Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно–інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Другий (магістерський)
(півень вищої освіти)

Розроблення компонентів системи комплектації для виробничого технологічного процесу
(тема)

Виконав:
студент II курсу, групи АУТПм-22-2

Закладний В. І.
(прізвище, ініціали)

Спеціальності 151 Автоматизація та комп'ютерно–інтегровані технології
(код і повна назва спеціальності)

Тип програми Освітньо–професійна
(освітньо–професійна або освітньо–наукова)

Освітня програма Автоматизоване управління технологічними процесами
(повна назва освітньої програми)

Керівник проф. Безкоровайний В. В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024 р.

Я, як студент ХНУРЕ (Закладний Владислав Ігорович), розумію і підтримую політику закладу із академічної доброчесності. Я не надав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Дата Закладний В. І.

13.01.2024 р 

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

Факультет *АКТ*
Кафедра *КІТАР*
Рівень вищої освіти *другий (магістерський)*
Спеціальність *151 Автоматизація та комп'ютерно–інтегровані технології*
Тип програми *Освітньо–професійна*
Освітня програма *Автоматизоване управління технологічними процесами*
(шифр і назва)

ЗАТВЕРДЖУЮ
Зав. кафедри КІТАР _____
(підпис)
«___» _____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Закладному Владиславу Ігоровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розроблення компонентів системи комплектації для виробничого технологічного процесу*
Затверджена наказом по університету від *03.11.2023 р. № 1286 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *25.01.2024 р.*
3. Вихідні дані до роботи *Об'єкт дослідження – система автоматизованої комплектації виробничого технологічного процесу. Предмет дослідження – процес комплектації замовлень для виробничого технологічного процесу. Функція системи – розподілення, розміщення і комплектація в складському приміщенні. Технічне забезпечення: IBM-сумісний персональний комп'ютер. Програмний засіб з веб-інтерфейсом.*

4. Перелік питань, що потрібно опрацювати в роботі *Вступ. Огляд предметної області. Необхідність впровадження систем складського обліку. Задачі оптимізації в управлінні складськими операціями. Опис задач комплектування. Постановка мети та задач кваліфікаційної роботи. Моделювання та конструювання програмного забезпечення. Сценарії використання системи. Програмна реалізація. Тестування та оцінка створеного програмного забезпечення. Охорона праці. Висновки. Перелік джерел посилання.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій *Графічний матеріал у вигляді презентації – 12–15 аркушів формату А4.*

6. Консультанти розділів роботи

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання та аналіз завдання*	*06.11.2023*	*Виконано*
2	*Огляд проблеми комплектації виробничих*		
	технологічних процесів	*13.11.2023*	*Виконано*
3	*Розробка математичного забезпечення задачі*	*20.11.2023*	*Виконано*
4	*Розробка програмного забезпечення задачі*	*27.11.2023*	*Виконано*
5	*Проведення експериментальних досліджень*	*30.11.2023*	*Виконано*
6	*Підготовка публікацій за результатами*		
	дослідження	*01.12.2023*	*Виконано*
7	*Оформлення пояснювальної записки*	*25.12.2023*	
8	*Подання закінченої роботи керівникові*	*28.12.2023*	*Виконано*
9	*Усунення зауважень наукового керівника*	*05.01.2024*	*Виконано*
10	*Підготовка презентації*	*08.01.2024*	*Виконано*
11	*Подання роботи на рецензування*	*10.01.2024*	*Виконано*
12	*Попередній захист*	*23.01.2024*	*Виконано*
13	*Подання роботи до екзаменаційної комісії*	*25.01.2024*	*Виконано*

Дата видачі завдання *06.11.2023 р.*

Студент _____ Закладний В. І.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Безкоровайний В. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 85 с., 1 табл., 35 рис., 2 дод., 39 джерел.

АВТОМАТИЗАЦІЯ ПРОЦЕСУ КОМПЛЕКТАЦІЇ, ВИРОБНИЧИЙ ТЕХНОЛОГІЧНИЙ ПРОЦЕС, МАРШРУТИЗАЦІЯ, ОПТИМІЗАЦІЯ, СИСТЕМА КОМПЛЕКТАЦІЇ, СКЛАДСЬКЕ ПРИМІЩЕННЯ.

Об’єкт дослідження – система автоматизованої комплектації виробничого технологічного процесу.

Предмет дослідження – процес комплектації замовлень для виробничого технологічного процесу.

Мета роботи – підвищення ефективності системи комплектації виробничого технологічного процесу за рахунок розробки засобу оптимізації маршрутів комплектувальників.

Методи дослідження – теорія систем, системний аналіз, теорія корисності, теорія прийняття рішень, теорія графів, методи сучасних інформаційних технологій.

Розроблено програмний засіб для системи комплектації виробничого технологічного процесу, що надає можливість підвищити ефективність складського обліку за рахунок автоматизованого формування маршрутів комплектації. Для розв’язання задачі використано алгоритм комбінаторної оптимізації.

Використання розробки дозволяє знижувати витрати часу та енергетичних ресурсів для комплектації виробничих технологічних процесів.

Галузь застосування – технології проєктування виробничих технологічних процесів та управління ними.

Результати кваліфікаційної роботи апробовані у тезах конференції.

ABSTRACT

The work contains: 85 p., 1 tabl., 35 pic., 2 app., 39 sources.

AUTOMATION OF THE ASSEMBLY PROCESS, PRODUCTION TECHNOLOGICAL PROCESS, ROUTING, OPTIMIZATION, ASSEMBLY SYSTEM, WAREHOUSE.

The object of research is a system of automated order picking for a production process.

The subject of research is the process of order picking for the production process.

Purpose - to increase the efficiency of the system of picking the production process by developing a means of optimizing the routes of pickers.

Research methods - systems theory, system analysis, utility theory, decision theory, graph theory, methods of modern information technology.

A software tool for the picking system of the production process has been developed, which makes it possible to increase the efficiency of warehouse accounting through the automated formation of picking routes. The combinatorial optimization algorithm was used to solve the problem.

The use of the development allows to reduce the time and energy costs for the picking of production processes.

The field of application is technologies for designing and managing production processes.

The results of the qualification work were tested in the conference abstracts.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

СУБД – Система Управління Базами Даних (Database Management System);

CRUD – Create, Read, Update, Delete (Створення, Читання, Оновлення, Видалення);

ERP – Enterprise Resource Planning (Планування ресурсів підприємства);

JSON – JavaScript Object Notation (Нотація Об'єктів JavaScript);

ORM – Object–Relational Mapping (Об'єктно–Реляційне Відображення);

SKU – Stock Keeping Unit (Складська Одиниця Зберігання);

SQL – Structured Query Language (Мова Структурованих Запитів);

TSP – Traveling Salesman Problem (Задача Комівояжера);

UML – Unified Modeling Language (Уніфікована Мова Моделювання);

VPS – Virtual Private Server (Віртуальний Приватний Сервер);

WMS – Warehouse Management System (Система Управління Складом);

WYSIWYG – What You See Is What You Get (Що Бачиш, Те й Отримуєш).

ВСТУП

Системи автоматизованої комплектації є невід'ємним елементом логістичного забезпечення технологічних процесів сучасних виробничих компаній. Вони передбачають автоматизовану координацію внутрішніх потоків ресурсів, матеріалів та інформації для оптимізації технологічних процесів і підвищення їх загальної ефективності [1].

Свідченням актуальності проблеми оптимізації логістики виробничо-збутової діяльності сучасних компаній є численні наукові публікації [1-7]. Виробнича логістика й управління складами є складовою виробничо-збутової логістики [3-5]. Важливою задачею управління складським господарством є вдосконалення процесу потоку комплектуючих з заводського складу до виробничої лінії. Відомо, що процес комплектування замовлень для виробничих технологічних процесів (ВТП) може досягати до 55 – 75 % від усіх трудовитрат на складі [6].

Об’єктом дослідження кваліфікаційної роботи є система автоматизованої комплектації виробничого технологічного процесу.

Предметом дослідження є процес комплектації замовлень для виробничого технологічного процесу.

Методи дослідження – теорія систем, системний аналіз, теорія корисності, теорія прийняття рішень, теорія графів, методи сучасних інформаційних технологій.

Мета кваліфікаційної роботи полягає у підвищення ефективності системи комплектації виробничого технологічного процесу за рахунок розробки засобу оптимізації маршрутів комплектувальників.

Необхідно розв’язати наступні задачі для досягнення мети кваліфікаційної роботи.

– виконати огляд сучасного стану проблеми комплектації виробничих технологічних процесів;

– провести аналіз існуючих систем автоматизованої комплектації

виробничих технологічних процесів;

– сформулювати постановку задачі розробки компонентів системи комплектації для оптимізації маршрутів комплектувальників;

– обрати методи та розробити алгоритми вирішення задачі оптимізації;

– провести тестування готової системи комплектації;

– оформити пояснювальну записку згідно з рекомендаціями [8], та вимогами ДСТУ 3008:2015 [9].

Результатом роботи є повнофункціональний програмний продукт, здатний втілити оригінальний підхід до оптимізації складських задач в частині комплектування за технологічними картами.

Результати кваліфікаційної роботи апробовані у тезах міжнародної конференції [7].

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Необхідність впровадження систем складського обліку

В умовах сучасної ринкової економіки кожне виробниче підприємство стикається з нагальною потребою в автоматизації своїх процесів. Ефективне управління ресурсами, оптимізація виробничих процесів та логістики – це ключові аспекти, які впливають на конкурентоспроможність підприємства. Один з вирішальних факторів успіху в даному аспекті – це впровадження інтегрованої системи складського обліку.

Сучасні виробничі компанії, які працюють переважно з бізнес-клієнтами, мають специфічні вимоги до управління запасами та логістики. Відсутність роботи з кінцевим споживачем вимагає більшої концентрації на оптимізації внутрішніх процесів, здатності швидко реагувати на запити бізнес-клієнтів та ефективності поставок [10].

Перше і, можливо, найголовніше – це точність. Наявність точної інформації про наявність комплектуючих на складі, їх місцезнаходження та рух – це фундаментальна потреба для забезпечення безперервності виробничих ланцюгів. Без точного складського обліку, підприємство ризикує втратити контроль над запасами, що може призвести до збоїв у ланцюжках постачання, зниження якості обслуговування клієнтів та, як наслідок, втрати ділової репутації.

Далі йде планування. Система складського обліку дозволяє вести детальний облік ресурсів, необхідних для виробництва, що, в свою чергу, уможливлює точне планування потреб у матеріалах та компонентах. Це також включає можливість прогнозування майбутніх потреб, аналіз тенденцій попиту та оптимізацію закупівлі необхідних компонентів та ресурсів.

Ефективне управління запасами також включає в себе зниження витрат. Система складського обліку дозволяє оптимізувати кількість запасів на складі, що знижує витрати на зберігання та мінімізує ризики пов'язані зі знеціненням

комплектуючих. З іншого боку, це допомагає уникнути ситуацій з надлишком або нестачею комплектуючих [11].

Треба враховувати і логістичний компонент. Систематичний облік дозволяє оптимізувати маршрутизацію вантажів, ефективно розподіляти товари між складами та точками відвантаження, а також скорочувати час на підготовку замовлень [12].

Перераховуючи приводи впровадження сучасних систем складського обліку, також варто згадати про аналітику. Сучасні системи складського обліку зазвичай містять інструменти для глибокого аналізу даних, що дозволяє виявити "вузькі місця" у виробничих та логістичних процесах, а також визначити напрямки для подальшого удосконалення.

Сучасні системи складського обліку не існують в ізоляції. Вони інтегровані з іншими бізнес–системами, такими як ERP (Enterprise Resource Planning) та CRM (Customer Relationship Management), що дозволяє синхронізувати всі процеси в компанії та отримувати повну картину бізнесу в реальному часі.

На закінчення, важливо зазначити, що впровадження сучасної системи складського обліку – це інвестиція в майбутнє підприємства. Точність обліку, оптимізація запасів і процесів, а також гнучкість є компонентами, що поруч з іншими забезпечує стале зростання бізнесу, а також приймає участь у максимізації прибутку шляхом оптимізації витрат.

В якості вдосконалення систем складського обліку можливо та доцільно розглядати також використання математичних алгоритмів, зокрема алгоритмів комбінаторної оптимізації, для оптимізації складських процесів. Базовим набором даних для таких алгоритмів можуть виступати саме облікові дані, що зберігаються в системі [13–14].

В подальшому розглянемо більш детально можливості оптимізації складських процесів з використанням наявних даних та математичних алгоритмів, а також визначимо які з них можуть бути найбільш корисними з практичної точки зору.

1.2 Задачі оптимізації в управлінні складськими операціями

Розглядаючи можливості оптимізації в складських операціях можна виділити наступні ключові можливості [15]:

а) оптимізація маршрутів переміщення комплектуючих в межах складу:

1) ефективні маршрути зменшують час, необхідний для переміщення комплектуючих між зонами приймання, зберігання, комплектації та відвантаження;

2) оптимізація внутрішньоскладських маршрутів знижує фізичні навантаження на працівників та збільшує продуктивність праці;

3) використання автоматизованих систем керування складом (Warehouse Management Systems, WMS) дозволяє точно планувати та оптимізувати маршрути, що веде до зменшення втрат часу та збільшення ефективності роботи складу;

б) управління запасами і запасними рівнями:

1) управління запасами дозволяє підтримувати оптимальну кількість комплектуючих на складі, запобігаючи надлишкам або дефіциту;

2) використання методів JIT (Just-In-Time) та JIS (Just-In-Sequence) сприяє зниженню витрат на зберігання та зменшенню потреби в оборотному капіталі;

3) аналіз історії замовлень та прогнозування попиту допомагає точно планувати закупівлі та оптимізувати робочий капітал;

в) оптимізація складського простору:

1) ефективне використання простору є ключовим для збільшення обсягу зберігання без необхідності розширення фізичних площ;

2) раціональне розташування комплектуючих за принципами ABC–аналізу або згідно частоти відвантаження може значно скоротити час на підбір замовлень;

3) введення вертикального зберігання та мобільних стелажних систем звільнить площу і підвищити продуктивність складських операцій;

г) оптимізація внутрішніх складських процесів:

1) автоматизація процесів приймання, зберігання, комплектації та відвантаження комплектуючих знижує ймовірність помилок та підвищує швидкість обробки замовлень;

2) впровадження системи управління складом (WMS) дозволяє оптимізувати роботу персоналу та використання технічних засобів;

3) застосування принципів Lean та Kaizen в складській логістиці сприяє постійному вдосконаленню та зниженню витрат;

д) мінімізація часу на обробку замовлень:

1) скорочення часу від моменту отримання замовлення до його виконання забезпечує більшу задоволеність клієнтів та здатність швидко реагувати на зміни попиту;

2) оптимізація процесу підбору комплектуючих (picking) через впровадження технологій штрих–кодування та RFID підвищує точність здійснення відвантаження;

3) використання автоматизованих систем сортування та конвеєрів може різко скоротити витрачений час на фізичні переміщення комплектуючих.

Виконання замовлень, включаючи їх пошук та комплектацію, є одними з найбільш трудовитратних процесів з одного боку, а з іншого займає до 75% усіх трудовитрат на складі. Ефективність цих операцій значно залежить від двох ключових факторів: оптимізації маршруту переміщення і розташування комплектуючих у складському просторі [13].

З урахуванням цього є доцільним розгляд реалізації задачі комплектувальника в рамках втілення оптимізаційних алгоритмів в системі складського обліку.

1.3 Огляд найбільш вживаних систем обліку роботи складу

Оглянемо деякі популярні системи складського обліку, що створені для малого та середнього бізнесу в Україні, Євросоюзі, Великобританії та США, Визначимо, чи дійсно притаманні описані у попередніх розділах властивості для цих широко вживаних підприємцями систем.

Zipppy склад – облікова система для автоматизації малого бізнесу. Це безкоштовне ПЗ з відкритим кодом та веб-інтерфейсом, призначене для автоматизації малого бізнесу. Облікова система, орієнтована на вітчизняний ринок, має звичний інтерфейс, проста в установці та налаштуванні, з модульною архітектурою, що дозволяє легко адаптувати її під вимоги бізнесу.

Стандартний документообіг, звичний для вітчизняних користувачів (особливо звиклих до 1С) інтерфейс, можливість багатокористувацької роботи та роботи через Інтернет з будь якого мобільного пристрою.

Метою проекту розробники вважають надання альтернативи платним та/або зарубіжним обліковим програмам.

При створенні проекту використано простіший та поширеніший технологічний стек (PHP/Mysql), що забезпечує кросплатформеність за рахунок web–формату додатку, але зберігає можливості установки на будь якому хостингу або офісному комп'ютері для індивідуального використання. З точки зору архітектури проект є платформою, де основні об'єкти – документи, довідники, звіти можуть додаватись, видалятись та замінюватись без внесення змін в ядро платформи та структуру бази даних.

Слід зауважити, що система дійсно проста для використання, на що розраховували розробники, але має лише базові функції обліку, необхідні з точки зору фіскальної політики держави. Будь-які інструменти оптимізації та управлінського обліку відсутні, рівно як у базових версіях прообразу, системи 1С: Торгівля та склад. Інтерфейс системи Zipppy-склад наведено на рисунку 1.1.

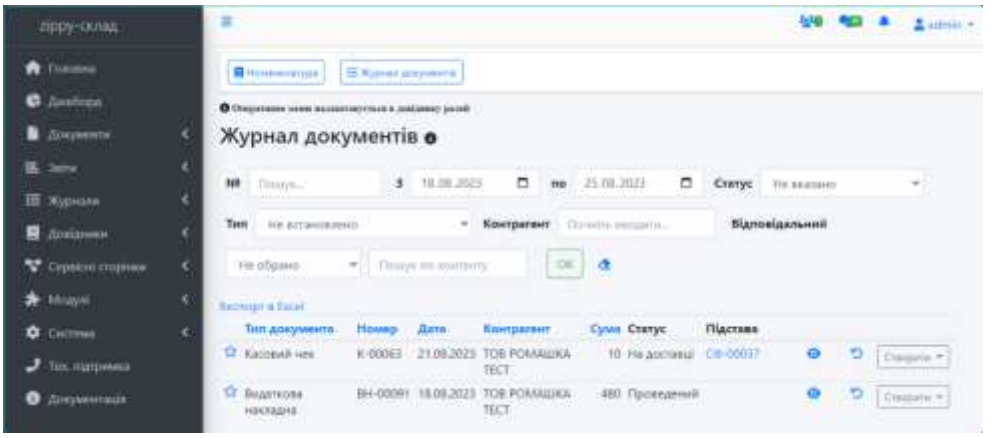

Рисунок 1.1 – Інтерфейс системи Zippy-склад

Система складського обліку LoMag. Розглядається платна версія, оскільки безкоштовна має обмеження, що унеможливлює її використання окрім як з метою ознайомлення з базовими функціями (рисунок 1.2).

