НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2023 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Комп'ютерні системи та мережі» зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Централізована система для віддаленого управління складськими приміщеннями»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІО-22мп
Зубець Микола Васильович _____

Керівник:
доц. каф. ОТ, к.т.н., доц
Ткаченко Валентина Василівна _____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., професор
Кулаков Юрій Олексійович _____

Рецензент:
проф. каф. ІСТ, д.т.н., професор,
Корнієнко Богдан Ярославович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність *123. Комп'ютерна інженерія*
(код і назва)

Освітньо-професійна програма *Комп'ютерні системи та мережі*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
«____» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Зубцю Миколі Васильовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Централізована система для віддаленого управління складськими приміщеннями

Науковий керівник дисертації Ткаченко Валентина Василівна, доцент, кандидат технічних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «07» листопада 2023 р. No 5168-с

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження розроблення засобів управління складськими приміщеннями

4. Предмет дослідження процес управління складськими приміщеннями

5. Перелік завдань, які потрібно розробити: дослідити роль автоматизації управління складськими приміщеннями, дослідити існуючі підходи у вирішенні проблеми управління складськими приміщеннями, визначити актуальність створення системи на основі проведеного дослідження, розробити централізовану систему для віддаленого управління складськими приміщеннями, створити модель розробленої системи і продемонструвати її роботу

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Ткаченко В.В., доц., к.т.н., доц.		
2.			
3.			
4.			

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	Затвердження теми дослідження.	1.09.2023	
2.	Вивчення та аналіз завдання	02.09. 2023-24.09.2023	
3.	Пошук матеріалів по темі роботи	25.09. 2023-12.10.2023	
4.	Розробка централізованої системи для віддаленого управління складськими приміщеннями	13.10. 2023-17.11.2023	
5.	Розробка програмного продукту	18.11.23-2023.11.2023	
6.	Оформлення пояснювальної записки	23.11. 2023-27.11.2023	
7.	Передзахист	10.12.2023	
8.	Захист	18.01.24	

Студент ___Микола ЗУБЕЦЬ_____ _____
(підпис)

Науковий керівник дисертації ___Валентина ТКАЧЕНКО___ _____
(підпис)

РЕФЕРАТ

Структура та обсяг роботи. Пояснювальна записка магістерської дисертації складається з чотирьох розділів, містить 93 сторінки, 44 рисунків, 26 таблиць та 36 джерел.

Актуальність теми. Актуальність цієї теми полягає в тому, що з розвитком підприємництва, інфраструктури та промисловості з’являється необхідність в менеджменті ресурсами та виробленою продукцією, адже порядок гарантує вчасно вироблену продукцію, наявність складського місця, можливість вести статистику і багато чого іншого, що позитивно впливає на підприємницькі процеси.

Раніше подібний облік вівся вручну з використанням паперових записів, після чого їм на зміну прийшли системи управління складом (WMS). Проте з подальшим розвитком та ростом підприємств з’явилась необхідність в одночасному обслуговуванні розподілених складських приміщень з використанням однієї системи. Вирішення цієї потреби може колосально вплинути на продуктивність використання складських приміщень.

Тема роботи. Централізована система для віддаленого управління складськими приміщеннями

Мета роботи. Метою цієї роботи є розробка централізованої системи управління складськими приміщеннями, що дозволить підвищити ефективність управління розподіленими складськими приміщеннями у віддаленому режим, а саме дозволить інтеграцію розподілених складів в одній системі, і подальшій їх взаємодії. Подібна система має знизити час виконання операцій зі складами

Предметом дослідження є процес управління складськими приміщеннями, який породжує проблему ефективного керування розподіленими складськими приміщеннями у віддаленому режимі. Проблема полягає у відсутності інтеграції розподілених складів в одній системі, і подальшій їх взаємодії, що значно підвищує час виконання операцій зі складами

Об’єкт дослідження – розроблення засобів управління складськими приміщеннями, призначених для ефективного керування складськими приміщеннями у віддаленому режимі.

Методи дослідження. спостереження, порівняння, аналіз

Задачі дослідження. Для досягнення поставленої мети, необхідно виконати наступні задачі:

- Виконати аналіз літературних джерел в області управління складськими приміщеннями, на підставі якого визначити проблематику ефективного керування ними і виконати постановку завдань дисертаційної роботи.
- Розробити архітектуру централізованої системи віддаленого управління складськими приміщеннями для ефективної організації віддаленого управління розподіленими складськими приміщеннями.
- Розробити централізовану систему для віддаленого управління складськими приміщеннями, що має на меті вирішити проблему керування розподіленими складами.

Наукова новизна.

Розроблено архітектуру централізованої системи для віддаленого управління складськими приміщеннями, яка відрізняється від відомих систем можливістю віддаленого управління розподіленими складами. За рахунок цього досягається підвищення ефективності керуванням складськими приміщеннями, що дозволяє зменшити час виконання операцій за складами.

Практичне значення отриманих результатів. Практичним результатом роботи є розробка централізованої системи для віддаленого управління складськими приміщеннями, яка може бути інтегрована в робочий процес підприємств, задля підвищення їх ефективності керування розподіленими складськими приміщеннями.

Ключові слова. СИСТЕМА УПРАВЛІННЯ. ВІДДАЛЕНА СИСТЕМА, СКЛАД, ІНВЕНТАРИЗАЦІЯ, ЦЕНТРАЛІЗОВАНА СИСТЕМА, АРХІТЕКТУРА СИСТЕМИ, АВТОМАТИЗАЦІЯ, УПРАВЛІННЯ ЗАПАСАМИ

ABSTRACT

Structure and scope of work. The explanatory note of the master's thesis consists of four sections, contains 93 pages, 44 figures, 26 tables and 36 referrences.

Relevance of the topic. The relevance of this topic is that with the development of entrepreneurship, infrastructure and industry there is a need for management of resources and produced products, because order guarantees products produced in time, the presence of warehouse, the ability to conduct statistics and more, which has a positive effect on entrepreneurial processes.

Previously, similar accounting was manually used using paper records, after which they were replaced by WMS control systems (WMS). However, with the further development and growth of enterprises, there was a need for simultaneous service of distributed warehouses using one system. The solution to this need can enhance the productivity of warehouses.

The topic of work. Centralized system for remote warehouse management

Purpose of research. The purpose of this work is to develop a centralized management system of warehouses, which will increase the efficiency of management of distributed warehouses into remote mode, namely allowing the integration of distributed warehouses in one system, and their further interaction. A similar system should reduce the time of operations with warehouses

Subject of research is the process of management of warehouses, which creates the problem of effective management of distributed warehouses in remote mode. The problem is the lack of integration of distributed warehouses in one system, and their subsequent interaction, which significantly increases the time of performing operations with warehouses

Object of research is the development of warehouses management facilities designed to effectively manage the warehouses remotely.

Research methods. observation, comparison, analysis

Research tasks. To achieve this goal, the following tasks must be completed:

- Perform the analysis of literary sources in the field of management of warehouses, on the basis of which to identify the problems of effective management of them and to perform the tasks of the dissertation.
- Develop an architecture of a centralized system for remote warehouse management for efficient organization of remote management of distributed warehouses.
- Develop a centralized system for remote warehouse management that aims to solve the problem of managing distributed warehouses.

Scientific novelty. The architecture of a centralized system for remote control of warehouses, which differs from known systems by the possibility of remote management of distributed warehouses, has been developed. This achieves an increase in the efficiency of managing warehouses, which allows to reduce the time of operations in warehouses.

Application value. The practical result of the work is the development of a centralized system for remote management of warehouses, which can be integrated into the working process of enterprises, to improve their efficiency of management of distributed warehouses.

Keywords. MANAGEMENT SYSTEMS, REMOTE SYSTEMS, WAREHOUSE, INVENTORY MANAGEMENT, CENTRALIZED SYSTEMS, SYSTEM ARCHITECTURE, AUTOMATION, RESOURCE MANAGEMENT

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

WMS – Warehouse management system – система управління складом

ERP – Enterprise resource planning – планування ресурсів підприємства

MVC – Model-view-controller – модель-вид-контролер

SOA – Service oriented architecture – сервісно-орієнтована архітектура

ACID – Atomic, consistency, isolation, durability – атомність, узгодженість, ізоляція, довговічність

SQL – Structured query language – мова структурних запитів

DBMS – **СУБД** – Система управління базами даних

RDBMS – Relational database management system – Реляційна система керування базами даних

JSP – Jakarta Servlet Pages

EL – Expression Language

DAO – Data Access Object – Об’єкт доступу до даних

ВСТУП

Зі стрімким розвитком підприємництва, збільшенням інфраструктури та розширенням промислових підприємств тема систем управління складами стала надзвичайно актуальною. У цьому контексті з’являється все більша потреба в ефективному управлінні ресурсами та виробництвом. Ефективний і організований облік ресурсів гарантує вчасне виготовлення продукції, відстеження наявності товарів на складі, можливість вести детальну статистику та багато іншого, що сприяє покращенню підприємницьких процесів.

Впровадження системи управління складськими приміщеннями може значно поліпшити продуктивність та ефективність використання складських приміщень, за рахунок автоматизації процесів обліку та стеження за предметами на складі, що в свою чергу впливає на конкурентоспроможність та прибутковість підприємства.

У минулому, цей процес обліку зазвичай виконувався вручну з використанням паперових записів. Проте з ростом підприємств та складності і кількості операцій з’явилася необхідність в удосконаленій системі керування складом. Через попит на подібне рішення та бажання підвищити ефективність використання складу та зменшити витрати пов’язані з ним, почали з’являтися відповідні системи.

Тенденція на збільшення використання систем управління складськими приміщеннями з’явилась в Америці та Європі, проте на сьогоднішній день подібні системи використовуються у всьому світі, адже вони значно підвищують ефективність підприємств.

Проте у подібних систем є ряд недоліків, тому ціллю цієї роботи є дослідження існуючих корпоративних рішень з розробки систем керування складськими приміщеннями, виокремлення їх основних переваг та недоліків. На основі проведеного аналізу пропонується створення централізованої системи для віддаленого управління складськими приміщеннями, яка може одночасно обслуговувати розподілені складські приміщень за допомогою єдиної інтегрованої системи.

У зв’язку з усіма розглянутими факторами, які вказують на високу актуальність теми роботи, метою цієї роботи є створення передової централізованої системи для віддаленого управління складськими приміщеннями.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ СИСТЕМ УПРАВЛІННЯ СКЛАДСЬКИМИ ПРИМІЩЕННЯМИ

1.1. Концепція управління складськими приміщеннями

Для управління складськими приміщеннями використовуються системи управління складом (warehouse management systems - WMS). WMS за Бартолді [1] це така система, основна функція якої це облік надходження та вибуття товарно-матеріальних цінностей. Відповідно подібна система допомагає організувати роботу складського приміщення найбільш ефективним чином та допомагає досягненню поставлених цілей. Теоретично електронні таблиці або фізичні носії такі як ручка та папір також можна вважати WMS, проте є очевидними недоліки подібного підходу.

Згідно [2] основні функції подібних систем розширюються і можуть включати в себе наприклад запис точного розташування предметів на складі, оптимізацію використання доступного простору або координацію завдань для досягнення максимальної ефективності.

Системи такого виду класифікують за декількома показниками:

- вартістю системи;
- способом налаштування під потреби конкретного складу-замовника;
- складністю процесів, які вона здатна автоматизувати;
- напрямком орієнтованості (клієнтоорієнтовані та продуктоорієнтовні);

1.1.1. Типи WMS

Згідно Томпкінса [3] WMS поділяються на 3 типи:

1. Базова WMS – така система, що підтримує інвентаризацію та контроль місцеперебування. Така система дозволяє вести статистику кількості предметів, які поступають на склад та покидають його за певний проміжок часу. Ця варіація WMS дуже схожа на систему контролю інвентарю.

2. Розширена WMS – модифікація базової системи, що дозволяє відслідковувати кількість предметів на складі та додатково відслідковувати ресурси та час витрачені на різну роботу. Це дозволяє системі визначити ефективність використання складу та знайти шляхи її покращення.

3. Контрольована WMS – система, що може обмінюватись даними з іншими системами, пов’язаними з галуззю роботи складу, такими як замовлення покупців, транспортування, потреби виробництва та інші. Зазвичай ця система пов’язана з інтегруванням IoT технологій.

Якщо розглянути WMS з позиції інтеграції, то їх поділяють на 2 типи: інтегрована та автономна система.

Інтегрована система управління складом зазвичай представляє собою розширення функціональності вже існуючої системи планування ресурсів підприємства (enterprise resource planning - ERP). ERP системи відповідають за процеси виставлення рахунків, обліку та відстеження запасів. З іншого боку, система управління складом відповідає за приймання замовлень, управління процесом комплектування замовлень, проведення інвентаризації, приймання та відвантаження продукції. Якщо ці системи можуть бути інтегровані в одну, то стає значно простіше відслідковувати, в які замовлення варто вкладати кошти.

Якщо є продукт, який має високий обсяг продажів, але приносить низький прибуток, можна вирішити замінити його на продукт із вищою прибутковістю, навіть якщо це призведе до менших обсягів продажів. За допомогою інтегрованої системи управління складом можна вести детальний фінансовий аналіз і приймати обґрунтовані рішення щодо товарних запасів.

Автономна система управління складом – це комплексне програмне забезпечення, яке в основному призначене для управління складськими процесами. Таким чином, воно може бути обмежена у функціональності для інших аспектів бізнесу, таких як інвентаризація чи бухгалтерський облік. Оскільки її головним призначенням є управління складом, цей тип системи управління складом може мати розширені функції стосовно звітності, які спрямовані на оптимізацію складських процесів.

Якщо ж розглянути ці системи згідно їх розташування, то тут їх також поділяють на 2 типи: локальні та хмарні системи управління складськими приміщеннями.

Локальна WMS — це система, у якій саме власник відповідає за розміщення та підтримку як апаратного, так і програмного забезпечення, пов’язаного з системою. Хоча це дає повний контроль над такими речами, як час безвідмовної роботи та безпека, але це також супроводжується великими початковими витратами, оскільки власник несе відповідальність за всі компоненти. Також саме власник має підтримувати цю систему.

Як альтернатива локальним системам, хмарні системи управління складом зазвичай працюють на основі підписки. Вони розташовані на віддаленому сервері. Задачі, такі як виправлення помилок і оновлення програмного забезпечення, покладаються на постачальника, і зазвичай після реєстрації власник отримує гарантований рівень надійності системи.

