Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Півня Івана Сергійовича*
(ПІБ)

академічної групи *122-20-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка комп'ютерної системи автоматизованого менеджменту промислових складів*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)
Алексєєв М.О.
(підпис) (прізвище, ініціали)
« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-2* (група) *Півня І.С.* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка комп'ютерної системи автоматизованого менеджменту промислових складів*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***03.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***10.06.2024 р.***

Завдання видав (підпис) *проф. Лактіонов І.С* (посада, прізвище, ініціали)
Завдання прийняв до виконання (підпис) *Півень І.С.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *17.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 65 с., 23 рис., 3 дод., 2 табл., 23 джерел.

Дана кваліфікаційна бакалаврська робота присвячена розробці автоматизованої системи менеджменту промислових складів. Метою розробки є оптимізація управління та обробки складських операцій. Система спрямована на відстеження запасів товарів і запасних частин, їх параметрів і характеристик, а також на інтеграцію додаткових підсистем для ефективного управління запасними частинами на складах, розташованих у віддалених регіонах. Об'єктом розробки є комплексна комп'ютерна система менеджменту складів, що складається з двох основних програмних компонентів: сервера для обробки даних зі складів та веб-сервера для надання користувачам візуального інтерфейсу для взаємодії та обробки запитів.

Кваліфікаційна робота включає детальний аналіз предметної області, обґрунтування обраних технологій, дизайн системи, деталі реалізації та економічну оцінку витрат на розробку програмного забезпечення. Результатом є гнучке рішення для управління складом, яке можна розгортати на різних пристроях та серверах, забезпечуючи сумісність та ефективність у промислових умовах. З наукової точки зору, це дослідження робить внесок у сферу комп'ютерних наук, інтегруючи передові методи обробки даних і хмарні обчислення в роботу складу. Використання розробленої системи дозволяє ефективно обробляти і аналізувати великі набори даних, полегшуючи прийняття рішень в режимі реального часу і оперативні коригування. Інтеграції, впроваджені в системі забезпечують безперебійне зберігання та доступність даних, сприяючи співпраці та обміну даними.

З переваг розробленої системи варто зазначити точність інвентаризації, що знижує операційні витрати та підвищує продуктивність за рахунок автоматизації рутинних завдань і надання інформації про складські операції в режимі реального часу. Крім того, розробка та впровадження системи підкреслюють важливість гнучких і масштабованих технологій в управлінні промисловістю.

Серверна частина системи менеджменту реалізована на мові Python з використанням таких бібліотек, як Flask для API-запитів, Pandas для обробки даних і gspread для управління хмарним сховищем через Google Sheets. Веб-сервер використовує фреймворк Next.js, побудований на React з TypeScript, з використанням shadcn/ui для графічних компонентів і Tailwind CSS для стилізації.

Ключові слова: система менеджменту складів, логістика, управління ланцюгами поставок, інтеграція даних, обробка даних, управління інвентарем, Python, Flask API, Pandas, Next.js.

ABSTRACT

Explanatory note: 65 pages, 23 fig., 3 appendix, 2 table, 23 sources.

This bachelor thesis is devoted to the development of an automated industrial warehouse management system. The purpose of the development is to optimize the management and processing of warehouse operations. The system is aimed at tracking stocks of goods and spare parts, their parameters and characteristics, as well as integration of additional subsystems for efficient management of spare parts in warehouses located in remote regions. The object of development is a complex warehouse management system consisting of two main software components: a server for processing data from warehouses and a web server for providing users with a visual interface for interaction and processing requests.

The work includes a detailed analysis of the subject area, justification of the selected technologies, system design, implementation details and economic evaluation of software development costs. The result is a flexible warehouse management solution that can be deployed across devices and servers, ensuring interoperability and efficiency in industrial environments. From a scientific perspective, this research contributes to the field of computer science by integrating advanced data processing techniques and cloud computing into warehouse operations. Using the developed system allows you to efficiently process and analyze large data sets, facilitating real-time decision-making and operational adjustments. Integrations implemented in the system ensure seamless data storage and availability, facilitating collaboration and data sharing.

Among the advantages of the developed system, it is worth noting the accuracy of the inventory, which reduces operational costs and increases productivity due to the automation of routine tasks and the provision of information about warehouse operations in real time. In addition, the development and implementation of the system emphasize the importance of flexible and scalable technologies in industrial management.

The server part of the management system is implemented in Python using such libraries as Flask for API requests, Pandas for data processing and gspread for cloud storage management through Google Sheets. The web server uses the Next.js framework, built on React with TypeScript, using shadcn/ui for graphical components and Tailwind CSS for styling.

Keywords: Warehouse Management System, Logistics, Supply Chain Management, Data Integration, Data Processing, Inventory Management, Python, Flask API, Pandas, Next.js.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

AR – Augmented Reality

CSS – Cascading Styles Sheets

ERP – Enterprise Resource Planning (System)

FTP – File Transfer Protocl

HTML – HypeText Markup Language

HTTP – HyperText Transfer Protocol

RFID – Radio Frequency Identification

TMS – Transport Management System

UI – User Interface

VPN – Virtual Private Network

WMS – Warehouse Management System

XML – (E)Xtensible Markup Language

БД – база даних

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

ЗМІСТ

РОЗІДЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Промислові склади - це будівлі, які організації використовують для зберігання товарів, зокрема продуктів харчування, меблів та одягу, складових запчастин. Склад - це частина складної логістичної мережі, яка дозволяє компаніям зберігати, захищати та переміщувати свою продукцію з місця на місце. Вони виконують не лише функцію зберігання, але і багато інших функцій, які допомагають компаніям оптимізувати свої операції та забезпечити постійну доступність і безпеку запасів.

Функція складу - зберігати товари, зазвичай тимчасово. На складах товари можуть зберігатися певний час - від кількох днів до місяців і років. Вільний доступ габаритних вантажів та транспорту – важлива їх особливість. Склади часто розташовані поблизу портів, аеропортів, залізничних ліній і автомагістралей, щоб підвищити доступність і полегшити своєчасне здійснення поставок і перевезень. Як правило, приміщення що функціонують як промислові склади розташовані у дуже великих будівлях, іноді з кількома поверхами і системами вентиляції для підтримання відповідної температури товарів. Організації використовують склади для зберігання товарів і виконання багатьох інших важливих операцій на них. Окрім зберігання товарів, логістичні команди проводять інвентаризацію та контроль якості на складі. Вони також використовують склад для захисту товарів і управління коливаннями цін на продукцію.

Призначенням складу є надання місця для зберігання товарів, зокрема інвентарю, обладнання та інших предметів. Наявність достатньої кількості складських приміщень для зберігання багатьох товарів зменшує кількість відходів, оскільки забезпечує безпечне зберігання товарів, які організація не продає одразу. На складі можна зберігати багато видів товарів, зокрема меблі та

одяг, протягом тривалого періоду, поки вони не будуть готові до купівлі або використання. Організації, що мають складські площі, які перевищують попит, мають більше можливостей реагувати на коливання ринку, ніж ті, що не мають надлишкових складських площ.

Складське зберігання можна розділити на два типи: планове зберігання та тривале зберігання. Заплановане зберігання - це зберігання, яке, за очікуваннями організації, необхідне для задоволення регулярного попиту клієнтів. Розширене зберігання - це зберігання, яке іноді може бути необхідним понад вимоги планового зберігання, наприклад, якщо попит іноді коливається через сезонні зміни або якщо організація тимчасово потребує додаткових запасів у зв'язку з великою акцією з просування продажів [1]. Дуже важливо, щоб логістичні лідери враховували потреби своєї організації як у плановому, так і в подовженому зберіганні при розподілі складських площ на складах.

Система управління складом - це програмне забезпечення, яке допомагає компаніям керувати та контролювати щоденні складські операції від моменту надходження товарів і матеріалів до розподільного центру або центру виконання робіт до моменту їх виходу. Програмні системи управління промисловими складами є ключовим компонентом управління ланцюгами поставок і забезпечують видимість у реальному часі всіх запасів компанії, як на складах, так і в дорозі. На додаток до управління запасами, системи пропонують інструменти для процесів комплектації та пакування, використання ресурсів, аналітики тощо.

Зараз, як ніколи раніше, оптовики, сторонні постачальники логістичних послуг і вантажовідправники перебувають під тиском необхідності виконувати і доставляти багатоканальні замовлення дуже швидко. Продажі в електронній комерції також різко зросли. Лише у 2019 році, за даними Statista [19], роздрібні продажі онлайн у США склали 343,15 мільярда доларів США - і, за прогнозами, до 2024 року вони сягнуть майже 476,5 мільярда доларів США. У той час як онлайн-продажі та очікування щодо швидкої доставки зростають, резерв робочої сили скорочується. І до початку пандемії COVID-19 в середині березня 2020 року

низький рівень безробіття в країні ускладнював пошук складських працівників. Хоча кількість вільних працівників тимчасово зросла, після COVID-19 складська робоча сила, ймовірно, знову стане дефіцитною. Всі ці фактори створюють потребу у швидшому та ефективнішому управлінні складом та логістичними процесами.

Сучасні системи менеджменту оптимізують кожен аспект управління складом - від процесів приймання, розміщення, комплектації, пакування та відвантаження до відстеження запасів і їх поповнення [2]. І вона організовує всі ці дії з єдиного інтерфейсу. Системи управління складом також інтегруються з іншими інструментами, включаючи такі базові, як сканування штрих-кодів і маркування RFID, більш досконалу робототехніку і переносні пристрої з доповненою реальністю (AR), а також з іншими критично важливими рішеннями, такими як системи управління транспортом (TMS), ERP і логістичне програмне забезпечення. Надійна цифрова система управління складом має важливе значення для будь-якого бізнесу з наявними запасами - і може допомогти заощадити гроші та отримати нові можливості для підвищення ефективності в багатьох сферах. З основних переваг систем менеджменту промислових складів можна перелічити [6]:

– Підвищення операційної ефективності: системи менеджменту автоматизують і впорядковують складські процеси від вхідних надходжень до вихідних відвантажень - для підвищення ефективності, безперебійної роботи і здатності обробляти більші обсяги. Вони зменшують кількість помилок при відборі та відвантаженні товарів і усувають дублювання та непотрібну роботу. Системи менеджменту також обмінюється даними з ERP-системами та системами управління транспортом, надаючи вам цілісну картину, яка виходить за межі вашого складу і допомагає прискорити переміщення товарів.

– Зменшення відходів і витрат: якщо у компанії є товари з обмеженим терміном придатності або ті, що швидко псуються, програмне забезпечення може визначити, які товари потрібно відбирати в першу чергу, а які, можливо, потрібно стимулювати продаж, щоб мінімізувати відходи. Воно також може

допомогти вам визначити найбільш ефективне використання складського простору, від розміщення запасів до оптимальних шляхів переміщення. Деякі системи пропонують просунуте моделювання для створення планів поверхів і розміщення палет, полиць і обладнання в найкращих місцях, щоб працювати з максимальною ефективністю та заощаджувати час і гроші.

– Контроль запасів у реальному часі: використовуючи штрих-кодування, RFID-мітки, датчики або інші методи відстеження місцезнаходження, система дає вам уявлення про ваші запаси в реальному часі, коли вони переміщуються на склад, навколо нього і в наступне місце. Завдяки такій видимості ви можете створювати більш точний попит

– Покращене управління трудовими ресурсами: програмне забезпечення може допомогти вам прогнозувати потреби в робочій силі, створювати графіки, оптимізувати час пересування по складу і призначати потрібні завдання потрібним працівникам на основі рівня кваліфікації, близькості та інших факторів. Хороша система може також сприяти підвищенню морального духу співробітників, створюючи більш спокійне, організоване і безпечне середовище, в якому працівники відчувають, що їхній час цінується і використовується з розумом.

– Кращі відносини з клієнтами та постачальниками: завдяки менеджменту складів клієнти отримують більш якісне виконання замовлень, швидшу доставку та менше неточностей, що підвищує їхню задоволеність і лояльність, а також покращує репутацію вашого бренду. Постачальники також можуть відчути скорочення часу очікування на вантажних майданчиках і в доках, що сприяє поліпшенню відносин.

Дослідження демонструють, як перехід від ручної до автоматизованої системи може покращити управління запасами, зменшити кількість людських помилок та оптимізувати складські площі. Для ілюстрації цих переваг використовується конкретний приклад провідного постачальника телекомунікаційних послуг в Йорданії [16]. Впровадження виробничої станції на складі для пакування, маркування та перепакування SIM-карток і передплачених

скретч-карток ще раз демонструє покращення використання простору та ефективності робочого процесу.

Однією з ключових переваг WMS є її здатність забезпечувати збір даних у реальному часі та оптимізувати звичайні складські завдання. Така автоматизація призводить до значної економії часу та підвищення точності роботи з запасами. Впровадження програмного забезпечення, яке управляє даними від отримання замовлення до доставки дилерам, забезпечує безпомилковість процесів і підвищує безпеку. WMS не лише підтримує сучасні складські процеси, такі як керований відбір та відвантаження, але й інтегрується з іншими системами, такими як ERP, для комплексного управління бізнесом [14].

Автоматизована WMS може значно покращити складські операції, підвищуючи ефективність, зменшуючи кількість помилок та оптимізуючи використання простору. Тематичне дослідження телекомунікаційного складу слугує практичним прикладом цих переваг, надаючи цінну інформацію для інших організацій, які прагнуть перейти від ручних до автоматизованих систем.