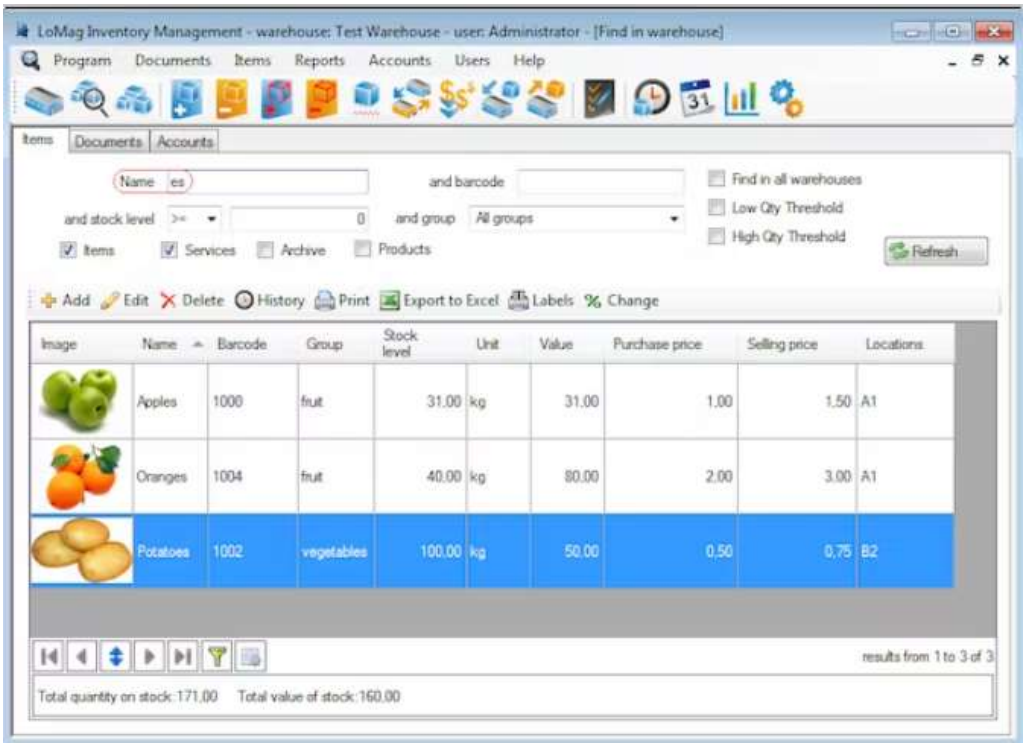

Рисунок 1.2 – Інтерфейс системи LoMag

Система має як десктоп версію, інтерфейс якої показано на рисунку 1.2., так й мобільний інтерфейс, але останній треба розглядати більш як додатковий, аніж у якості основного попри порівняний функціонал.

До основного функціоналу додатку можна віднести:

– відстеження кількості комплектуючих на складі в реальному часі;

– підтримку роботи з серійними номерами і термінами придатності комплектуючих;

– можливість проведення інвентаризації та коригування залишків;

– створення та друк основних складських документів, таких як накладні, прибуткові та видаткові ордери;

– підтримку сканування штрих-кодів для швидкої роботи з товарами;

– можливість генерації та друку власних штрих-кодів для комплектуючих і місць зберігання;

– генерацію різних звітів щодо руху комплектуючих, залишків і фінансових результатів;

– аналіз оборотності комплектуючих і визначення мертвих запасів;

– інтеграцію із зовнішніми пристроями, як-от принтери для друку етикеток і термінали збору даних.

Додаток можна визначити як такий, що має певну кількість аналітичних звітів та інтегрується із засобами автоматизація складського обліку, але в додатку знов відсутній блок націлений на оптимізацію процесів складського обліку.

JTL WaWi (WarenWirtschaftsSystem) – система складського обліку Інтернет-майданчика, що інтегрується із Shopify. Система націлена реагувати на виклики, що створює зазвичай великий асортимент інтернет – магазинів, та інтегрується із Shopify, стаючи його функціональним доповненням для back – офісу підтримки продажів. Якщо Shopify націлений надати максимально зручний вибір комплектуючих покупцеві, то JTL WaWi, приклад UI якого наведено нижче, відповідає за автоматичну актуалізацію переліку

комплектуючих, відстеження збирання та відправки комплектуючих, оплати замовлень, тощо. Саме великий асортимент створює правила, за якими працює система та пояснює її функціональність (рисунок 1.3).

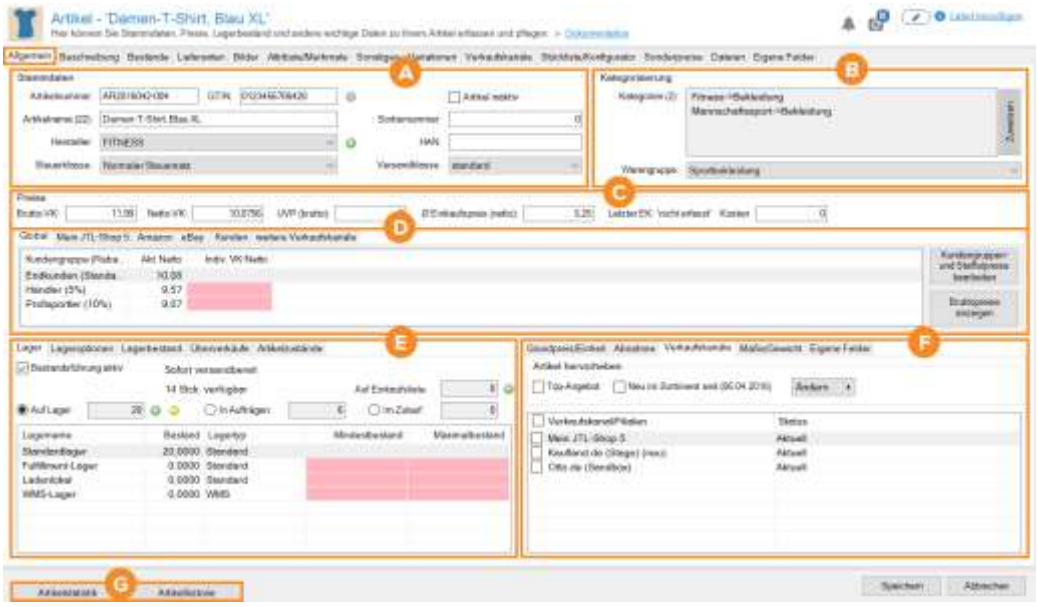

Рисунок 1.3 – Робочий екран JTL WaWi

Без подібної автоматизації завдання підтримання високого рівня сервісу інтернет – магазину із широким різноманітним асортиментом стає майже неможливим для персоналу. В додатку ми вже зустрічаємо карту збору замовлень, але вони розташовані за алфавітом. В якості альтернативи можна сортувати за складськими комірками, але враховується лише положення комірки в довіднику, а не їх реально взаємне розташування.

Якщо коротко виділити основний функціонал додатку:

– управління товаром (Product Information Management – PIM);
– управління цінами на товар;
– зберігання;
– доставка;
– виставлення рахунків за товари.

Отже, акцент зроблено саме на управління товарами, що пояснюється

орієнтацією на підприємства з численним асортиментом.

Fishbowl Inventory – американська система управління складом з 20-річним досвідом та інтеграцією з бухгалтерським програмним забезпеченням.

Відмінності американської системи управління складом – врахування особливостей складського обліку виробничого підприємства та інтеграція с бухгалтерським програмним забезпеченням, що має велику популярність в Сполучених Штатах – QuickBook. Такий тандем можна вважати майже ERP – системою для невеликих бізнесів, де не йде мова про створення потужних індивідуальних систем обліку та управління (рисунок 1.4).

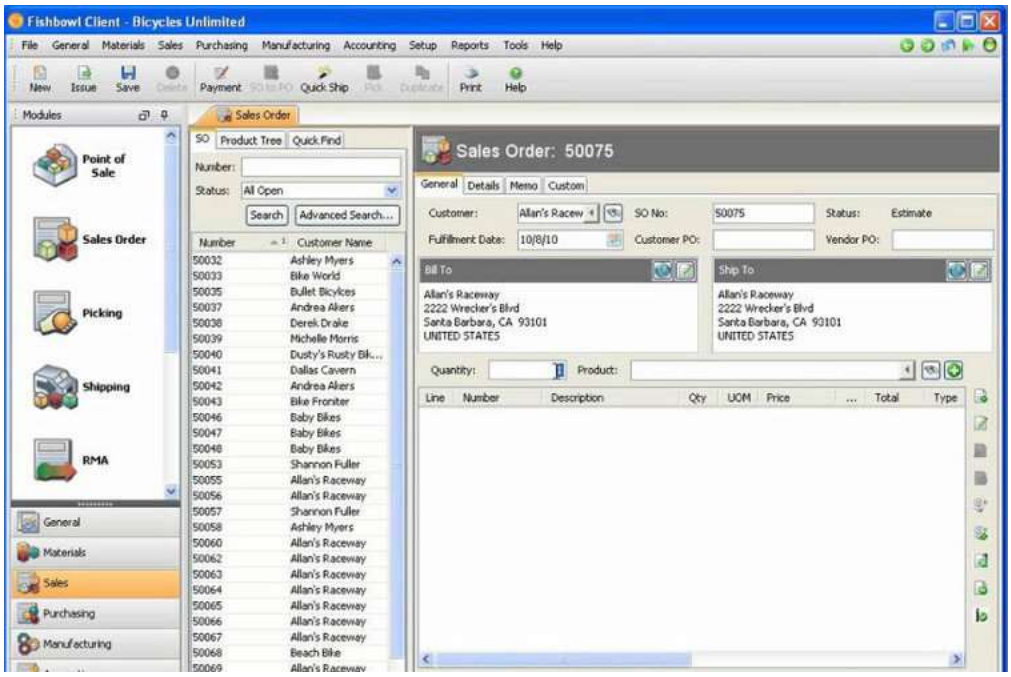

Рисунок 1.4 – Робочий екран Fishbowl Inventory

На відміну від попередньо оглянутих облікових систем, Fishbowl Inventory дозволяє управляти складанням на базі технологічних карт. Тобто, система має інформацію, з яких компонентів складається той чи інший виробничий зразок, дозволяє відстежувати рівень запасів необхідних для складання, враховуючі технологічні карти, та звичайно виводити перелік місць розташування компонентів, необхідних для складання. Але знов таки

збиральні карти використовують лише базові алгоритми сортування у переліку, не оперуючи інформацією з фактичного розташування того чи іншого компонента на складі. Також система не генерує інформацію про оптимальне розташування комплектуючих, що надходять на склад виробничого підприємства.

Підводячи підсумки огляду програмного забезпечення, націленого на складську логістику, управління складом, можна зробити наступні висновки.

– типові програмні рішення для малого та середнього бізнесу націлені, в першу чергу, на задоволення базових потреб складського обліку: підтримання актуального стану інформації про асортимент, наявність та кількість комплектуючих; задоволення потреб, пов’язаних з фіскальними вимогами у відповідних країнах; друк накладних та інвойсів;

– вирішення задач оптимізації на математичному базисі, зокрема з використанням комбінаторних алгоритмів, машинного навчання та штучних нейронних мереж є частиною більш складних та коштовних систем, а також систем, що розробляються підприємствами індивідуально та на замовлення;

– найбільш простою ланкою, що поєднує внутрішню складську логістику та виробничі процеси є технологічна карта складання. Саме карта складання може бути завданням – відправною точкою задачі комплектувальника, що була згадана в розділі 1.2.

1.4 Опис задачі комплектування

Одним з простих, але дієвих та ефективних шляхів оптимізації внутрішньої логістики можна вважати оптимізацію технології комплектування, що була згадана у підрозділі 1.2.

Оптимізаційний підхід у задачі маршрутизації комплектувальника відноситься до розробки ефективного маршруту в межах складських операцій, зокрема для збору комплектуючих з різних місць на складі. Це важливо для зниження часу на збір замовлення і підвищення ефективності складських

процесів.

Сформулюємо задачу математично та проілюструємо на прикладі моделі складу.

Нехай $S = \{s_1, s_2, \ldots, s_n\}$ представляє собою множину локацій на складі, де s_i – це конкретна локація. Реальним аналогом локації є комірка, в якій зберігаються комплектуючі (товар).

Комплектувальник повинен відвідати підмножину цих локацій $P \subset S$, щоб зібрати всі необхідні товари. Множина P визначається, наприклад, технологічною картою, що містить перелік необхідних комплектуючих для складання певного виду продукції.

Визначимо функцію відстані $d(s_i, s_j)$, яка вимірює відстань між локаціями s_i та s_j на складі.

Задача полягає у визначенні послідовності відвідувань локацій P_{seq} такої, що мінімізується загальна пройдена відстань, тобто мінімізується (1.1)

$$D = \sum_{i=0}^{n+1} d(p_i', p_{i+1}'), \tag{1.1}$$

де $P_{seq} = \{p_1', p_2', \ldots, p_{n+1}'\}$, $p_i' \in P$.

Зауважимо, що у послідовності P_{seq} першим та останнім елементом є точка входу. Будемо вважати також, що відбувається збірка відносно малогабаритних продуктів, а значить не накладається додаткових обмежень на вагу та об’єм комплектуючих, що можуть бути зібрані комплектувальником за один тур: збираються усі компоненти згідно технологічної карти.

Звісно, ще більшої оптимізації можна досягти за рахунок поєднання кількох складальних списків, але це фактично зводиться до поєднання декількох списків у один з подальшим зворотнім розподілом на кілька груп. Тому будемо розглядати підходи до вирішення базової задачі оптимізації збирання одиночного переліку комплектуючих (компонентів) у заданій

кількості та розташуванні. При цьому один вид комплектуючих може бути розташований відразу у декількох місцях складу.

Схематично модель складу, що використовується в роботі, показана на рисунку 1.5. Рух складальника або автомата, що його замінює, починається на точці входу, що співпадає з початком координат, та закінчується у тій самій точці.

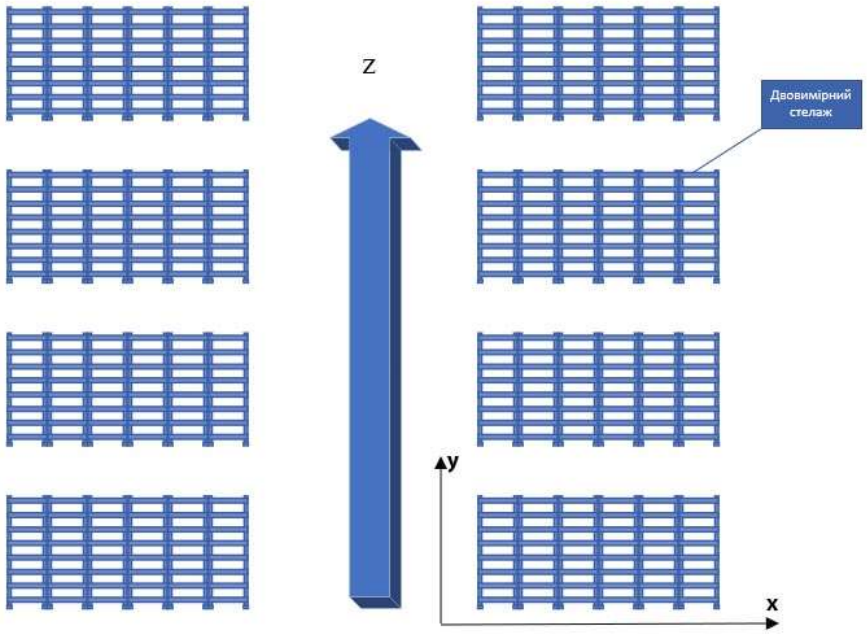

Рисунок 1.5 – Схема моделі складу

Відстань між комірками зберігання $d(s_i, s_j)$ визначається як сума відстаней від проходу до комірок та відстані між стелажами, що показано в (1.2)

$$d(s_i, s_j) = d_i^{xy} + d_z + d_j^{xy}, \quad (1.2)$$

де d_i^{xy}, d_j^{xy} – відстань від проходу до першої та другої комірки відповідно;

d_z – відстань між стелажами.

Зрозуміло, що розрахунок (1.2) представлений для деякого умовного складу, а в реальності відстань між комірками може бути результатом більш складного розрахунку. В такому випадку відбудуться заміна функції відстані $d(s_i, s_j)$, а інші елементи алгоритму не зазнають змін, що відповідає принципу інкапсуляції в розробці програмного забезпечення.

Отже, сформульовано одну з задач оптимізації складської логістики, що є доволі актуальною, а саме комбінаторну задачу комплектації. Саме цю задачу змоделюємо та виконаємо під час створення MVP програмного забезпечення управління складом з використанням оптимізаційних алгоритмів.

В наступному підрозділі виконаємо постановку задачі, а також сформулюємо функціональні та нефункціональні вимоги до програмного забезпечення, що має бути сконструйовано та розроблено для вирішення сформульованої задачі комплектації в рамках системи складського обліку та управління складом.

1.5 Постановка мети та задач кваліфікаційної роботи

Виконаний огляд сучасного стану проблеми комплектації виробничих технологічних процесів й аналіз існуючих систем автоматизації відповідних процесів дозволив зробити висновок щодо необхідності подальшого підвищення їх ефективності.

Сформульована мета кваліфікаційної роботи полягає у підвищення ефективності системи комплектації виробничого технологічного процесу за рахунок розробки засобу оптимізації маршрутів комплектувальників.

Для досягнення сформульованої мети необхідно:

– сформулювати постановку задачі розробки компонентів системи комплектації для оптимізації маршрутів комплектувальників;

– обрати методи та розробити алгоритми вирішення задачі оптимізації;

– провести тестування готової системи комплектації.