Відповідно можна виділити основний мінус усіх цих систем, а саме відсутність інтеграції подібних систем в бізнеси та підприємства більше ніж з одним складським приміщенням. Частково цей недолік можна вирішити використанням контрольованих WMS з зовнішнім налаштуванням інтеграції, проте подібний підхід має свої мінуси, пов’язані з необхідністю налаштування WMS окремо для кожного приміщення, ускладненою логікою зовнішнього модулю, складністю інтеграції нових приміщень, або видалення старих.

1.1.2. Функції WMS

Основне завдання WMS системи – автоматизувати склад, прискорити робочі процеси та мінімізувати будь-які помилки, пов’язані з людським фактором.

Рис.1.1 Функції, які може виконувати WMS [4]

Функції, зображені на рис.1.1, які найчастіше наявні в подібних системах, описані в [5], включають в себе:

- Оптимізація складського приміщення. Необхідно внести в систему інформацію щодо розміру складу та докладну інформацію про наявні запаси, такі як розміри палет, кількість існуючого обладнання та товарів. Програмне забезпечення обробить ці дані і автоматично створить графічну діаграму, яка сприятиме кращій оптимізації розміщення виробничих потужностей та більш ефективному використанню доступного простору.
- Контроль запасів. У такій системі чітко відображено, які товари є в дефіциті і потребують додаткового замовлення. Є можливість вибору "поповнення одиниць товару, палету або коробки". Крім того, ця функція сприяє уникненню перевищення запасів на складі, оскільки товар замовляється у необхідній кількості, а не навмання.
- Оптимізація графіка роботи. Подібна система аналізує передбачувані операції компанії у майбутньому та, зважаючи на кількість персоналу, здатна

створювати оптимальний розклад роботи. Наприклад, вона може визначати, скільки працівників необхідно для вчасного виконання певного замовлення. Крім того, WMS система може легко взаємодіяти з транспортними компаніями та передчасно планувати час завантаження та розвантаження товарів в транспортний засіб.

- Моніторинг та звітність. Систематичний моніторинг та контроль робочих процесів дозволяють виявляти проблеми на складі та вирішувати їх. Система надає інформацію, чи є достатньо персоналу, і, можливо, підкаже, чи можна зменшити його чисельність. Також, є можливість відстежувати продуктивності окремих працівників. Крім того, це спрощує контроль за ключовими показниками ефективності бізнесу, встановленням цілей для команди та відстеженням їх виконання.
- Оптимізація зберігання. Система розподіляє товари відповідно до різних параметрів, таких як вага, потреба чи строк придатності. Це означає, що товари з великим оборотом будуть розташовані ближче до зони відвантаження.
- Управління відвантаженням та завантаженням. Система управління складом вміє отримувати та розпізнавати інформацію про товари, які надходять на склад, генерувати штрих-коди, перевіряти дані про вантаж та виявляти відмінності в описі чи характеристиках товару.
- Інвентаризація, яка проводиться без зупинки робочих процесів.
- Комплектація. Процес збору замовлень значно прискорюється, оскільки комплектувальники використовують заздалегідь підготовлені пакувальні списки, на яких вже зазначені всі необхідні елементи замовлення та їх місцезнаходження.
- Збільшення кількості клієнтів. Завдяки оптимізації складських операцій, підприємство може співпрацювати з більшою кількістю клієнтів і розширювати свою клієнтську базу.
- Документообіг. Після впровадження системи управління складом, склад та підприємство повністю відмовляються від використання паперових

звітів та документації, переходячи виключно до електронного формату, що виявляється надзвичайно зручним і ефективним.

1.2. Архітектура WMS

Згідно [6], зазвичай, архітектура WMS складається з трьох основних рівнів: інтерфейсу, бази даних і бізнес-логіки, що зображено на рис.1.2.

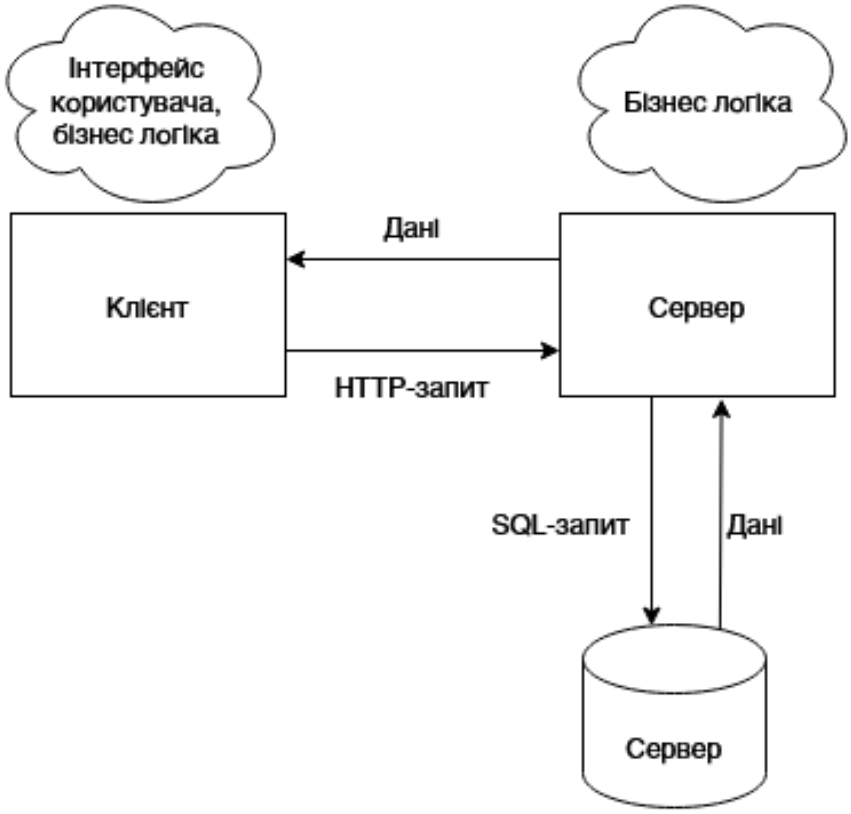

Рис.1.2 Базова архітектура WMS системи [6]

Рівень інтерфейсу відповідає за взаємодію користувача з додатком і відображення інформації користувачу. Основні завдання цього рівня включають в себе:

- Відображення інтерфейсу користувача
- Обробку введених даних користувача і відправлення цих даних на рівень бізнес-логіки для обробки.
- Передачу результатів роботи рівня бізнес-логіки назад на відображення користувачу.

Рівень бази даних відповідає за зберігання та управління даними, необхідними для роботи додатку. Основні функції цього рівня:

- Зберігання даних у відповідних структурах (таблиці, колекції тощо).
- Отримання, оновлення або видалення даних за запитами бізнес-логіки.
- Забезпечення безпеки та конфіденційності даних.

Рівень бізнес-логіки містить основну функціональність додатку, яка визначає, як додаток взаємодіє з даними і виконує конкретні завдання. Основні завдання цього рівня:

- Обробка бізнес-логіки, наприклад розрахунки, валідація даних і прийняття рішень.
- Керування взаємодією з рівнями інтерфейсу та бази даних.
- Забезпечення логіки додатку, яка відповідає за бізнес-процеси.

Ці три рівні взаємодіють між собою, створюючи функціональний та структурний фундамент системи. Подібна модель архітектури має переваги над іншими, включаючи розподіл відповідальності, полегшення розробки та супроводу, підвищення масштабованості і зручність модифікацій.

1.3. Корпоративні системи управління складськими приміщеннями

Наразі існує багато корпоративних систем управління складськими приміщеннями, у кожної з них є свої переваги та недоліки.

Серед найбільш популярних систем можна виділити Sortly, NetSuite WMS, Fishbowl Inventory, Oracle та інші.

1.3.1. Sortly

Ця WMS є однією із найкращих програм для візуального керування складом на основі фотографій для малого бізнесу. Згідно [7] за допомогою цього програмного забезпечення можна отримати систему, яка може відстежувати будь-який об’єкт або пов’язану з ним деталь у різних місцях.

Користувачі цієї WMS можуть додавати спеціальні поля, квитанції та кілька зображень до кожного товару, щоб полегшити відстеження активів. Користувачі також можуть створювати та друкувати QR-коди та штрих-коди, які можна сканувати за допомогою сканеру. Приклад робочого вікна можна побачити на рис.1.3.

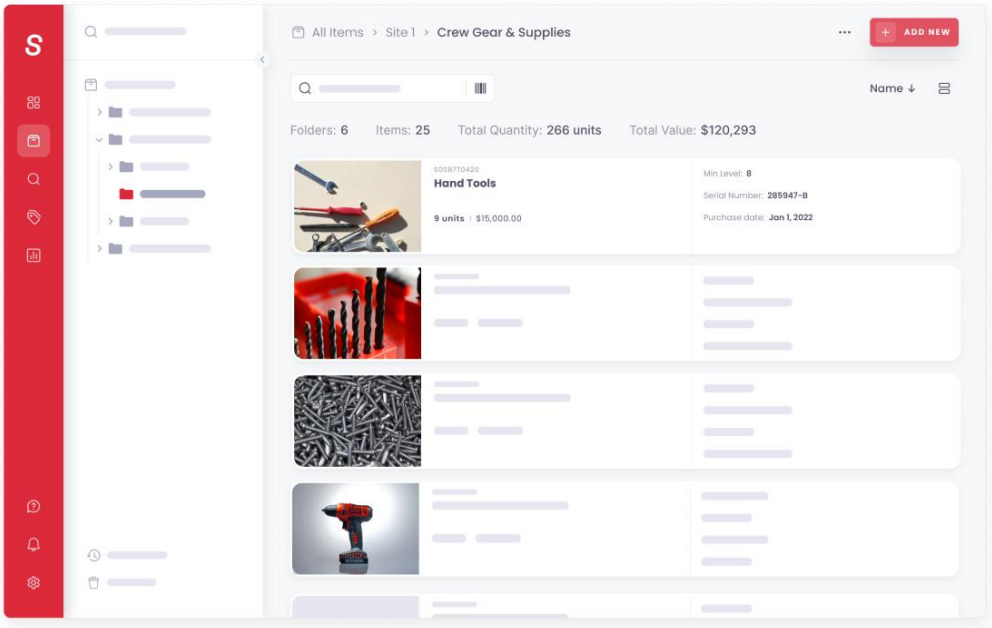

Рис.1.3 Приклад вікна Sortly [8]

Переваги Sortly:

- Сканування й оновлення елементів за допомогою QR-кодів і штрих-кодів.
- Створення автоматичних сповіщень для відстеження кількості продукції
- Призначайте ролі користувачів і керування правами доступу
- Відстеження запасів продукції та активності користувачів
- Створення власних звітів CSV і PDF.

Недоліки Sortly:

- Відсутність інтеграцій
- Складний інтерфейс
- Це програмне забезпечення, яке треба встановлювати на пристрої

1.3.2. NetSuite WMS

NetSuite WMS – система, що допомагає оптимізувати складські та виробничі операції. Відповідно до [9], ця система допомагає користувачам виконувати найважливіші функції складу, такі як отримання, зберігання та відправлення товарів. Приклад вікна програми можна побачити на рис.1.4.

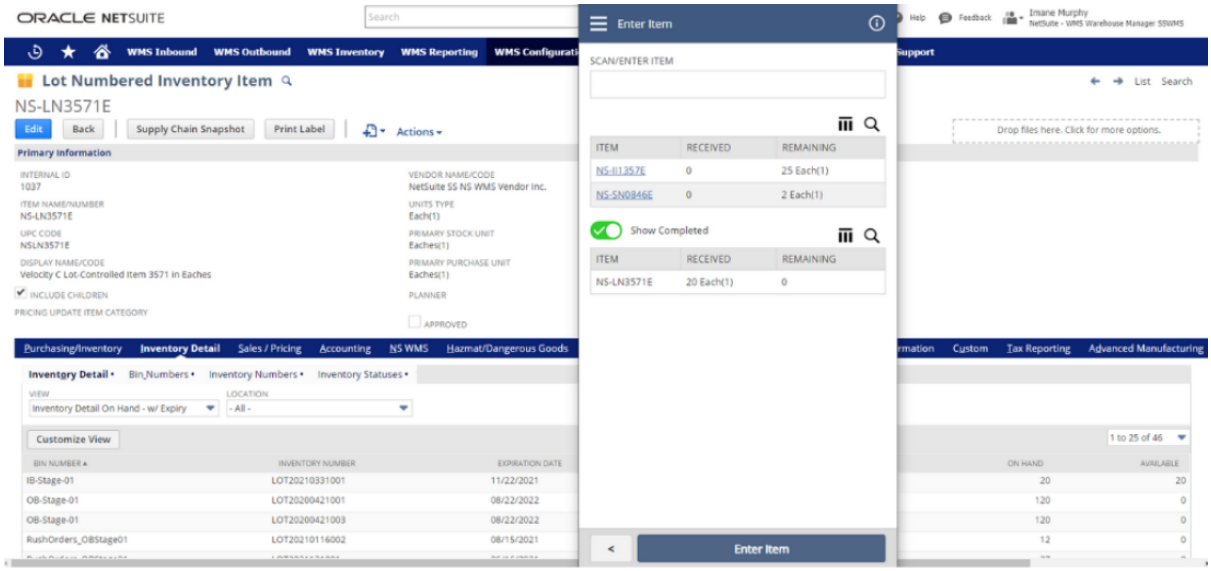

Рис.1.4 Приклад вікна NetSuite WMS [10]

Переваги NetSuite WMS:

- Сканування й оновлення елементів за допомогою QR-міток і штрих-кодів.
- Визначення стратегії прибирання предметів
- Управління завданнями
- Відстеження запасів продукції та активності користувачів
- Створення власних звітів

Недоліки NetSuite WMS:

- Присутні деякі функції ERP, проте система не є повною та ефективною ERP-системою.
- Програмне забезпечення, що треба встановлювати

1.3.3. Fishbowl Inventory

Ця WMS є інтеграційною складовою середовища QuickBooks, до якого також входять і інші ERP модулі та елементи управління підприємством. Приклад вікна програми можна побачити на рис. 1.5. *[11]*