1.2. Призначення розробки та галузь застосування

Метою розробки комп'ютерної системи автоматизованого менеджменту промислових складів є відслідковування наявних на складах товарів та запчастин, їх параметрів та характеристик та додаткових під-систем для ефективного менеджменту запасних запчастин у складах, що знаходяться у віддалених регіонах. Провівши аналіз щодо існуючих рішень, даних та режимів доступу до них було сформовано наступні вимоги:

– Під-система розрахунку запасних деталей на основі заданих критеріїв: завдяки створенню системної документації та маркуванню є можливість відслідковування наявних запасних частин на головних складах та рівень їх забезпеченості на інших. Для підтримання ефективної діяльності підприємства є нагальна вимога розробки системи, що швидко та зручно звітує

про необхідність доповнення кількості запасних запчастин на віддалених складах.

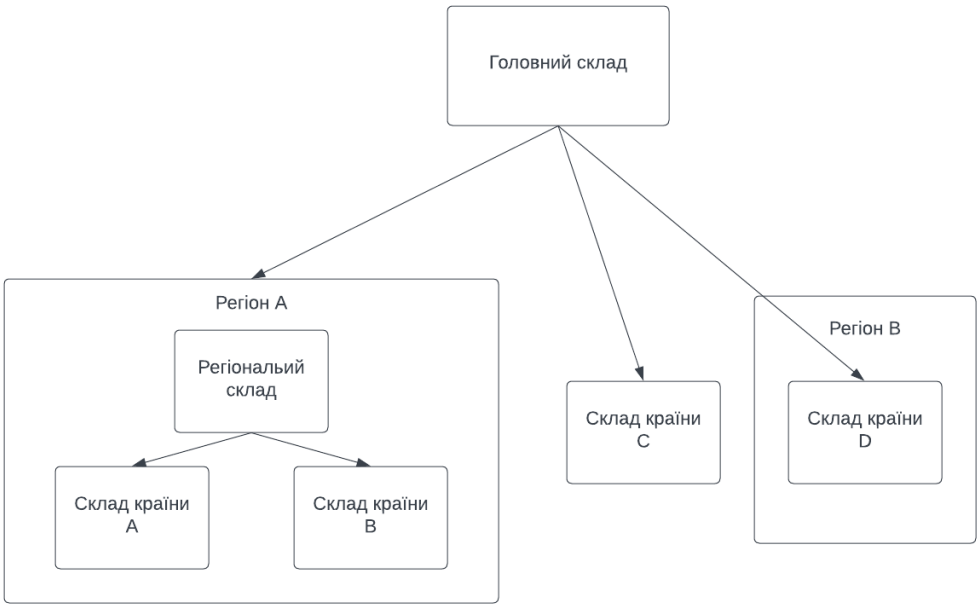


Рис. 1.1. Приклад ієрархії складів, демонструючий різні конфігурації організації

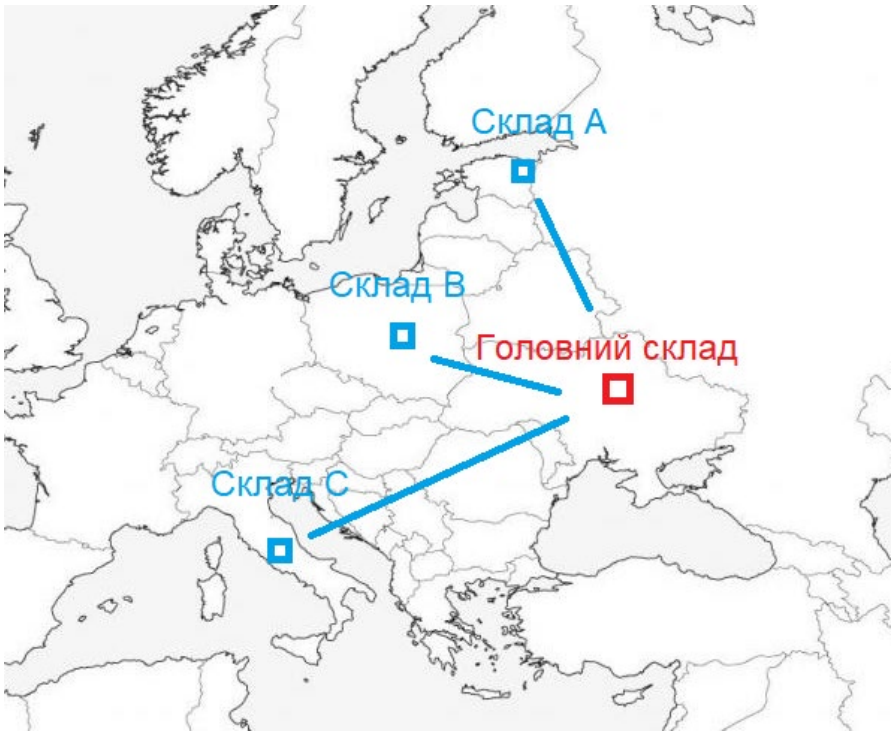


Рис. 1.2. Приклад розташування промислових складів компанії.

– Конфігурація джерел надходження даних з промислових складів: так як інформації щодо рівня забезпеченості складів є доволі несталою та завжди змінюється, система повинна включати в себе можливість налаштування джерел даних, їх обробка та трансформування.

– Безпека даних: так як дані про склад деталей, використання та їх кількість є конфіденційною інформацією компанії, система менеджменту повинна забезпечувати безпеку даних на максимальному рівні, наприклад шифрування та контроль доступу до неї.

Загалом, це лише кілька речей, які повинні бути включені у розробку системи. При проектуванні такої системи важливо уважно враховувати потреби підприємства та співпрацювати з фахівцями в цій сфері, щоб переконатися, що вона відповідає всім вимогам.

Застосування такої системи відносно широке, оскільки її можна використовувати для будь-якого промислового підприємства, що працює з великими обсягами товарів та деталей до них.

1.3. Підстава для розробки

В кінці навчання, студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою.

Підставою для розробки кваліфікаційної роботи на тему «Розробка комп'ютерної системи автоматизованого менеджменту промислових складів» є наказ по Національному технічному університету «Дніпровська політехніка» №469-с від 23.05.2024.

1.4. Постановка завдання

Метою даної кваліфікаційної бакалаврської роботи є підвищення ефективності процесу менеджменту товарів, деталей та запчастин у промислових

складах. Виходячи з поставленої мети, були сформовані наступні завдання: розробка системи швидкого доступу до інформації про наявність складів, моніторинг наповненості складів, відстеження рівня критичних запчастин на окремих складах, формування списків поповнення. Об'єктом розробки є автоматизована система управління промисловим складом, призначена для оптимізації цих операцій управління та обробки.

Основні суб'єктами вхідної інформації системи перелічені у таблиці 1.1.

Таблиця 1.1

Основні суб'єкти вхідної інформаційної системи

Промислові склади	Приміщення, розташовані в різних регіонах заданої території, що зберігають в собі вироблені товари, запчастини для їх виготовлення та запасні деталі у разі поломки.
Запасні деталі	Для кожної запчастини виготовленого продукту є маркування, чи є ця деталь запасною та швидкозмінною у разі поломки та рівень її критичності. Кожен промисловий склад у віддалені локації повинен мати певний рівень запасних деталей

Вихідна інформація з системи менеджменту складів наведена у наступній таблиці 1.2.

Таблиця 1.2

Вихідна інформація з системи менеджменту промислових складів

Дані наповненості складів	Стан наповненості складів товарами та запасними деталями відповідно до них станом на відправку запиту користувачем системи.

Продовж. табл. 1.2.

Звіт, щодо кількості деталей, що не вистачає	Згідно з конфігурацією системи та відповідно наявній кількості продуктів організації, розміщеній у певному регіоні, система менеджменту повинна генерувати звіт, що зазначає поточну кількість запчастин та нестачу, яку треба заповнити.
Сумісність деталей	Система повинна мати змогу при вводі товарів, їх під-версій – представляти перелік запасних деталей, які сумісні до них.

Кожна вихідна інформація повинна також містити додаткову інформацію щодо деталей, наприклад:

- Під-тип деталі
- Короткий її опис
- Серійний номер, якщо зазначено
- Параметри (висота, ширини, довжина)
- Вага запчастини
- Місцеположення деталі у складах

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблена комп’ютерна система має наступні функціональні характеристики:

– Система постійно надсилає запити до промислових складів щодо їх наповнення

– Користувач системи обирає потрібний тип експорту, зазначаючи цікавлений тип продуктів та їх кількість

– Система відповідно зіставляє введений тип введеної продукції з відповідними запасними деталями

– За формулою встановленою у конфігурації, система вираховую відповідну нестачу запчастин

– Формується вихідний файл, включаючи додаткові дані про деталі

– Користувач має змогу передивлятися та завантажувати вихідних файл.

1.5.2. Вимоги до інформаційної безпеки

Через сутність та чутливість інформації, з якою оперує комп'ютерна система – вони повинна мати належний рівень безпеки. Для його забезпечення – сервер системи розміщений на віртуальній машині у відокремленій комп'ютерній мережі. Цей сервер здійснює запити щодо наповнення складів використовуючи унікальний ключ до інтерфейсу API. Даний підхід унеможливлює доступ звичайних користувачів до повної інформації для звичайних користувачів.

Також до системи встановлено режим доступу тільки певній групі працівників організації. Доступ керується командою системних адміністраторів організації шляхом надання сертифікатів для використання VPN (Virtual Private Network), де розташовано веб-сервер який надає графічний інтерфейс взаємодії.

1.5.3. Вимоги до складу та параметрів технічних засобів

Клієнт системи автоматичного менеджменту промислових складів має змогу використовувати будь-якій пристрій, який має змогу під'єднання до комп'ютерної мережі з використанням VPN (комп'ютер, ноутбук, смартфон, планшет) та має змогу відображати веб-застосунки. Сервер системи менеджменту повинен забезпечити безперебійну роботу, швидкий час відповіді,

підтримувати безпечні протоколи передачі даних. Резервне копіювання для забезпечення цілісності та безпеки даних необхідно для таких типів розробок.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для забезпечення належного функціонування системи, серверне середовище має відповідати наступним критеріям: операційна система має бути сумісна з програмами, написаними на Python і Node.js. Підтримувані операційні системи включають системи на базі Unix і Microsoft Windows 7, 8 або 10.

Користувачі повинні мати обладнання, яке підтримує веб-браузери. Інтерфейс системи доступний через сучасні веб-браузери, такі як Google Chrome, Opera, Mozilla Firefox і Microsoft Edge.

Завдяки забезпеченню роботи системи на операційних системах Unix та Windows, це розширює її крос-платформну функціональність та діапазон потенційних середовищ розгортання, роблячи її більш універсальною та доступною. Обрані бібліотеки та фреймворки мають специфічні залежності, які повинні бути дотримані базовою операційною системою для коректної роботи.

Сумісність з браузерами: враховуючи залежність користувацького інтерфейсу від веб-технологій, забезпечення сумісності з широко використовуваними браузерами гарантує, що користувачі зможуть отримати доступ до системи без проблем, незалежно від того, якому браузеру вони надають перевагу.

Дотримуючись цих вимог сумісності, система управління складом може бути надійно розгорнута в різних середовищах, забезпечуючи оптимальну продуктивність і зручність для користувачів.

Висновок: аналіз підкреслює невід’ємну роль складів у логістиці та їхні багатогранні функції, окрім простого зберігання. Впроваджуючи систему для забезпечення швидкого доступу до даних інвентаризації, контролю рівня зберігання, відстеження критичних компонентів і оптимізації процесів

поповнення, організації можуть досягти кращої точності інвентаризації, зменшити операційні витрати та підвищити продуктивність завдяки автоматизації. Цей розділ закладає основу для наступних етапів проектування та розробки, підкреслюючи важливість гнучкого та масштабованого рішення в управлінні промисловістю. Загалом необхідність такої системи цілком виправдана, враховуючи складність і вимоги сучасних складських операцій.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Проаналізувавши наявні приклади, функціональне призначення даної системи полягає у створення системи автоматизованого менеджменту промислових складів. Розглянемо більш детального функціональне призначення:

– Збір та агрегування даних. Одним з головних призначень системи є централізований збір та агрегування даних про кількість деталей на промислових складах організації та відповідно їх параметри.

– Система розрахунку запасних деталей. Завдяки відслідковуванню наявних запасних частин на головних складах та рівень їх забезпеченості на інших – у користувача є можливість автоматичного моніторингу критично важливого рівня забезпеченості складів.

– Графічний інтерфейс та експорт даних. Перегляд стану наповненості складів здійснюється через веб-додаток. Завдяки ньому ж є і можливість експорту критичних рівнів складів у зручні формати табличних даних.

