':
                picking.wms_state = 'in_transit'
            else:
                picking.wms_state = 'in_work'
        else:
            picking.wms_state = 'in_work'

order = fields.Char(string='Order')
volume = fields.Float('Volume', compute='_cal_volume', digits='Volume')
volume_uom_name = fields.Char(string='Volume unit of measure label',
                      default=lambda self: self.env.ref('uom.product_uom_cubic_meter').name)
sequence_code = fields.Char(related='picking_type_id.sequence_code', store=True)

divergences_ids = fields.One2many('deb.divergence', 'picking_id')

volume_actual = fields.Float('Volume Actual', compute='_cal_val_actual', digits='Volume')
weight_actual = fields.Float('Weight Actual', compute='_cal_val_actual', digits='Stock Weight')

@api.depends('move_ids')
def _cal_val_actual(self):
    vols = []
    weights = []
    for move in self.move_ids.filtered(lambda x: x.state != 'cancel'):
        vols.append(move.product_id.volume * move.quantity_done)
        weights.append(move.product_id.weight * move.quantity_done)
    self.volume_actual = sum(vols)
    self.weight_actual = sum(weights)

@api.depends('move_ids')
def _cal_volume(self):
    for picking in self:
        picking.volume = sum(move.volume for move in picking.move_ids if move.state != 'cancel')

@api.depends('state', 'wms_state')
def _compute_show_validate(self):
    super(StockPiking, self)._compute_show_validate()
    for picking in self:
        if picking.wms_state == 'in_transit':
            if picking.show_validate:
                picking.show_validate = False

def set_arrived(self):
    if self.wms_state not in ['in_transit']:
        return
    self.write({
        'wms_state': 'arrived'
    })

def generate_nonconf_act_number(self):
    noncof_act_number = self.env['ir.sequence'].next_by_code('stock.picking')
    self.write({
        'noncof_act_number': noncof_act_number
    })
    return noncof_act_number

def create_product_table(self, package_id=False):
```

```
    if not package_id:
        package_count = self.move_line_nosuggest_ids.mapped('result_package_id').mapped('id')
    else:
        package_count = package_id
    res = []
    for i in package_count:
        package = self.env['stock.quant.package'].browse(i).name
        by_package = self.move_line_ids_without_package.filtered(lambda x: x.result_package_id.id == i)
        if by_package:
            done_qty = sum(by_package.mapped('qty_done'))
            total_weight = sum(by_package.mapped('product_id').mapped('weight'))
            res.append([package, by_package, done_qty, total_weight])

    return res

def update_defect(self, create_scrap=False):
    # create defect
    if self.defect_ids:
        self.defect_ids.unlink()
    for line in self.move_line_nosuggest_ids:
        if line.quality_status:
            self.defect_ids.create({
                'product_id': line.product_id.id,
                'quality_status': line.quality_status.id,
                'defect_count': line.defect_count,
                'picking_id': self.id,
                'move_id': line.id
            })
            scraps = self.env['stock.scrap']
            if create_scrap:
                find_lot = False
                if line.lot_name:
                    find_lot = self.env['stock.lot'].search(
                        [('name', '=', line.lot_name), ('product_id', '=', line.product_id.id)])
                    if not find_lot:
                        find_lot = self.env['stock.lot'].create({
                            'product_id': line.product_id.id,
                            'name': line.lot_name,
                            'company_id': self.company_id.id,
                            'expiration_date': str(line.expiration_date) if line.expiration_date else False
                        })
                scrap = self.env['stock.scrap'].create({
                    'product_id': line.product_id.id,
                    'scrap_qty': line.qty_done,
                    'location_id': line.location_dest_id.id,
                    'scrap_location_id': line.quality_status.defect_zone.id,
                    'picking_id': self.id,
                    # 'move_id': line.move_id.id,
                    'product_uom_id': line.product_id.uom_id.id,
                    'origin': line.picking_id.origin,
                    'package_id': line.result_package_id.id,
                    'owner_id': self.owner_id.id,
                    'lot_id': find_lot.id if find_lot else False
                })
                scraps |= scrap
                # move = self.env['stock.move'].create(scrap._prepare_move_values())
                # scrap.write({
                #     'move_id': move.id
                # })

                transport_order = self.env['mo.transport.order'].search(
                    [('package_id', '=', line.result_package_id.id), ('for_scrap', '=', True)])
                if transport_order:
```