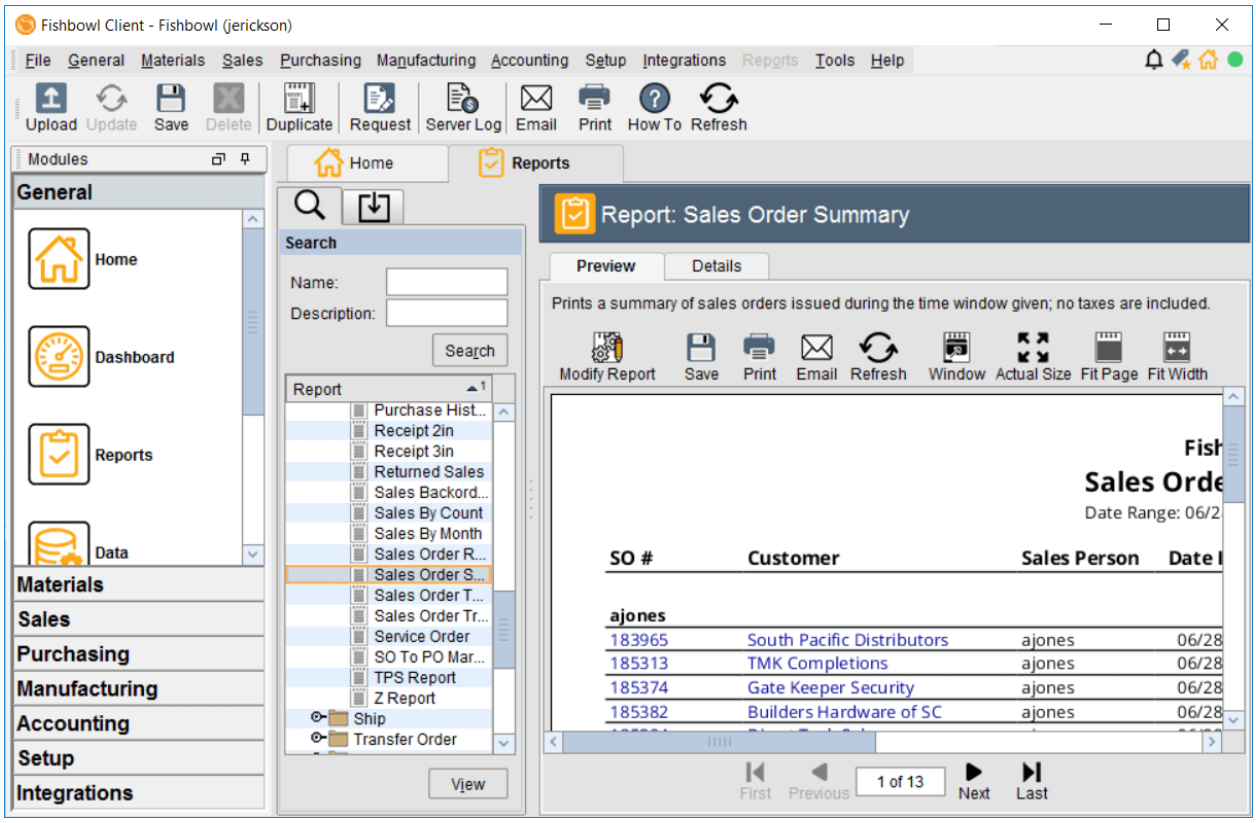

Рис. 1.5 Приклад вікна Fishbowl Inventory [12]

Переваги Fishbowl Inventory:

- Можливість формувати звіти
- Управління замовленнями
- Інтегрована купівля та продаж
- Відслідковування доставок

Недоліки Fishbowl Inventory:

- Відсутність інтеграції між складами
- Незручний інтерфейс
- Модуль до програмного забезпечення, що треба встановлювати

У цілому на прикладі цієї системи можна звернути увагу на масивний недолік, пов’язаний з тим що це модуль для ERP системи. Це означає що таке

рішення підходить лише для крайніх випадків, дороге, та не працює автономно. Тому під час розробки централізованої системи для віддаленого управління складськими приміщеннями важливо зробити її не настільки прив'язаною до інших великих систем, а особливо, не настільки залежною від них.

1.3.4. Manhattan WMS

Manhattan WMS, або більше відома як Manhattan SCALE – це гнучка, функціональна WMS-система на платформі Microsoft.Net. Згідно з [13], рішення розроблено компанією Manhattan Associates, провідним розробником IT-рішень для логістики в світі. Рішення Manhattan Associates мають широкий функціонал і можливість масштабування системи, дозволяють автоматизувати повний комплекс складських операцій (від приймання до відвантаження).

Ця WMS-система, зображена на рис.1.6, підходить для автоматизації складів з процесами середнього ступеня складності, розподільних центрів, зберігання складів з великою кількістю номенклатурних позицій, температурних зон і т.д.

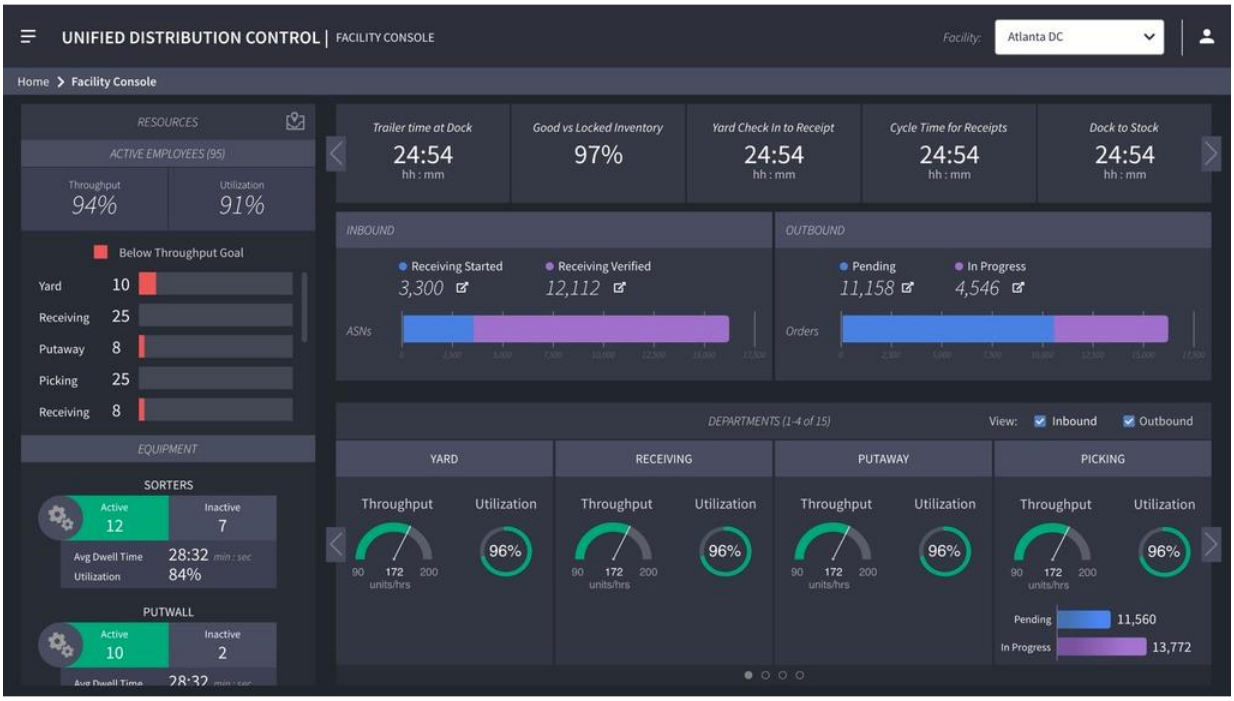

Рис.1.6 Приклад вікна Manhattan WMS [14]

Переваги Manhattan WMS:

- Сучасна архітектура
- Хмарна система
- Кастомізація
- Масштабування системи
- Легкість у встановленні та керуванні

Недоліки Manhattan WMS:

- Відсутність інтеграції між складами

У цілому ця система управління складськими приміщеннями є потужною та новітньою, проте їй не вистачає можливості керувати одночасно декількома складськими приміщеннями, що не дозволяє великим компаніям ефективно використовувати їх склади без додаткових інтеграцій.

1.4. Переваги використання системи управління складськими приміщеннями

Надійна цифрова система управління складом є важливим елементом для будь-якого підприємства з наявністю товарних запасів. Вона сприяє збереженню коштів і підвищенню продуктивності в багатьох галузях. Переваги, використання подібних систем включають описані в роботі [15]:

- Підвищення операційної ефективності. З метою підвищення продуктивності, забезпечення безперебійності роботи та здатності обробляти великі обсяги товару системи управління складськими приміщеннями автоматизують та оптимізують усі складські процеси, починаючи від приймання товарів та закінчуючи відправкою. Вони допомагають зменшити помилки під час комплектування та відвантаження товарів і усувають неефективну та зайву роботу. Паралельно системи WMS обмінюються інформацією з системами ERP і управління транспортом, надаючи комплексне уявлення, яке виходить за межі складських операцій і сприяє прискоренню переміщення товарів.
- Зменшення відходів і витрат. Якщо на складі є запаси із обмеженим терміном придатності або швидкопсувні товари, програмне забезпечення

управління складом може визначити, які позиції слід обирати в першу чергу, або які можуть потребувати стимулювання продажу, з метою мінімізації витрат. Крім того, це може допомогти визначити найефективніше використання складського простору, включаючи розташування запасів і оптимальні маршрути переміщення. Деякі системи навіть пропонують розширене моделювання для створення оптимальних планів розміщення товарів та обладнання на різних рівнях складу, з метою підвищення продуктивності та економії часу та ресурсів.

- Видимість запасів у режимі реального часу. З використанням штрих-кодування, RFID-кодів, датчиків та інших засобів відстеження. Система управління складом надає інформацію щодо місцезнаходження запасів, коли вони рухаються на склад, всередині нього та між різними регіонами в реальному часі. Завдяки цій видимості можна створювати більш точні прогнози попиту, ефективно впроваджувати стратегії своєчасної інвентаризації та поліпшувати контроль, особливо важливий у випадку потреби в відкликанні товарів.
- Покращене управління робочою силою. Система управління складом може сприяти прогнозуванню потреб у робочій силі, створенню графіків роботи, оптимізації часу переміщення на складі та автоматичному призначенню завдань правильним працівникам на основі їх кваліфікації, місцезнаходження та інших чинників. Система управління складом також може сприяти підвищенню морального духу співробітників, створюючи більш спокійне, організоване та безпечне робоче середовище, де працівники відчувають, що їх час цінується і використовується раціонально.
- Кращі відносини з клієнтами та постачальниками. Завдяки системі управління складом, клієнти можуть насолоджуватися швидшим виконанням своїх замовлень, швидшою доставкою та зменшеною кількістю помилок. Це позитивно впливає на їхнє задоволення і лояльність, а також сприяє покращенню репутації бренду. Постачальники також можуть відчути скорочення часу очікування на вантажних майданчиках та вантажних доках.

1.5. Сфера застосування WMS

Найчастіше подібні системи використовують комерційні організації зі складами. Компанії, які мають склад або розподільний центр для зберігання та керування запасами, зазвичай отримують найбільшу користь від систем управління складом. Це стосується роздрібних торговців, виробників, оптовиків та інших підприємств, яким потрібно зберігати, відстежувати та відправляти великі обсяги товарів. Як зазначено в роботі [16] основні галузі використання цих систем включають в себе:

- Роздрібна торгівля. Окрім програмного забезпечення для керування роздрібною торгівлею, роздрібні продавці використовують програми для керування складами, щоб керувати запасами та замовленнями у своїх онлайн-магазинах або звичайних підприємствах.
- Електронна комерція – системи управління складами електронної комерції допомагають підприємствам ефективно відстежувати замовлення та виконувати відправлення для своїх онлайн-магазинів, щоб вони могли задовольняти швидкі запити клієнтів.
- Оптова торгівля. Іншою галуззю, яка використовує програмне забезпечення WMS, є оптові торговці та дистриб’ютори. Це допомагає їм оптимізувати завдання інвентаризації, відстежувати відправлення та оптимізувати виконання замовлень.
- Виробництво. Системи управління складськими приміщеннями є популярним технічним засобом у виробничих програмних рішеннях, таких як відстеження виробничих процесів і управління ланцюгами поставок.
- Логістика та транспортування – для логістичних і транспортних компаній WMS може планувати транспортування та відстежувати відправлення, коли вони просуваються через ланцюжок поставок.

Система управління складом виступає важливим елементом управління ланцюгом поставок. В роботі [17] показано як подібні системи сприяють оптимізації переміщення та зберігання товарів на складі або у розподільному центрі, виступаючи ключовою ланкою між діяльністю на верхньому та

нижньому рівнях ланцюга поставок. Крім того, програмне забезпечення системи управління складом відстежує рівень запасів, стан товарів на складі та надає дані в реальному часі для всіх учасників ланцюга поставок.

Отже, WMS забезпечує наявність необхідного продукту в необхідній кількості, в необхідний час і в необхідному місці на складі. Це допомагає підприємствам знизити витрати на зберігання запасів, скоротити час циклу замовлення та покращити обслуговування клієнтів. Крім того, це надає можливість приймати кращі рішення щодо управління запасами, планування складу та розподілу робочої сили.

Висновки до розділу 1

Перший розділ роботи присвячений докладному дослідженню систем управління складськими приміщеннями. В цьому розділі ретельно розглянуто їх концепцію, різновиди, функції та архітектуру.

Виконано аналіз типів систем управління складами, включаючи їх основні характеристики та призначення, а також проведено аналіз сфер застосування таких систем, де вони можуть бути корисними та необхідними.

Під час аналізу були розглянуті найбільш популярні комерційні рішення на ринку, що надало можливість отримати об’єктивну картину стану технології та можливостей, які існують для вирішення завдань, пов’язаних із управлінням складами.

Виконано аналіз джерел в області управління складськими приміщеннями:

1. Виконано дослідження сучасного стану розвитку систем управління складськими приміщеннями, розглянуто базову архітектуру таких систем та функції які вони найчастіше всього виконують.
2. Визначені основні переваги та недоліки існуючих комерційних рішень, що дозволяє розробити систему з урахуванням відповідних недоліків.
3. Аргументовано необхідність розробки системи управління складами, яка дозволяє віддалено керувати розподіленими складами.
4. Було розглянуто сфери застосування систем керування складськими приміщеннями, що аргументує доцільність розробки подібних систем через високий на них попит.

На підставі виконаного аналізу визначена проблематика в області керування розподіленими складськими приміщеннями:

1. Відсутність можливості керування розподіленими складами з однієї системи;
2. Відсутність можливості керування складськими приміщеннями віддалено;
3. Складність інтеграції систем управління складськими приміщеннями.

Беручи до уваги проблематику ефективного управління складськими приміщеннями, встановлено завдання цієї роботи:

1. Розробити архітектуру централізованої системи віддаленого управління складськими приміщеннями для ефективної організації віддаленого управління розподіленими складськими приміщеннями.

2. Розробити централізовану систему для віддаленого управління складськими приміщеннями, що має на меті вирішити проблему керування розподіленими складами.

РОЗДІЛ 2

РОЗРОБКА МОДЕЛІ ЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ ДЛЯ ВІДДАЛЕНОГО УПРАВЛІННЯ СКЛАДСЬКИМИ ПРИМІЩЕННЯМИ

2.1. Вибір шаблону архітектури системи

2.1.1. Клієнт-серверна архітектура

Однією з основних визначених проблем є відсутність можливості керування складськими приміщеннями віддалено, тому пропонується розглянути клієнт-серверну архітектуру задля досягнення поставлених завдань.