Рис. 2.1. Візуалізація функціоналу системи менеджменту складів

2.2 Опис застосованих математичних методів

При написанні даної кваліфікаційної бакалаврської роботи використовувалися наступні математичні методи: теорія систем масового обслуговування та статистика. Формалізований опис використаних методів є наступним:

– Для виробів типу A:
  - кількість деталей категорії P (поставляються разом з виробом) (формула 2.1);
  - кількість деталей категорії C (повинні знаходитись у країні, де розташовані вироби) (формула 2.2);
  - кількість деталей категорії R (деталі у регіоні, де розташовані вироби) (формула 2.3);
– Для виробів типу B:
  - кількість деталей категорії P (поставляються разом з виробом) (формула 2.4);
  - кількість деталей категорії C (повинні знаходитись у країні, де розташовані вироби) (формула 2.5);
  - кількість деталей категорії R (деталі у регіоні, де розташовані вироби) (формула 2.6);

$$Pa(n) = \lfloor \frac{n-1}{10} \rfloor + 1, \tag{2.1}$$

де P_a – необхідна мінімальна кількість запасних деталей типу P на складі країни,

n – кількість виробів у певній країні.

$$Ca(n) = \lfloor \frac{n-1}{10} \rfloor + 1, \quad (2.2)$$

де C_a – необхідна мінімальна кількість запасних деталей типу С на складі країни,

n – кількість виробів у певній країні.

$$Ra(n) = \lfloor \frac{n-1}{10} \rfloor + 1, \quad (2.3)$$

де R_a – необхідна мінімальна кількість запасних деталей типу R на складі регіону,

n – кількість виробів у певному регіоні.

$$P_b(n) = 0, \quad (2.5)$$

де P_b – необхідна мінімальна кількість запасних деталей типу Р на складі країни,

n – кількість виробів у певній країні.

$$C_b(n) = \lfloor \frac{n-1}{20} \rfloor + 1, \quad (2.6)$$

де C_b – необхідна мінімальна кількість запасних деталей типу С на складі країни,

n – кількість виробів у певній країні.

$$R_b(n) = \left\lfloor \frac{n-1}{100} \right\rfloor, \quad (2.7)$$

де R_b – необхідна мінімальна кількість запасних деталей типу R на складі країни, n – кількість виробів у певному регіоні.

2.3 Опис використаних технологій та мов програмування

При написанні даної кваліфікаційної бакалаврської роботи застосовується мова програмування Python. Перевагою цією мови є наявність готових бібліотек та модулей для швидкої та зручної взаємодії з табличними даними, керуванням, відправленням та отриманням HTTP-запитів, а також можливість розгортання веб-додатків. Для взаємодії з табличними даними був обраний пакет Pandas [17]. За відправлення та отримання HTTP-запитів відповідає вбудований пакет requests [18]. Так як відповідь з інтегрованої системи обліку складів відповідь приходять у форматі XML-документів: для їх форматування використаний пакет xmltodict [22]. Для звертання до файлів Google Sheets використовується модуль gspread [9] у тандемі з бібліотекою google-auth [7] та google-auth-oauthlib [8]. Для обслуговування клієнтської частини та відповіді на її запити використовується фреймворк Flask [21].

Для клієнтської частини (веб-додатку) використовується T3 Stack, що фокусується на модульності, простоті та типостійкості. Основні частини цього стеку є фреймворк Next.js [4] (що побудований на базі React), використанні мови TypeScript [5] та Tailwind CSS [20] – фреймворк, що дозволяє налаштовувати каскадні стилі HTML-сторінок без написанні великих об’ємів CSS. У якості основи для компонентів використані shadcn/ui [11] – модульні частини компонентів, основані на базі компонентної бібліотеки Radix UI [10]. Для збереження типізації та валідації типів при відправленні запитів через форму у веб-застосунку використовується бібліотека zod [23].

2.4 Опис структури системи та алгоритмів її функціонування.

Узагальнена схема роботи комп'ютерної системи автоматизованого менеджменту промислових складів наведена нижче (рис. 2.2).

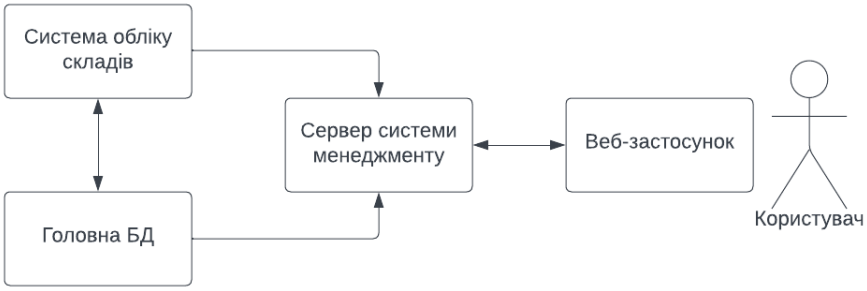


Рис. 2.2. Схема роботи системи автоматизованого менеджменту промислових складів

Загальний опис складових системи менеджменту складів:

- Система обліку складів. Використовується працівниками організації фізично розташованих у віддалених промислових складах для обліку наявних запчастин та запасних деталей.
- Головна БД. Зберігає перелік усіх запасних деталей та запчастин організації з їхніми фізичними показниками.
- Сервер системи менеджменту. Постійно запитує актуальну інформацію з системи обліку та головної БД для кешування. Відповідає на запити веб-застосунку щодо перегляду наявних запчастин у віддалених промислових складах; формування списків деталей, що не вистачає згідно заданим користувачем параметрам; генерування вихідних даних у форматах для перегляду табличних даних: CSV, Microsoft Excel Document.
- Веб-застосунок. Використовується користувачами для взаємодії з системою автоматизованого менеджменту промислових складів. Надає змогу перегляду наявних запчастин та запасних деталей на віддалених складах станом на момент відправки запиту; представляє гнучкий вибір параметрів для формування списків деталей, що не вистачає на віддалених складах; є шлюзом для отримання файлів табличного типу.

Принцип роботи користувацького інтерфейсу наведений нижче:

- Алгоритм отримання переліку промислових складів станом на зараз наведено на рис. 2.3.

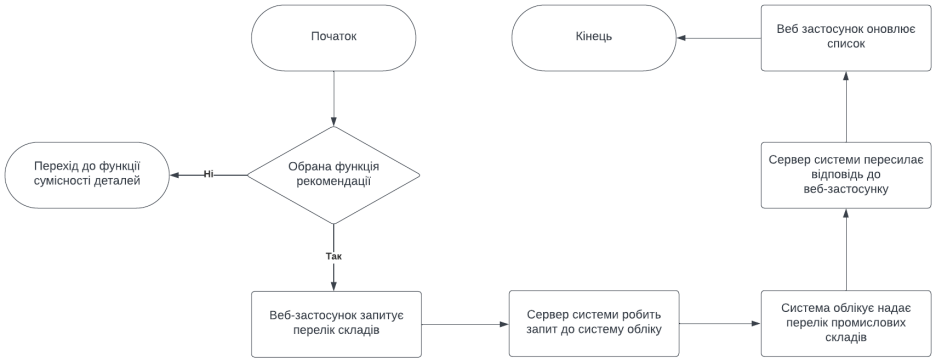


Рис. 2.3. Алгоритм отримання актуального переліку промислових складів

- Алгоритм відображення фільтрів за категоріями товарів (рис. 2.4).

Рис. 2.4. Алгоритм відображення опції фільтрування запасних деталей для користувацького інтерфейсу

Сервер системи автоматизованого менеджменту промислових складів побудований з використанням архітектури REST. Тобто, у замкненому постійному циклі сервер асинхронно відповідає на запити веб-застосунку. Алгоритми цих відповідей та обробки даних наведені нижче:

- Процес отримання переліку наявних запасних деталей та запчастин на промислових складах наведено на рис. 2.5.

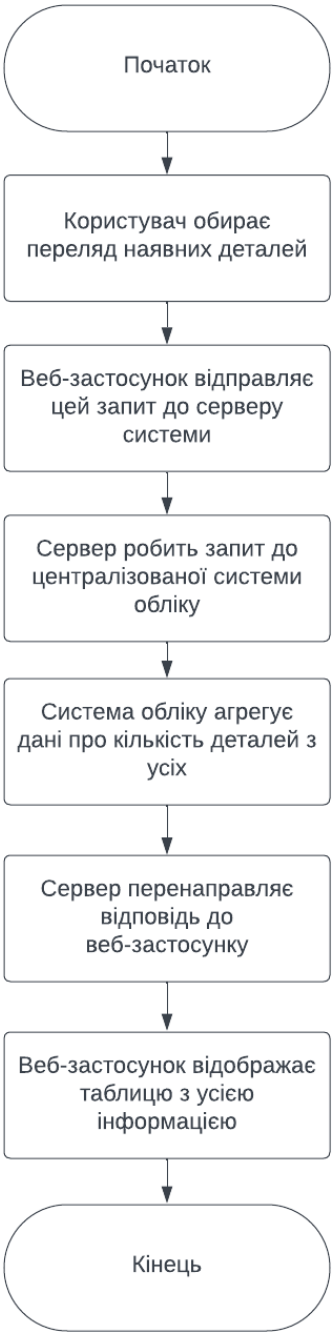


Рис. 2.5. Алгоритм отримання користувачем переліку усіх наявних запчастин та деталей на віддалених промислових складах

- Відображення таблиці сумісності запчастин та запасних деталей відповідно до обраних користувачем фільтрів категорії, типу та версії продукції організації описано на рис. 2.6.

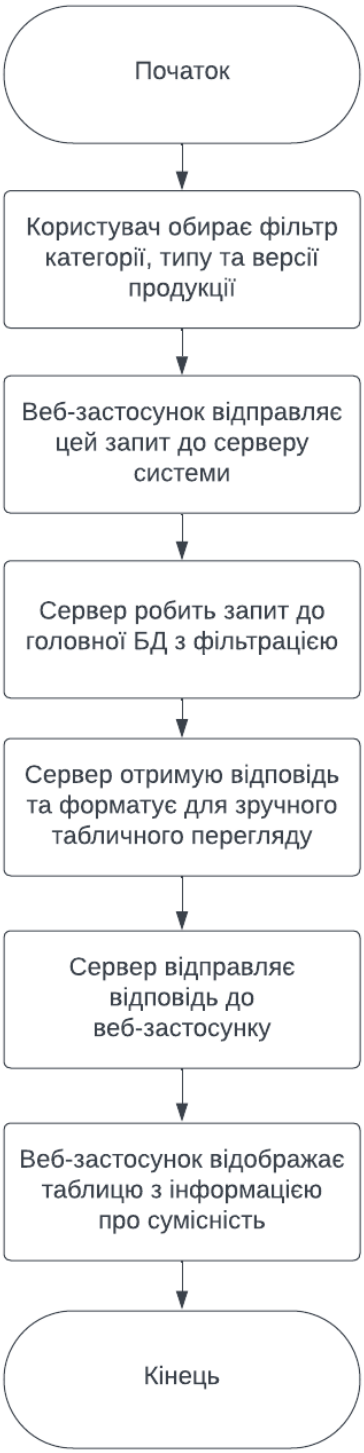


Рис. 2.6. Алгоритм отримання переліку деталей та запасних запчастин, їх характеристик та сумісності за заданим користувачем фільтром

- Алгоритм отримання рекомендації щодо поповнення критично важливих деталей рівнів R, C та P у віддалених промислових складах (рис. 2.7).

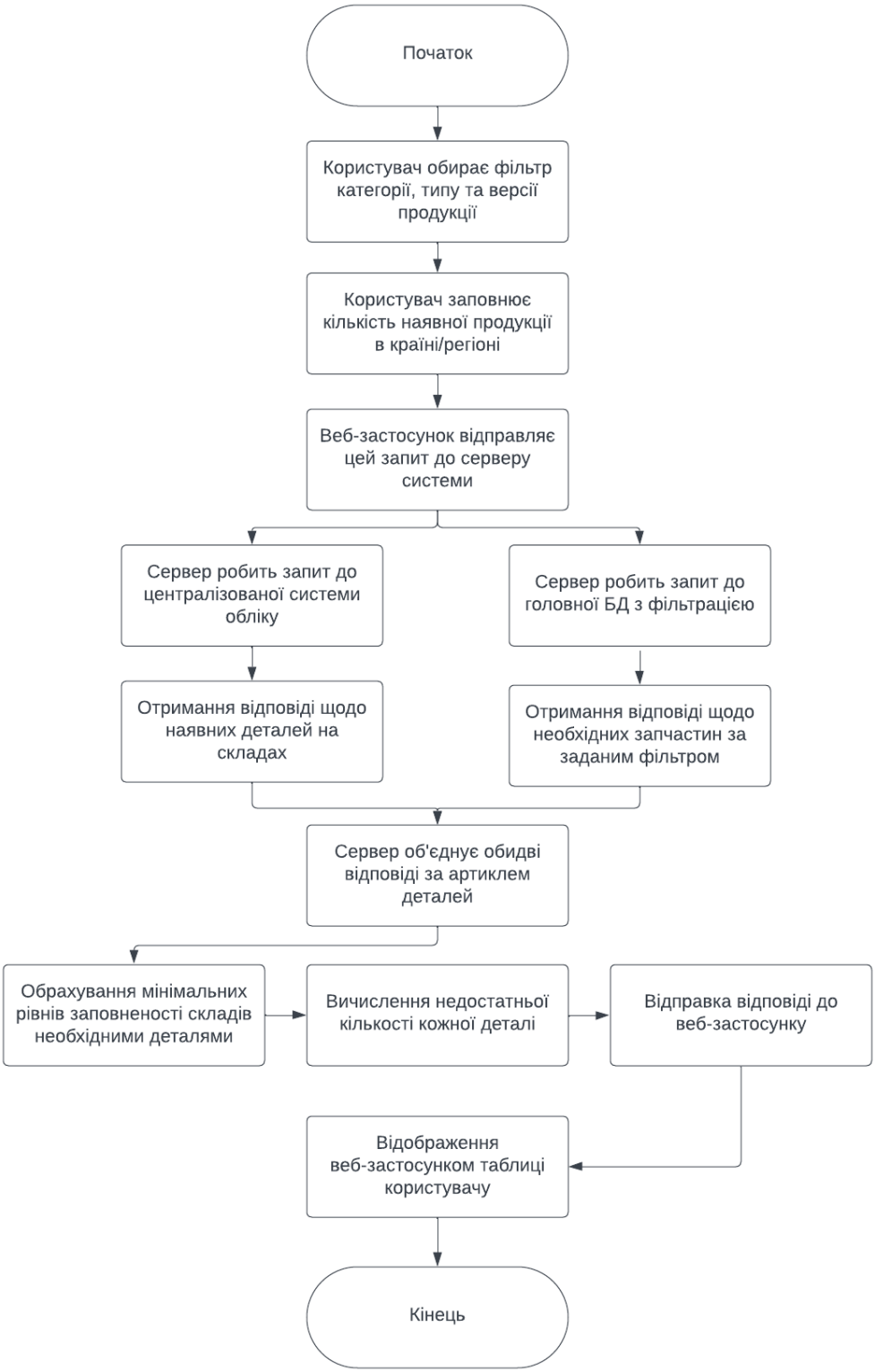


Рис. 2.7. Алгоритм рекомендації заповнення запасних деталей та запчастин у віддалених промислових складах

2.5 Обґрунтування та організація вхідних та вихідних даних програми

У створеній системі автоматизованого менеджменту промислових складів вхідними даними являються:

– Перелік промислових складів (включаючи головний та віддалені);

– Головна база даних. Містить у собі увесь перелік артиклів існуючих запчастин та запасних деталей в організації та відомості про них, такі як: вартість, розміри, вага, необхідна мінімальна кількість, ієрархічні відносини тощо. Приклад записів у головні базі даних наведено нижче (рис. 2.8).