Характеристика комплексу задач:

а) призначення комплексу задач. Комплекс задач призначений для ведення кількісного обліку комплектуючих на складі, місць їх розташування (комірок), а також оптимізації маршрутів комплектувальника на складі, що забезпечує більш ефективний процес збору комплектуючих;

б) перелік об’єктів:

1. комплектуючі (SKU – stock keeping unit);

2. складські зони для зберігання комплектуючих – комірки (cells);

3. комплектувальники та інші співробітники, механізми та роботи, що автоматизують процес комплектування;

в) періодичність та тривалість рішення. Рішення використовується безперервно під час роботи складу. Нові записи генеруються кожен раз, коли відбувається надходження та вибуття SKU зі складу. Оптимізований маршрут комплектування формується кожен раз коли виникає потреба в комплектуванні нової партії комплектуючих згідно відповідної технологічної карти;

г) умови припинення рішення. Умови припинення рішення визначаються технологією використання системи;

д) зв’язок з іншими комплексами задач. Комплекс задач, що розглядається, тісно пов'язаний з іншими підсистемами системи управління підприємством ERP, а за її відсутності – з іншими підсистемами бухгалтерського та управлінського обліку на підприємстві;

е) посади та підрозділи. Підрозділ, в якому розробляється та виконується задача автоматизації – склад. Керівництво складу визначає політику оптимізації. Оператор ПК складу відповідає за облік приходу та вибуття SKU, генерацію маршрутів збирання відповідно до запитів, що надходять, підтримання технологічних карт в актуальному стані. Комплектувальники виконують збір замовлень, що складаються з певного

переліку SKU, за згенерованим системою маршрутом;

ж) розподіл дій:

1) приход SKU виконується оператором ПК складу;

2) після надходження запиту на комплектування з виробничого підрозділу оператор ПК складу генерує відповідний маршрут збирання;

3) комплектувальник виконує збирання SKU за технологічною картою;

4) оператор ПК реєструє вибуття комплектуючих по факту закінчення збирання комплектуючих.

Вихідна інформація:

– оптимізований маршрут збирання. Складається з переліку пар (комірка, SKU), що розташовані в алгоритмічно оптимізованій послідовності, що мінімізує маршрут збирання. Для кожного рядку (пари) вказуються координати комірки (стелаж, полиця, стовпчик) та найменування SKU, що має бути додано до комплекту, що збирається;

– відомість товарних залишків. До відомості включаються усі SKU, залишок яких на складі ненульовий. Для кожного SKU проставляється відповідний залишок, рядки сортуються у прямому алфавітному порядкує.

Вхідна інформація:

– запит на збирання за відповідною технологічною картою. Запит на збирання містить інформацію щодо номеру технологічної карти та кількості комплектів, що мають бути зібрані;

– приходна накладна. Приходна накладна містить перелік комплектуючих, що мають бути додані до складських залишків та кількість кожного такого товару.

Сформулюємо тепер функціональні та нефункціональні вимоги до програмного забезпечення автоматизації складського обліку з урахуванням огляду типових складських систем та поставленої задачі оптимізації задачі комплектувальника. Зауважимо, що формульовані вимоги відповідають моделі програмного забезпечення, що є макетом WMS системи з алгоритмами

оптимізації. Програмне забезпечення, що створюється, не є повноцінною та повнофункціональною WMS системою.

Функціональні вимоги до програмного забезпечення:

– ведення складського обліку в розрізі SKU (Stock Keeping Unit);

– незалежне ведення обліку місць розташування товару, можливість розташування однієї складської одиниці товару (SKU) в декількох місцях розташування;

– зберігання інформації про додаткові властивості комплектуючих, що визначаються відповідними довідниками;

– забезпечити додавання, редагування, видалення SKU, їх властивостей, місць зберігання, технологічних карт та іншої довідкової інформації в графічному інтерфейсі користувача;

– забезпечити автоматичне генерування складального списку на підставі технологічної карти з врахуванням місць розташування компонентів (SKU), можливої відсутності або недостатньої кількості компонентів;

– забезпечити можливості налаштування та зміни параметрів складу;

– відтворювати технологічні карти у вигляді ієрархічного списку;

– відтворювати інформацію про складські залишки.

Нефункціональні вимоги до програмного забезпечення:

– в якості інтерфейсу додатку використовувати десктоп – додаток з графічним інтерфейсом;

– забезпечити зберігання даних облікової системи незалежно від самої системи та можливість віддаленого зберігання даних на сервері, в хмарі, тощо;

– при створенні архітектури програмного забезпечення передбачити можливість використання оптимізаційного алгоритму незалежно від графічного інтерфейсу програми, що в подальшому дозволить створити інші інтерфейси користувача системи – мобільний та web.

1.6 Висновки до розділу 1

На початку розділу 1 було оглянуто та перелічено основні характеристики та переваги систем внутрішньої складської логістики, що дозволяють не тільки вести облік комплектуючих, але й виконувати задачі з оптимізації руху комплектуючих, що в свою чергу створює додаткову економію для підприємства. Але на практиці такі складні функції притаманні коштовним та розробленим на замовлення комплексним ERP-системам у той час, як системи спрямовані на аудиторію малого та середнього бізнесу виконують лише базові облікові функції та допомагають підприємствам виконувати фіскальні вимоги. Серед систем забезпечення внутрішньої логістики, створених розробниками різних країн, лише дві системи підтримують роботу з великим асортиментом комплектуючих та дозволяють працювати з списками збирання (складання), й лише одна з систем має певні інструменти обліку складу виробничого підприємства. При цьому в жодній з систем не вирішується хоча б одна з численних задач оптимізації внутрішньої логістики шляхом імплементації алгоритмів комбінаторної оптимізації або будь-яких інших.

На підставі огляду доступного програмного забезпечення, спрямованого на автоматизацію внутрішньої логістики, було поставлено задачу створення прототипу програмного забезпечення, що не тільки дозволяє вести облік складських одиниць та місць зберігання, але й вирішує задачу оптимізації маршруту комплектувальника за допомогою інструментарію комбінаторної оптимізації. Оптимізація маршрутів збирання дозволяє, в свою чергу, оптимізувати ресурси, що витрачаються на збирання компонентів.

В завершення розділу зроблено постановку задачі на розробку автоматизованого рішення, а також сформульовано функціональні та нефункціональні вимого до нього. В подальших розділах буде розглянуто математичний базис вирішення відповідної задачі комбінаторної оптимізації, а також змодельовано, сконструйовано та реалізовано програмне

забезпечення, що відповідає поставленій в розділі 1.5 задачі та сформульованим в ньому функціональним та нефункціональним вимогам.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Математична модель задачі оптимізації маршрутів комплектувальника

Поставлена задача оптимізації є інтерпретацією добре відомої задачі комбінаторної оптимізації, що має назву «задача комівояжера» (TSP – Traveling Salesman Problem). В класичному формулюванні задача полягає в пошуку найкоротшого можливого маршруту, який проходить через кожне місто зі списку рівно один раз і повертається у вихідне місто. Завдання комівояжера належить до класу NP-важких завдань, що означає, що не існує відомого ефективного способу знайти точне рішення для великих списків міст [15–17].

Задача комівояжера вперше була сформульована в 1800-х роках, а математичне вивчення цієї проблеми почалося в 1930-х роках. Вона була широко обговорюваною в галузі математики та економіки, особливо у зв'язку з плануванням і логістикою [18–19].

У інтерпретації, що розглядається в рамках задач внутрішньої логістики змінюється лише спосіб розрахунку відстаней між «містами», якими в даному випадку становляться комірки зберігання відповідних комплектуючих. Окрім того, маршрут завжди починається та закінчується в нульовій точці, що співпадає з входом на склад. В іншому задача повністю повторює класичне формулювання задачі комівояжера, що полягає у пошуку мінімальної відстані між певними пунктами.

Під якістю того чи іншого методу найчастіше розуміють співвідношення мінімальної відстані, що може бути знайдена повним перебором, та середньої відстані, що знаходить алгоритм в межах певної кількості ітерацій. Усереднення необхідне оскільки, на відміну від повного перебору, алгоритми не генерують ідемпотентні рішення.

Найвідоміші методи вирішення задачі комівояжера включають: Метод гілок та меж; методи наближеного вирішення.

а) Метод гілок та меж (Branch and Bound) [18]:

1) метод систематично досліджує всі можливі часткові рішення задачі;

2) ефективно зменшує область пошуку за допомогою "меж", які визначають мінімально можливе рішення всередині підмножини;

3) метод дозволяє генерувати рішення досить високої якості, але ефективність сильно знижується зі збільшенням розміру задачі.

б) Методи наближеного вирішення (Heuristics) [19]:

1) жадібний алгоритм (Greedy Algorithm): швидкий, але часто знаходить підоптимальні рішення, засновані на локальних, не враховуючи більш широкий контекст;

2) алгоритм найближчого сусіда (Nearest Neighbor Algorithm): простий та інтуїтивно зрозумілий, починається з випадкового міста і на кожному кроці переходить до найближчого міста, яке ще не було відвідано. Діє швидко, але генерує результати відносно низької якості;

3) алгоритм Кристофідеса (Christofides Algorithm): забезпечує рішення, яке не гірше, ніж в 1,5 рази довше оптимального шляху, використовуючи мінімальне остовне дерево і досконале паросполучення. Діє набагато швидше ітеративних алгоритмів, що відноситься до його переваг.

в) Метаевристичні алгоритми [20]:

1) імітований відпал (Simulated Annealing): імітація процесу відпалу, під час якого система поступово охолоджується, і є ймовірність тимчасового погіршення рішення для уникнення локальних оптимумів;

2) генетичний алгоритм (Genetic Algorithm): використовує механізми природного відбору, такі як мутація, кросовер і селекція для пошуку оптимального рішення;

3) оптимізація мурашиних колоній (Ant Colony Optimization):

натхненний поведінкою мурах, цей метод використовує багато агентів (мурахів), які досліджують простір рішень і комунікують за допомогою феромонів для знаходження оптимальних шляхів.

г) Інтелектуальні методи (Intelligent Techniques) [19]:

1) пошук з табу (Tabu Search): застосовується список заборон (табу), який запобігає повторенню раніше відвіданих рішень;

2) пошук зі змінною околицею (Variable Neighborhood Search): систематично змінює структуру околиці в пошуках кращого рішення.

ґ) Точні алгоритми [17]:

1) динамічне програмування (Dynamic Programming): розбиває проблему на менші підзадачі і вирішує їх послідовно, але експоненційно зростає з збільшенням розміру вхідних даних;

2) цілочисельне лінійне програмування (ILP – Integer Linear Programming): формулює задачу у вигляді цілочисельного лінійного програмування та вирішує за допомогою спеціалізованих оптимізаційних програм.

В ході роботи використаємо алгоритм Кристофідеса [21] як такий, що дозволяє за короткий час згенерувати рішення, що не більше аніж в 1.5 рази гірше оптимального. Це означає, що це рішення буде набагато кращим у порівняння із довільним списком, але впорядкуванням за алфавітом. При цьому будуватимемо архітектура програмного забезпечення таким чином, що обраний алгоритм оптимізації може бути замінений іншим без зміни інших модулів програмного коду. Для цього використаємо адаптер, як буде показано в частині, що присвячена програмній реалізації.

Алгоритм Кристофидеса був запропонований у 1976 році і є одним із найефективніших евристичних методів для задачі комівояжера. Далі надамо покроковий опис алгоритму, припустивши для наочності, що необхідний пошук мінімальної відстані між п’ятьома точками. Відстані позначено на графі, що представлений на рисунку 2.1, вони є вагами ребер.

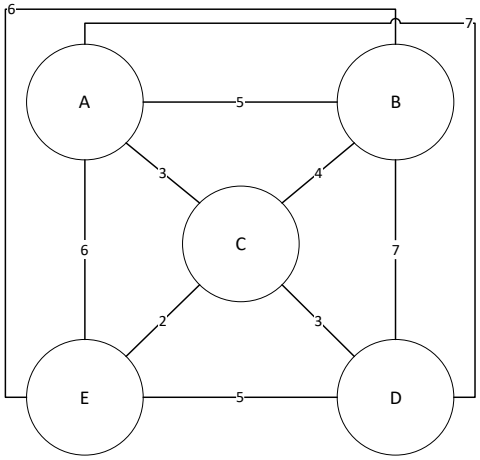

Рисунок 2.1 – Початковий граф відстаней

Побудова мінімального охоплюючого дерева (МОД) для графу. Використовуючи алгоритм Пріма чи Крускала, знаходимо мінімальне охоплююче дерево. Охоплююче дерево є підграфом, який з'єднує всі вершини оригінального графу без утворення циклів і має мінімальну можливу суму ваг ребер. Приклад мінімального охоплюючого дерева наведено на рисунку 2.2.

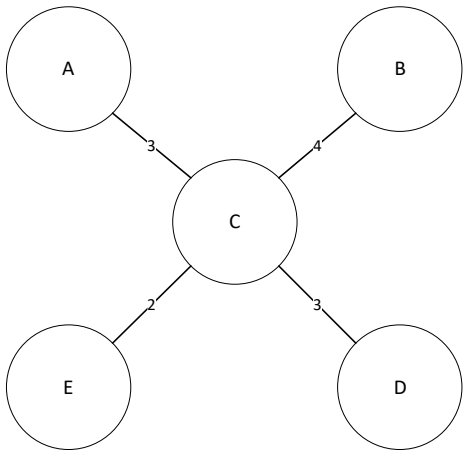

Рисунок 2.2 – Приклад мінімального охоплюючого дерева

Визначення множини вершин з непарним ступенем у МОД. У мінімальному охоплюючому дереві деякі вершини матимуть непарний ступінь

(кількість ребер, що ведуть до вершини). Знаходимо ці вершини, тому що для отримання ейлерового циклу (циклу, який проходить через кожне ребро рівно один раз) кожна вершина повинна мати парний ступінь (рисунок 2.3).

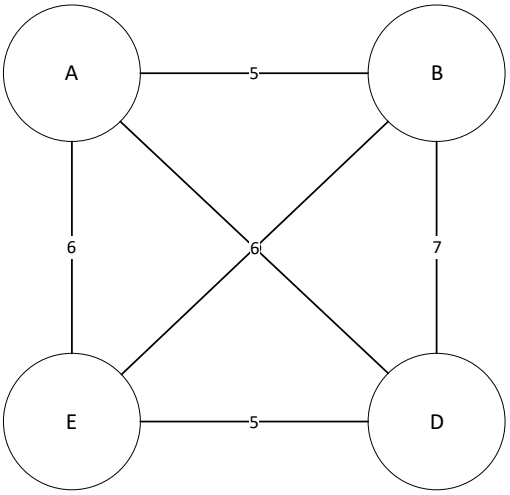

Рисунок 2.3 – Проекція основного графа (тільки непарні вершини)

Побудова мінімального паросполучення для вершин з непарним ступенем. Використовуємо алгоритм для пошуку мінімального паросполучення, який з'єднає вершини з непарним ступенем парами з мінімальною загальною вагою ребер. Це гарантує, що додаванням цих ребер до МОД кожна вершина отримає парний ступінь (рисунок 2.4).

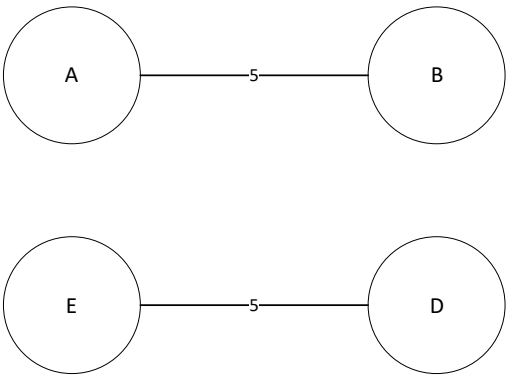

Рисунок 2.4 – Довершене паросполучення

Об'єднання МОД з мінімальним паросполученням для створення багатократного графу. Додаємо ребра з мінімального паросполучення до МОД, створюючи багатократний граф, в якому всі вершини мають парний ступінь, як на рисунку 2.5. Тепер можливо знайти в цьому графі ейлерів цикл.

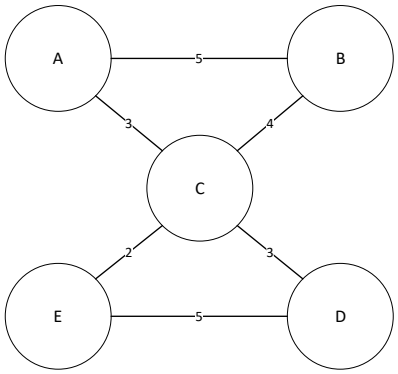

Рисунок 2.5 – Паросполучення і дерево, що стягує

Перетворення ейлерового циклу в гамільтонів цикл. Виходимо з ейлерового циклу і перетворюємо його в гамільтонів, представлений на рисунку 2.6. Отримуємо обхід, що включає кожну вершину рівно один раз, що і є рішенням задачі комівояжера.

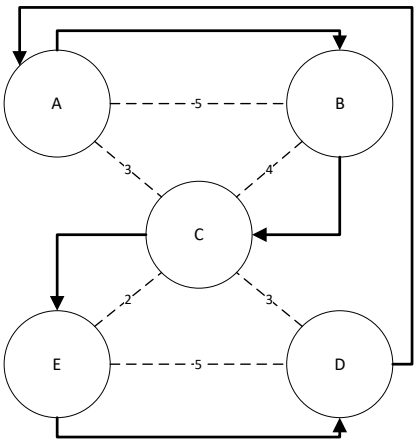

Рисунок 2.6 – Гамільтонів обхід

Окрім точок маршруту збірки в ньому присутня початкова точка, що збігається з кінцевою – це точка входу на склад/виходу зі складу. Оскільки протяжність маршруту не залежить від початкової точки, як і сам алгоритм формування оптимального маршруту за Кристофідесом, а задача комівояжера передбачає повернення в початкову точку, достатньо після формування маршруту переформувати його таким чином, що початкова/кінцева точка збігалася з точкою входу на склад.

2.2 Сценарії використання системи

Діаграма сценаріїв використання (Use Case Diagram) – це графічне представлення взаємодій між користувачами системи та її функціональністю, яке використовується при моделюванні програмного забезпечення за допомогою UML [22].

Розглянемо основні сценарії використання системи та відобразимо їх графічно шляхом побудови діаграм сценаріїв використання. На рисунку 2.7 показаний сценарій використання ієрархічного переліку технологічних карт. На верхньому рівні ієрархії знаходяться продукти, що збираються, на нижньому – товарні одиниці (SKU), з яких відбувається збирання.