Клієнт-серверна архітектура є однією з основних моделей розподіленого обчислення в інформаційних технологіях. Як описано в [18] в цій архітектурній моделі відбувається розподіл функціональності між двома типами комп'ютерів: клієнтами та серверами, що зображено на рис. 2.1.

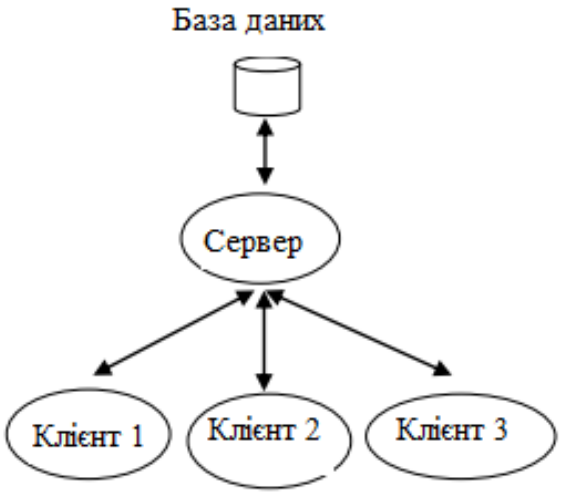

Рис.2.1 Клієнт-серверна архітектура [19]

Клієнт-серверна модель – це спосіб організації програмного середовища, в якому задачі та навантаження розподіляються між постачальниками ресурсів або послуг (серверами) і користувачами, які називаються клієнтами. У цій архітектурі клієнтський комп'ютер надсилає запит на отримання інформації серверу через мережу, після чого сервер обробляє цей запит і надсилає відповідні дані назад клієнту. Клієнти самі не надають ресурси для інших користувачів. Прикладами використання клієнт-серверної моделі є електронна пошта, веб-

браузери та інші схожі системи, де користувачі отримують доступ до інформації та послуг, які зберігаються на серверах через мережу.

Ключовою перевагою клієнт-серверної моделі є зручність забезпечення безпеки даних завдяки її централізованій структурі, яка дозволяє здійснювати контроль доступу на рівні серверів шляхом встановлення правил безпеки. Додатково, ця модель не залежить від операційних систем клієнтів та серверів, оскільки комунікація між ними відбувається за допомогою стандартизованих клієнт-серверних протоколів.

Відповідно до [19] серйозним недоліком клієнт-серверної моделі є можливість перевантаження сервера, якщо одночасно надійде занадто багато запитів від клієнтів. Це може викликати надмірне навантаження на мережу та призвести до відмови у обслуговуванні користувачів. У контексті поставленої задачі імовірність перевантаження системи дуже низька, через низьку потребу в запитах та відносно незначній кількості клієнтів.

Клієнти зазвичай взаємодіють з серверами через набір протоколів TCP/IP, як видно на рис.2.2. Протокол TCP – це спосіб передачі даних, який акцентує на забезпеченні стійкого зв’язку між клієнтом і сервером. Іншими словами, згідно [20] цей протокол встановлює і підтримує з’єднання між обчислювальними пристроями до тих пір, поки програми, які працюють на цих пристроях, не завершать обмін інформацією.

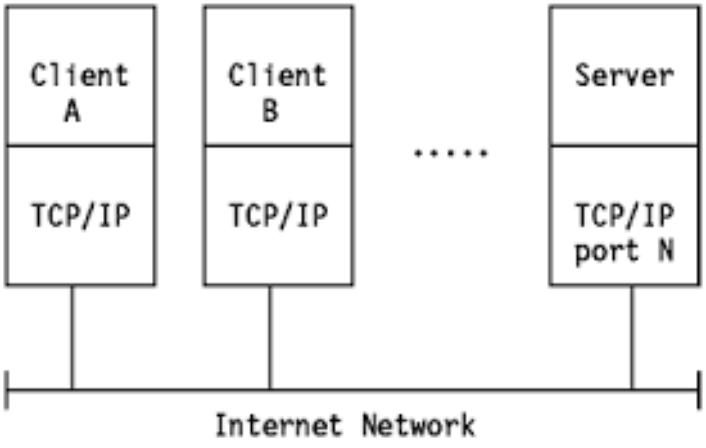

Рис.2.2 TCP/IP протокол в клієнт-серверній архітектурі [20]

З огляду на переваги цієї архітектури, та в зв’язку з поставленим завданням створення централізованої системи для віддаленого управління складськими приміщеннями, саме вона і підходить для вирішення поставленого завдання найкраще.

2.1.2. Сервісно-орієнтована архітектура

При розробці подібної системи варто розглянути існуючий підхід до розробки віддалених систем – сервісно-орієнтовану архітектуру, оскільки теоретично вона могла б допомогти розв’язати поставлені задачі.

Як описано в [21] - у сервісно-орієнтованій архітектурі, зображеній на рис.2.3, сервіси використовують певні стандарти зв’язку для передачі і обробки повідомлень, і ці стандарти описуються у метаданих. Ці метадані не лише розкривають функціональні можливості сервісу та його якість обслуговування, але й дозволяють користувачам об’єднувати великі фрагменти функціональності для створення додатків. Це відбувається за рахунок використання існуючих сервісів і за потреби їх комбінацій. Сам сервіс надає споживачу простий інтерфейс, який приховує внутрішню будову, діючи при цьому, умовно, як чорний ящик. Крім того, інші користувачі також можуть отримувати доступ до цих незалежних сервісів, не потребуючи знань про їх внутрішню реалізацію.

Відповідно до [21], основні переваги SOA включають в себе:

- Підвищена модульність: SOA сприяє створенню незалежних модулів або служб, які можуть бути легко розширені та підтримуються окремо.
- Повторне використання коду: Сервіси можуть бути повторно використані в різних додатках, що зменшує дублювання коду і сприяє ефективному використанню ресурсів.
- Легкість інтеграції: SOA дозволяє легко інтегрувати різні системи і додатки, оскільки вони можуть взаємодіяти через стандартизовані інтерфейси.
- Гнучкість: Здатність комбінувати та змінювати налаштування сервісів надає можливість швидко реагувати на зміни в бізнес-вимогах.

- Спрощена підтримка: Сервіси можуть бути підтримані і оновлені окремо без впливу на інші частини системи.

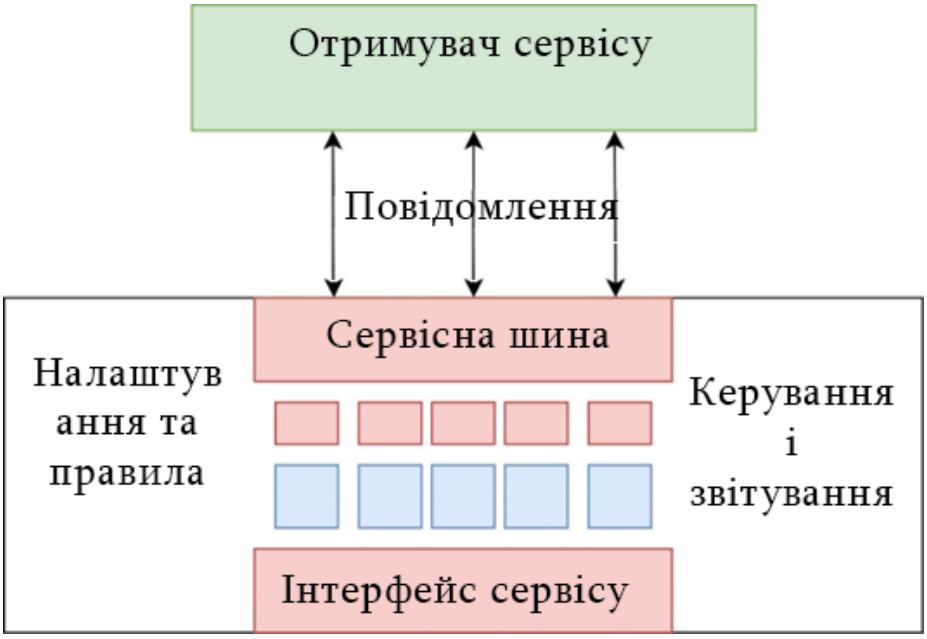

Рис.2.3 Сервісно-орієнтована архітектура [22]

Також згідно [21], недоліками SOA є:

- Складність управління: При розширенні SOA-системи може зростати складність управління великою кількістю сервісів та їх взаємозв’язками.
- Потенційні проблеми з безпекою: Вимоги до безпеки складніше впровадити в середовищі, де існує багато сервісів та точок доступу.
- Передача даних і продуктивність: Передача даних між сервісами може вимагати додаткових витрат, а це може вплинути на продуктивність системи.
- Залежність від мережі: SOA передбачає взаємодію між різними сервісами через мережу, і це може призвести до проблем при недоступності мережі.
- Витрати на розгортання та інфраструктуру: Розгортання та підтримка інфраструктури для SOA може призвести до додаткових витрат.

З огляду на зазначені вище недоліки архітектури, можна визначити, що вона не підходить для розробки централізованої системи віддаленого управління складськими приміщеннями.

2.1.3. Архітектура модель-вид-контролер

Логічним є розгляд існуючого архітектурного підходу розробок систем – модель-вид-контролер, через те, що сутності поставленої задачі можна розділити на основні блоки цієї моделі без зайвих проблем.

MVC є популярним підходом до структурування програмного коду. Основна ідея в MVC полягає в розділенні коду на різні функціональні блоки, зображені на рис.2.4, кожен з яких відповідає за конкретне завдання. Згідно з [23] одна частина коду відповідає за збереження даних в програмі (модель), інша частина робить програму дружньою для користувача і відображає інформацію (вид), а третя частина відповідає за керування логікою програми і взаємодію між моделлю та представленням (контролер).

MVC допомагає організувати основні функції коду у відокремлені та добре структуровані блоки. Це спрощує процес розробки програми, полегшує її аналіз і дозволяє зрозуміло комунікувати з іншими розробниками та користувачами програми.

Рис.2.4 Архітектура модель-вид-контролер [24]

- Модель – це частина коду, яка відображає абстрактні або реальні об’єкти та концепції, що використовуються у програмі. Вона може містити необроблені дані або визначати основні складові програми. Зазвичай визначається базою даних.
- Вид – це частина коду, яка відповідає за те, як програма відображається і взаємодіє з користувачем. Вона містить всі функції і елементи, які користувач бачить на екрані. Код цього блоку визначає зовнішній вигляд програми та сприйняття користувачем інтерфейсу.
- Контролер – це частина коду, яка виступає як посередник між моделлю і представленням. Вона отримує дані від користувача та вирішує, що робити з цими даними в контексті програми. Контролер відповідає за взаємодію між моделлю і видом, реалізуючи логіку та операції, необхідні для правильного функціонування програми.

Відповідно до [24], переваги MVC архітектури включають в себе:

- Можливість спільної роботи кількох розробників над різними частинами програми (модель, контролер, вид).
- Логічне групування пов’язаних функцій в контролері та представленнях. Представлення для конкретної моделі також організовані спільно.
- Можливість використовувати різні види моделей в одній програмі.
- Загальні компоненти програми зручно керуються і мають менше залежностей, що дозволяє підвищити відмовостійкість програми.

Недоліки подібної архітектури:

- Складність навігації в структурі, оскільки MVC вводить нові рівні абстракції та вимагає від розробників адаптації до методів розбиття програми на компоненти.
- Вимога до розробників мати знання багатьох технологій. Використання MVC передбачає розуміння та володіння різними технологіями, що може стати перепоною для розробників.

Для систем управління складом модель MVC може бути досить вигідною через наступні причини:

- Розподіл функціональності. WMS система складається з різних компонентів, таких як управління запасами, відправка, отримання, інвентаризація і т. д. Використання моделі MVC дозволяє логічно розділити ці функції між моделлю, контролером і видом.
- Гнучкість та масштабованість: WMS системи можуть вимагати постійних змін та розширень. Використання моделі MVC дозволяє легко внести зміни у різних частинах системи без впливу на інші компоненти. Це полегшує підтримку, розвиток та масштабування системи.
- Спільна робота розробників: Великі WMS системи можуть розроблятися різними командами розробників. Використання моделі MVC дозволяє різним групам розробників працювати над окремими частинами системи паралельно, забезпечуючи взаємодію через визначені інтерфейси контролерів.
- Підтримка та тестування: Використання моделі MVC полегшує тестування і супровід системи, оскільки окремі компоненти можуть бути протестовані незалежно, а також легко заміненими або оновленими без зміни інших частин системи.

У підсумку, модель MVC може полегшити розробку та підтримку систем керування складськими приміщеннями, оскільки вона допомагає логічно розділити функціональність та забезпечити більшу гнучкість і масштабованість системи.

2.2. Вибір бази даних

Оскільки в моделі MVC одним з трьох основних блоків є модель, тому варто обрати таку базу даних, яка буде ефективно підходити для неї. Це дозволить створити надійну MVC модель, що дасть змогу вирішити поставлені у роботі задачі.

2.2.1. NoSQL

Як зазначено в [25], NoSQL – це підхід до проектування баз даних, який акцентує увагу на створенні механізму для зберігання та пошуку даних, які представлені у формі, відмінній від звичайних табличних взаємозв'язків, що характерні для реляційних баз даних. Замість стандартної табличної структури, яка властива реляційним базам даних, бази даних NoSQL містять дані в одній об'єднаній структурі даних.

Згідно [25] бази даних NoSQL надають ряд переваг у порівнянні зі звичайними реляційними базами даних:

- Гнучка модель даних: В базах даних NoSQL, схеми зазвичай є дуже гнучкими. Гнучка схема дозволяє з легкістю вносити змін у базу даних при зміні вимог до даних. Можна швидко адаптувати та інтегрувати нові функції додатків, для надання користувачам нового функціоналу.
- Горизонтальне масштабування: В більшості реляційних даних, при перевищенні вимог поточного сервера, доводиться його масштабувати вертикально (переходити на більш потужний та дорожчий сервер). Більшість баз даних NoSQL дозволяють горизонтальне масштабування, тобто можливість додавати додаткові, менш потужні сервери при потребі.
- Швидкі запити: Запити в базах даних NoSQL можуть бути значно швидшими, ніж у SQL-базах даних. Одна з причин полягає в тому, що дані в SQL-базах зазвичай нормалізуються. Це означає, що дані про один об'єкт або сутність розподіляються по різних таблицях. Це вимагає об'єднання таблиць при виконанні запитів, що може навантажувати систему, особливо при великих об'ємах даних. В базах даних NoSQL дані зазвичай зберігаються в оптимізованому для запитів форматі, що допомагає виконувати запити дуже швидко.
- Легкість для розробників: Деякі бази даних NoSQL, такі як MongoDB, дозволяють відображати структури даних з бази даних у структури з мов програмування. Це дозволяє розробникам зберігати дані так само, як вони

використовуються у коді. Ця можливість дозволяє розробникам писати менше коду, що прискорює розробку і зменшує кількість помилок.