Article	Description	Price	Replacable?	Smart Part?	Visible?	Height (m)	Width (m)	Depth (m)	Weight (kg)
F0001	Aku 12V/12AH-P	€94.60	Y	N	N	0.013	0.014	0.013	0.0008
F0448	Display Controlle	€30.00	Y	N	N	0.013	0.014	0.013	0.0008
1076	Console Right Sli	€126.25	Y	Y	Y				
152230.R1	Console Left Slid	€28.13	Y	N	N				2.2610
X1085	Service Door Ass	€26.10	Y	N	N	0.030	0.139	0.093	0.1600
7001	Serial cable D9M	€30.09	Y	N	Y	0.018	0.084	0.072	0.3200
166309	Button Cell Batte	€15.85	Y	Y	N				
176428	Loader 04.04 (L2	€49.70	Y	N	Y	0.001	0.015	0.011	0.0000
X2637	Serial cable D9M	€94.60	N	N	Y	0.030	0.139	0.093	0.0760
PL0032	Photoelectric Sen	€195.00	Y	N	Y	0.019	0.615	0.538	2.0030
X1340	Carrier LYNX Tele	€52.23	Y	Y	Y				0.0490
127733	Console tray cove	€117.58	Y	N	N				0.3230
X1544	Aluver Door VU te	€23.10	Y	N	N	0.050	5.485	0.197	0.0000
1481	Roof Beam	€112.88	Y	N	N				
X3035	Lock Kerong KR-	€24.72	Y	N	Y	0.020	0.094	0.073	0.0480
X0929	Micro Link 3 USB	€121.38	N	Y	N				
X1342	Eccentric Bushing	€159.75	Y	N	N	0.019	0.153	0.031	0.0130
X1341	Bolt Concentric L	€112.88	Y	N	N	0.018	0.153	0.028	0.0110
CL-SC	Molex Micro Fit s	€112.88	Y	N	N	0.034	0.136	0.077	0.0800
130963	Concrete Screw V	€8.00	Y	N	Y	0.027	0.044	0.031	0.0500
X1286	Temperature sens	€112.88	Y	Y	N				0.0710
153873.R0	Electric Bolt Morti	€58.08	Y	N	N	0.017	0.060	0.053	0.0890
X0904	Outdoor Router H	€482.90	N	Y	N				0.1110
113221	5604-2109 T4 - P	€72.72	Y	Y	N	0.019	0.720	0.023	0.2480
163159.R1	Southco-EM-05-1	€118.01	Y	N	N	0.261	0.750	0.495	1.3350
176279-K1.R0	Switch D-Link 5-p	€54.75	Y	N	N	0.030	1.960	0.050	2.1170
BR650MI	Aluver Door VU te	€25.00	Y	Y	N	0.091	0.313	0.190	7710.0000
X1023	Sliding piece inac	€54.75	Y	N	N	0.012	0.030	0.029	0.0170
DR22	Tray guide - rev 0	€54.75	Y	N	N	0.018	0.022	0.022	0.0090
132639	Dahua DH-PFS3	€110.88	N	N	Y				0.0050
145218	Southco-EM-05-1	€48.95	Y	N	Y				0.0290
106758	ARC-1232-610B1	€70.43	Y	N	Y	0.014	0.023	0.023	0.0020
147146.R6	Concrete Screw V	€70.35	Y	N	Y	0.054	0.194	0.140	0.8760
X2938	Aluver Door VU te	€574.00	Y	Y	Y	0.075	0.105	0.091	0.0550
127880	TOP Hinge 55037	€161.43	Y	N	N	0.015	0.688	0.088	0.0420
113305	AC Servo Drive 1	€845.42	Y	N	N	0.078	0.154	0.108	1.1780
130963	Finder Semicondu	€154.83	Y	Y	Y	0.078	0.154	0.108	1.1560
126991	Roof Beam	€281.98	Y	Y	Y				1.3760
143279	toiteplokk Meanw	€7.05	Y	N	N	0.020	0.719	0.122	0.4410
X0976	Label printer BT-U	€25.00	Y	Y	N				0.1110
X0880	UPS APC Smar	€28.66	Y	Y	N	0.045	0.112	0.099	300.0000
RB750	Outdoor Router H	€128.00	Y	Y	N	0.030	0.113	0.089	0.2660
X1113	Photoelectric Sen	€45.90	Y	Y	Y	0.019	0.087	0.064	0.0080
163173	LoaderÂ´s side ri	€28.00	Y	Y	Y	0.056	0.323	0.262	3.4940
163850-K1	LoaderÂ´s side as	€28.00	Y	Y	N				0.6810
174623-K2.R0	Micro Link 3 USB	€23.85	Y	N	N	0.027	0.253	0.041	0.1060
MGN12CZ0HM	UPS APC Smar	€926.67	Y	N	N	0.010	0.035	0.027	0.0070
144964	Outdoor Router H	€19.77	Y	N	N				0.0000

Рис. 2.8. Приклад записів усіх запасних деталей та запчастин у головній базі даних

На виході користувач має змогу отримати такі дані, як:

– Перелік активних промислових складів;

– Наявні категорії виробів організації та їх версії;

– За запитом, перелік усіх запасний деталей та запчастин, що підходять до зазначених типів виробів та їх версій;

– Перелік деталей та запчастин, що знаходяться на віддалених промислових складах, їх кількість та кількість, яку необхідно відправити для відповідності мінімальним вимогам.

Article	Type	Description (ENG)	location	Price
X1778	C	BT10 Left loader front beam transmitter	LOADERS	€112.75
136313.R3	C	Loader telescope home sensor flag - rev 3	LOADER	€29.70
136313.R3	C	Loader telescope home sensor flag - rev 3	LOADERS	€29.70
136313.R3	C	Loader telescope home sensor flag - rev 3	LOADERS	€29.70
136313.R3	C	Loader telescope home sensor flag - rev 3	LOADERS	€29.70
F0001	C	Plastic rivet 8x12 DRR013-BLK	FRAME	€0.00
F0001	C	Plastic rivet 8x12 DRR013-BLK	FRAME	€0.00
F0001	C	Plastic rivet 8x12 DRR013-BLK	FRAME	€0.00
143279	C	402 Loader side - single - rev 1	LOADERS	€15.00

Рис. 2.9. Приклад візуальної репрезентації вихідних даних системи автоматизованого менеджменту промислових складів у веб-застосунку

2.6 Опис розробленої системи

2.6.1 Використані технічні засоби

Під час розробки системи автоматизованого менеджменту промислових складів було використано:

– Персональний комп’ютер. Використовується для локальної розробки та відловлювання помилок в процесі розробки ПЗ.

– Віртуальна машина. Віддалений сервер у хмарі, який розташовує на собі готову версію систему, включаючи як сервер так і веб-застосунок.

2.6.2 Використані програмні засоби

Локальна розробка системи автоматизованого менеджменту промислових складів була проведена на персональному комп’ютері з використанням операційної системи Microsoft Windows 10.

Віртуальна машина, на якій відбувається запуск готової версії системи менеджменту використовує операційну системи дистрибутиву Linux з відкритим кодом – Debian.

Безпосередня розробка виконувалась у інтегрованому середовище для програмування Visual Code від компанії Microsoft.

Для розробки серверної частини програми була використана систему керування пакунками Python – pip. Вона ж слугує для контролювання віртуальних середовищ розробки.

Керування пакунками мови програмування JavaScript було здійснено з використанням менеджеру npm. Він включає в себе клієнт для командного рядка, а також онлайн базу даних публічних пакунків.

Передавання файлів до віртуальної машини у хмарі здійснюється через FileZilla – вільний багатоплатформний клієнт з відкритим кодом.

2.6.3 Виклик та завантаження програми

Перед розгортанням системи автоматизованого менеджменту промислових складів на сервері, необхідно створити віртуальне середовище для мови Python та встановити пакунки, що використовувались при розробці програмного продукту. Для цього знаходячись в директорії з серверним кодом необхідно виконати команду: pip install -r requirements.txt

Для розгортання веб-частини системи менеджменту необхідно виконати процес збирання веб-застосунку. Знаходячись в директорії веб-серверу необхідно виконати команду next build.

Виклик програми виконується з використанням менеджерів пакунків pip для Python та npm для JavaScript:

– Серверна частина програми запускається командою: flask --app app run

– Веб-сервер застосунку запускається командою: next start

Інтерфейсом системи для користувача є веб-застосунок. Він доступний за публічною IP-адресою віртуального серверу у хмарі через сучасні браузери:

– Chrome версії 64 та вище;

– Edge версії 79 та вище;

– Firefox версії 67 та вище;

– Opera версії 51 та вище;

– Safari версії 12 та вище.

2.6.4 Опис інтерфейсу користувача

Візуальний користувацький інтерфейс розроблений з використанням готових гнучких компонентів shadcn/ui. Під час розробки були модифіковані та використані наступні компоненти:

– Кнопка (рис. 2.10).

Рис. 2.10. Компонент типу кнопка

– Компонент прапорцю (рис. 2.11).

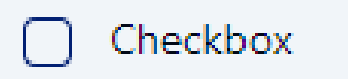


Рис. 2.11. Компонент типу кнопка

– Група прапорців (рис. 2.12).

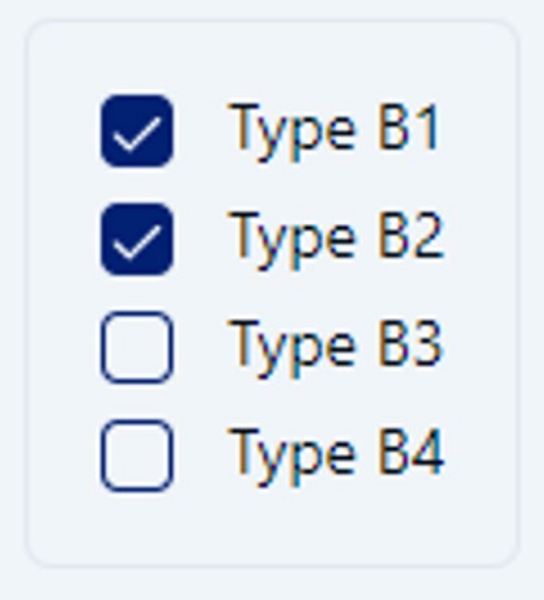


Рис. 2.12. Група прапорців

– Поле для вводу (рис. 2.13).

Рис. 2.13. Поле для цифрового вводу

– Група перемикачів radio-button (рис. 2.14).

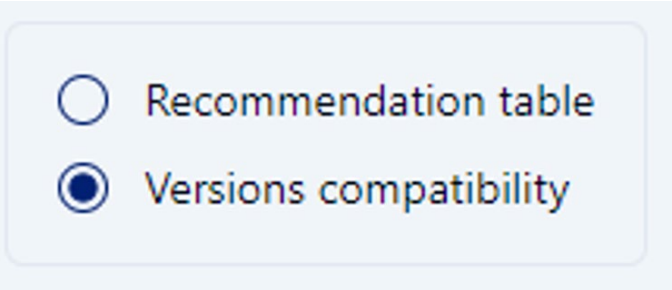


Рис. 2.14. Компонент radio-button

– Компонент форми був використаний для структурування зазначених вище компонентів та валідації їх типів (рис. 2.15).

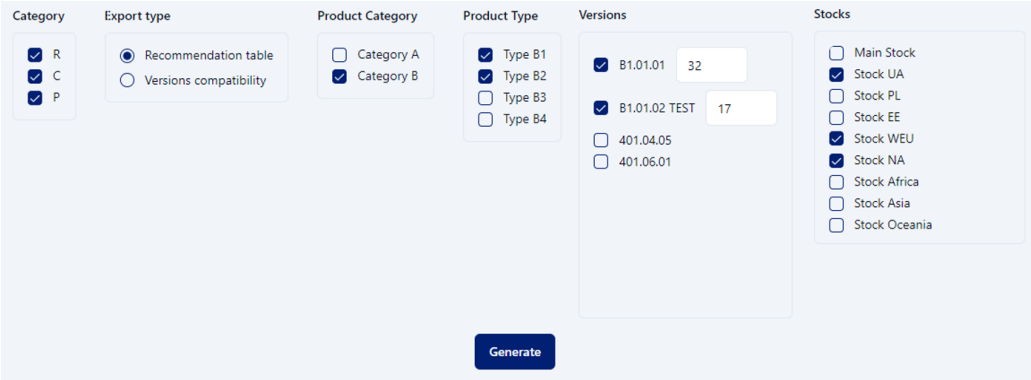


Рис. 2.15. Використання компоненту форми

– Компонент таблиці (рис. 2.16).