Окремо розглядаємо сценарій використання, що відповідає CRUD – операціям з SKU та довідниками, що визначають їх властивості та ознаки. Під властивостями розуміємо ті характеристики товару, що вимірюються. Під ознаками ті, що визначається символьним виразом. Наприклад, до властивостей відноситься «ємність» акумулятору, а до ознак його «колір».

На рисунках 2.8 та 2.9 представлено сценарії використання, що пов’язані з редагуванням довідників. На рисунку 2.8 відображено редагування довідника SKU, а на рисунку 2.9 – редагування довідників списком.

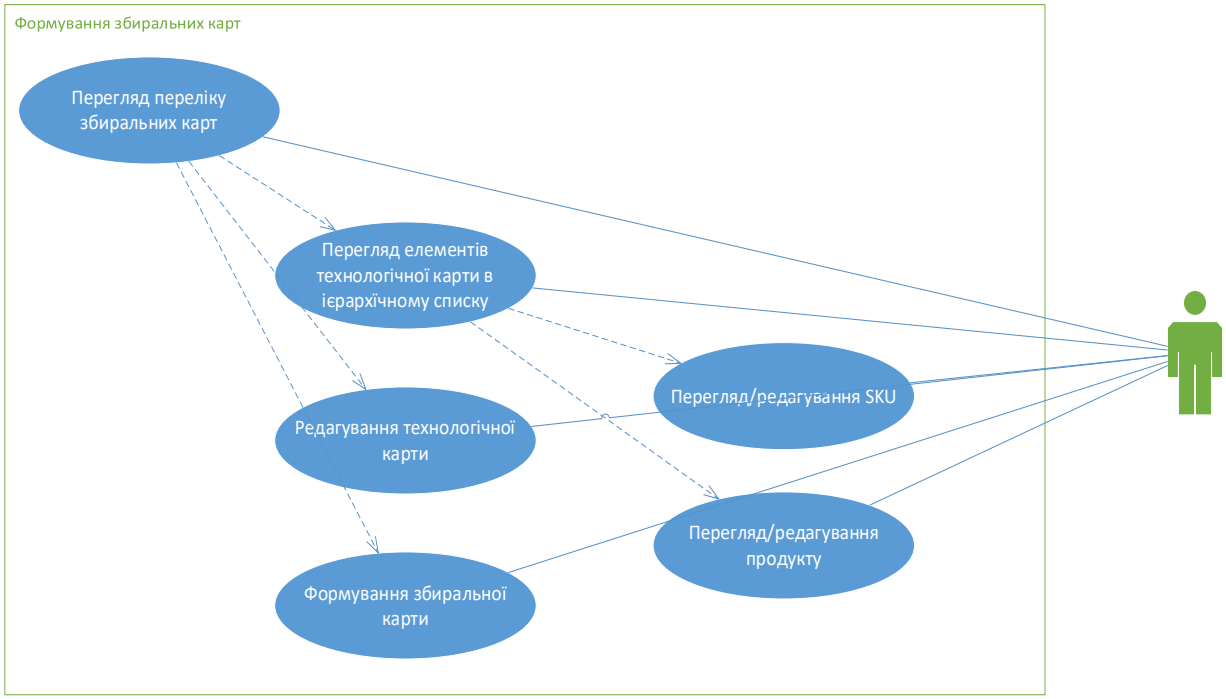

Рисунок 2.7 – Сценарій використання переліку збиральних карт

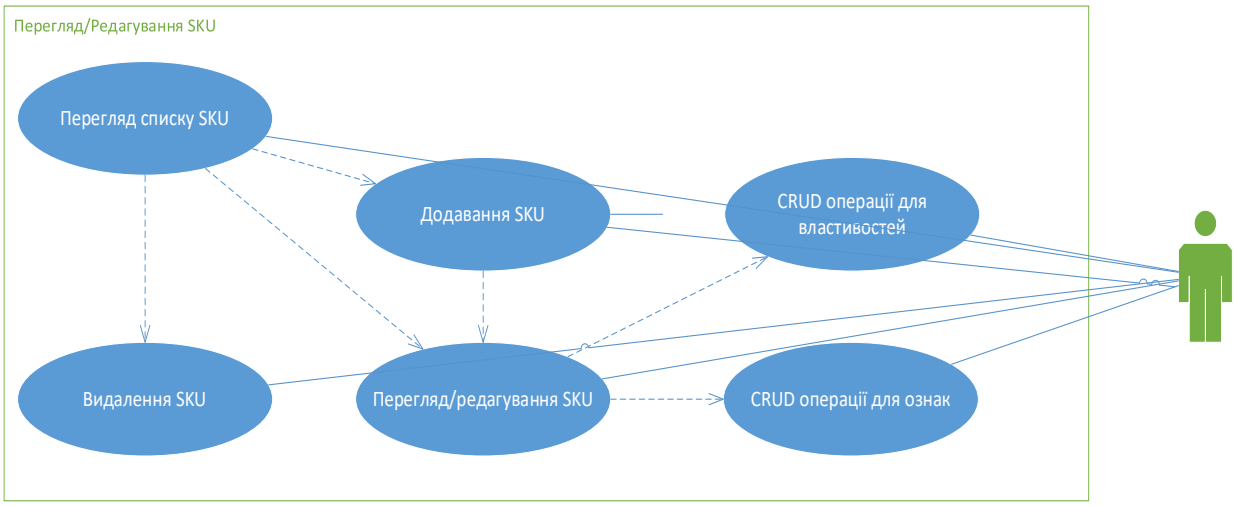

Рисунок 2.8 – Перегляд списку та редагування SKU

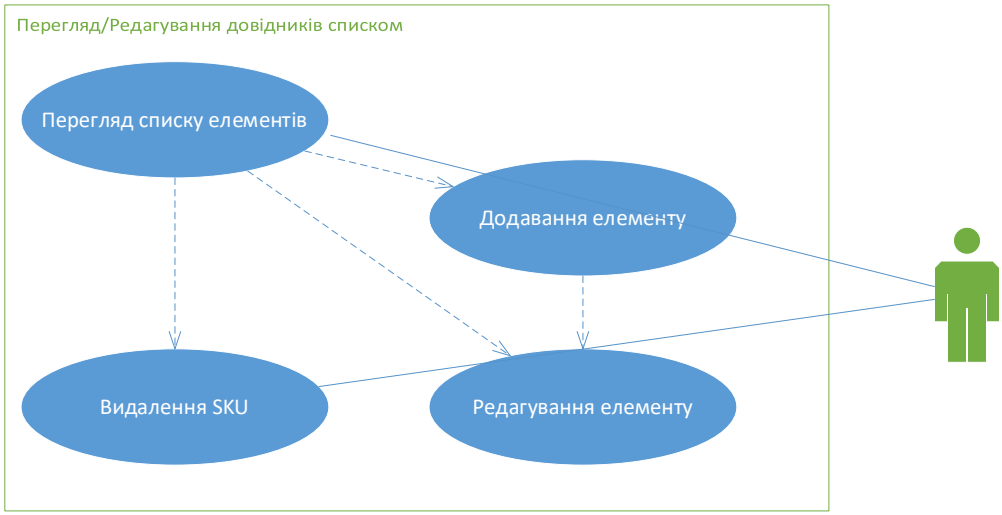

Рисунок 2.9 – CRUD операції інших довідників

2.3 Діаграми діяльності для основних сценаріїв використання

При моделюванні проектованої системи часто необхідно деталізувати особливості алгоритмічної та логічної реалізації операцій, що виконуються системою. Зазвичай для цього використовуються блок–схеми.

У мові UML моделювання процесу виконання операцій використовуються діаграми діяльності (Activity Diagrams).

Кожен стан на діаграмі діяльності відповідає виконанню деякої елементарної операції. Саме діаграми діяльності дозволяють реалізувати у мові UML особливості процедурного та синхронного управління. Метамодель UML надає для цього необхідні терміни та семантику. Основним напрямом використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно уявити алгоритми їх виконання. При цьому кожен стан може бути виконанням операції деякого класу або її частини, дозволяючи використовувати діаграми діяльності для опису реакцій на внутрішні події системи.

Стан дії (Action State) є спеціальним випадком стану з деякою вхідною дією і принаймні одним переходом, що виходить зі стану. Стан дії не може

мати внутрішніх переходів, оскільки є елементарним. Дія може бути записана природною мовою, деяким псевдокодом або мовою програмування.

Перш за все розглянемо основні функції та алгоритми системи, залишивши поза увагою послідовності виконання кроків для типових CRUD операцій.

На рисунку 2.10 наведено діаграму діяльності, що відповідає алгоритму Кристофідеса, а на рисунку 2.11 – діаграму діяльності, що відповідає сценарію використання «Формування збиральної карти». Алгоритм Кристофідеса відтворено відповідно до першого розділу роботи, де викладено послідовність дій в ньому. Він використовується в 2.11 на кроці «Формування оптимального шляху.

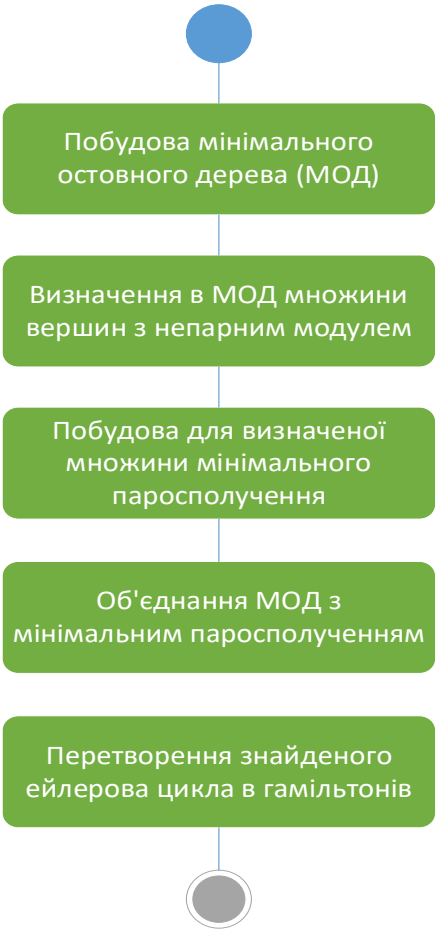

Рисунок 2.10 – Діаграма алгоритму Кристофідеса

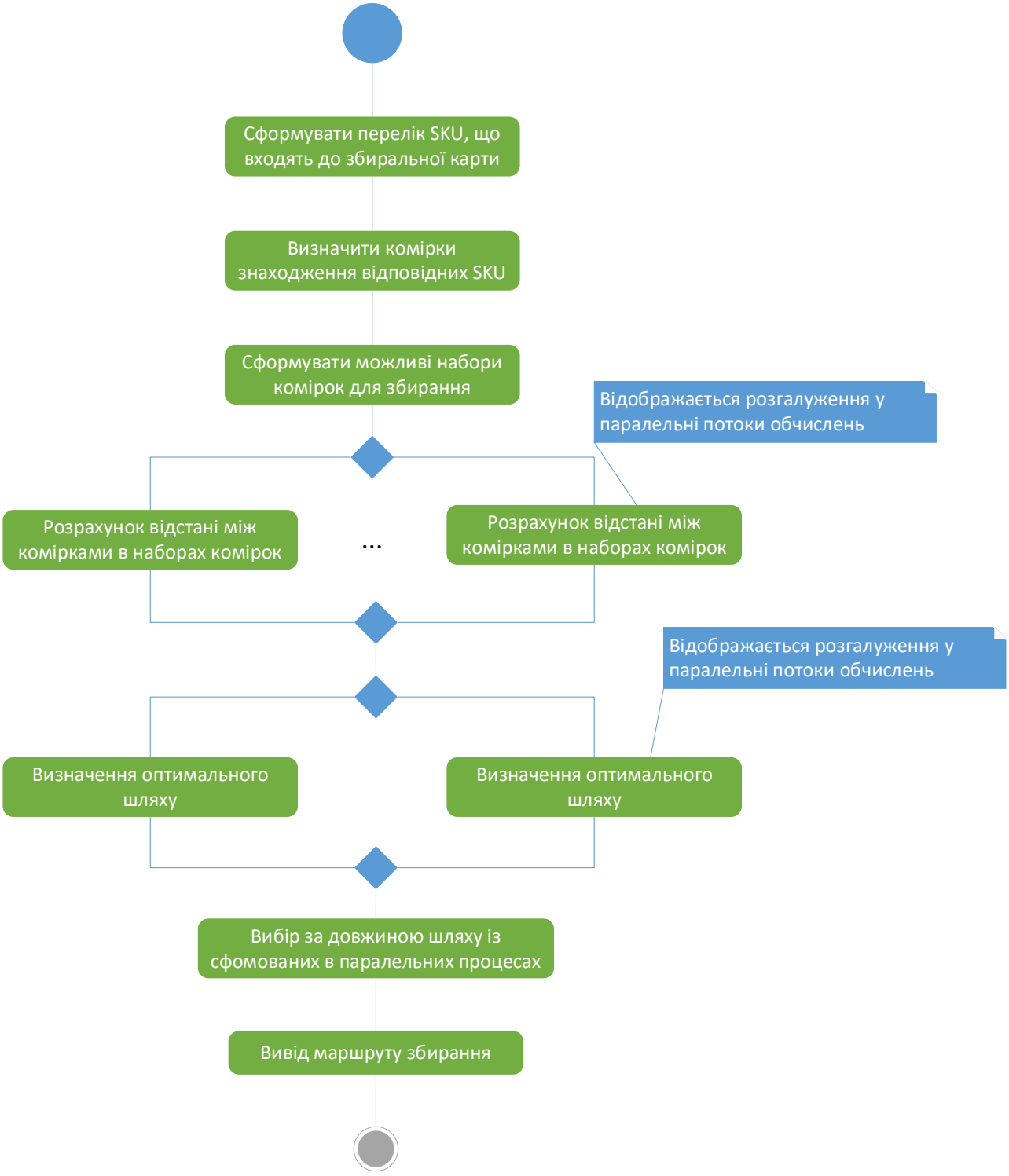

Рисунок 2.11 – Діаграма діяльності «Формування збиральної карти»

Для основної діаграми діяльності, що наведена на 2.11, необхідно пояснити певні аспекти роботи. За рахунок того, що одна й та сама складська

одиниця SKU може зберігатись в декількох комірках складу, на кроці «Сформувати можливі набори комірок» може бути сформовано декілька наборів комірок. Для кожного з цих наборів необхідно прорахувати відстані між комірками та на підставі цієї інформації визначити оптимальний порядок обходу комірок.

Оскільки на кожному кроці алгоритму Кристофідеса використовуються результати, отримані на попередньому, схема його виконання – лінійна.

2.4 Діаграма сутностей системи

Діаграма сутностей (або діаграма сутностей-зв'язків, ER діаграма – Entity–Relationship diagram) – це тип діаграми, яка використовується для візуалізації структури бази даних у формі абстрактної моделі. Вона допомагає визначити дані, які будуть зберігатися в базі даних, та зв'язки між різними сутностями. Вона використовується для моделювання даних, виявлення сутностей та зв’язків між ними. Також при побудові діаграми сутностей визначаються атрибути сутностей, первинні та зовнішні ключи.

Типи зв'язків сутностей в моделі сутностей-зв'язків (Entity-Relationship Model) визначають взаємини між сутностями в базі даних. Основні типи зв'язків:

– один до одного (1:1): кожен екземпляр однієї сутності асоційований з одним екземпляром іншої сутності;

– один до багатьох (1:M або 1..*): один екземпляр сутності А може бути асоційований з багатьма екземплярами сутності Б, але кожен екземпляр сутності Б асоційований лише з одним екземпляром сутності А. В деяких джерелах від цього типу зв’язку відокремлюють зв’язок «Багато до одного», що знаходить обґрунтування в практиці роботи з базами даних;

– багато до багатьох (M:N або *:*): екземпляри однієї сутності можуть бути асоційовані з багатьма екземплярами іншої сутності, і навпаки.

Ці зв'язки дозволяють розробникам баз даних точно визначати

структуру даних і є ключовими для нормалізації бази даних, яка допомагає зменшити дуплікацію даних, підвищивши їх цілісність. У фізичному дизайні бази даних, зв'язки "багато до багатьох" реалізовуються через створення додаткової таблиці, яка зв'язує ключі обох основних таблиць.

Діаграма сутностей використовується також при створення моделей в ORM, наприклад, ORM Django або SQLAlchemy. ORM дозволяє працювати з базами даних на більш високому рівні абстракції. Для системи, що моделюється, діаграма сутностей наведена на рисунку 2.12.