Недоліки NoSQL баз даних включають в себе:

- Одним з найбільш поширених недоліків баз даних NoSQL є їх відсутність підтримки транзакцій, відомих як ACID для операцій між кількома записами. При належному структуруванні схеми бази даних, атомарність одного запису може бути прийнятною для багатьох програм, проте існують додатки, які вимагають забезпечення ACID для групи записів.
- Оскільки моделі даних у NoSQL базах даних зазвичай оптимізовані для операцій читання та запису, а не для зменшення дублювання даних, то бази даних NoSQL можуть виявитися більшими за SQL бази даних. Проте на сьогодні витрати на зберігання даних настільки доступні, що ця проблема вже не є такою суттєвою.
- Залежно від обраного типу NoSQL бази даних, може виникнути обмеження в можливості задовольнити всі потреби в одній системі. Наприклад, графові бази даних ідеально підходять для аналізу взаємозв’язків у інформації, але вони можуть не бути оптимальним варіантом для пошуку даних, наприклад за запитом з діапазоном.

З огляду на недоліки NoSQL баз даних, а особливо відсутність ACID операцій між кількома записами, можна дійти висновку, що подібний тип баз даних не підходить для виконання поставленої задачі з розробки централізованої системи віддаленого управління складськими приміщеннями. Подібні системи потребують підтримки транзакцій, навіть для такої банальної функції як журналювання.

2.2.2. Реляційна база даних та SQL

Структурована мова запитів – це мова, яка спеціалізується на взаємодії з даними, які зберігаються у реляційних базах даних або обробці потокових даних в системах керування потоками даних. Ця мова особливо корисна при роботі з

даними, які мають структуровану форму, включають в себе взаємозв’язки між об’єктами та змінними.

Відповідно до [26] реляційна модель бази даних організовує дані шляхом їх структурування у вигляді однієї або декількох таблиць, що зображена на рис.2.5, де кожна таблиця має стовпці та рядки. Кожен рядок має унікальний ключ, який ідентифікує його. Рядки також називаються записами або кортежами. Стовпці ж відомі як атрибути. Зазвичай, кожна таблиця представляє один "тип сутності," такий як клієнт або продукт. Рядки в таких таблицях відображають конкретні екземпляри цих сутностей, наприклад, імена, або назви, а стовпці містять значення, пов’язані з цими екземплярами, наприклад, адресу або ціну.

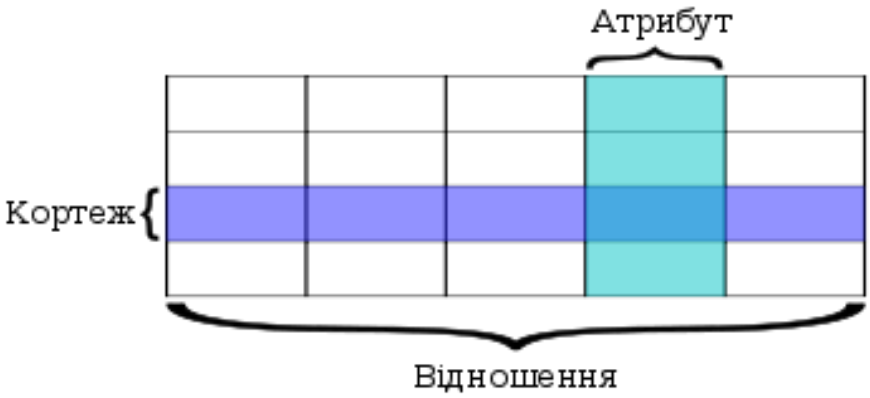

Рис.2.5 Умовна схема реляційної бази даних

Згідно [27] систему керування реляційною базою даних по іншому можна назвати системо керування базою даних, побудованою на основі реляційної моделі. Більшість сучасних баз даних базуються саме на цій моделі. Реляційні системи управління базами даних вже давно є популярним вибором для зберігання інформації в базах даних, які включають в себе фінансові записи, виробничу та матеріально-технічну інформацію, дані про персонал та інше.

Реляційні бази даних стали альтернативою застарілим ієрархічним та мережевим базам даних, оскільки RDBMS було простіше впроваджувати та адмініструвати.

Проте, завдяки поширенню технологій, таких як горизонтальне масштабування комп'ютерних кластерів, бази даних NoSQL стали популярними альтернативами реляційним базам даних.

Функціональні можливості, які надає DBMS, можуть суттєво варіюватися. Основною функціональністю є можливість зберігання, пошуку та оновлення даних. Наприклад схему роботи RDBMS можна побачити на рис.2.6.

Коннолі пропонує [28] наступний перелік функцій та сервісів, які повинна надавати повноцінна СУБД загального призначення:

- Зберігання, пошук і оновлення даних.
- Наявність для користувача каталогу або словника даних, який містить метадані та опис об'єктів бази даних.
- Підтримка транзакцій та можливість паралельної обробки запитів.
- Можливість відновлення бази даних в разі її пошкодження або втрати.
- Механізми авторизації доступу та оновлення даних з метою забезпечення безпеки.
- Підтримка можливості отримання доступу до бази даних з віддалених місць.
- Можливість застосування обмежень для забезпечення відповідності даних певним правилам.

Крім того, очікується, що СУБД надаватиме набір корисних підпрограм, необхідних для ефективного адміністрування бази даних. Ці утиліти включають в себе інструменти для імпорту та експорту даних, моніторингу використання ресурсів, дефрагментації та аналізу бази даних. Основна частина СУБД, яка взаємодіє між базою даних та інтерфейсом програми, іноді називається механізмом бази даних.

Часто СУБД також мають параметри конфігурації, які можна налаштовувати як статично, так і динамічно, наприклад, обсяг пам'яті на сервері, який може бути використаний базою даних. Тенденція полягає в автоматизації

конфігурації, а для вбудованих баз даних важливим є нульовий рівень адміністрування.

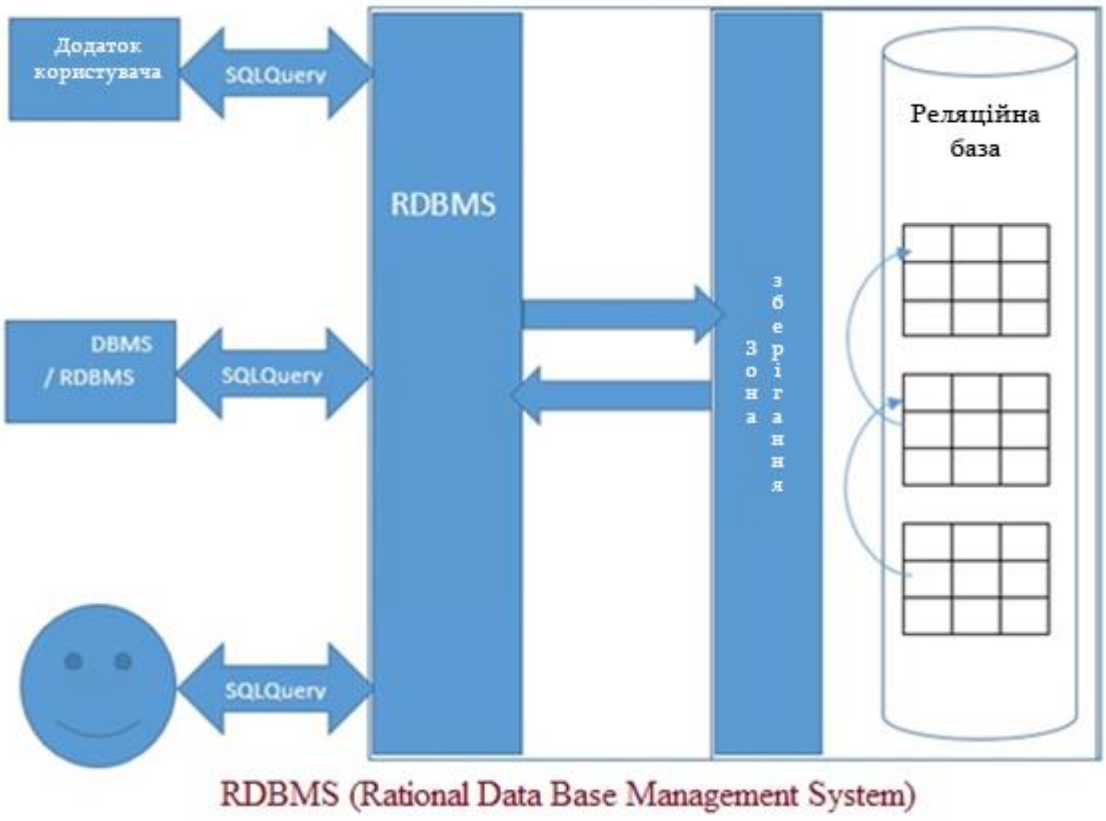

Рис.2.6 Схема роботи системи управління реляційної бази даних [29]

Переваги SQL включають в себе:

- Незалежність від конкретної системи управління базами даних. Більшість SQL-запитів, включаючи запити на створення та модифікацію даних, можуть бути легко перенесені з однієї СУБД на іншу, навіть з урахуванням різниці в діалектах та синтаксисі. Деякі системи з самого спочатку розроблені для використання з різними СУБД. Проте, при специфічних характеристиках моделі даних перенесення на рівні СУБД може бути важким завданням.
- Наявність стандартів. Існує стандарт, що стосується синтаксису та сумісності. Набори тестів для визначення сумісності та відповідності конкретної реалізації SQL загальноприйнятому стандарту дозволяє зробити мову більш універсальною. Проте, слід зауважити, що сам стандарт може бути надто формалізованим і об’ємним.

- Декларативність. SQL є декларативною мовою. Тобто програміст описує лише дані, які потрібно отримати або змінити. Точні дії виконує СУБД при обробці SQL-запиту. Цей принцип не є абсолютно універсальним, оскільки програмісту корисно знати, як СУБД інтерпретує його запит, особливо при роботі з великими базами даних та складними запитами. Складні запити можуть мати багато можливих реалізацій з різною швидкістю та з різним використанням ресурсів, але з однаковим результатом.
- Транзакції. SQL мови підтримують транзакції, що означає підтримку принципу ACID, тобто можливість неперервного запису при виконанні декількох запитів поспіль.

До недоліків SQL можна віднести:

- Складний інтерфейс: SQL має складний інтерфейс, що може викликати незручності для деяких користувачів при роботі з базою даних.
- Складність у налаштуванні та керуванні: SQL бази даних можуть бути складними для налаштування та управління, тому для забезпечення оптимальної продуктивності та збереження цілісності даних необхідні кваліфіковані адміністратори баз даних.
- Висока вартість: Деякі версії SQL є дорогими, тому програмісти можуть не мати можливості отримати до них доступ.
- Обмежений контроль: Бізнес-правила встановлені розробником можуть приховувати повний контроль над базою даних.
- Відсутність реального часу для аналітики: Бази даних SQL призначені для пакетної обробки та не підтримують аналітику в реальному часі, що може бути недоліком для програм, які потребують обробки даних в реальному часі.
- Менша гнучкість: SQL бази даних менш гнучкі у порівнянні з NoSQL базами даних, коли мова йде про обробку неструктурованих даних. Це відбувається через вимогу до структурованості даних у таблицях і стовпцях.
- Обмежена продуктивність запитів: SQL бази даних можуть виявити обмежену продуктивність при роботі з великими обсягами даних, оскільки

запити можуть потребувати більше часу для обробки, ніж нереляційні бази даних, що функціонують у пам’яті.

З огляду на розглянуті переваги та недоліки, очевидним є вибір реляційної бази даних та SQL мови запитів як основи для моделі в архітектурі MVC для вирішення задачі розробки централізованої системи для віддаленого управління складськими приміщеннями.

2.3. Авторизація

Оскільки поставленою задачею є створення віддаленої системи, тому додавання авторизації в систему є важливим кроком, який є необхідним для забезпечення безпеки та контролю доступу до функціоналу та даних системи. Авторизація визначає, які користувачі мають право увійти до додатку та який обсяг доступу до різних його функцій та ресурсів вони отримають.

Очевидним є рішення про створення користувачів, проте через віддаленість системи важливо подбати про безпеку збереження даних користувача.

Зберігання паролю в чистому вигляді, або в лише захешованому не є оптимальним. Спочатку зловмисник може спробувати провести атаку за допомогою словника. Він використовує заздалегідь підготовлений список слів, такий як записи з англійського словника, разом із їхніми обчисленими хешами, і порівнює кожен хеш зі списком хешів, які були витягнуті з таблиці вкрадених паролів. Якщо відбувається збіг, зловмиснику вдається розгадати пароль.

Через це, як запропоновано в [30], важливо додати ще один рівень захисту – сіль. За допомогою прийому, який включає в себе додавання випадкової послідовності даних до пароля перед його обробкою алгоритмом хешування, пароль стає унікальним і складнішим для зламу, як видно з рис.2.7.

Коли і якщо користувачі вводять однаковий пароль та використовується як хешування, так і додавання солі, кожному паролю додаються випадкові символи. В результаті генеруються різні хеші, що допомагає запобігти можливому злому паролів та облікових записів.

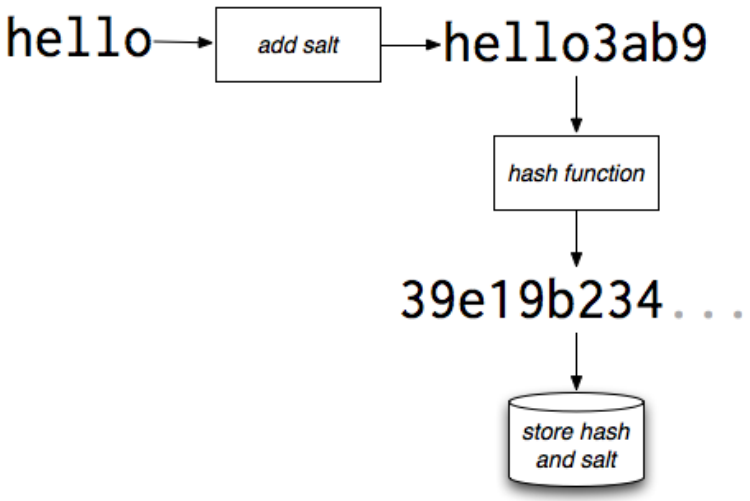

Рис.2.7 Концепція додавання солі перед хешуванням [31]

Для власне валідації користувача збережена в базі даних сіль додається до введеного паролю та результат хешування цієї комбінації порівнюється з хешем збереженим у базі даних. У випадку співпадіння користувачу надається доступ до ресурсу.