Article	The recommended amount of spare parts	Main Stock	Stock UA	Stock UA to replenish	Category	Type	Compatibility with version
X1778	40	95257	124	0	Category A	Type A1	A1.01.01
136313	2	-8	0	2	Category A	Type A1	A1.01.02
136313.R3	3	16	11	0	Category B	Type B2	B2.01.01
136313.R3	2	11	0	2	Category A	Type A1	A1.01.01
136313.R3	1	28	0	1	Category A	Type A1	A1.01.02
F0001	1	2	1	0	Category B	Type B2	B2.01.01
F0001	3	16	12	0	Category B	Type B2	B2.01.01
F0001	2	2988	0	2	Category A	Type A1	A1.01.01
143279	1	4	1	0	Category A	Type A1	A1.01.02
143279	1	1	1	0	Category A	Type A1	A1.01.02

Рис. 2.16. Використання компоненту таблиці

При звертання до веб-інтерфейсу системи автоматизованого менеджменту промислових складів користувач опиняється на головній сторінці. Візуальне відображення наведене на рис. 2.17.

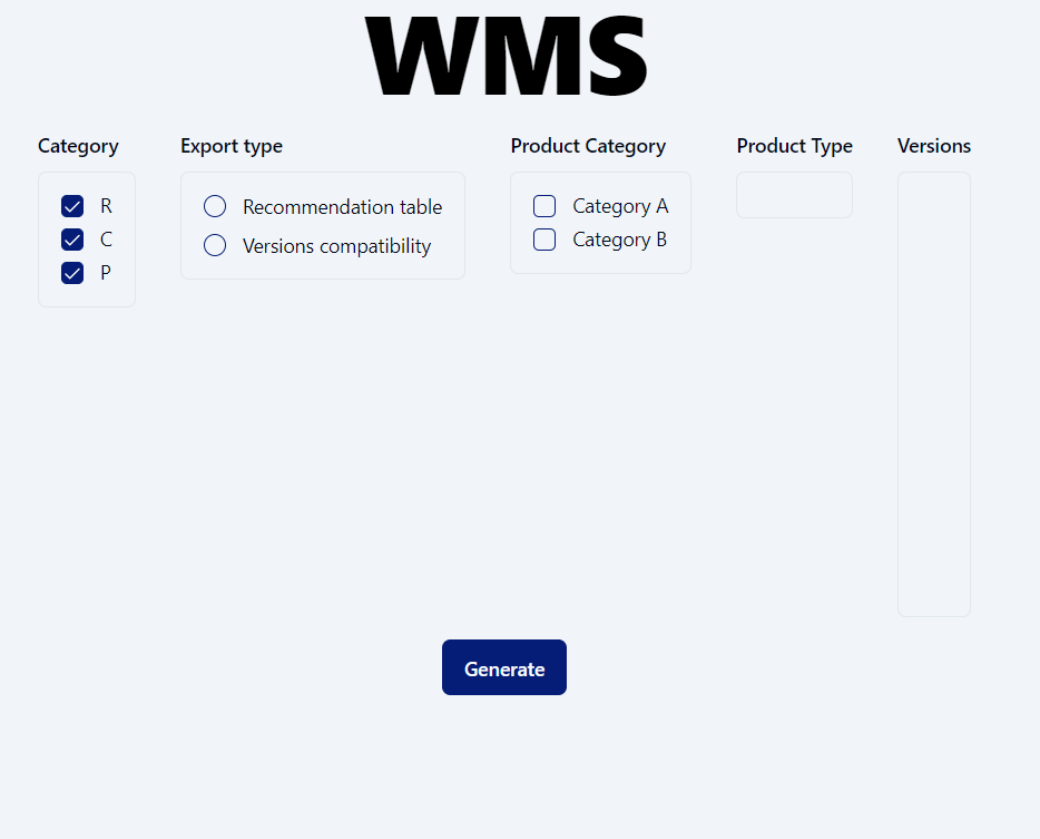


Рис. 2.17. Головна сторінка системи автоматизованого менеджменту промислових складів

Далі користувач система має змогу вибрати необхідний наразі йому тип дії. Обирання функції перегляду усіх запасних деталей та запчастин, їх відомостей та фізичних характеристик продемонстровано на рис. 2.18.

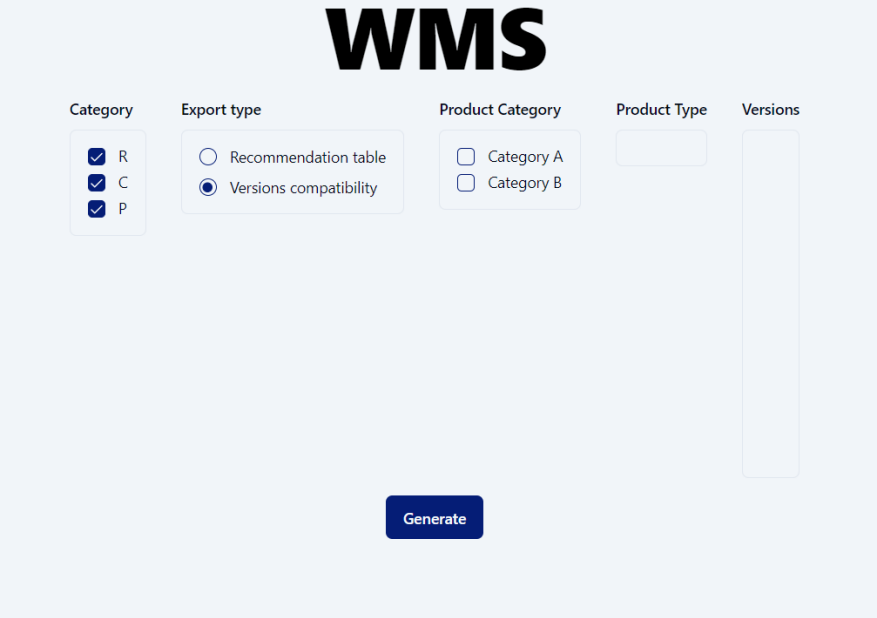


Рис. 2.18. Варіант обирання перегляду сумісності запасних деталей та запчастин

Приклад обирання рекомендаційної підсистеми для виведення переліку деталей, що необхідно доставити у віддалені промислові склади наведено на рис. 2.19.

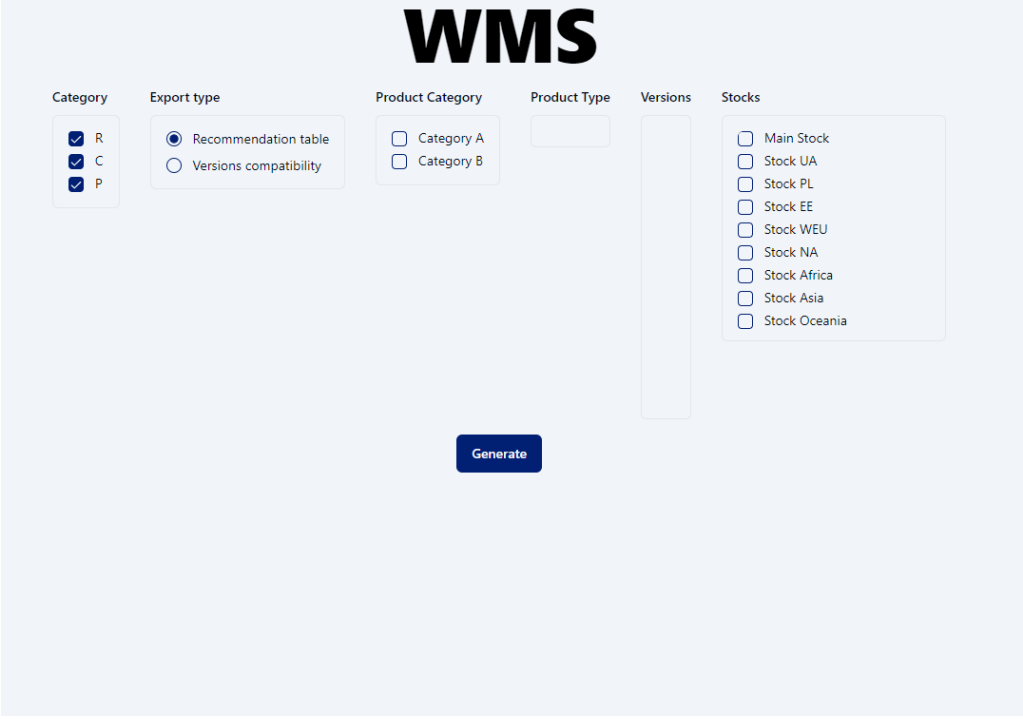


Рис. 2.19. Варіант обирання перегляду рекомендаційної підсистеми щодо заповнення віддалених промислових складів

Після обирання одного з двох варіантів, користувач має налаштувати фільтр запиту, щодо яких виробів організації він бажає отримати відповідну інформацію. Це відбувається шляхом обирання фільтрів для категорії продукту, типу та версій. Приклад зображено на рис. 2.20.

Рис. 2.20. Приклад обирання категорії, типів для перегляду сумісності компонентів до відповідних версій продуктів організації

Для формування перегляду рекомендацій, щодо наповнення промислових складів необхідними критичними запчастинами та запасними деталями відповідно наявній кількості виробів організації у країні/регіоні, користувач після обирання версій продукту повинен ввести вручну їх відповідну кількість. Приклад користувацького вводу є на рис. 2.21.

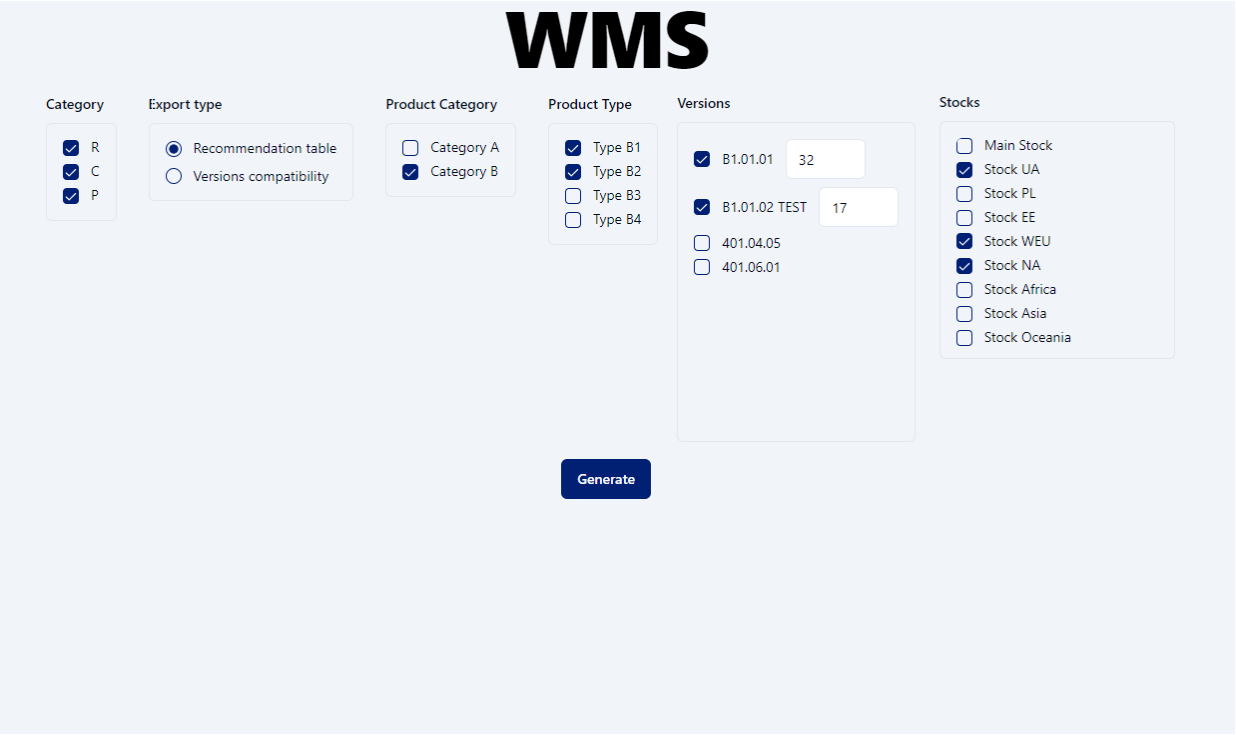


Рис. 2.21. Приклад обирання категорії, типів та версій продукту для перегляду рекомендацій щодо наповнення промислових складів