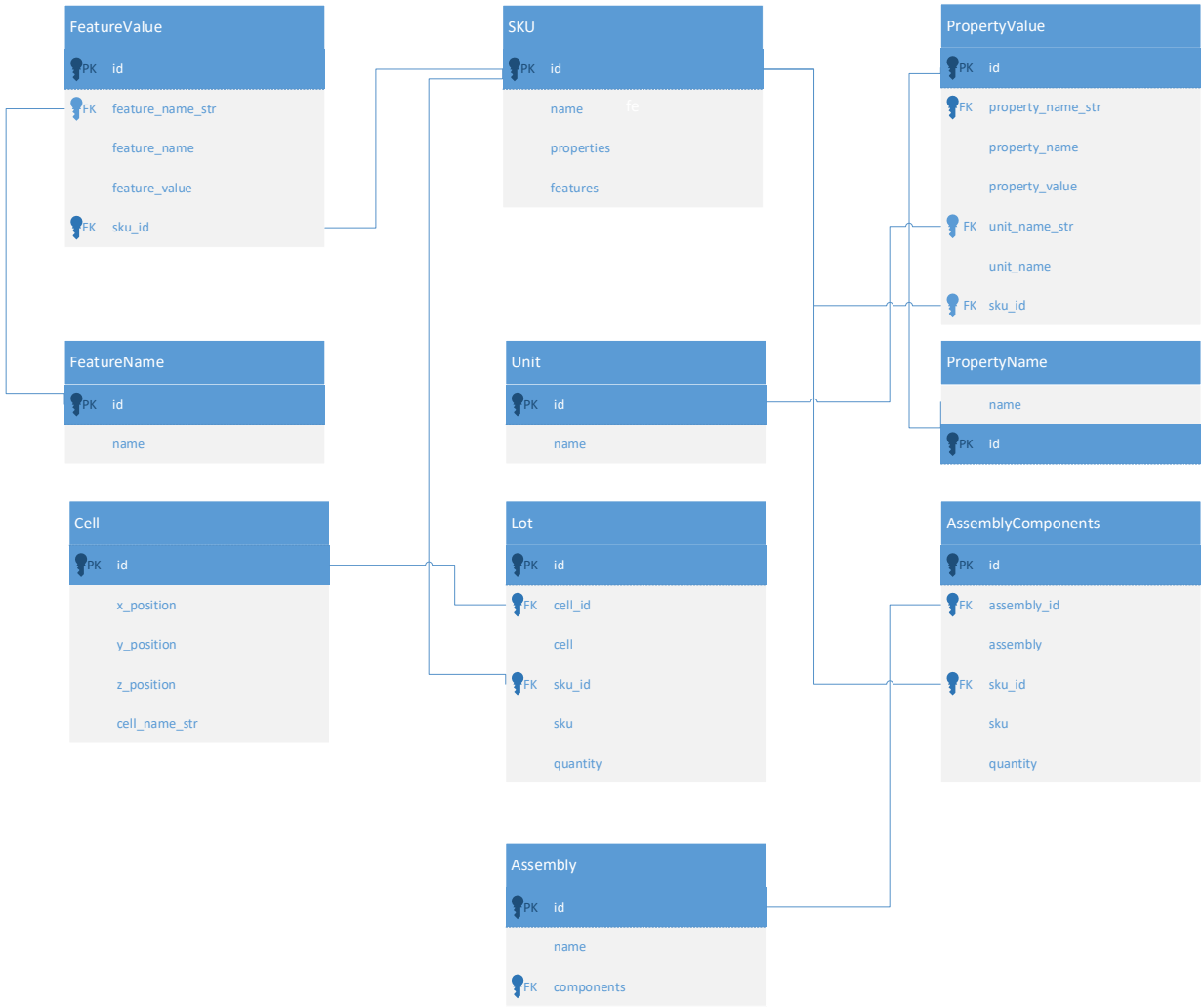

Рисунок 2.12 – Діаграма сутностей системи

Основними сутностями є SKU, Lot та Assembly. SKU, як вже визначалося раніше, відтворює складську одиницю, що має певні

характеристики та властивості. Для урахування можливості зберігання однієї складської одиниці в декількох комірках додатково введено сутність Lot, яка відтворює партію зберігання SKU. Кожний екземпляр цієї сутності ідентифікується складською одиницею та коміркою зберігання.

Зберігання технологічних карт реалізовано сутностями Assembly та AssemblyComponent, між якими зв'язок 1…n. Таким чином зберігаємо інформацію про перелік складських одиниць, потрібних для збирання того чи іншого продукту. Компоненту збірки поставлена у відповідність складська одиниця, а також для нього визначена необхідна кількість складських одиниць. Сутності PropertyValue та FeatureValue відповідають за зберігання значень властивостей та ознак складських одиниць. Механіка 1..n дозволяє зберігати довільну кількість різних наборів ознак та властивостей для кожної складської одиниці. Сутності PropertyName, FeatureName, Unit є довідниками й допомагають визначити довільні набори можливих ознак, властивостей та одиниць зберігання.

2.5 Діаграма компонентів системи

Діаграма компонентів дозволяє визначити склад програмних компонентів, ролі яких може виступати вихідний, бінарний і виконуваний код, і навіть встановити залежності з-поміж них.

При розробці діаграм компонентів переслідуються цілі:

– специфікація загальної структури вихідного коду системи;

– специфікація можливого варіанта системи.

Ця діаграма забезпечує узгоджений перехід від логічного до фізичного уявлення системи як програмних компонентів. Одні компоненти можуть бути тільки на етапі компіляції програмного коду, інші – на етапі його виконання. Основними елементами діаграми є компоненти, інтерфейси та залежності між ними. Крім того, на ній можуть відображатися ключові класи, що входять до компонентів.

Компонент (англ. component) – це фізична частина системи. Компоненти, що являють собою файли з вихідним кодом класів, бібліотеки, модулі тощо, які повинні володіти узгодженим набором інтерфейсів. Для їхнього графічного уявлення використовуються стандартизовані графічні символи.

Діаграму компонентів додатку представлено на рисунку (2.13).

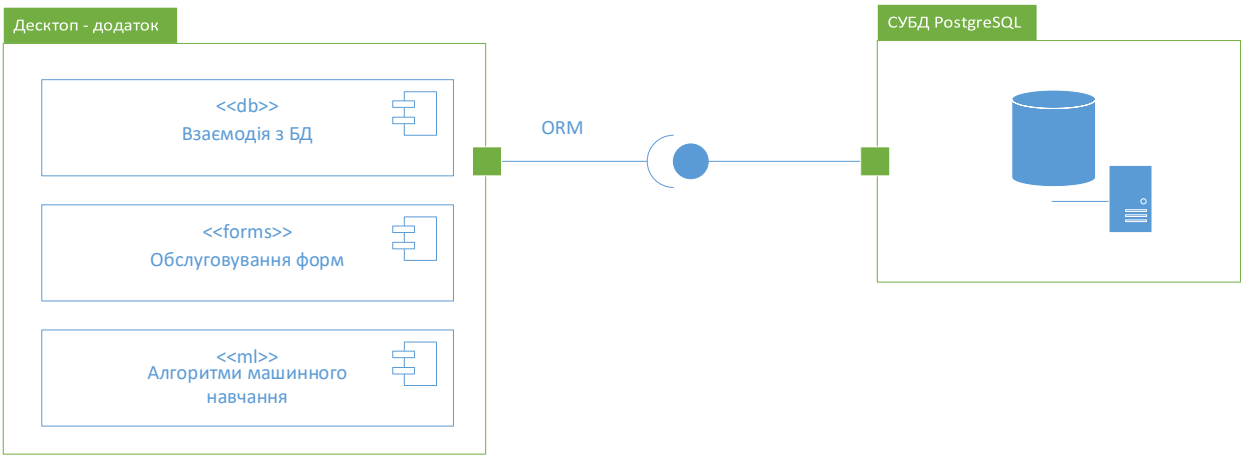

Рисунок 2.13 – Діаграма компонентів

Система складається з двох компонентів: десктоп – додатку та системи управління базою даних. СУБД є незалежним компонентом, база даних може бути розташована на локальному комп’ютері користувача, в локальній мережі або на хмарному сервісі. Від цього не залежить взаємодія десктоп – додатку з базою даних. Інтерфейсом взаємодії між компонентами є ORM, що надає більш високий рівень абстракції у порівнянні з SQL–запитами.

ORM (Object-Relational Mapping) – техніка програмування, яка дозволяє керувати базами даних з реляційними таблицями як об'єктами програмного коду. ORM автоматично конвертує (мапує) дані між реляційними базами даних і об'єктно–орієнтованими мовами програмування. До переваг ORM відносять:

– продуктивність розробки: завдяки використанню об'єктно-

орієнтованого синтаксису розробники можуть швидше писати код, оскільки ORM фреймворк бере на себе багато рутинних завдань;

– абстракція бази даних: ORM дозволяє розробникам зосередитися на роботі з об'єктами у їх рідній мові програмування, не переймаючись особливостями SQL. Це також призводить до зменшення кількості помилок;

– масштабування: ORM дозволяє легше управляти змінами у базі даних, адже зміни в структурі об'єктів автоматично відображаються на структуру бази даних;

– незалежність від конкретної СУБД: багато ORM фреймворків підтримують різні СУБД, що робить можливим зміну СУБД без необхідності переписування коду.

2.6 Діаграма розгортання системи

Діаграми розгортання використовуються для візуалізації апаратних процесорів/вузлів/пристроїв системи, каналів зв'язку між ними та розміщення програмних файлів на цьому апаратному забезпеченні.

Діаграма розгортання – це тип UML-діаграми, яка показує архітектуру виконання системи, включаючи такі вузли, як апаратні або програмні середовища виконання, а також проміжне програмне забезпечення, яке їх з'єднує.

Діаграми розгортання зазвичай використовуються для візуалізації фізичного апаратного та програмного забезпечення системи. Використовуючи його, можна зрозуміти, як система буде фізично розгорнута на апаратному забезпеченні.

Діаграми розгортання допомагають моделювати апаратну топологію системи порівняно з іншими типами UML-діаграм, які здебільшого описують логічні компоненти системи.

Створений десктоп – додаток розгортається на локальному комп’ютері

користувача. З оглядом на необхідність встановлення також бібліотек підтримки ORM для роботи з PostgreSQL доцільно використовувати для розгортання додатку на локальному комп’ютері користувача пакет інсталяції програмного забезпечення.

Діаграму розгортання системи, що будується, показано на рисунку 2.14.

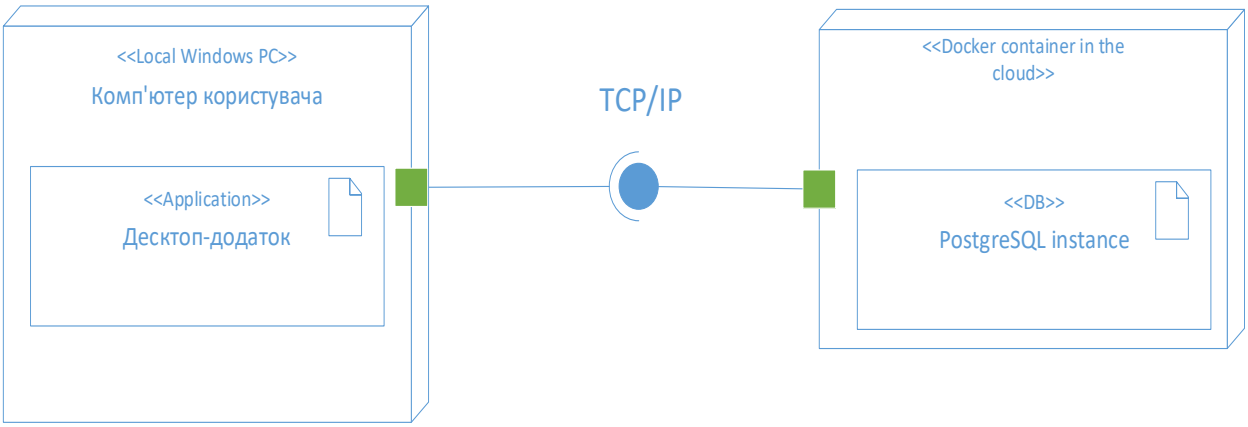

Рисунок 2.14 – Діаграма розгортання створеної системи

Додаток, розташований на локальному комп'ютері користувача, взаємодіє з СУБД PostgreSQL на віддаленому VPS сервері за протоколом PostgreSQL Frontend/Backend Protocol. Це спеціалізований прикладний протокол, який використовується PostgreSQL для комунікації між клієнтом і сервером. Комунікація здійснюється по TCP/IP мережі.

2.7 Висновки до розділу 2

На початку розділу викладено математичний базис задачі оптимізації комплектування. Для розв’язання використовується відома задача комбінаторної оптимізації, а саме – задача комівояжера. До задачі в XX столітті були запропоновані численні розв’язки. Для практичної реалізації з різноманіття розв’язків обрано алгоритм Кристофідеса, що за прийнятний час

генерує маршрут, довжина якого перевищує оптимальний не більше ніж на 50%.

В послідуючих підрозділах виконане моделювання та конструювання програмного забезпечення за допомогою UML 2.0. Комплексне проектування системи охопило всі етапи розробки. До цього процесу слід також включити й формулювання функціональних та нефункціональних вимог до програмного забезпечення, що було зроблено в підрозділі постановки задачі 4розділу 1. Використання UML 2.0 як основного інструментарію для проектування дозволило детально пропрацювати архітектуру системи, забезпечило глибоке розуміння бізнес–процесів і взаємодій усередині програмного продукту.

У результаті, було створено цілісне бачення проекту, що включає:

– діаграми варіантів використання, що ідентифікують основні функції системи та їх користувачів;

– діаграми діяльності, що визначили послідовності та розгалуження виконання алгоритмів основних сценаріїв використання системи. Частина алгоритмів рутинних операцій залишилися за межами розгляду, що дозволило сфокусуватись на особливостях системи, а не стандартних операціях;

– діаграма сутностей дозволила точно визначити структуру даних і виконати нормалізацію бази даних, яка допомагає підвищити їх цілісність.

– діаграми компонентів та розгортання відобразили фізичну структуру системи, включаючи апаратне та програмне забезпечення, що є критично важливим для розуміння того, як система буде реалізована та впроваджена у робоче середовище.

Розробка, заснована на UML, сприяє зниженню кількості помилок у процесі кодування та спрощує фазу тестування, оскільки кожен аспект системи чітко описаний та визначений до початку написання коду.

Імплементація програмного забезпечення на основі створених мовою UML моделей дозволяє досягти високого рівня відповідності між вимогами до програмного забезпечення та функціональними можливостями продукту, а також дозволяє легко контролювати та демонструвати цю відповідність.

Таким чином, використання UML 2.0 у процесі проектування програмного забезпечення має доведену ефективність. Основні переваги – зрозумілість та прозорість процесів розробки, здатність до швидкої адаптації під зміни вимог.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір середовища, мови та інших інструментів розробки

Python є однією з найпопулярніших високорівневих мов програмування, яка відома своєю простотою, читабельністю та широкою підтримкою спільноти [22–28]. Вона містить велику кількість бібліотек та фреймворків, що дозволяє розробникам швидко створювати ефективні рішення для різноманітних завдань, включаючи управління складами.

PyQT– біндинг для бібліотеки Qt, яка дозволяє створювати графічний інтерфейс користувача (GUI) для додатків на Python. Основною мовою Qt є C++ [27, 29–31].

PostgreSQL – це потужна, відкрита і багатофункціональна реляційна база даних, яка підтримує як SQL, так і JSON для нереляційних запитів [32–34].

На користь використання означеного стеку інструментів говорить наступне [30, 35].

Python:

– Python має чистий і виразний синтаксис, що робить процес написання коду швидким і ефективним. Це дуже важливо при створенні складних систем;

– Python пропонує великий вибір готових модулів і бібліотек, що покривають усі необхідні потреби при створені сконструйованого додатку, включаючи роботу з графами та таблицями;

– завдяки високому рівню абстракції Python дозволяє швидко створювати прототипи, що є ключовим для динамічного процесу розробки;

– Python є кросплатформенною мовою, що дозволяє запускати додатки на різних операційних системах без значних змін коду;

– Python має велику спільноту розробників та достатню кількість онлайн документації, що дозволяє в обмежені терміни вирішувати складні питання, виникаючи під час розробки.

PyQT [30–31]:

– PyQT пропонує велику колекцію елементів управління та інструментів для створення професійного та сучасного інтерфейсу користувача;

– Qt Designer – потужний інструмент для візуального проектування інтерфейсів по типу WYSIWYG, що значно спрощує процес їх розробки. На основі згенерованих форм відбувається автоматична генерація Python – кода для їх створення у додатку;

– PyQT дозволяє легко інтегрувати нові компоненти і розширювати існуючі, що надзвичайно важливо для розробки адаптивних систем. Це включає також зміну візуалізації та поведінки елементів графічного інтерфейсу;

– PyQT дозволяє запускати додатки на основних операційних системах, таких як Windows, macOS і Linux.

PostgreSQL [33, 36]:

– PostgreSQL відома своєю високою надійністю, підтримкою транзакцій, відкатом і журналізацією, що забезпечує безпеку даних;

– PostgreSQL ефективно обробляє великі обсяги даних, що є важливим для додатків управління складом, де може бути велика кількість транзакцій;

– підтримка складних SQL запитів та об'єднання даних дозволяє створювати гнучкі та потужні запити для аналізу та управління запасами;

– PostgreSQL має широку підтримку розширень, що збільшують можливості СУБД.

З урахуванням вищевказаних аргументів, вибір Python, PyQT та PostgreSQL для розробки десктопного додатку управління складом є обґрунтованим та цілком виправданим. Це комбінація технологій забезпечує гнучкість, швидкість розробки, високу продуктивність та здатність до масштабування.

3.2 Реалізація проекту системи

Відповідно до обраних інструментів реалізацію системи виконано мовою Python. Відповідальність компонентів системи, у відповідності до діаграми компонентів 2.13, розподілена на взаємодію з базою даних, що використовує ORM SQLAlchemy [36], обслуговування форм графічного інтерфейсу користувача та модуль, що відповідає за імплементацію розв’язку комбінаторної задачі комівояжера.

Загальну структуру додатку наведено на рисунку 3.1.

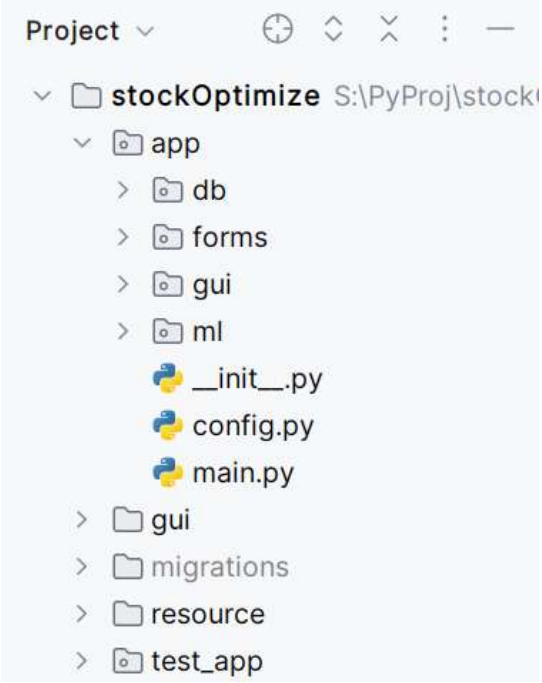

Рисунок 3.1 – Загальна структура додатка

Необхідні налаштування додатку згруповані у файлі конфігурації config.cfg, обгорткою якого є config.py. Точкою входу в додаток є manage.py, що виконує підключення до бази даних та створює екземпляр класу QApplication.

Структура пакету db показана на рисунку 3.2.