```
                transport_order.write({
                    # 'scrap_move_ids': [(4, scrp.move_id.id) for scrp in scrap] if scrap else False,
                    'scrap_ids': [(4, scrap.id)],
                })
            else:
                transport_order = self.env['mo.transport.order'].create({
                    # 'scrap_move_ids': [(4, scrap.move_id.id)],
                    'package_id': line.result_package_id.id,
                    'picking_ids': [(4, line.move_id.move_dest_ids.picking_id.id)],
                    'scrap_ids': [(4, scrap.id)],
                    'for_scrap': True
                })
                line.move_id.move_dest_ids.picking_id.write({
                    'state': 'placing',
                    'transfer_order_ids': [(4, transport_order.id)]
                })
            # srp.action_validate()
        if scraps:
            self.env.context = dict(self.env.context)
            self.env.context.update({'scrap_create': scraps})

@api.model_create_multi
def create(self, vals_list):
    # self.env.context = dict(self.env.context)
    # for vals in vals_list:
    #     if 'picking_type_id' in vals:
    #         self.env.context.update({'default_picking_type_id': vals['picking_type_id']})
    res = super().create(vals_list)
    return res

def write(self, vals):
    if 'wms_state' in vals:
        if vals['wms_state'] == 'in_work':
            vals['start_date'] = str(datetime.datetime.utcnow().replace(microsecond=False))
    res = super().write(vals)
    return res

def _action_done(self):
    for picking in self:
        picking.write({
            'wms_state': 'done',
            # 'date_done': str(datetime.datetime.utcnow().replace(microsecond=False))
        })
    res = super()._action_done()
    return res

def action_cancel(self):
    for picking in self:
        picking.write({
            'wms_state': 'cancel'
        })
    res = super().action_cancel()
    return res

@api.depends("canceled_by_routing")
def _compute_state(self):
    super()._compute_state()
    for picking in self:
        if picking.state == 'cancel' and picking.wms_state != 'cancel':
            picking.wms_state = "cancel"
            # picking.sudo().unlink()

def check_entire_pack(self):
```

```
    self._check_entire_pack()

def action_assign(self):
    to_remove = self.env['stock.picking']
    if self._context.get('manual_assign', False):
        for picking in self:
            if picking.picking_type_id.sequence_code == 'INT':
                picking = picking.with_context(ignore_reserve_rule=True)

                super(StockPiking, picking).action_assign()
                to_remove |= picking

    if to_remove:
        self -= to_remove

    if not self:
        return

    res = super(StockPiking, self).action_assign()
    if not res:
        return res

    # for move in self.move_ids:
    #     if move.move_dest_ids:
    #         for dest in move.move_dest_ids:
    #             if dest.product_uom_qty != move.reserved_availability:
    #                 dest.write({
    #                     "product_uom_qty": move.reserved_availability
    #                 })
    #     else:
    #         dest_pickings_move = \
    #         self.move_ids.filtered(lambda x: x.move_dest_ids).mapped('move_dest_ids')[0]
    #
    #         arr_pick = self.env['stock.picking']
    #
    #         def get_dest_pickings(dest_move):
    #             if dest_move.picking_id == self.picking_id:
    #                 return
    #
    #         def process_dest(dest_move, update_move):
    #             pick = dest_move.picking_id
    #             new_move = move.copy({
    #                 'product_id': move.product_id.id,
    #                 'picking_id': pick.id,
    #                 'state': 'waiting',
    #                 'product_uom_qty': move.reserved_availability,
    #                 "product_uom": move.product_uom.id,
    #                 'name': move.product_id.name,
    #                 'description_picking': move.product_id.name,
    #             })
    #             update_move.write(
    #                 {"move_dest_ids": [(4, new_move.id)]}
    #             )
    #             new_move.write({"move_orig_ids": [(4, update_move.id)]})
    #             new_move._merge_moves()
    #             new_move._recompute_state()
    #
    #             if dest_move.move_dest_ids:
    #                 process_dest(dest_move.move_dest_ids[0], new_move)
    #
    #         process_dest(dest_pickings_move, move)

    return res
```

```
def _check_entire_pack(self):
    for picking in self:
        origin_packages = picking.move_line_ids.mapped("package_id")
        for pack in origin_packages:
            if picking._check_move_lines_map_quant_package(pack):
                package_level_ids = picking.package_level_ids.filtered(lambda pl: pl.package_id == pack)
                move_lines_to_pack = picking.move_line_ids.filtered(
                    lambda ml: ml.package_id == pack and not ml.result_package_id and ml.state not in ('done', 'cancel'))

                disable_domain = safe_eval(picking.picking_type_id.disable_entire_pack_domain)
                if disable_domain:
                    move_lines_not_to_pack = move_lines_to_pack.filtered_domain(disable_domain)
                    move_lines_to_pack -= move_lines_not_to_pack
                    if not move_lines_to_pack:
                        continue