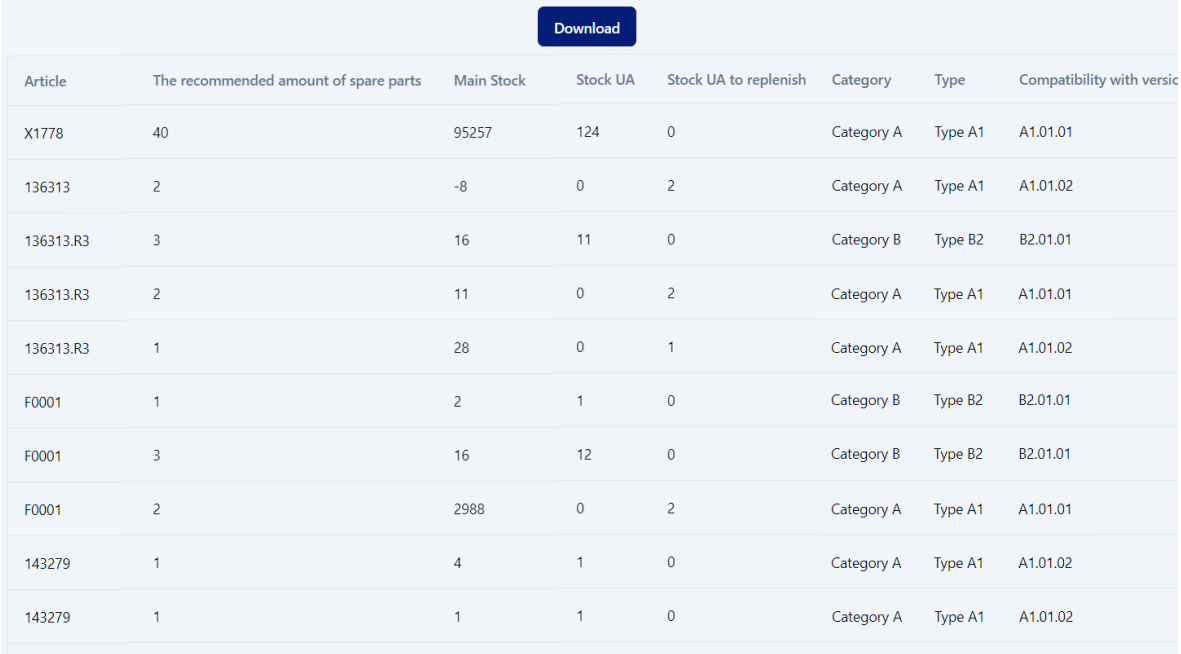
Приклад перегляду сумісності компонентів виробів до відповідно обраних категорій, типів та версій продуктів організації наведено на рис 2.22.

Download

Article	Type	Description (ENG)	Depth (m)	Width (m)	Height (m)	Weight (kg)	Category	Type	Compatibility with version
X1778	C	IME2S12-08N4DC0 safety inductive sensor	0.010	0.044	0.004	0.0110	Category A	Type A1	A1.01.01
136313	C	IME2S12-08N4DC0 safety inductive sensor	0.018	0.029	0.009	0.0010	Category A	Type A1	A1.01.02
136313.R3	C	Inductive Sensor, PNP-NO,65mm, (Blue tip)	0.018	0.029	0.009	0.0010	Category B	Type B2	B2.01.01
136313.R3	C	PM-K65-P Panasonic Sensor	0.018	0.029	0.009	0.0010	Category A	Type A1	A1.01.01
136313.R3	C	PM-K65-P Panasonic Sensor	0.018	0.029	0.009	0.0010	Category A	Type A1	A1.01.02
F0001	C	Ultra-slim PCB Relay, 24V, 6A, 1 SPDT-CO 34.51.7.024.0010	0.013	0.014	0.013	0.0008	Category B	Type B2	B2.01.01
F0001	C	PM-K65-P Panasonic Sensor	0.013	0.014	0.013	0.0008	Category B	Type B2	B2.01.01
F0001	C	Plate for inductive sensor - rev 0	0.013	0.014	0.013	0.0008	Category A	Type A1	A1.01.01
143279	C	Ultra-slim PCB Relay, 24V, 6A, 1 SPDT-CO 34.51.7.024.0010	0.122	0.719	0.020	0.4410	Category A	Type A1	A1.01.02
143279	C	Ultra-slim PCB Relay, 24V, 6A, 1 SPDT-CO 34.51.7.024.0010	0.122	0.719	0.020	0.4410	Category A	Type A1	A1.01.02

Рис. 2.22. Приклад перегляду сумісності деталей у системі менеджменту промислових складів

Приклад візуальної репрезентації табличних даних підсистеми рекомендації щодо наповнення віддалених промислових складів наведено на рис. 2.23.



Article	The recommended amount of spare parts	Main Stock	Stock UA	Stock UA to replenish	Category	Type	Compatibility with versic
X1778	40	95257	124	0	Category A	Type A1	A1.01.01
136313	2	-8	0	2	Category A	Type A1	A1.01.02
136313.R3	3	16	11	0	Category B	Type B2	B2.01.01
136313.R3	2	11	0	2	Category A	Type A1	A1.01.01
136313.R3	1	28	0	1	Category A	Type A1	A1.01.02
F0001	1	2	1	0	Category B	Type B2	B2.01.01
F0001	3	16	12	0	Category B	Type B2	B2.01.01
F0001	2	2988	0	2	Category A	Type A1	A1.01.01
143279	1	4	1	0	Category A	Type A1	A1.01.02
143279	1	1	1	0	Category A	Type A1	A1.01.02

Рис. 2.23. Приклад перегляду рекомендацій щодо наповнення промислових складів

Висновок: було спроектовано та розроблено програмне забезпечення для автоматизованого управління промисловими складами, зосереджуючись на його функціональному призначенні та застосованих математичних методах. Функціональне призначення підкреслює автоматизований збір і агрегацію даних для ефективного управління запасами. Також підкреслюється використання сучасних веб-технологій і мов програмування, які забезпечують сумісність системи з різними платформами та браузерами. Структура системи розроблена як модульна, що забезпечує масштабованість і легкість обслуговування. Ефективно організовуючи вхідні та вихідні дані, розроблена система забезпечує точне й надійне відстеження запасів і керування ними. Загалом у цій главі представлено вичерпний огляд технічної основи та практичної реалізації системи управління складом, створюючи надійну основу для її та експлуатації.

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1815;
2. коефіцієнт складності програми – 1,5;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста – 475 грн/год;

За проведеним опитуванням зими 2024 спільнотою розробників DOU, середня заробітна плата розробника становить 80 тисяч гривень. Робочий графік розробника складається з 21 робочого дня по 8 годин на день. Отже, погодинна ставка становитиме (80 тисяч гривень / 8 годин) / 21 робочий день ≈ 475 гривень на годину.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,1;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,4;
7. вартість машино-години ЕОМ – 0,61 грн/год.

У ході написання кваліфікаційної роботи єдиними витратами, які вплинули на підрахунок є витрати на інтернет з’єднання та витрати на електроенергію під час роботи за ноутбуком. Згідно за офіційними даними України на момент написання роботи, вартість 1 кВт/год становила 3,6 гривень. Домашній інтернет коштував 180 гривень (за стабільне з’єднання 100 Мбіт/сек) на місяць. Ноутбук споживав 70 Вт, тож вартість електроенергії за місяць використання в робочий

час становила 70 * 3,6 * 8 * 21 / 1000 = 42,34 гривень. Таким чином, вартість машино-години комп'ютера становила (60 + 42,34) / 168 = 0,61 гривень на місяць.

Трудомісткість розроблення ПЗ може бути розрахована з використанням системи моделей, що пропонують різні рівні точності оцінки. Трудомісткість розроблення програмного забезпечення може бути визначена за такою формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1815);

C - коефіцієнт складності програми (1,5);

p - коефіцієнт корекції програми в ході її розробки (0,2).

Після підставлення значень умовне число операторів дорівнює:

$$Q = 1815 \cdot 1{,}5 \cdot (1 + 0{,}2) = 3267$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.2). Після підставлення значень маємо:

$$t_u = (3267 \cdot 1{,}2) \,/\, (75 \cdot 1{,}2) = 43{,}56 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25) \cdot k}, \text{ людино-годин, } (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 3267 \,/\, (20 \cdot 1{,}2) = 136{,}125 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_{\text{п}} = \frac{Q}{(20...25) \cdot k}, \text{ людино-годин.}$$

$$t_п = 3267 \,/\, (25 \cdot 1{,}2) = 108{,}9 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5)\cdot k} \text{, людино-годин.}$$

Підставивши значення:

$$t_{отл} = 3267 \,/\, (5 \cdot 1{,}2) = 544{,}5 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t^{k}_{\text{отл}} = 1{,}5 \cdot t_{\text{отл}}\text{, людино-годин.}$$

$$t^{k}_{отл} = 1{,}5 \cdot 544{,}5 \ = 816{,}75 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до} \text{ , людино-годин,}$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{\text{др}} = \frac{Q}{(15..20)\cdot k} \text{ , людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0{,}75 \cdot t_{\text{др}} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 3267 \,/\, (19 \cdot 1{,}2) = 143{,}29 \text{ людино-годин.}$$

$$t_{до} = 0{,}75 \cdot 143{,}29 = 107{,}47 \text{ людино-годин.}$$

$$t_{д} = 143{,}29 + 107{,}47 = 250{,}76 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 43{,}56 + 136{,}125 + 108{,}9 + 544{,}5 + 250{,}76 = 1133{,}85 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 475 грн / год, отримуємо:

$$З_{ЗП} = 1133{,}85 \cdot 475 = 538\,578{,}75 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (0,61 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 544{,}5 \cdot 0{,}61 = 332{,}15 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 538\,578{,}75 + 332{,}15 = 538\,910{,}9 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин);

t – загальна трудомісткість, людино годин()

Звідси витрати на створення програмного продукту:

$$T = 1133{,}85 \,/\, 1 \cdot 176 \approx 6{,}44 \text{ міс.}$$

Висновок: остаточні витрати на розробку комп'ютерної системи автоматизованого менеджменту промислових складів становлять 538 910,9 гривень. Розрахунки показують, що термін створення програмного продукту буде становити 6,44 місяців. У цей період включено час на виправлення помилок та уточнення завдань у зв'язку з їхньою неточністю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alias, C., Salewski, U., Ortiz Ruiz, V.E., Adapting warehouse management systems to the requirements of the evolving era of industry 4.0. : American Society of Mechanical Engineers, 2017. 50 с.

2. Deng, M., Mao, J., Gan, X., Development of automated warehouse management system : EDP Sciences. 2018. 6 с.

3. Docs | Next.js: URL https://nextjs.org/docs (дата звернення 04.06.2024)

4. Documentation - TypeScript for JavaScript Programmers: URL https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html (дата звернення 04.06.2024)

5. Fauzan, R., Shiddiq, M.F., Raddlya, N.R., The designing of warehouse management information system. : IOP Publishing, 2020. 8 с.

6. google-auth — google-auth 1.30.0 documentation: URL https://google-auth.readthedocs.io/en/master/дата звернення 04.06.2024)

7. google-auth-oauthlib - Read the Docs: URL https://google-auth-oauthlib.readthedocs.io/en/latest/ (дата звернення 04.06.2024)

8. gspread – gspread 6.0.0 documentation: URL https://docs.gspread.org/en/v6.0.0/ (дата звернення 04.06.2024)

9. Introduction – Radix Primitives: URL https://www.radix-ui.com/primitives/docs/overview/introduction (дата звернення 04.06.2024)

10. Introduction – shadcn/ui: URL https://ui.shadcn.com/docs (дата звернення 04.06.2024)

11. Ivgantius, T.Z., Andry, J.F., Development of warehouse management system using Extreme Programming : International Journal of Engineering and Information Systems, 2019. 46 с.

12. Khan, M.G., Huda, N.U., Smart Warehouse Management System: Architecture, Real-Time Implementation and Prototype Design : Machines, 2022. 150 с.

13. Lee, C.K., Lv, Y., Ng, K.K., Design and application of Internet of things-based warehouse management system for smart logistics : International Journal of Production Research. 2018. 13 c.

14. Mao, J., Xing, H., Zhang, X., Design of intelligent warehouse management system : Wireless Personal Communications, 2018. 14 c.

15. Min, H., The applications of warehouse management systems: an exploratory study : International Journal of Logistics, 2006. 14.c.