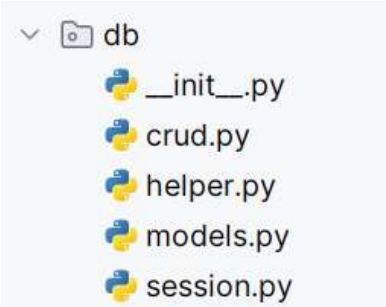

Рисунок 3.2 – Структура модуля db

В модулі crud містяться методи, що відповідають за усі необхідні CRUD [36] операції з базою даних: додавання, читання, оновлення та видалення записів. В модулі також реалізовано заповнення бази даних тестовими даними, що зручно при тестуванні системи до релізу. В модулі models містять усі моделі SQLAlchemy. Приклад моделі, що відповідає сутності SKU, наведений на рисунку 3.3.

```
class SKU(StockBase):
    __tablename__ = 'sku'

    id: Mapped[int] = mapped_column(primary_key=True)
    name: Mapped[str] = mapped_column(String(120), unique=True, nullable=False)
    properties: Mapped[List["PropertyValue"]] = relationship()
    features: Mapped[List["FeatureValue"]] = relationship()

    def __repr__(self):
        return f'SKU [{self.name}]'

    def __str__(self):
        return f'SKU [{self.name}]'
```

Рисунок 3.3 – Визначення сутності SKU

В класі визначаються метадані – назва таблиці, що використовується при створенні таблиць з використанням engine SQLAlchemy, а також при виконанні міграцій за допомогою alembic. Кожне поле таблиці бази даних описується в декларативному стилі SQLAlchemy з використанням типу

Mapped. Relationship() вказує на зв’язок з іншими таблицями. У випадку, що розглядається, це зв’язок один-до-багатьох з властивостями та ознаками: кожен екземпляр SKU може мати необмежену кількість властивостей та ознак.

Для зручності розробки клас також має текстові представлення __str__ та __repr__. Перше використовується, наприклад, при виконанні друку у консоль, а друге – при перегляді значення у відладнику.

Модуль session відповідає за створення engine (підключення до бази даних) та створення екземплярів сесій підключення до db – використовується генератор (Generator).

Типовий метод модулю crud представлено на рисунку 3.4.

```
def get_sku_by_id(self, sku_id):
    stmt = select(SKU).where(SKU.id == sku_id)
    return self._db.scalar(stmt)
```

Рисунок 3.4 – Типовий CRUD-метод

Метод складається з формування запиту в форматі SQLAlchemy та повернення результатів виконання цього запиту в скалярній формі. В наведеному прикладі результат запиту заздалегідь містить не більше одного екземпляру сутності, тому використовується єдине число скаляру.

Відображення графічного інтерфейсу користувача реалізовано двома пакетами – forms та gui. Пакет gui містить автоматично згенеровані модулі форм, що були створені у PyQT Designer. Приклад редагування форми наведено на рисунку 3.5.

Designer дозволяє створення форм у стилі WYSIWYG та налаштування їх поведінки у випадку зміни розміру вікна. Для деяких форм розмір фіксований, для інших – задано мінімальний з можливістю максимізації до розмірів батьківського вікна.

В пакеті forms згруповано класи, успадковані від QDialog та QWidget.

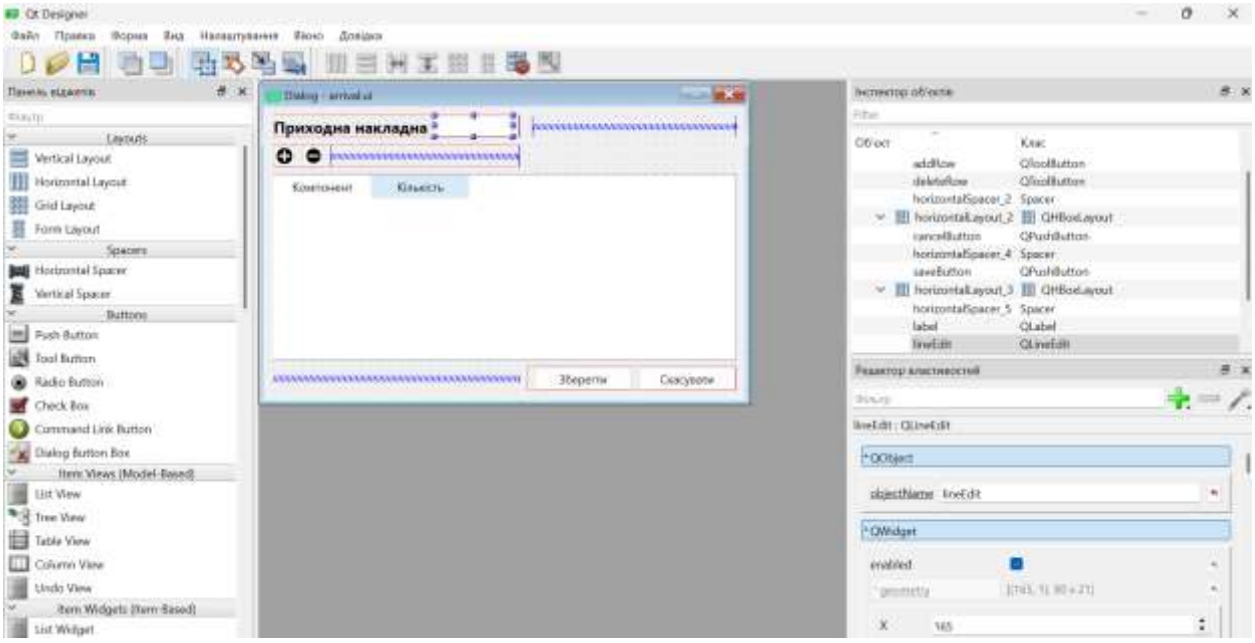

Рисунок 3.5 – Редагування форми у Designer

‘QDialog’ є спеціалізованим підкласом ‘QWidget’, призначеним для створення модальних та немодальних діалогових вікон у графічному інтерфейсі користувача. Він зазвичай використовується для отримання від користувача інформації, відповіді або вибору. З іншого боку, ‘QWidget’ служить як базовий клас для всіх об'єктів графічного інтерфейсу в Qt, надаючи функціональність для створення та управління вікнами, а також взаємодії з користувачем. ‘QWidget’ виступає як контейнер для інших віджетів, формуючи складні інтерфейси користувача.

Кожен з модулів в пакеті відповідає за логіку роботи окремої форми: містить обробники подій натискання кнопок, додавання та редагування рядків таблиць, початку та закінчення редагування окремих полів, тощо.

Для створення нових екземплярів вікон використовується механіка модальних вікон, що імплементована класом QMdiSubWindow. Це дозволяє додавати необмежену кількість модальних вікон в область основного вікна додатку, а користувачеві – переключатись між вікнами на кшталт того, як це відбувається в звичних MS Word або Excel. Це дозволяє користуватись

інформацією у декількох вікнах одночасно без необхідності закриття одного вікна перед відкриттям іншого. Приклад відкриття декількох вікон й діалогу поверх них наведено на рисунку 3.6.

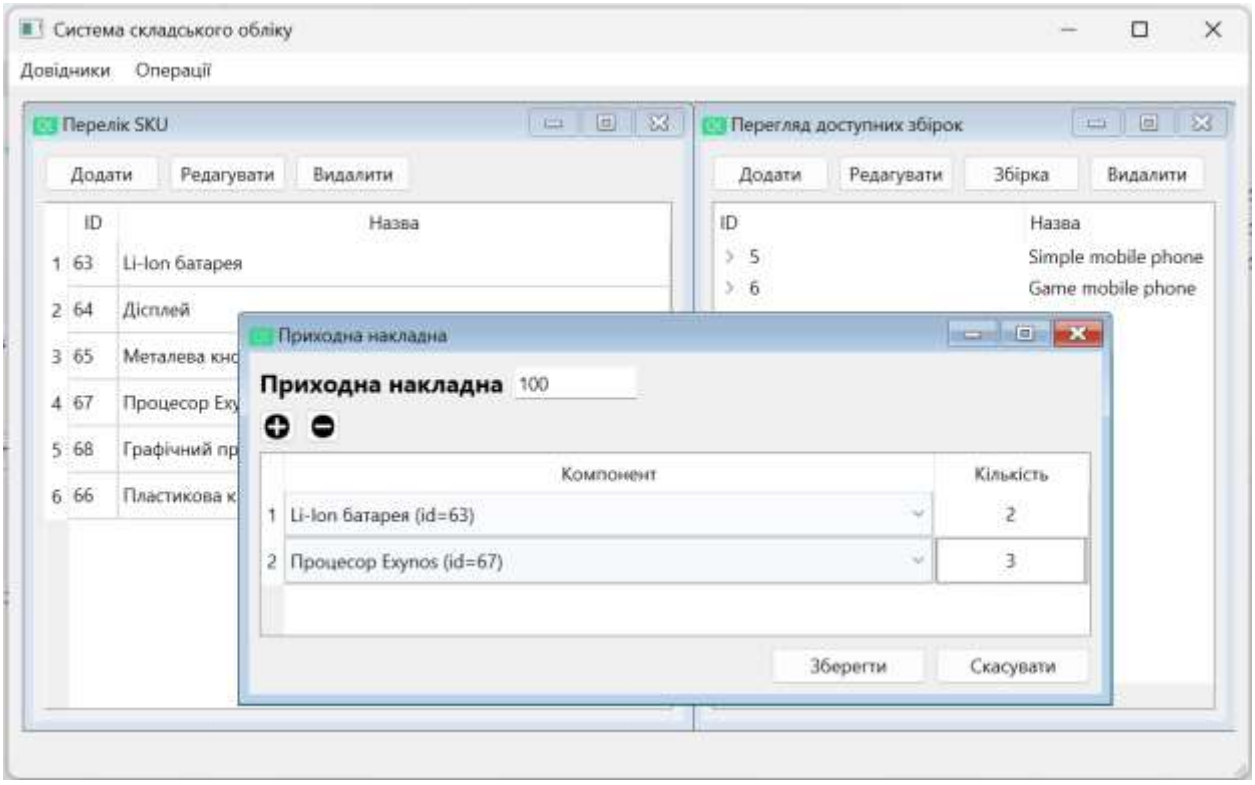

Рисунок 3.6 – Модальні та діалогові вікна створеного додатку

Реалізований алгоритм комбінаторної оптимізації – алгоритм Кристофідеса, адаптер результатів роботи алгоритму та моделі даних, що використовуються адаптером знаходяться в пакеті ml. Призначення адаптеру – забезпечення сумісності формату даних, що використовує алгоритм та типів даних, що використовуються в інших модулях системи.

Моделі включають в себе:

– Route – маршрут збирання;

– AssemblyData – дані збирання відповідно до певної технологічної карти;

– Stock – модель складу, що створює екземпляр складу на базі даних,

завантажених з JSON [37] файлу, та дозволяє обчислювати відстані між комірками.

Модуль assembly_analyzer складається з класу адаптеру AssemblyAdapter, призначення якого – побудова інтерфейсу між різними типами даних, що обробляються. В додатку використовується модель складу з комірками, між якими є відстані, що можна розрахувати, а реалізація алгоритму Кристофідеса використовує двовимірну матрицю відстаней між точками. Для формування матриці та інтерпретації результату роботи комбінаторного алгоритму в термінах моделі складу використовується створений адаптер.

Клас, що управляє пошуком оптимального маршруту – AssemblyAnalyzer. При створенні класу використовується патерн «медіатор»: методи класу є статичними, сам клас немає стану (state), уся необхідна для опрацювання інформаціє подається в екземплярі датакласу AssemblyData, що подається на вхід методу AssermblyAnalyzer. define_best_stock_route(). Другим вхідним параметром вказаного методу є екземпляр датакласу Stock, що описує конфігурацію поточного складу.

Безпосередньо реалізація алгоритму Кристофідеса міститься у модулі christ_algorithm.py. Реалізація алгоритму повністю відповідає наведеному у розділі 2.1. Для операцій з графами такими, як виділення остовного дерева, створення дводольного графу, використовується спеціалізована бібліотека NetworkX [38–39], що призначена для операцій з графами та спрощує виконання рутинних операцій.

Як згадувалось раніше, на вхід подається матриця відстаней між точками, що понумеровані від 0 до n-1 (n – кількість точок). Вихідними даними є послідовність порядкових номерів точок, наприклад, [1, 3, 8, 7, 2, 4, 5, 6, 1]. Тому для поєднання з даними у форматі інших частин системи використовується адаптер [26].

3.3 Тестування та оцінка створеного програмного забезпечення

Процес тестування можна поділити на кілька ключових етапів: модульне тестування, функціональне тестування, та перевірка на відповідність функціональним та нефункціональним вимогам.

На етапі модульного тестування (unittest) основна увага приділялася індивідуальним компонентам програми. Мета цього етапу – забезпечити, щоб кожен ізольований модуль програми працює належним чином. Використання Python та його бібліотеки unittest дозволило створити автоматизовані тести для перевірки кожної функції та методу в програмі.

Для модулів, що працюють з базою даних (SQLAlchemy+PostgresSQL), були розроблені тести, які перевіряли коректність запитів, інтеграцію з базою даних, а також правильність обробки даних. Це дозволило виявити та виправити помилки, пов'язані з базою даних, на ранній стадії розробки. Для методів, що спрямовані на обробку результатів запитів, створювались макети результатів запитів (mock-ups), що дозволило проводити тестування без підключення до бази даних. Тестування методів, що потребують підключення до бази даних, було винесено в окрему групу інтеграційних тестів. Загальне покриття коду тестами склало більше 80% за розрахунками відповідних метрик в інтегрованому середовищі розробки PyCharm.

Модульні тести включають, в тому числі, тестування коректної роботи алгоритму й якості маршрутів, що генеруються. Оскільки алгоритм евристичний й не гарантує ідемподентної відповіді, проводилось порівняння згенерованої довжини маршруту та довжини оптимального маршруту, що створений шляхом повного перебору комбінацій.

Після модульного тестування наступним етапом було функціональне тестування. Цей етап зосереджений на перевірці, як додаток функціонує в цілому. Тут перевірялися всі основні функції додатку: від введення даних до генерації оптимізованого маршруту збирання.

Останнім етапом тестування була перевірка додатку на відповідність

функціональним та нефункціональним вимогам, що сформульовані в розділі 1.5. Фактично це допомагає перевірити відповідність програмного забезпечення, що сконструйовано та реалізовано, початковим вимогам до нього, що деталізують мету створення програмного забезпечення. Результати перевірки відповідності функціональним та нефункціональним вимогам наведено в таблиці 3.1.

Таблиця 3.1 – Відповідність функціональним та нефункціональним вимогам

№	Вимога	Результат перевірки
Функціональні вимоги		
1	Ведення складського обліку в розрізі SKU (Stock Keeping Unit)	Облік ведеться в розрізі SKU.
2	Незалежне ведення обліку місць розташування товару, можливість розташування однієї складської одиниці товару (SKU) в декількох місцях розташування	Ведення місць розташування комплектуючих (компонентів) ведеться незалежно від обліку SKU. Можливе розташування одного компоненту в декількох місцях. Таке розташування коректно опрацьовується в усіх сценаріях використання.
3	Зберігання інформації про додаткові властивості комплектуючих, що визначаються відповідними довідниками	Реалізовано зберігання ознак та властивостей SKU. Найменування ознак й властивостей довільні, зберігаються в окремих довідниках.
4	Забезпечити додавання, редагування, видалення SKU, їх властивостей, місць зберігання,	Всі необхідні CRUD операції реалізовані у графічному інтерфейсі.

Продовження таблиці 3.1

№	Вимога	Результат перевірки
	технологічних карт та іншої довідкової інформації в графічному інтерфейсі користувача	
5	Забезпечити автоматичне генерування складального списку на підставі технологічної карти з врахуванням місць розташування компонентів (SKU), можливої відсутності або недостатньої кількості компонентів	Автоматична генерація складальної карти з використанням комбінаторного алгоритму реалізована. Проводиться перевірка достатньої кількості компонентів.
6	Забезпечити можливості налаштування та зміни параметрів складу	Параметри складу налаштовуються в окремому JSON файлі.
7	Відтворювати технологічні карти у вигляді ієрархічного списку	Технологічні карти відтворюються у вигляді ієрархічного списку.
8	Відтворювати інформацію про складські залишки	Інформація про складські залишки відтворюється у вигляді списку – у переліку SKU
Нефункціональні вимоги		
1	В якості інтерфейсу додатку використовувати десктоп – додаток з графічним інтерфейсом	Додаток має графічний інтерфейс.

Продовження таблиці 3.1

№	Вимога	Результат перевірки
2	Забезпечити зберігання даних облікової системи незалежно від самої системи та можливість віддаленого зберігання даних на сервері, в хмарі, тощо	Дані зберігаються окремо, на сервері PostgresSQL. Він може бути розгорнутий локально, в контейнерів, в локальній мережі, в хмарі.
3	При створенні архітектури програмного забезпечення передбачити можливість використання оптимізаційного алгоритму незалежно від графічного інтерфейсу програми, що в подальшому дозволить створити інші інтерфейси користувача системи – мобільний та web	Оптимізаційний алгоритм реалізовано окремим модулем. Модуль може буте використаний в інших реалізаціях додатку: мобільний додаток, web-додаток, тощо.

На підставі таблиці 3.1 можна зробити висновок, що функціональні та нефункціональні вимого до додатку виконані при проектуванні та розробці додатку в повному обсязі.

3.4 Опис контрольного прикладу

В рамках контрольного прикладу проведемо наступні операції в обліковій системі.

- Додавання SKU разом із додавання властивостей та ознак;
- Приход компонентів на склад;
- Розміщення комплектуючих в комірках зберігання;

– Складання технологічної карти;
– Генерація оптимального маршруту збирання компонентів;
– Вибуття компонентів зі складу.

Оскільки система оперує певною кількістю SKU, приклад розпочинається із додавання SKU до облікової системи. Перелік SKU показано на рисунку 3.7.