2.4. Розробка схеми бази даних моделі

Розробка схеми бази даних є важливою частиною створення централізованої системи для віддаленого управління складськими приміщеннями. Цей процес передбачає створення структури, яка визначає, як дані будуть зберігатися, як вони організовані та як взаємодіють один з одним. Особливо важливо створити схему з урахуванням того факту, що обрано модель архітектури MVC. Тобто модель, створена на основі цієї схеми має мати відображення у реальності.

При створенні схеми користувача, зображеної на рис.2.8, логічним є додавання його ролі, адже створений аккаунт має мати різний рівень доступу як для безпеки даних так і для зручності користування системою.

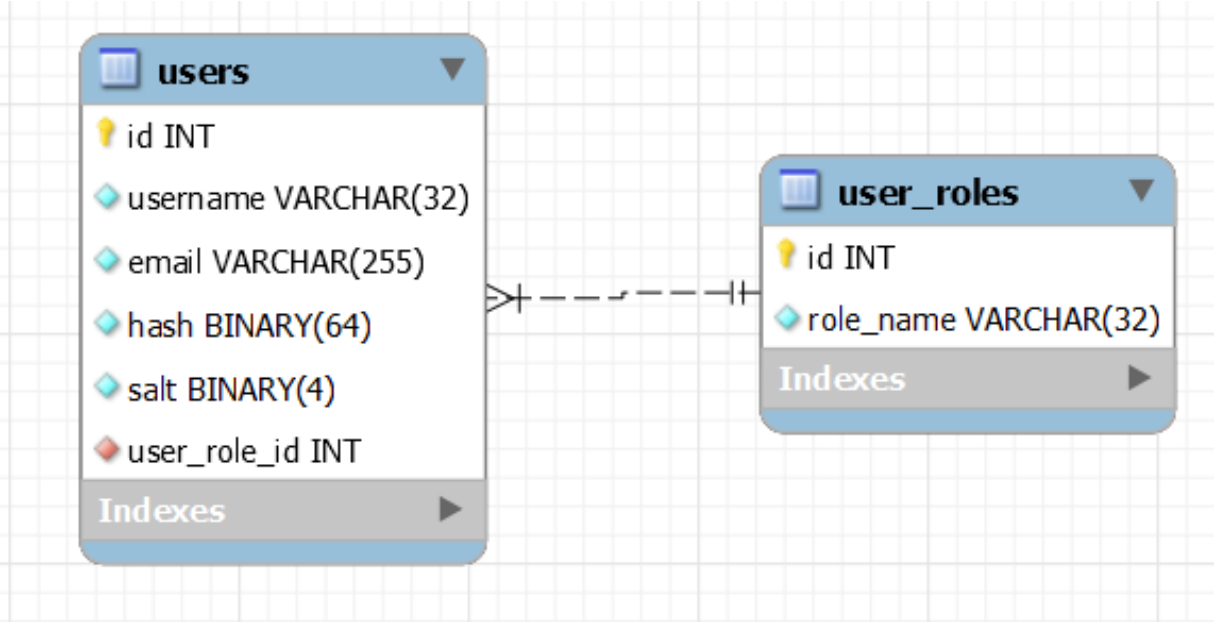

Рис.2.8 Схема моделі користувача

Пропонується створити таблицю з ролями, детальніше описану в табл. 2.1, яка міститиме назву ролі. Власне поведінкою в залежності від ролі користувача опікуватиметься контролер в моделі MVC

Таблиця 2.1.

Опис таблиці ролей користувачів в базі даних

Кортеж	Тип атрибуту	Опис
id	INT	Унікальний ідентифікатор ролі
role_name	VARCHAR(32)	Назва ролі, на основі якої відбувається керування доступом

Таблиця власне користувач, описана в табл. 2.2, мусить містити окрім його даних також посилання на його роль сіль і хеш, що необхідні для авторизації з алгоритмом використання солі та хешування.

Таблиця 2.2

Опис таблиці користувача в базі даних

Кортеж	Тип атрибуту	Опис
id	INT	Унікальний ідентифікатор користувача
username	VARCHAR(32)	Ім’я користувача, яке він вводить під час реєстрації
email	VARCHAR(255)	Поштова адреса користувача, яку він вводить під час реєстрації
hash	BINARY(64)	Хеш, що базується на захешованому під час реєстрації паролі
salt	BINARY(4)	Сіль, використана під час хешування паролю під час реєстрації
user_role_id	INT	Посилання на роль користувача

Модель складу, зображена на рис.2.9, містить таблиці зі складами, предметами, приналежністю предмету до складу.

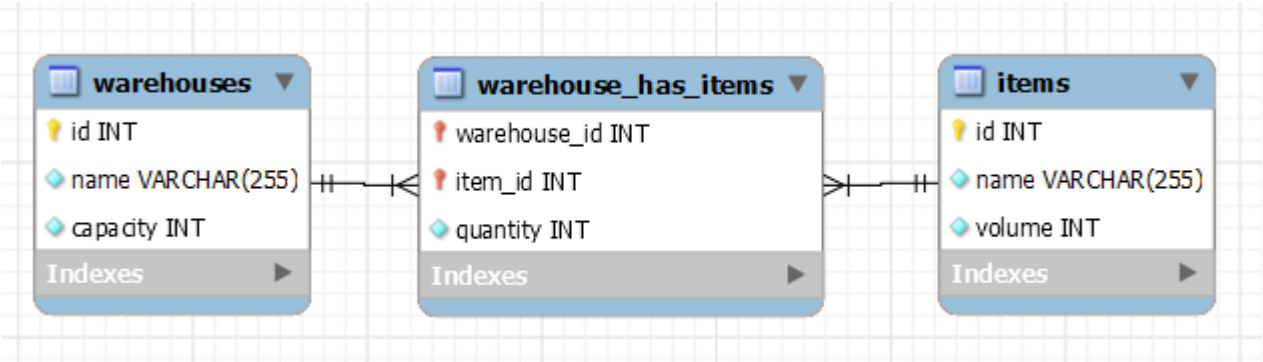

Рис.2.9 Схема моделі складу

Важливо звернути увагу на таблицю приналежності предмету до складу, детальніше описану в табл. 2.3.

Таблиця 2.3

Опис таблиці належності предмету до складу в базі даних

Кортеж	Тип атрибуту	Опис
warehouse_id	INT	Унікальний ідентифікатор пов’язаного складу
item_id	INT	Унікальний ідентифікатор пов’язаного предмету
quantity	INT	Кількість предметів на складі

Метою специфічної структури цієї моделі є відслідковування змін пов’язаних з додаванням або видаленням предметів до інвентарю складського приміщення.

Важливим є також додавання історії змін складу, що можна досягти за рахунок моделі історії, зображеної на рис.2.10.

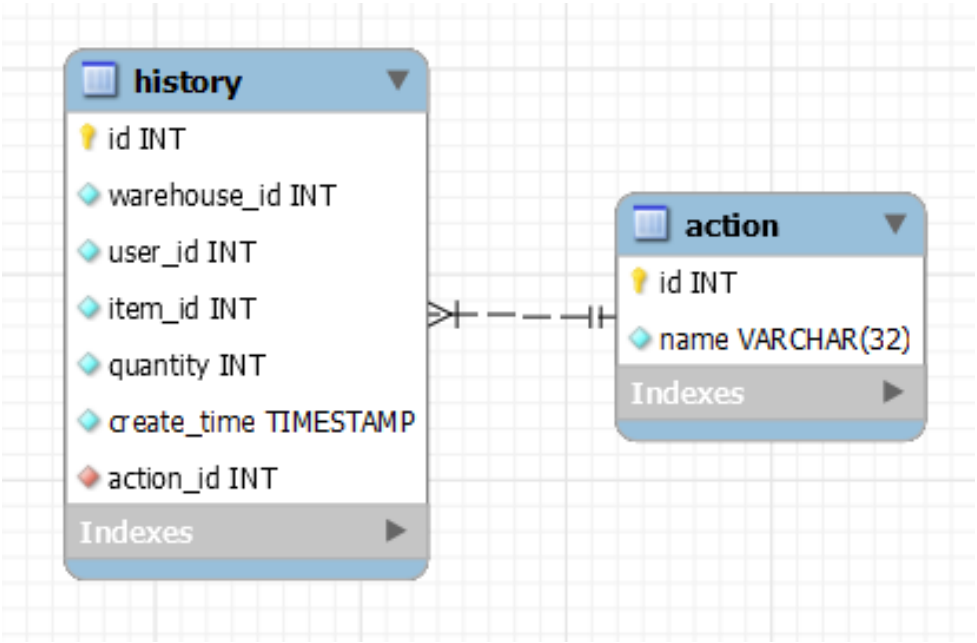

Рис.2.10 Схема моделі історії

Схема містить дві таблиці: власне історію та таблицю можливих дій, пов’язаних з додаванням відповідного запису до історії. Детальніше таблицю історії розглянуто в табл. 2.4.

Таблиця 2.4

Опис таблиці історії в базі даних

Кортеж	Тип атрибуту	Опис
id	INT	Унікальний ідентифікатор запису
warehouse_id	INT	Унікальний ідентифікатор пов’язаного складу
user_id	INT	Унікальний ідентифікатор користувача
item_id	INT	Унікальний ідентифікатор предмету, що зазнав змін
quantity	INT	Кількість предметів, яка зазнала змін
create_time	TIMESTAMP	Час додавання запису до історії
action_id	INT	Унікальний ідентифікатор дії, пов’язаної з додаванням запису

Організація схеми складу в базі даних, де кожен склад та його дані обробляються і зберігаються в єдиній структурі, дозволяє здійснювати централізоване управління низкою складських приміщень. В цей же час наявність користувачів з відповідними ролями, які мають різні рівні доступу, допомагає забезпечити безпеку доступу до даних в системі.

Висновки до розділу 2

У цьому розділі було проведено поетапну розробку моделі централізованої системи для віддаленого управління складськими приміщеннями, яка вирішує поставлені у роботі завдання. А саме процес розробки включив наступні кроки:

1. Розглянуто можливі для реалізації архітектури системи, та на основі розглянутих переваг і недоліків обґрунтовано вибір клієнт-серверної MVC архітектури. Цей вибір пояснюється тим, що він дозволяє ефективно розділити відповідальності між компонентами системи, забезпечуючи легкість розширення та обслуговування.

2. Під час аналізу можливих баз даних, були детально розглянуті реляційні та NoSQL бази даних. Враховуючи їх сильні та слабкі сторони обрано базу даних для реалізації моделі в архітектурі MVC. Власне обрано реляційну базу даних з SQL. Ключовим фактором цього вибору стала підтримка транзакцій для забезпечення ACID принципу під час множинних запитів до бази даних.

3. Розглянуто важливий для централізованої системи віддаленого управління процес авторизації. Власне обґрунтовано необхідність використання системи шифрування паролів користувачів шляхом хешування з додаванням солі.

4. На основі обраної бази даних розроблено схему бази даних для моделі централізованої системи для віддаленого управління складськими приміщеннями. Було детально описано найбільш важливі та нестандартні таблиці присутні у цій схемі.

РОЗДІЛ 3

СТВОРЕННЯ РОЗРОБЛЕНОЇ МОДЕЛІ

3.1. Вимоги до програмного продукту

Для демонстрації роботи запропонованої системи, було створено модель централізованої системи для віддаленого управління складськими приміщеннями. Розроблена програма має бути проста, зі слушною структурою, зрозуміла у використанні для користувача та забезпечувати виконання наступних завдань:

1. Можливість реєстрації та подальшої авторизації, обмеження доступу для неавторизованих користувачів.
2. Додавання та видалення предметів до бази даних через інтерфейс.
3. Додавання та видалення складів до бази даних через інтерфейс.
4. Перегляд інформації про окремий склад.
5. Можливість додавати та змінювати кількість предметів на складі.
6. Можливість видаляти склади та предмети
7. Можливість переслати предмети з одного складу до іншого.
8. Облік ємності складу.
9. Ведення історії операцій та можливість її перегляду.

Для виконання поставленого завдання було обрано вебсервер Apache Tomcat версії 10.1.2. Цей тип серверу реалізує специфікацію сервлетів і підтримує JSP. Власне ця версія серверу підтримує специфікацію сервлетів – 6.0, JSP – 3.1, EL – 5.0 та середовище виконання Java версії 11 та вище. Для написання JSP сторінок, та внутрішньої структури використано мову програмування Java специфікації 11.

JSP – це метод для створення веб-сторінок, що забезпечує можливість генерації HTML, XML та інших веб-елементів цих сторінок у реальному часі. Ця технологія дозволяє вставляти код Java у статичний вміст сторінок. Також використовуються бібліотеки JSP-тегів для вбудовування їх у JSP-сторінки.

Сторінки JSP компілюються у сервлети за допомогою JSP-компілятора, що перетворює їх у класи Java. В подальшому ці класи компілюються в байт-код та виконуються на сервері в середовищі віртуальної машини Java.

Для гнучкого логування стану системи використано фреймворк log4j версії 2.19.

Для відображення використано Bootstrap 5 – фреймворк який містить шаблони CSS та HTML для форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript.

Для коректної роботи програми, було визначено наступні мінімальні характеристики ПК:

- Двох-ядерний процесор з тактовою частотою 1ГГц
- 2ГБ RAM
- Вільне місце на диску 500Мб, або більше
- Операційна система Windows 10 або новіше

3.2. Структура програмного продукту

Структуру проекту перед компіляцією для розміщення на вебсервері Tomcat можна побачити на рис.3.1.

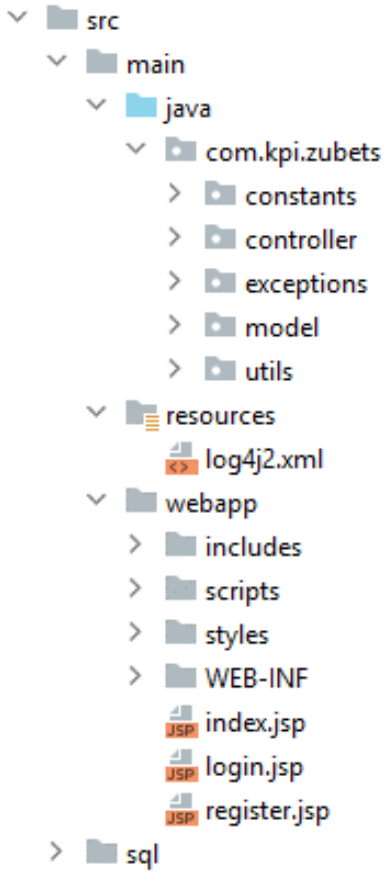

Рис.3.1 Структура проекту

Java код, що має виконуватися на сервері розміщено в теці main/java.*. В теці main/resources розміщено файл для налаштування інструменту для логування.