16. pandas 2.2.2 documentation: URL https://pandas.pydata.org/docs/ (дата звернення 04.06.2024)

17. Requests – PyPI: URL https://pypi.org/project/requests/ (дата звернення 04.06.2024)

18. Revenue of the e-commerce industry in the U.S. 2019-2029: URL https://www.statista.com/statistics/272391/us-retail-e-commerce-sales-forecast/ (дата звернення 19.05.2024)

19. Tailwind Documentation: URL https://v2.tailwindcss.com/docs (дата звернення 04.06.2024)

20. Welcome to Flask – Flask Documentation (3.0.x): URL https://flask.palletsprojects.com/en/3.0.x/ (дата звернення 04.06.2024)

21. xmltodict - PyPI: URL https://pypi.org/project/xmltodict/ (дата звернення 04.06.2024)

22. Zod | Documentation : URL https://zod.dev/ (дата звернення 04.06.2024)

23. Žunić, E., Delalić, S., Hodžić, K., Smart warehouse management system concept with implementation : Neural Networks and Applications (NEUREL), 2018. 5 c.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

app.py

```
# Import necessary modules and functions
from flask import Flask, request, jsonify, send_file
from flask_cors import CORS

from data.sheets import getMajorSeries, getSeries, getVersions, getExport
from data.warehouses import getAllWarehouses

# Initialize Flask app and configure CORS
app = Flask(__name__)
CORS(app)
app.config["CORS_HEADERS"] = "Content-Type"
app.config["CORS_HEADERS"] = "Access-Control-Allow-Origin"

# Define route to get major series
@app.route("/majorSeries", methods=["GET"])
def majorSeries():
    return jsonify(getMajorSeries())

# Define route to get series data
@app.route("/series", methods=["GET", "POST", "OPTIONS"])
def series():
    seriesBody = request.get_json(silent=True)
    print(seriesBody)
    return jsonify(getSeries(seriesBody))

# Define route to get versions data
@app.route("/versions", methods=["GET", "POST", "OPTIONS"])
def versions():
    versionsBody = request.get_json(silent=True)
    print(versionsBody)
    result = getVersions(versionsBody)
    return jsonify(result)

# Define route to get all warehouses data
@app.route("/warehouses", methods=["GET"])
def wareohuses():
    return jsonify(getAllWarehouses())

# Define route to export data
@app.route("/export", methods=["OPTIONS", "POST"])
def export():
    if request.method == "POST":
        params = request.get_json(silent=True)
        export = getExport(params)
        print("res", export)
        return export
    return []
```

warehouses.py

```
# Import necessary modules and load environment variables
import requests
import os
from dotenv import load_dotenv
import xmltodict

load_dotenv()

# Initialize stock level dictionary
```

```
stockLevelDict = None

# Function to fetch and parse stock levels from an external URL
def getStockLevel():
    GET_STOCK_LEVEL_URL = (
        "REPLACE WITH YOUR URL"
        + os.getenv("DIRECTO_KEY")
    )
    response = requests.get(
        url=GET_STOCK_LEVEL_URL, headers={"Accept-Encoding": "utf8"}
    )
    return xmltodict.parse(response.text)

# Fetch stock level data
stockLevelDict = getStockLevel()

# Function to extract all unique warehouse stock levels
def getAllWarehouses():
    warehouses = []
    for article in stockLevelDict["transport"]["stocklevels"]["stocklevel"]:
        if article["@stock"] not in warehouses:
            warehouses.append(article["@stock"])
    return warehouses
```

sheets.py

```
# Import necessary modules
import gspread
import pandas as pd
from server.data.warehouses import getStockLevel

# Authenticate with Google Sheets using service account
gc = gspread.service_account("venv/service_account.json")

# Open the Google Sheet by key and read data into DataFrame
RCP_helper_sheet = gc.open_by_key("REPLACE WITH YOUR SHEET")
version_to_series_wks = RCP_helper_sheet.worksheet("REPLACE WITH YOUR SHEET")
version_to_series_df = pd.DataFrame(version_to_series_wks.get_all_records())
version_to_series_df["version"] = version_to_series_df["version"].astype(str)

# Placeholder function
def getRCP():
    return ["R", "C", "P", "V", "T"]

# Get unique major series
def getMajorSeries():
    return list(version_to_series_df["product_type"].unique())

# Get series for the given major series
def getSeries(majorSeries=None):
    if majorSeries:
        return list(
            version_to_series_df.loc[
                version_to_series_df["product_type"].isin(majorSeries)
            ]["RCP series"].unique()
        )
    else:
        return []

# Get versions for the given series
def getVersions(series=None):
    if series:
        result = list(
            version_to_series_df.loc[version_to_series_df["RCP series"].isin(series)][
```

```
            "version"
        ].unique()
    )
    print("getVer", result)
    return result
  else:
    return []

# Export function to generate and return the export data based on parameters
def getExport(params):
  print("funct json", params)
  categoryRCPSelected = params["categoryRCP"]
  exportTypeSelected = params["exportType"]
  recommend = True if exportTypeSelected == "recommendation" else False

  majorSeriesSelected = params["majorSeries"]
  seriesSelected = params["series"]
  versionsSelected = {
    key: value for item in params["versions"] for key, value in item.items()
  }

  stocksSelected = params["stocks"] if "stocks" in params else None

  # Fetch and process data from Google Sheets
  RCP_list_wks = gc.open_by_key(
    "REPLACE WITH YOUR SHEET"
  ).worksheet("REPLACE WITH YOUR SHEET")
  RCP_list_array = RCP_list_wks.get("A1:DN3000")
  RCP_list_df = pd.DataFrame(RCP_list_array[1:], columns=RCP_list_array[0])
  RCP_list_df.dropna(subset="Article", inplace=True)
  RCP_list_df = RCP_list_df[RCP_list_df["Article"] != ""]

  RCP_helper_sheet = gc.open_by_key("REPLACE WITH YOUR SHEET")
  RCP_guide_wks = RCP_helper_sheet.worksheet("SPG")
  RCP_B_guide_array = RCP_guide_wks.get("B15:F21")
  RCP_B_guide_df = pd.DataFrame(
    RCP_B_guide_array[1:], columns=RCP_B_guide_array[0]
  ).astype(int)

  RCP_A_guide_array = RCP_guide_wks.get("B24:F92")
  RCP_A_guide_df = pd.DataFrame(
    RCP_A_guide_array[1:], columns=RCP_A_guide_array[0]
  ).astype(int)

  version_to_series_wks = RCP_helper_sheet.worksheet("Version -> RCP series")
  version_to_series_df = pd.DataFrame(version_to_series_wks.get_all_records())
  version_to_series_df["version"] = version_to_series_df["version"].astype(str)

  # Select relevant columns and reshape data
  selected_general_info_df = RCP_list_df[
    [
      "Article",
      "Description (ENG)",
      "Price",
      "Serial #\n(YES/-)",
      "Depth (m)",
      "Width (m)",
      "Height (m)",
      "Weight (kg)",
    ]
  ]

  selected_RCP_df = RCP_list_df.melt(
```

```
    id_vars="Article", var_name="RCP series", value_name="RCP"
)
selected_RCP_df = selected_RCP_df[selected_RCP_df["RCP"] != "-"]
selected_RCP_df.dropna(subset="RCP", inplace=True)

selected_versions_df = RCP_list_df.melt(
    id_vars="Article", var_name="version", value_name="version_count"
)
selected_versions_df["version_count"] = pd.to_numeric(
    selected_versions_df["version_count"], errors="coerce"
)
selected_versions_df.dropna(subset="version_count", inplace=True)
selected_versions_df = pd.merge(
    left=selected_versions_df, right=version_to_series_df, on="version"
)
selected_versions_df = pd.merge(
    left=selected_versions_df, right=selected_RCP_df, on=["Article", "RCP series"]
)

selected_locations_df = RCP_list_df.melt(
    id_vars="Article", var_name="RCP series", value_name="location"
)
selected_locations_df = selected_locations_df[
    selected_locations_df["location"] != ""
]
selected_locations_df = selected_locations_df[
    selected_locations_df["location"] != "-"
]
selected_locations_df["RCP series"] = "RCP " + selected_locations_df[
    "RCP series"
].str.replace(" Location", "")

selected_RCP_location_df = pd.merge(
    left=selected_RCP_df, right=selected_locations_df, on=["Article", "RCP series"]
)

full_df = pd.merge(
    selected_RCP_location_df,
    selected_versions_df,
    on=["Article", "RCP series", "RCP"],
)
full_df = pd.merge(full_df, selected_general_info_df, on=["Article"])
full_df = full_df.groupby(
    [
        "Article",
        "Description (ENG)",
        "Serial #\n(YES/-)",
        "Price",
        "Depth (m)",
        "Width (m)",
        "Height (m)",
        "Weight (kg)",
        "Product Type",
        "RCP",
        "location",
        "version",
    ]
).agg("last")

initial_df = full_df.copy()

# Filter data based on selected criteria
filtered_product_type = initial_df
```

```
if majorSeriesSelected:
    filtered_product_type = initial_df.loc[
        initial_df.index.get_level_values("product_type").isin(majorSeriesSelected)
    ]

filtered_RCP_series = filtered_product_type
if seriesSelected:
    filtered_RCP_series = filtered_RCP_series.loc[
        filtered_RCP_series.index.get_level_values("RCP series").isin(
            seriesSelected
        )
    ]

filtered_RCP = filtered_RCP_series
if seriesSelected:
    filtered_RCP = filtered_RCP_series.loc[
        filtered_RCP_series.index.get_level_values("RCP").isin(categoryRCPSelected)
    ]

filtered_versions = filtered_RCP
if versionsSelected.keys():
    filtered_versions = filtered_versions.loc[
        filtered_versions.index.get_level_values("version").isin(
            list(versionsSelected.keys())
        )
    ]
    versions_input_df = (
        pd.DataFrame.from_dict(versionsSelected, orient="index")
        .reset_index()
        .set_axis(["version", "APM_count"], axis=1)
    )
    with_versions_count_df = pd.merge(
        left=filtered_versions.reset_index(),
        right=versions_input_df,
        on="version",
        how="left",
    )
    filtered_versions = with_versions_count_df.groupby(
        ["Article", "product_type", "RCP"]
    ).agg(
        max_amount_versions=("version_count", "max"),
        applied_APM_sum=("APM_count", "sum"),
    )
else:
    filtered_versions = filtered_versions.groupby(
        ["Article", "product_type", "RCP"]
    ).agg(max_amount_versions=("version_count", "max"))

# Function to apply coefficients based on guides
def apply_coef(row_max_df):
    R_coef = None
    C_coef = None
    P_coef = None

    if row_max_df.name[1] == "A":
        if row_max_df["applied_APM_sum"] < 0:
            raise ValueError("The count can't be below zero")

        elif row_max_df["applied_APM_sum"] == 0:
            R_coef = 0
            C_coef = 0
            P_coef = 0
```

```
        elif row_max_df["applied_APM_sum"] > RCP_A_guide_df["max"].max():
            last_row = RCP_A_guide_df.loc[RCP_A_guide_df["max"].idxmax()]
            R_coef = last_row["R"]
            C_coef = last_row["C"]
            P_coef = last_row["P"]

        else:
            for index, row in RCP_A_guide_df.iterrows():
                if row["min"] <= row_max_df["applied_APM_sum"] <= row["max"]:
                    R_coef = row["R"]
                    C_coef = row["C"]
                    P_coef = row["P"]

    elif row_max_df.name[1] == "B":
        if row_max_df["applied_APM_sum"] < 0:
            raise ValueError("The count can't be below zero")

        elif row_max_df["applied_APM_sum"] == 0:
            R_coef = 0
            C_coef = 0
            P_coef = 0

        elif row_max_df["applied_APM_sum"] > RCP_B_guide_df["max"].max():
            last_row = RCP_B_guide_df.loc[
                RCP_B_guide_df["max"].idxmax()
            ]
            R_coef = last_row["R"]
            C_coef = last_row["C"]
            P_coef = last_row["P"]

        else:
            for index, row in RCP_B_guide_df.iterrows():
                if row["min"] <= row_max_df["applied_APM_sum"] <= row["max"]:
                    R_coef = row["R"]
                    C_coef = row["C"]
                    P_coef = row["P"]

    if row_max_df.name[2] == "R":
        return row_max_df[0] * R_coef
    elif row_max_df.name[2] == "C":
        return row_max_df[0] * C_coef
    elif row_max_df.name[2] == "P":
        return row_max_df[0] * P_coef

Directo_dict = getStockLevel()

if recommend:
    filtered_versions["recommended_amount"] = filtered_versions.apply(
        apply_coef, axis=1
    )
    join = pd.merge(
        filtered_versions,
        initial_df.reset_index(),
        on=["Article", "product_type", "RCP"],
    )
    join = pd.merge(versions_input_df, join.reset_index(drop=True), on="version")
    join.rename(
        columns={
            "recommended_amount": "The recommended amount of spare parts",
            "RCP": "Type",
            "applied_APM_sum": "Total APM-s",
            "version": "Compatibility of spare parts with products",
            "APM_count": "APM - qty",
```

```
        },
        inplace=True,
    )

    to_replenish_columns_ordered = []
    if (len(stocksSelected) != 0) and ("None" not in stocksSelected):
        warehouses_count = get_article_by_code(
            join["Article"].unique().tolist(), stocksSelected
        )
        join = pd.merge(
            left=join,
            right=warehouses_count,
            how="left",
            left_on="Article",
            right_on="code",
        ).fillna(0)

        for stock in stocksSelected:

            def count_to_replenish(row):
                result = row["The recommended amount of spare parts"] - row[stock]
                return max(0, result)

            to_replenish_column = f"{stock} to replenish"
            to_replenish_columns_ordered.extend([stock, to_replenish_column])
            join[to_replenish_column] = join.apply(count_to_replenish, axis=1)

        join = join.groupby(
            [
                "Article",
                "The recommended amount of spare parts",
                *to_replenish_columns_ordered,
                "Type",
                "Description (ENG)",
                "Serial #\n(YES/-)",
                "Depth (m)",
                "Width (m)",
                "Height (m)",
                "Weight (kg)",
                "Total APM-s",
                "max_amount_versions",
                "product_type",
                "RCP series",
                "Price",
                "Compatibility of spare parts with products",
            ]
        ).agg("last")
        join = join.loc[
            join.index.get_level_values("The recommended amount of spare parts") > 0
        ]
        join = join.loc[
            join.index.get_level_values(
                "Compatibility of spare parts with products"
            ).isin(list(versionsSelected.keys()))
        ]
else:
    join = pd.merge(
        filtered_versions,
        initial_df.reset_index(),
        on=["Article", "product_type", "RCP"],
    )
    join.rename(
        columns={
```

```
                    "RCP": "Type",
                    "version": "Compatibility of spare parts with products",
                },
                inplace=True,
            )
            join = join.groupby(
                [
                    "Article",
                    "Type",
                    "Description (ENG)",
                    "location",
                    "Price",
                    "Serial #\n(YES/-)",
                    "Depth (m)",
                    "Width (m)",
                    "Height (m)",
                    "Weight (kg)",
                    "product_type",
                    "RCP series",
                    "Compatibility of spare parts with products",
                ]
            ).agg("last")
            join = join.loc[join.index.get_level_values("RCP series").isin(seriesSelected)]
            join.drop(columns=["max_amount_versions"], inplace=True)

        join.sort_values("Type", inplace=True)
        print(join)
        join.to_excel("data/export.xlsx")
        return join.reset_index().to_json(orient="records")
```