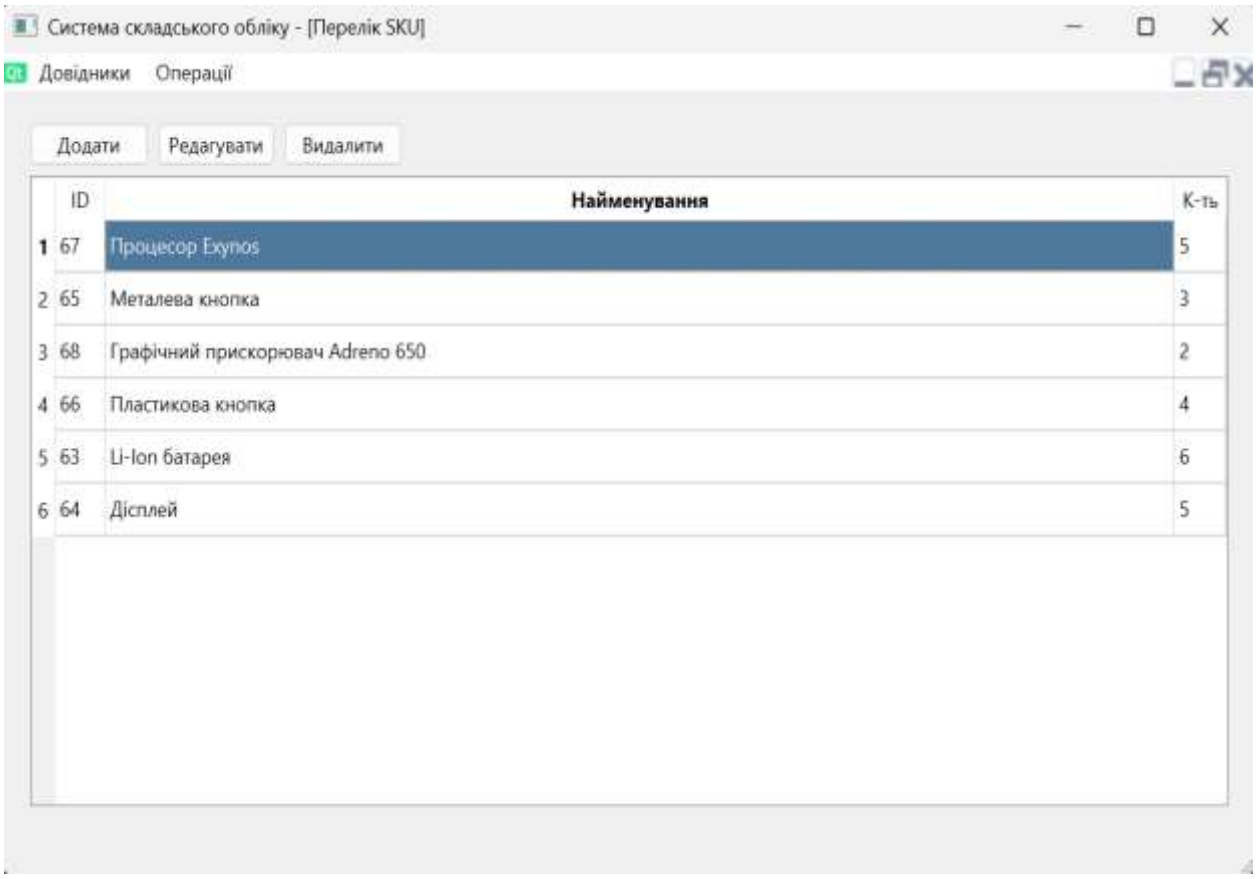

Рисунок 3.7 – Перелік SKU

В діалозі редагування SKU доступні дві вкладки для редагування властивостей та ознак. Обидві вкладки показані на рисунку 3.8.

У прикладі додано пластикову кнопку, що має одну ознаку – вагу та дві властивості – колір та матеріал.

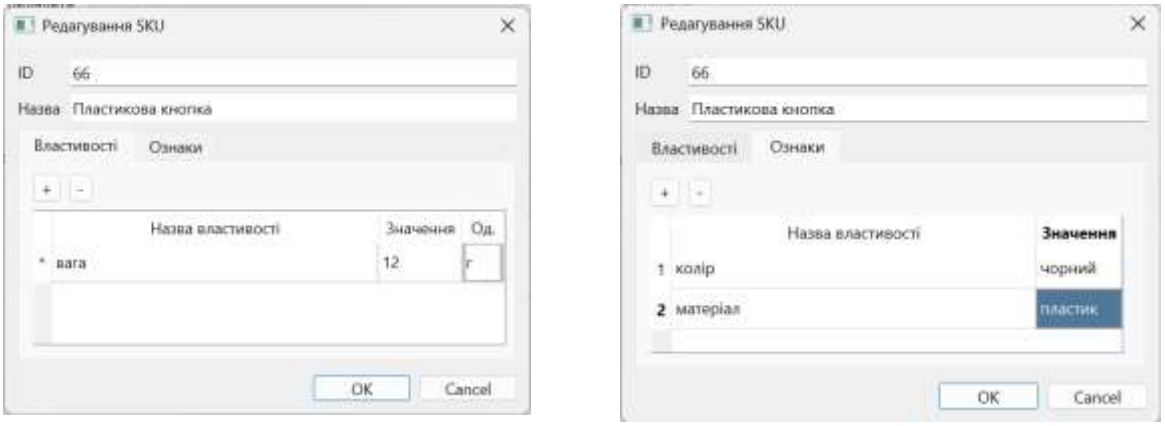

Рисунок 3.8 – Редагування SKU

Додавання та видалення комірок відбувається в таблиці. Якщо комірці при редагуванні не надано індивідуальне найменування, надається найменування формату «Комірка (x;y;z)». Таблицю комірок наведено на рисунку 3.9.

Додати Видалити

	ID	Полиця	Стовпчик	Стелаж	Назва
1	81	1	1	1	Комірка (1;1;1)
2	82	1	1	2	Комірка (1;1;2)
3	83	1	2	1	Комірка (1;2;1)
4	84	1	2	2	Комірка (1;2;2)
5	85	1	3	1	Комірка (1;3;1)
6	86	1	3	2	Комірка (1;3;2)
7	87	2	1	1	Комірка (2;1;1)
8	88	2	1	2	Комірка (2;1;2)
9	89	2	2	1	Комірка (2;2;1)
10	90	2	2	2	Комірка (2;2;2)
11	91	2	3	1	Комірка (2;3;1)

Рисунок 3.9 – Таблиця комірок

Ще однією таблицею є таблиця розміщення комплектуючих. Розміщення комплектуючих відбувається автоматично при проведенні

документу «Приходна накладна». При вибутті комплектуючих (документу «Вибуття комплектуючих» кількість комплектуючих у відповідних комірках зменшується. Вид таблиці розміщення наведений на рисунку 3.10. Окрім розміщення та вибуття комплектуючих відповідними документами доступний режим ручного редагування, додавання та видалення.

Рисунок 3.10 – Розміщення комплектуючих

Документ приходу. Він складається з заголовку з номером документу та табличної частини, в якій вказуються товари, що мають бути додані, комірки розміщення та кількість, що розміщується. Якщо один товар повинен бути доданий до двох різних комірок, в документ слід додати два рядки з одним товаром, але різними комірками призначення. При натисканні на кнопку «Зберегти» відбувається запис до інформаційної бази про зміну кількості комплектуючих. Відповідно інформація відображається й для кількості у відповідній комірці.

Для перевірки коректності запису у базу даних можна використовувати переглядач таблиць, що вбудований в PyCharm Professional, або пропрієтарне

програмне забезпечення PostgreSQL – pgAdmin. Документ приходу подано на рисунку 3.11. Перегляд вмісту таблиці з використанням pgAdmin показано на рисунку 3.12.

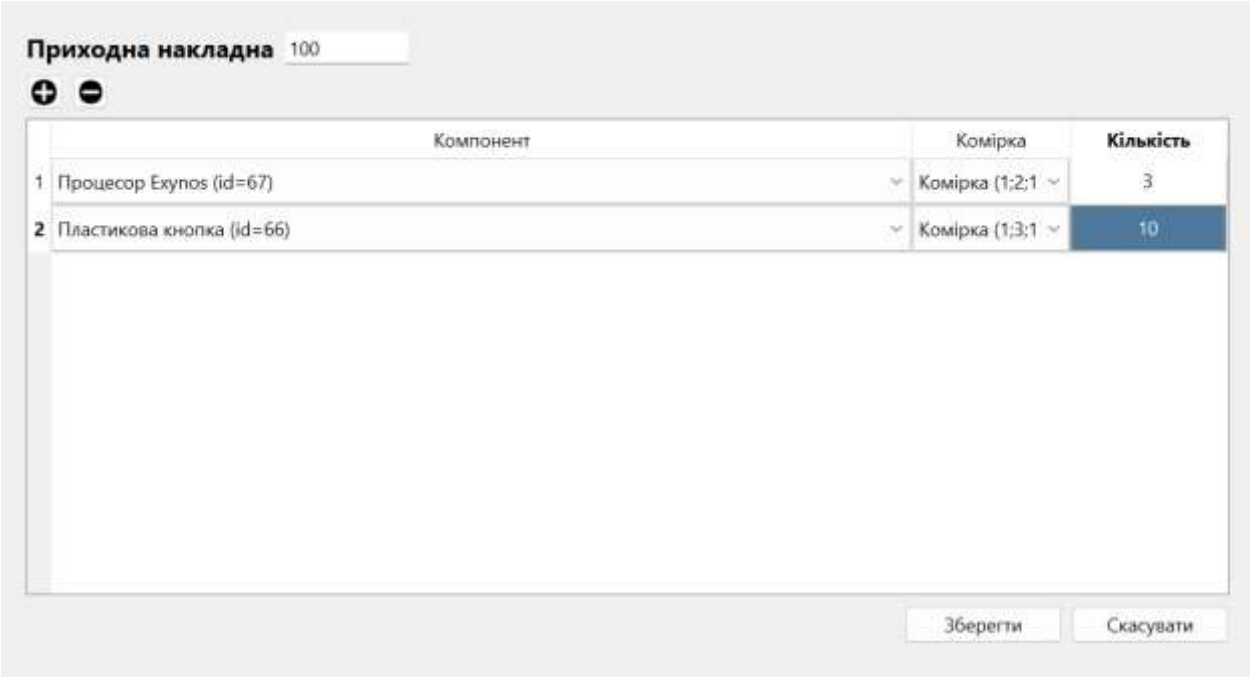

Рисунок 3.11 – Приходна накладна

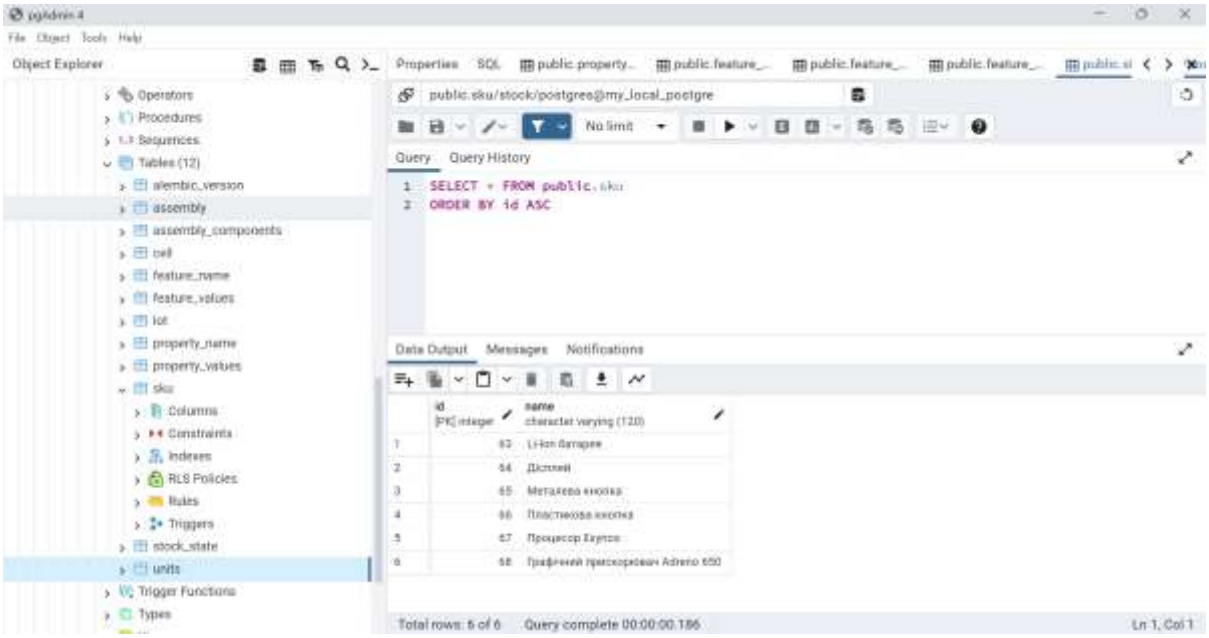

Рисунок 3.12 – Перегляд даних в pgAdmin

Редагування технологічних карт, на базі яких формуються маршрути збирання, відбувається в табличному ієрархічному списку, як показано на рисунку 3.13. Верхній рівень ієрархії – продукти, що збираються з компонентів, розташованих на другому рівні ієрархії. Як притаманно іншим ієрархічним спискам, користувач може відкривати або закривати певні набори елементів другого рівня, що відповідають одному елементу першого рівня.

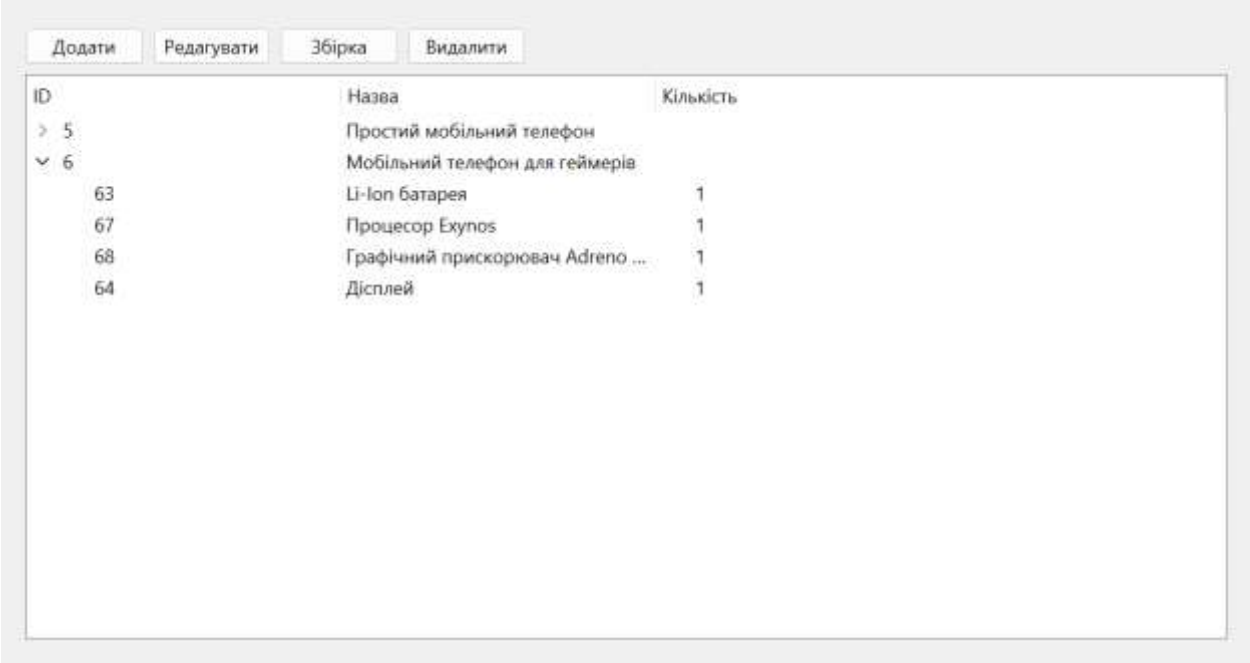

Рисунок 3.13 – Редагування технологічних карт

При натисканні на рядок ієрархії першого рівня відкривається вікно редагування технологічної карти, а при натисканні на рядок другого рівня – вікно редагування SKU. За допомогою кнопок, розміщених над таблицею, можна додавати, видаляти окремі технологічні карти та генерувати маршрути збирання – кнопка «Збірка». Вікно редагування технологічної карти показано на рисунку 3.14. Технологічна карта складається з назви продукту, що збирається та табличного переліку компонентів з вказаними кількостями, що необхідні для збирання одного екземпляру продукту.

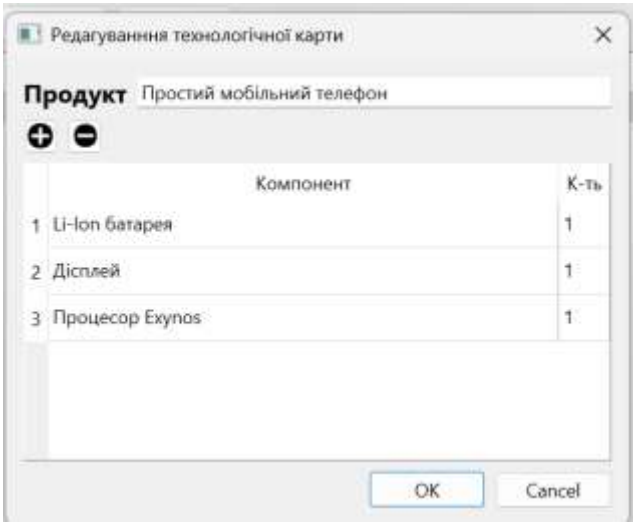

Рисунок 3.14 – Редагування технологічної карти

Натискання кнопки «Збірка» активує генерацію маршрути збирання з використання комбінаторного алгоритму. Результат генерація маршруту представляється у текстовому вигляді. Маршрут представляє собою перелік пар (компонент; комірка розташування компоненту). Починається та закінчується маршрут – точкою входу та точкою виходу. Приклад виводу маршруту показаний на рисунку 3.15.

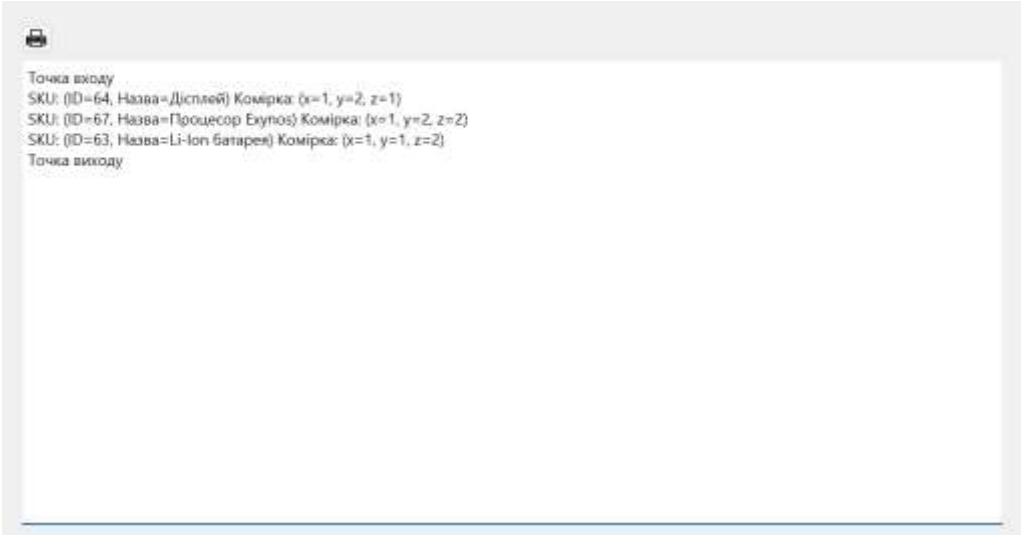

Рисунок 3.15 – Приклад генерації маршруту

На базі маршруту збирання може бути сформована видаткова накладна, приклад якої показаний на рисунку 3.16. Значення комірок та компоненти підставляються з карти збирання компонентів.