                if not package_level_ids:
                    self.env['stock.package_level'].create({
                        'picking_id': picking.id,
                        'package_id': pack.id,
                        'location_id': pack.location_id.id,
                        'location_dest_id': self._get_entire_pack_location_dest(
                            move_lines_to_pack) or picking.location_dest_id.id,
                        'move_line_ids': [(6, 0, move_lines_to_pack.ids)],
                        'company_id': picking.company_id.id,
                    })
                    # Propagate the result package in the next move for disposable packages only.
                    if pack.package_use == 'disposable':
                        move_lines_to_pack.with_context(
                            bypass_reservation_update=self._get_entire_pack_location_dest(move_lines_to_pack)
                        ).write({'result_package_id': pack.id})
                else:
                    move_lines_in_package_level = move_lines_to_pack.filtered(
                        lambda ml: ml.move_id.package_level_id)
                    move_lines_without_package_level = move_lines_to_pack - move_lines_in_package_level
                    for ml in move_lines_in_package_level:
                        ml.write({
                            'result_package_id': pack.id,
                            'package_level_id': ml.move_id.package_level_id.id,
                        })
                    move_lines_without_package_level.write({
                        'result_package_id': pack.id,
                        'package_level_id': package_level_ids[0].id,
                    })
                    for pl in package_level_ids:
                        pl.location_dest_id = self._get_entire_pack_location_dest(
                            pl.move_line_ids) or picking.location_dest_id.id

def _spilt_by_kit(self):
    group_by_bom = {}
    for move in self.move_ids:
        if move.bom_line_id:
            bom_id = move.bom_line_id.bom_id
            if bom_id.type != 'phantom':
                continue
            if not group_by_bom.get(bom_id):
                group_by_bom.update({
                    bom_id: [move]
                })
            else:
                moves = group_by_bom[bom_id]
                moves.append(move)
```

```
                group_by_bom.update({
                    bom_id: moves
                })

        if not group_by_bom:
            return

        for bom in group_by_bom:
            moves = group_by_bom[bom]

            move_ids = [move.id for move in moves]

            vals = {
                "kit_picking": True,
                "move_ids": [(4, move) for move in move_ids],
                "bom_id": bom.id
            }
            if self.move_line_ids:
                lines = self.move_line_ids.filtered(lambda x: x.move_id.id in move_ids)
                if lines:
                    vals['move_line_ids'] = [(4, move.id) for move in lines]

            new_picking = self.copy(vals)

    # def action_put_in_pack(self):
    #     self.ensure_one()
    #
    #     picking_move_lines = self.move_line_ids
    #     if (
    #             not self.picking_type_id.show_reserved
    #             and not self.immediate_transfer
    #             and not self.env.context.get('barcode_view')
    #     ):
    #         picking_move_lines = self.move_line_nosuggest_ids
    #     move_line_ids = picking_move_lines.filtered(lambda ml:
    #                                         float_compare(ml.qty_done, 0.0,
    #                                                 precision_rounding=ml.product_uom_id.rounding) > 0
    #                                         and not ml.result_package_id
    #                                         )
    #     if not move_line_ids:
    #         move_line_ids = picking_move_lines.filtered(lambda ml: float_compare(ml.reserved_uom_qty, 0.0,
    #                                                             precision_rounding=ml.product_uom_id.rounding) > 0 and
float_compare(
    #             ml.qty_done, 0.0,
    #             precision_rounding=ml.product_uom_id.rounding) == 0)
    #
    #     res = super().action_put_in_pack()
    #     if res:
    #         return res
    #     else:
    #         if not self._context.get('skip_check_package_type'):
    #             package_id = move_line_ids.result_package_id
    #             res = self._check_package_type(package_id)
    #             if res:
    #                 return res
    #
    #     return res

    def _check_package_type(self, package_id):
        if not package_id.package_type_id:
            view_id = self.env.ref('stock.stock_package_destination_form_view').id
            wiz = self.env['deb.check.package.type'].create({
                'picking_id': self.id,
```

```
            'location_dest_id': move_line_ids[0].location_dest_id.id,
        })
        return {
            'name': _('Choose destination location'),
            'view_mode': 'form',
            'res_model': 'deb.check.package.type',
            'view_id': view_id,
            'views': [(view_id, 'form')],
            'type': 'ir.actions.act_window',
            'res_id': wiz.id,
            'target': 'new'
        }
    else:
        return {}
```