_app.tsx

```
import { type AppType } from "next/dist/shared/lib/utils";

import "~/styles/globals.css";

const MyApp: AppType = ({ Component, pageProps }) => {
  return <Component {...pageProps} />;
};

export default MyApp;
```

index.tsx

```
import Head from "next/head";
import { useEffect, useState } from "react";
import GenerateForm from "~/components/general/generate";
import {
  Table,
  TableBody,
  TableCaption,
  TableCell,
  TableHead,
  TableHeader,
  TableRow,
} from "src/components/ui/table";
import { ScrollArea, ScrollBar } from "~/components/ui/scroll-area";
import { Button } from "~/components/ui/button";
import { Loader } from "lucide-react";
import { useRouter } from "next/router";

export default function Home() {
  // State variables to manage data fetching and display
  const [exportTable, setExportTable] = useState([]); // Stores the table data to display
  const [tableHeader, setTableHeader] = useState<string[]>([]); // Stores the headers of the table
```

```
  const [isFetching, setIsFetching] = useState(false); // Indicates if data is currently being fetched

  const router = useRouter(); // Next.js router instance

  // Function to generate table data based on parameters
  async function generateTable(params: any): Promise<any> {
    console.log(JSON.stringify(params)); // Log the parameters for debugging
    setIsFetching(true); // Start fetching data
    const result = await fetch("http://localhost:5000/export", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(params),
    });
    const exportJSON = await result.json(); // Parse the fetched JSON data

    setExportTable(exportJSON); // Update the table data state
    setTableHeader(Object.keys(exportJSON[0])); // Update the table headers based on fetched data
    setIsFetching(false); // Finish fetching data
  }

  // Effect to log exportTable whenever it changes
  useEffect(() => {
    console.log(exportTable); // Log the current export table data
    console.log(tableHeader); // Log the current table headers
  }, [exportTable]); // Run this effect when exportTable changes

  return (
    <>
      <Head>
        <title>WMS</title>
        <meta name="description" content="Generated by create-t3-app" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className="flex min-h-screen flex-col items-center bg-slate-100">
        {/* Header section */}
        <h1
          onClick={() => {
            router.reload(); // Reload the page on header click
          }}
          className="select-none text-5xl font-extrabold tracking-tight text-black hover:cursor-pointer sm:text-
[5rem]"
        >
          <span className="">WMS</span>
        </h1>
        {/* Form to generate table */}
        <div className="mt-4">
          <GenerateForm generate={generateTable}></GenerateForm>
        </div>
        <div className="mt-6 flex w-3/4 grow-0 flex-col items-center">
          {/* Loading indicator */}
          {isFetching && (
            <span className="flex justify-center align-middle">
              Loading...
              <Loader className="ml-2 h-6 w-6 animate-spin" />
            </span>
          )}
          {/* Display table if data is fetched */}
          {exportTable.length > 0 && (
            <>
              {/* Download button */}
              <Button
```

```
                    variant="default"
                    onClick={() => {
                      const downloadFile = async () => {
                        const result = await fetch(
                          "http://localhost:5000/download",
                        );
                        const blob = await result.blob(); // Retrieve blob data

                        const url = window.URL.createObjectURL(blob); // Create object URL for blob
                        const a = document.createElement("a");
                        a.style.display = "none";
                        a.href = url;
                        document.body.appendChild(a); // Append <a> element to the body
                        a.click(); // Simulate click to trigger download
                        window.URL.revokeObjectURL(url); // Release object URL resources
                      };
                      downloadFile(); // Initiate file download on button click
                    }}
                    className="mb-2"
                  >
                    Download
                  </Button>
                  {/* Scrollable area for the table */}
                  <ScrollArea className="mb-8 w-full whitespace-nowrap rounded-md border">
                    <ScrollBar orientation="horizontal" />
                    {/* Table component */}
                    <Table className="">
                      <TableHeader>
                        {/* Render table headers */}
                        <TableRow>
                          {tableHeader.map((header) => (
                            <TableHead key={header}>{header}</TableHead>
                          ))}
                        </TableRow>
                      </TableHeader>
                      <TableBody>
                        {/* Render table rows */}
                        {exportTable.map((item, index) => (
                          <TableRow key={index}>
                            {tableHeader.map((key) => (
                              <TableCell key={key}>{item[key]}</TableCell>
                            ))}
                          </TableRow>
                        ))}
                      </TableBody>
                    </Table>
                  </ScrollArea>
                </>
              )}
            </div>
          </main>
        </>
      );
}}
```

generate.tsx

```
"use client";

import { useState, useEffect } from "react"; // Import React hooks for state and side effects

import * as z from "zod"; // Import Zod for schema validation
import { zodResolver } from "@hookform/resolvers/zod"; // Import Zod resolver for React Hook Form
import { useForm } from "react-hook-form"; // Import useForm hook from React Hook Form
```

```
import { Button } from "src/components/ui/button"; // Import UI components for the form
import {
  Form,
  FormControl,
  FormDescription,
  FormField,
  FormItem,
  FormLabel,
  FormMessage,
} from "src/components/ui/form"; // Import UI components for the form
import { Checkbox } from "src/components/ui/checkbox"; // Import UI components for checkboxes
import { RadioGroup, RadioGroupItem } from "src/components/ui/radio-group"; // Import UI components for
radio buttons
import { ScrollArea } from "src/components/ui/scroll-area"; // Import UI components for scrollable areas
import { Input } from "src/components/ui/input"; // Import UI components for input fields

// Define the schema for the form validation using Zod
const formSchema = z.object({
  category: z
    .array(z.string())
    .refine((value) => value.some((item) => item)), // Validate and refine category as non-empty array of strings
  exportType: z.enum(["recommendation", "compatibility"] as const), // Validate exportType as one of two
specific string values
  majorSeries: z
    .array(z.string())
    .refine((value) => value.some((item) => item)), // Validate and refine majorSeries as non-empty array of
strings
  series: z.array(z.string()).refine((value) => value.some((item) => item)), // Validate and refine series as non-
empty array of strings
  versions: z.array(z.record(z.number())), // Validate versions as an array of records with numeric values
  stocks: z.array(z.string()).optional() || [undefined], // Validate stocks as an optional array of strings or
undefined
});

export default function GenerateForm({ generate = (f: any) => f }) {
  // State variables for form options and selections
  const [RCPItems, setRCPItems] = useState(["R", "C", "P", "V", "T"]);
  const [majorSeriesItems, setMajorSeriesItems] = useState(["A", "B"]);
  const [series, setSeries] = useState([]);
  const [versions, setVersions] = useState([]);
  const [warehouses, setWarehouses] = useState([]);

  // Effect to fetch major series and warehouses data on component mount
  useEffect(() => {
    const fetchMajorSeries = async () => {
      const result = await fetch("http://localhost:5000/majorSeries");
      const majorSeries = await result.json();
      setMajorSeriesItems(majorSeries); // Set fetched major series items
    };
    fetchMajorSeries();

    const fetchWarehouses = async () => {
      const result = await fetch("http://localhost:5000/warehouses");
      const warehouses = await result.json();
      setWarehouses(warehouses); // Set fetched warehouses
    };
    fetchWarehouses();
  }, []); // Empty dependency array ensures this effect runs only once on mount

  // Form submission handler
  function onSubmit(values: z.infer<typeof formSchema>) {
    generate(values); // Call generate function with form values
```

```
}

// Initialize useForm hook with formSchema for validation and default values
const form = useForm<z.infer<typeof formSchema>>({
  resolver: zodResolver(formSchema), // Use Zod resolver for form validation
  defaultValues: {
    categoryRCP: ["R", "C", "P"], // Default values for form fields
    exportType: undefined,
    majorSeries: [],
    series: [],
    versions: [],
    stocks: [],
  },
});

// Effect to fetch series data based on selected major series
const selectedMajorSeries = form.watch("majorSeries");
useEffect(() => {
  const fetchSeries = async () => {
    const result = await fetch("http://localhost:5000/series", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(selectedMajorSeries),
    });
    const series = await result.json();
    setSeries(series); // Set fetched series

    // Update form values for series to match fetched series items
    form.setValue(
      "series",
      form.getValues("series").filter((item) => series.includes(item)),
    );
  };
  fetchSeries();
}, [selectedMajorSeries]); // Run this effect whenever selectedMajorSeries changes

// Effect to fetch versions data based on selected series
const selectedSeries = form.watch("series");
useEffect(() => {
  const fetchVersions = async () => {
    const result = await fetch("http://localhost:5000/versions", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(selectedSeries),
    });
    const versions = await result.json();
    setVersions(versions); // Set fetched versions

    // Filter and update form values for versions to match fetched versions
    function filterRecords(
      records: { [key: string]: number }[],
      keysToInclude: string[],
    ) {
      return records.filter((record) => {
        const key = Object.keys(record)[0];
        return keysToInclude.includes(key!);
      });
    }
```

```
    form.setValue(
      "versions",
      filterRecords(form.getValues("versions"), versions),
    );
  };
  fetchVersions();
}, [selectedSeries]); // Run this effect whenever selectedSeries changes

// Get currently selected exportType from form
const selectedExportType = form.watch("exportType");

// Render the form with the following structure and components
return (
  <>
    <Form {...form}>
      <form
        onSubmit={form.handleSubmit(onSubmit)} // Handle form submission with onSubmit function
        className="flex flex-col justify-center"
      >
        <div className="mb-4 flex flex-row space-x-8">
          {/* Render categoryRCP field */}
          <FormField
            control={form.control}
            name="categoryRCP"
            render={() => (
              <FormItem>
                <FormLabel>Category</FormLabel>
                <div className="h-auto space-y-2 rounded-md border p-4">
                  {RCPItems.map((item) => (
                    <FormField
                      key={item}
                      control={form.control}
                      name="categoryRCP"
                      render={({ field }) => {
                        return (
                          <FormItem
                            key={item}
                            className="flex flex-row items-start space-x-3 space-y-0"
                          >
                            {/* Render checkboxes for RCP items */}
                            <FormControl>
                              <Checkbox
                                checked={field.value?.includes(item)}
                                onCheckedChange={(checked) => {
                                  return checked
                                    ? field.onChange([...field.value, item])
                                    : field.onChange(
                                        field.value?.filter(
                                          (value) => value !== item,
                                        ),
                                      );
                                }}
                              />
                            </FormControl>
                            <FormLabel className="font-normal">
                              {item}
                            </FormLabel>
                          </FormItem>
                        );
                      }}
                    />
                  ))}
                </div>
```

```
    </FormItem>
  )}
/>

{/* Render exportType field */}
<FormField
  control={form.control}
  name="exportType"
  render={({ field }) => (
    <FormItem className="space-y-2">
      <FormLabel>Export type</FormLabel>
      <FormControl>
        <div className="h-auto space-y-2 rounded-md border p-4">
          {/* Render radio buttons for exportType */}
          <RadioGroup
            onValueChange={field.onChange}
            className="flex flex-col space-y-1"
          >
            <FormItem className="flex items-center space-x-3 space-y-0">
              <FormControl>
                <RadioGroupItem value="recommendation" />
              </FormControl>
              <FormLabel className="font-normal">
                Recommendation table
              </FormLabel>
            </FormItem>
            <FormItem className="flex items-center space-x-3 space-y-0">
              <FormControl>
                <RadioGroupItem value="compatibility" />
              </FormControl>
              <FormLabel className="font-normal">
                Versions compatibility
              </FormLabel>
            </FormItem>
          </RadioGroup>
        </div>
      </FormControl>
    </FormItem>
  )}
/>

{/* Render majorSeries field */}
<FormField
  control={form.control}
  name="majorSeries"
  render={() => (
    <FormItem>
      <FormLabel>Major series</FormLabel>
      <div className="h-auto space-y-2 rounded-md border p-4">
        {majorSeriesItems.map((item) => (
          <FormField
            key={item}
            control={form.control}
            name="majorSeries"
            render={({ field }) => {
              return (
                <FormItem
                  key={item}
                  className="flex flex-row items-start space-x-3 space-y-0"
                >
                  {/* Render checkboxes for major series items */}
                  <FormControl>
                    <Checkbox
```

```
                checked={field.value?.includes(item)}
                onCheckedChange={(checked) => {
                  return checked
                    ? field.onChange([...field.value, item])
                    : field.onChange(
                        field.value?.filter(
                          (value) => value !== item,
                        ),
                      );
                }}
              />
</FormControl>
```

ДОДАТОК Б

ВІДГУК

Керівника економічного розділу

на кваліфікаційну роботу бакалавра на тему:

Розробка комп'ютерної системи автоматизованого менеджменту промислових складів

Студента групи 122-20-2 Півня Івана Сергійовича

Керівник економічного розділу

доц. каф. ПЕП та ПУ, к.е.н **Л.В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_Півень_ІС.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_Півень_ІС.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Software.rar	Архів. Містить коди програми
Презентація	
Презентація_Півень.ppt	Презентація дипломного проекту