В теці sql розміщено скрипти для створення схеми бази даних, заповнення статичного контенту та допоміжні скрипти для генерації динамічного контенту.

Згідно з запропонованою моделлю використано архітектуру MVC, де власне моделлю є таблиці в базі даних, їх зв’язки та вміст, представлення – це JSP сторінки, стилі, та JavaScript файли, а контролер – код серверу, що оперує моделлю та надає відображення користувачу.

Цю архітектуру відображено в структурі проекту, а саме: модель знаходиться в теці main/java.com.kpi.zubets.model.*, контролер в теці main/java.com.kpi.zubets.controller.*, а представлення в теці main/webapp/*.

3.2.1. Модель

Оскільки для роботи з базою даних використовується JDBC, який дозволяє лише надсилати запити до бази даних, але не опікується її структурою, тому для роботи з моделлю використано шаблон проектування DAO.

Відповідно до [32], об’єкт доступу до даних представляє собою шаблон, що утворює абстрактний спосіб взаємодії з певним типом бази даних або іншим механізмом зберігання інформації. DAO відображає виклики додатків на власне рівень бази даних. Це дозволяє виконувати операції з даними без викриття конкретних деталей реалізації на рівні бази даних. Ця ізоляція відокремлює процес доступу до даних до предметно-орієнтованих об’єктів і типів даних, які описуються в публічному інтерфейсі DAO. Це дозволяє приховати як саме задовольняються потреби DAO певною системою управління базами даних.

Використання об’єктів доступу до даних має значну перевагу у виключенні прямих зв'язків між двома частинами програми, які можуть функціонувати незалежно одна від одної без знання подробиць роботи іншої. Це дозволяє незалежну зміну цих частин програми без взаємного впливу. Якщо вносяться

зміни у бізнес-логіку, це може стосуватися лише інтерфейсу DAO, тоді як зміни в логіці виконання на рівні бази даних не впливають на користувачів DAO.

У сфері програмування на Java реалізація DAO може бути виконана по-різному. Це може бути простий інтерфейс, що відділяє доступ до даних від логіки програми, або використання фреймворків та комерційних рішень.

Для створення системи обрано підхід використання інтерфейсів. Як можна побачити на рис. 3.2, відповідні інтерфейси DAO знаходяться в теці main/java.com.kpi.zubets.model.dao

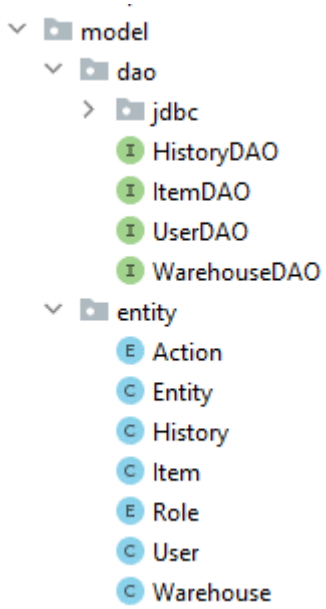

Рис.3.2 Структура моделі в проекті

Ці інтерфейси опікуються об’єктами-відображеннями, які знаходяться в теці main/java.com.kpi.zubets.model.entity.*, а власне логіка роботи JDBC знаходиться в теці main/java.com.kpi.zubets.model.dao.jdbc.*

3.2.2. Контролер

При створенні контролера, структуру якого зображено на рис. 3.3, використано шаблон програмування Command.

Шаблон command є способом проектування, де основний об’єкт команди містить усю необхідну інформацію для виконання певної дії у майбутньому. Ця

інформація включає назву методу, об’єкт, до якого відноситься метод, та значення параметрів методу.

Згідно до [33], у цьому шаблоні проектування є чотири ключових елементи – це команда, отримувач, виклик та клієнт, як видно з рис.3.4. Об’єкт команди знає про отримувача та активує його метод. Значення параметрів методу отримувача зберігаються в команді. Також в об’єкті команди зберігається об’єкт отримувача, який виконує ці методи. Отримувач виконує дії при виклику методу .execute() в команді. Об’єкт виклику команди знає як виконати команду, але нічого не знає про конкретну команду, він взаємодіє лише з командним інтерфейсом. Об’єкт виклику, об’єкти команд і об’єкти отримувача зберігаються в об’єкті клієнта, який вирішує, які об’єкти отримувача пов’язати з об’єктами команд та які команди відправити на виконання.

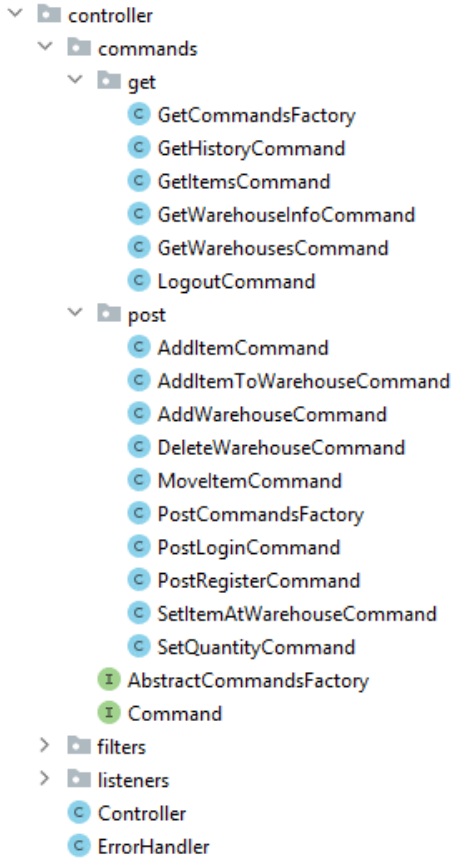

Рис.3.3 Структура контролера в проекті

Використання об’єктів команд спрощує створення загальних компонентів, які повинні виконувати дії чи викликати методи у певний момент часу, не маючи

необхідності знати конкретний клас методу чи його параметри. Об’єкт команди може реалізувати різні режими виконання команд, без необхідності витоку деталей виконання до клієнта.

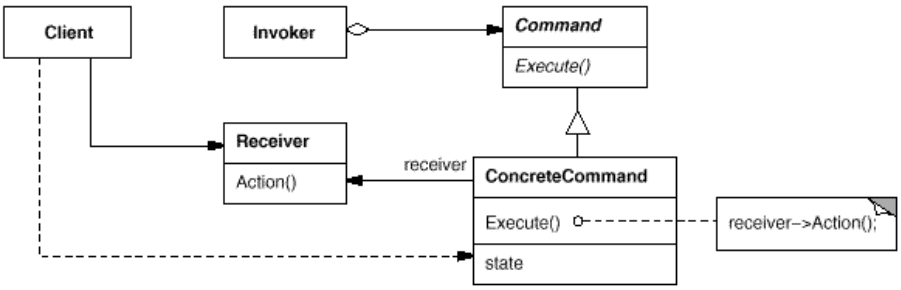

Рис.3.4 UML-діаграма шаблону Command [34]

Оскільки команди розділяються на два різних типи: post ta get з однаковими методами і структурою, то абсолютно логічним є також використання шаблону програмування Abstract Factory.

Як зазначено в [35], шаблон абстрактної фабрики – це метод, який дозволяє створювати групу пов'язаних об'єктів без прив'язки до конкретних класів. Це досягається шляхом створення окремих фабрик, які мають спільний контекст, у межах одного інтерфейсу. Коли клієнт використовує цей шаблон, він спочатку створює конкретну реалізацію абстрактної фабрики, а потім використовує загальний інтерфейс цієї фабрики для створення конкретних об'єктів, які належать до цієї групи. Клієнт не знає, які саме об'єкти він отримує від кожної внутрішньої фабрики, оскільки використовує лише загальні інтерфейси їх продуктів.

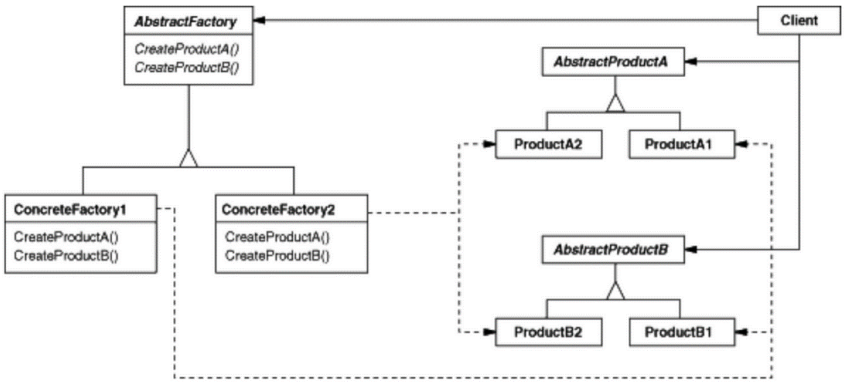

Рис.3.4 UML-діаграма шаблону AbstractFactory [36]

Цей шаблон дозволяє змінювати конкретні реалізації без внесення змін у код, який їх використовує. Проте його використання може призвести до зайвої складності та більшої роботи на етапі написання коду. Збільшення рівня абстракції також може зробити систему складнішою для розуміння та підтримки.

3.2.3. Представлення

Структура представлення, що зображена на рис 3.5, досить проста. Налаштування контейнеру сервлетів Tomcat знаходяться в файлі main/webapp/web.xml.

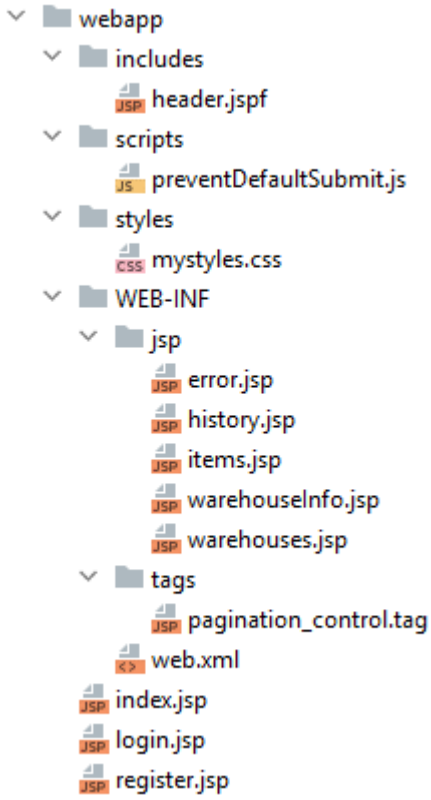

Рис.3.5 Структура представлення в проекті

JSP сторінки знаходяться в теках main/webapp/ та main/webapp/WEB-INF/jsp/*. Таке розділення необхідно для того щоб відокремити сторінки з різним доступом. JSP-фрагменти – файли JSP які не є самостійними сторінками, а використовуються для підключення до інших JSP сторінок, та мають розширення .jspf, знаходяться в теці main/webapp/includes/*. JavaScript скрипти знаходяться в теці main/webapp/scripts/*. CSS стилі в теці main/webapp/styles/*.

Файли з розширенням .tag, тобто користувацькі скриптовані елементи для використання в JSP знаходяться в теці main/webapp/WEB-INF/tags/*.

3.3. Розробка програмного продукту

3.3.1. Меню

Для зручності користування системою варто одразу подбати про меню, яким користувач буде проводити навігацію у системі.

По-перше, меню, зображене на рис.3.6, є JSP-фрагментом, тобто .jspf файлом. Це означає, що воно підключається до кожної окремої JSP -сторінки.

Рис.3.6 Меню системи в залежності від авторизації користувача

По-друге, в цей JSP-фрагмент закладена логіка перевірки користувача на авторизованість, що дозволяє відобразити відповідні пункти меню в залежності від того чи він зайшов у систему, чи ні.

Рис.3.7 Меню системи у закритому, відкритому вигляді та в залежності від авторизації користувача

По-третє, як можна побачити на рис 3.7, за допомогою фреймворку Bootstrap 5 додано підтримку для вікна при збільшенні його масштабу, або при використанні екрану з меншою роздільною здатністю.

3.3.2. Авторизація

Одним з основних елементів роботи системи є авторизація. У випадку, якщо користувач не є авторизованим, йому не будуть показані елементи меню, та він не зможе отримати доступ до сторінок, які вимагають статус авторизованого користувача.

Для того щоб стати авторизованим користувачем необхідно пройти реєстрацію та увійти до аккаунту.

У вікні реєстрації, зображеному на рис 3.8, необхідно ввести логін, пароль та пошту. Після чого спочатку на рівні клієнта відбуваються перевірки на правильність введених даних.

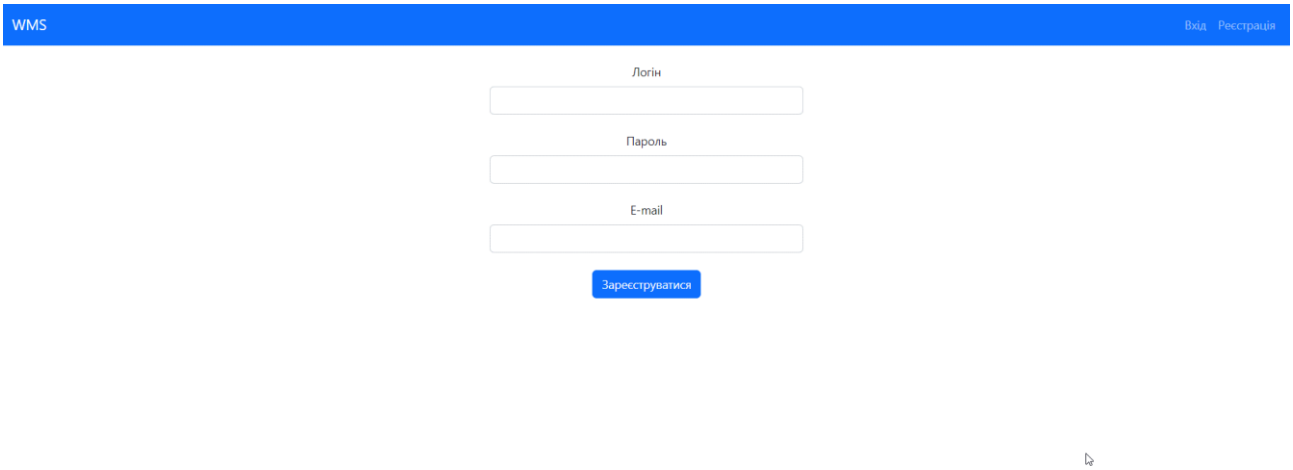

Рис.3.8 Вікно реєстрації

Якщо користувач неправильно ввів дані, йому відображаються підказки, зображені на рис.3.9. Якщо ж дані введені правильно, то форма відправляє запит на сервер, де відбувається повторна валідація даних та створюється запис в базі даних про користувача, у випадку проходження валідації.