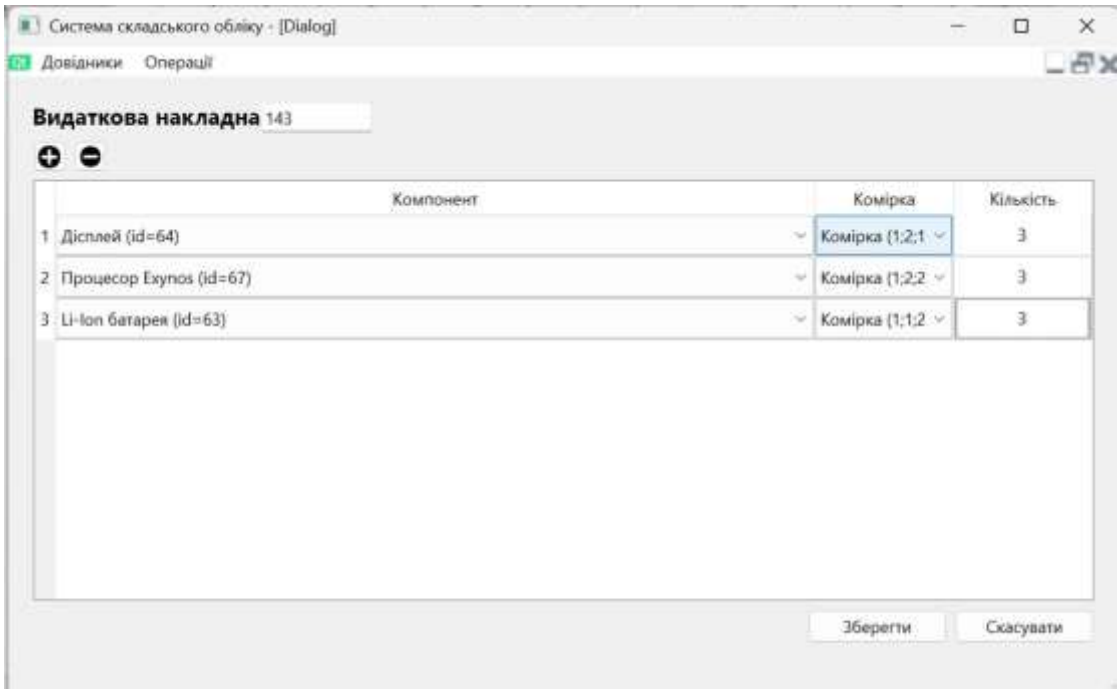

Рисунок 3.16 – Видаткова накладна

Таким чином, в рамках контрольного прикладу коректно виконані усі операції, що мають виконуватись обліковою системою згідно розділу 2, в тому числі функціональним вимогам та сценаріям використання. Отже, тестування системи можна вважати завершеним в повному обсязі.

3.5 Висновки до розділу 3

У розділі описано розробку додатку для складського обліку, що інтегрує оптимізацію комплектації. Розробка математичної бази, моделювання та проектування додатку було виконано в попередньому розділі, розробка ґрунтується саме на ньому.

Вибір Python як основної програмної мови зумовлений його високою

продуктивністю та гнучкістю, що дозволяє ефективно вирішувати задачі обробки даних та втілення комбінаторних алгоритмів. PySide 6 забезпечувало високий рівень інтерактивності і адаптивності графічного інтерфейсу користувача, а PostgreSQL, вибрана в якості системи управління базами даних, гарантувала надійне зберігання та швидкий доступ до великих обсягів інформації.

Процес розробки сфокусований на створенні модульної архітектури, що дозволяє масштабувати додаток та впроваджувати нові функції. Основні модулі включають операції з базою даних, обслуговування графічного інтерфейсу та реалізацію обраного алгоритму комбінаторної оптимізації. Алгоритм оптимізації, що використовується, націлений на мінімізацію шляху, що долає комплектувальник або відповідний робот під час збирання товару. Економія шляху відображається у економії ресурсів підприємства, що є кінцевою метою.

У рамках тестування програмного забезпечення проводилась перевірка основних функцій, стабільність з’єднання та операцій з базою даних, виконано наскрізний кейс використання системи. Тестування проводилося в операційній системі Windows 11, базу даних було розгорнуто в docker – контейнері. Додаток демонструє стабільну роботу та відповідність поставленій у підрозділі 1.5 задачі, у тому числі функціональним та нефункціональним вимогам до додатку.

Аналіз контрольного прикладу відображав усі етапи – від приходу комплектуючих до вибуття в результаті збирання за згенерованим маршрутом. Увесь графічний інтерфейс додатку продемонстрований в рамках контрольного прикладу. Це дозволило оцінити зручність інтерфейсу, швидкість та коректність роботи вбудованого алгоритму комбінаторної оптимізації.

4 ОХОРОНА ПРАЦІ

Охорона праці є надзвичайно важливою аспектом будь-якого виробничого технологічного процесу, включаючи його складське приміщення. Складські приміщення мають свої особливості і вимоги, які повинні дотримуватися для забезпечення безпеки працівників та запобігання нещасним випадкам.

Однією з перших вимог до охорони праці на складському приміщенні є належне освітлення. Світло повинно бути достатньо яскравим і рівномірно розподіленим по всьому приміщенню, щоб уникнути тіней та забезпечити видимість працівникам. Крім того, важливо, щоб освітлення було захищене від пошкоджень та складалося з вогнестійких матеріалів.

Ще одним аспектом охорони праці є правильне організування простору в складському приміщенні. Стелажі та полиці повинні бути стійкими, надійно закріпленими та не завантаженими зайвою вагою. Також важливо, щоб прогін між стелажами був достатнім для безпечного руху працівників та вантажівок. Важливо, щоб продукція була зберігана відповідно до правил безпеки, зокрема, вбудовувати вибухонебезпечні чи легкозаймисті матеріали.

Однією з найбільш важливих аспектів охорони праці на складському приміщенні є правильне використання підйомно-транспортного обладнання. Процедури безпеки мають бути заздалегідь визначені, а працівники повинні бути навчені та сертифіковані щодо безпечного користування таким обладнанням. Також, слід встановити межі швидкості руху вантажних машин і смуги руху, щоб уникнути зіткнень та інших нещасних випадків.

Крім початкової інструктажу та навчання працівників, охорона праці на складському приміщенні вимагає постійного стеження та контролю. Регулярні перевірки технічного стану обладнання, оцінка ризиків та впровадження необхідних заходів для запобігання нещасним випадкам є ключовими елементами безпечної роботи на складі.

Найчастішими загрозами на складському приміщенні є падіння

предметів, посковзнення та спотикання, вибухи та пожежі, отруєння, травми, а також відпрацювання і перевитрата сил працівників.

Для запобігання падінню предметів необхідно використовувати належно закріплені стелажі, розміщувати важкі предмети на нижніх полицях, а легкі на верхніх рівнях. Також слід використовувати приховані системи кріплення, які не дозволяють предметам зламатися чи випасти при руху обладнання чи вантажуванні або розвантаженні.

Для запобігання посковзненню та спотиканню важливо забезпечити правильну гідроізоляцію підлоги, встановити антиповзаючі покриття та забезпечити відведення вологості. Також важливо регулярно очищувати підлогу від сміття та інших перешкод, які можуть призвести до нещасних випадків.

Для запобігання вибухам та пожежам необхідно дотримуватися правил зберігання та операцій з вибухонебезпечними матеріалами. Заборонять доступ до вогню та джерел іскри, а також використовувати вогнезахисне обладнання та системи пожежогасіння.

Для запобігання отруєнням слід правильно зберігати хімічні речовини та додержуватися всіх правил щодо їх утилізації. Працівники повинні бути навчені про заходи безпеки при роботі з отруйними речовинами та додатково забезпечені необхідними засобами індивідуального захисту.

Щодо травматизму, важливо передбачити ергономічне обладнання та меблі, які зменшують фізичне навантаження на співробітників. Також слід наймати достатню кількість працівників на складі, щоб не відпрацьовувати та не перевитрачувати силу працівників.

Контроль за відведенням вологості є також важливим аспектом безпеки. Перевищена вологість може призвести до утворення збитків на виробах, пошкоджень обладнання та спричинити небезпечні умови праці. Використання вентиляційних систем, дезінфекція та контроль вологості є ефективними способами регулювання вологості у приміщенні.

Також слід мати план евакуації та знати вимоги щодо поведінки у разі

надзвичайних ситуацій, таких як пожежа або вибух. Прописання рятувальних евакуаційних шляхів, розміщення вогнегасників та навчання працівників дієвим заходам допоможуть забезпечити безпеку у випадку нещасних випадків.

Необхідно також підтримувати відповідні навички працівників шляхом проведення регулярних навчань і тренінгів з питань безпеки праці, а також публікувати інформацію та інструкції щодо безпеки на видному місці

Враховуючи всі ці аспекти охорони праці, складське приміщення виробничого технологічного процесу буде безпечним і забезпечить оптимальні умови для працівників. Важливо ніколи не забувати про виявлення потенційних загроз, постійне оновлення процедур безпеки та постійне навчання працівників.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи отримано рішення завдання підвищення ефективності системи комплектації виробничого технологічного процесу за рахунок розробки засобу оптимізації маршрутів комплектувальників.

У роботі створено десктопний додаток для управління складом, що є внеском у сферу автоматизації складської логістики. Створене програмне забезпечення не тільки виконує стандартні функції складського обліку, але й покращує складські процеси за рахунок оптимізації шляху комплектувальника.

Використання Python, PySide 6, та PostgreSQL для реалізації цього проекту дозволило забезпечити високу продуктивність, гнучкість та надійність системи. Через це, додаток може стати простим та ефективним інструментом у руках фахівців складської логістики.

Програмний продукт, розроблений в рамках роботи, не тільки відповідає сучасним вимогам до складського обліку, але й включає інноваційний модуль комбінаторної оптимізації. Цей модуль застосовує алгоритм Кристофідеса для оптимізації маршрутів збору комплектуючих, забезпечуючи значну економію ресурсів підприємства. Економія ресурсів відбувається через зниження витрат на комплектування відповідної партії компонентів.

Архітектура програмного забезпечення була ретельно спроектована з використанням UML діаграм, що забезпечило її модульність та масштабованість. Процес розробки був сфокусований на створенні гнучкої системи, яка могла би легко адаптуватися до змінних потреб складської логістики. Основні модулі додатку включали операції з базою даних, обслуговування графічного інтерфейсу та реалізацію алгоритму комбінаторної оптимізації.

Тестування та валідація програмного забезпечення виявили високий рівень його стабільності та надійності. Реалізовані функціональні та

нефункціональні вимоги до додатку були ретельно перевірені, що підтвердило його придатність для використання в реальних умовах складу. Контрольний приклад продемонстрував ефективність реалізованого алгоритму оптимізації.

Кваліфікаційна робота виявилася успішною у втіленні оригінального підходу до оптимізації процесів складської логістики. Розроблений додаток є не тільки технічно продуктивним, але й практично значущим для підприємств, що прагнуть до оптимізації своїх складських операцій. Він забезпечує інтегроване рішення для управління запасами, планування маршрутів збирання та виконання повсякденних складських задач. Таким чином, програмний продукт відіграє важливу роль у поліпшенні оперативного управління складськими процесами, підвищуючи їх ефективність та продуктивність.

Висновки цієї роботи підкреслюють важливість інтеграції сучасних технологій та алгоритмічних рішень у розробці програмного забезпечення для управління складами. Програмний продукт, розроблений в рамках цієї роботи, є яскравим прикладом того, як інноваційні підходи можуть трансформувати традиційні методи управління та сприяти підвищенню ефективності бізнес–процесів. Розширення функціональності додатку може збільшити його практичну корисність. Реалізація розширення не є складною задачею через правильно проектування та модульну архітектуру.

За темою кваліфікаційної роботи опубліковано тези в матеріалах Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених [7].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Завитій О., Дідоренко Т., Кондрюк Л. Логістичні витрати виробничих підприємств як об’єкти обліку, та контролю // Інститут бухгалтерського обліку, контроль та аналіз в умовах глобалізації. 2019. Вип. 1-2. С. 49–73.

2. Sabo-Zielonka A., Tarczyński G. Porównanie czasów kompletacji zamówień dla różnych sposobów wyznaczania trasy magazynierów na przykładzie dużego centrum logistycznego // Ekonometria. 2014. № 2 (44). P. 62–81.

3. Beskorovainyi V., Sudik A. Optimization of topological structures of centralized logistics networks in the process of reengineering // Innovative technologies and scientific solutions for industries, 2021. No. 1 (15). P. 23–31.

4. Безкоровайний В. В. , Нефьодов Л. І., Русскін В. М. Математична модель структурно-топологічної оптимізації логістичних мереж», Вісник Харківського національного автомобільно-дорожнього університету, 2021. Вип. 95. С. 178–184.

5. Beskorovainyi V., Kuropatenko O., Gobov D. Optimization of transportation routes in a closed logistics system // Innovative Technologies and Scientific Solutions for Industries. 2019. No. 4 (10). P. 24–32.

6. Кучерук О. Я., Драч І. В. Оптимізаційний підхід в задачі маршрутизації комплектувальника // Вісник Черкаського державного технологічного університету, 2021. №3. С. 59–68.

7. Закладний В. І., Безкоровайний В. В. Оптимізація маршрутів у системі комплектації виробничого технологічного процесу // Матеріали всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених / Харківський національний автомобільно-дорожній університет; Харків: ХНАДУ, 2023. С. 151–154. URL: https://dspace.khadi.kharkov.ua/handle/123456789/17839 (дата звернення: 29.12.2023).

8. Методичні вказівки з підготовки та захисту кваліфікаційної роботи

здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2021. 55 с.

9. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП "УкрНДНЦ". 2016. 30 с.

10. Gwynne R. Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse. Kogan Page, 2011. 344 p.

11. Paul A. Myerson, Supply Chain and Logistics Management Made Easy: Methods and Applications for Planning, Operations, Integration, Control and Improvement, and Network Design, Pearson FT Press, 2015. 352 p.

12. Zhong-huan Wu, Hong-jie Chen, Jia-jia Yang, Optimization of Order–Picking Problems by Intelligent Optimization Algorithm, Mathematical Problems in Engineering, 2020.

13. Veronika Lesch, Patrick B.M. Müller, Moritz Krämer, Samuel Kounev, Christian Krupitzer, A Case Study on Optimization of Warehouses. 2021. URL: https://www.researchgate.net/publication/357267571_A_Case_Study_on_Optimization_of_Warehouses (дата звернення: 09.12.2023).

14. Gianpaolo Ghiani, Gilbert Laporte, Roberto Musmanno, Introduction to Logistics Systems Management, 2nd Edition, Wiley, 2013. 480 p.

15. Alan Rushton, Phil Croucher, Peter Baker, The Handbook of Logistics and Distribution Management, Kogan Page, 2010. 664 p.

16. Christos H. Papadimitriou, Kenneth Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Dover Publications, 1998. 528 p.

17. William J. Cook, William H. Cunningham, William R. Pulleyblank, Alexander Schrijver, Combinatorial Optimization, Springer Cham, 2022. 402 p.

18. Ding-Zhu Du, Panos M. Pardalos, Handbook of Combinatorial Optimization, Springer New York, 1998. 2406 p.

19. Bernhard Korte, Jens Vygen, Combinatorial Optimization: Theory and Algorithms, Springer Publishing Company, 2018. 698 p.

20. Michael T. Goodrich, Roberto Tamassia – "Algorithm Design and Applications", Wiley, 2014. 800 p.

21. Nicos Christofides, Graph Theory: An Algorithmic Approach, London: Academic Press, 1975. 400 p.

22. Miles R., Hamilton K., Learning UML 2.0: A Pragmatic Introduction to UML: 1st edition, O'Reilly media, 2006. 290 p.

23. Michael Dawson, Programming with Python, Course Technology, 2020. 455 p.

24. Zed Shaw, Learn Python 3 the Hard Way, Course Technology, 2017. 320 p.

25. Dan Bader, Python Tricks: A Buffet of Awesome Python Features, 2017. 301 p.

26. David Beazley, Brian K. Jones, Python Cookbook: Recipes for Mastering Python 3, O'Reilly Media, 2017. 708 p.

27. Joshua Willman, Beginning PyQt: A Hands–on Approach to GUI Programming with PyQt6, Apress, 2nd edition 2022. 543 p.

28. Python. URL: https://docs.python.org/3/ (дата звернення: 11.12.2023).

29. Michael Herrmann, Python and Qt: The Best Parts, 2022. URL: https://payhip.com/b/YwDa (дата звернення: 15.12.2023).

30. Martin Fitzpatrick, Create GUI Applications with Python & Qt6, 2022. 796 p.

31. PySide [Документація] URL: https://wiki.qt.io/Qt_for_Python (дата звернення: 12.12.2023).

32. Luc Perkins, Eric Redmond, Jim Wilson, Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement, Pragmatic

Bookshelf, 2018, ISBN: 978–1680502534, 360 p.

33. Simon Riggs, Gianni Ciolli, PostgreSQL 14 Administration Cookbook, Packt Publishing, 2022. 608 p.

34. Hans–Jürgen Schönig, "Mastering PostgreSQL 13 – Fourth Edition", Packt, 2020. 476 p.

35. Васильєв О., Програмування мовою Python, Богдан, 2020. 504 с.

36. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 14.12.2023)

37. JSON. URL: https://www.json.org/json–en.html (дата звернення: 15.12.2023).

38. Coelho L., Richert W. Building Machine Learning Systems with Python. Packt, 2018. 290 p.

39. Teoh T., Rong Z., Artificial Intelligence with Python. Springer Nature, Singapore, 2022. 336 p.