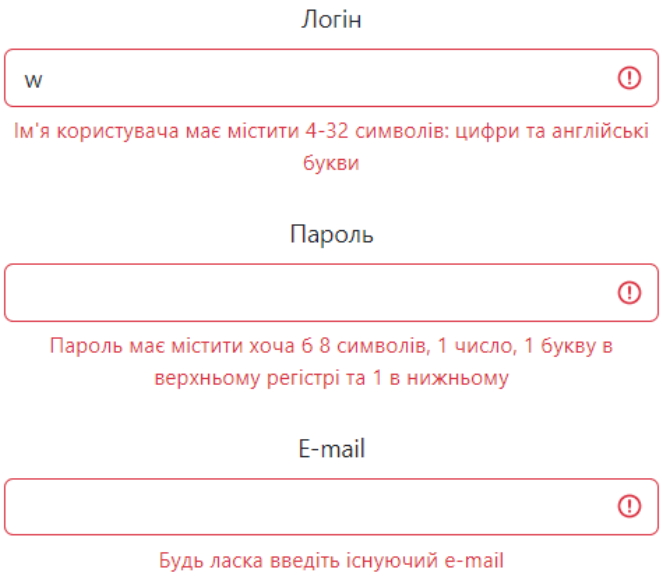

Рис.3.9 Форма реєстрації при некоректно введених даних

Варто зауважити що дані які поступають на сервер валідуються за допомогою допоміжного класу Validator, структуру якого можна побачити на рис.3.10. Цей клас містить методи для підтвердження проходження рядками перевірки необхідних умов. В першу чергу ці умови стосуються формату збереження даних в базі даних, тому дуже важливо перевірити їх виконання.

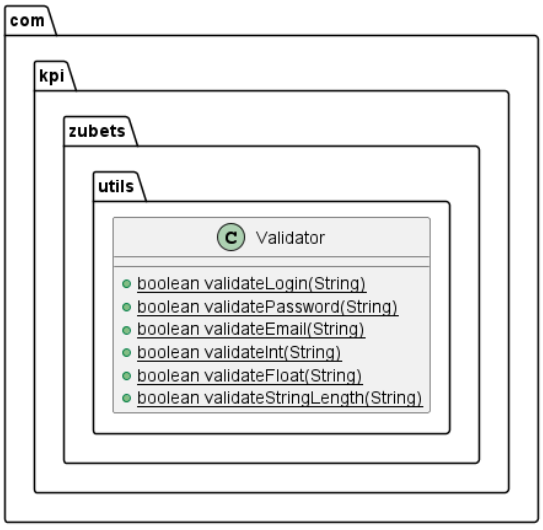

Рис.3.10 UML-діаграма класу Validator

У відповідності до методів створення запропонованої моделі пароль не зберігається в базі даних у чистому вигляді. А саме відбувається його хешування з додаванням солі. Цим процесом опікується клас Passwords, зображений на

рис.3.11. Його методи дозволяють згенерувати сіль, захешувати пароль та перевірити відповідність паролю шляхом хешування введеного паролю з додаванням солі.

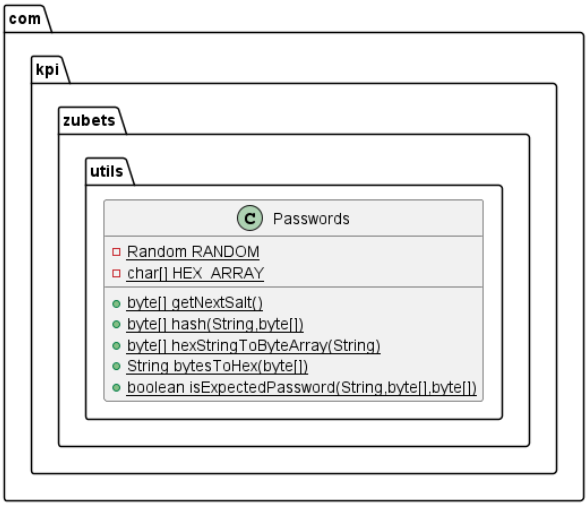

Рис.3.11 UML-діаграма класу Passwords

У випадку успішного створення нового користувача, відбувається переадресація клієнта на вікно входу, зображеного на рис. 3.12.

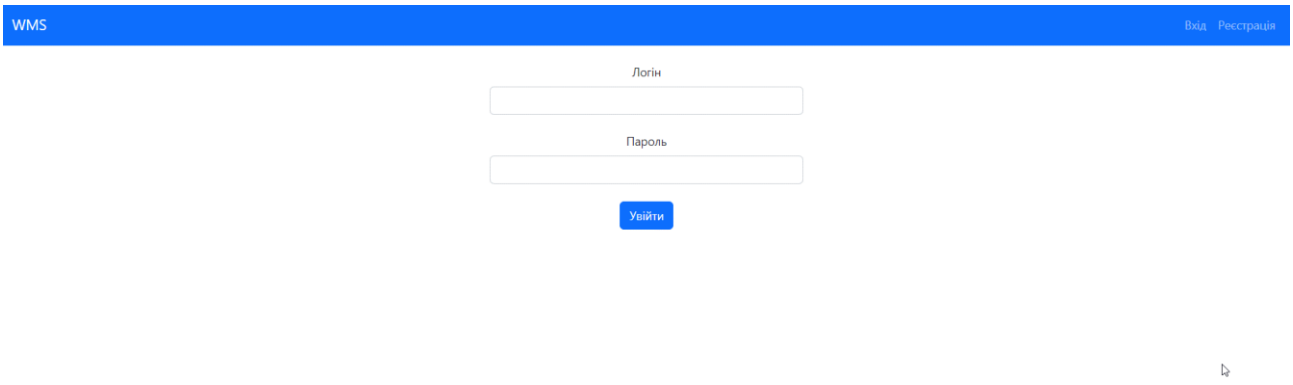

Рис.3.12 Вікно входу в систему

В цьому випадку перевірка на валідність введених даних перевіряється лише на сервері з використанням тих же класів Validator та Passwords. У випадку

успішного підтвердження даних користувачу відкривається сесія та відбувається переадресація на головну сторінку програми.

Якщо ж дані введені неправильно, то відбувається переадресація на сторінку входу з відображенням інформації про некоректно введені дані, що зображено на рис. 3.13.

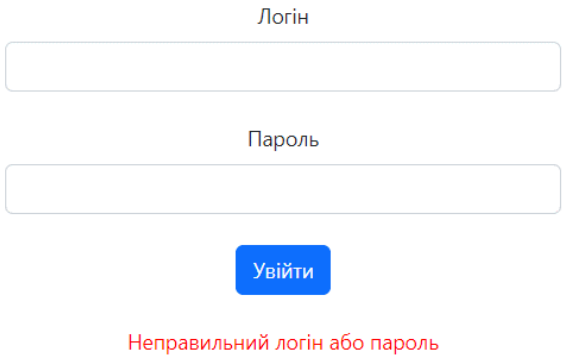

Рис.3.13 Форма входу при некоректних даних

З точки зору перевірки авторизованості на сервері, тобто логіки контролера для обмеження доступу до сторінок системи, недоступних для неавторизованих користувачів, основним елементом є клас ControllerFilter, зображений на рис.3.14.

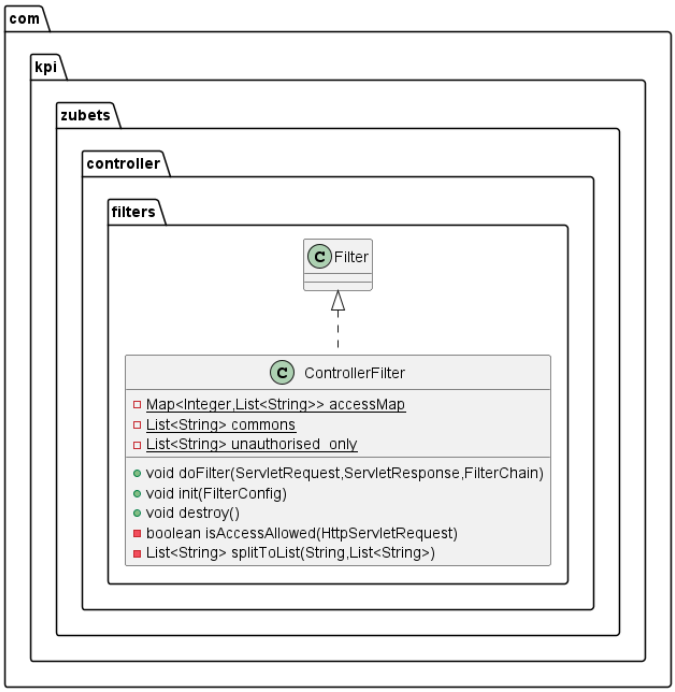

Рис.3.14 UML-діаграма класу ControllerFilter

Цей клас реалізує інтерфейс Filter, який є складовою Tomcat, за допомогою нього можна у відповідності до заданого шаблону шляху виконувати певні дії над запитом користувача, до його обробки контролером.

У випадку саме цього фільтру перевіряється власне наявність доступу користувача до різних команд та сторінок в залежності від його рівня авторизації.

3.3.3. Пагінація

Оскільки сторінки з динамічною кількістю елементів можуть містити їх велику кількість, то важливо додати пагінацію, задля адекватного відображення вмісту сторінки та зменшення навантаження на мережу.

У веб-дизайні пагінація використовується для поділу великої кількості контенту на декілька окремих сторінок, щоб полегшити навігацію користувачам. Зазвичай це представлено у вигляді набору пронумерованих кнопок, які дозволяють переходити між сторінками. Наприклад, на веб-сайтах із списками товарів або результатами пошуку, пагінація відображає сторінки 1, 2, 3 і так далі, дозволяючи користувачам переміщатись між різними частинами списку.

Пагінацію, зображену на рис.3.15, реалізовано на сервері з використанням користувацьких тегів для JSP. Подібний підхід дозволяє використати цю технологію на різних сторінках системи з їх різним вмістом та наповненням.

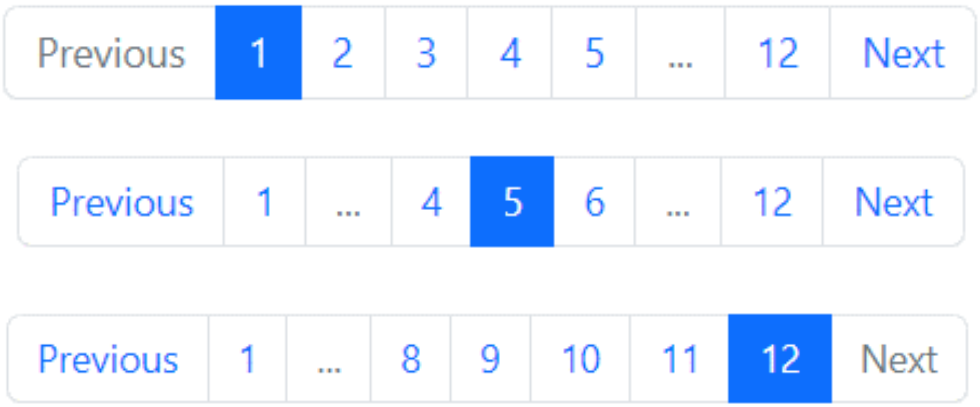

Рис.3.15 Модуль пагінації

Пагінацію, зображену на рис.3.15, реалізовано на сервері з використанням користувацьких тегів та Bootstrap 5. Подібний підхід дозволяє використати цю технологію на різних сторінках додатку з їх різним вмістом та наповненням.

Необхідні для обчислень значення, а саме номер поточної сторінки, та загальна кількість сторінок є атрибутами запиту, які надає контролер, а атрибут тегу вказує яку команду необхідно встановити на гіперпосиланнях.

3.3.4. Сторінка предметів

При відображенні інформації про наявні в базі даних предмети, варто взяти до уваги поля цієї бази даних, а саме id предмету, його назву та ємність одиниці. Тому подання цієї інформації у вигляді таблиці, зображеної на рис.3.17, є досить влучним підходом, беручи до уваги незначну кількість полів в базі даних.

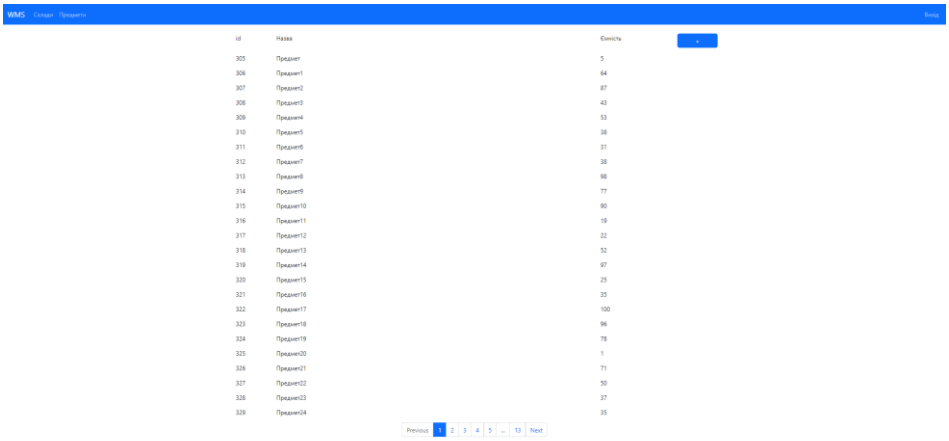

Рис.3.17 Сторінка предметів

Для додавання предметів до бази даних наявна кнопка, яка викликає модальне вікно, зображене на рис.3.18.

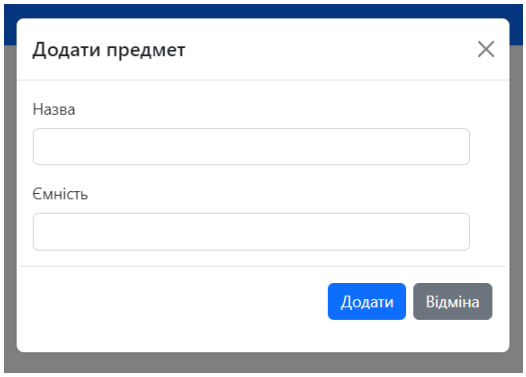

Рис.3.18 Вікно додавання предмету

При введені даних на стороні користувача відбувається їх валідація і у випадку неправильно введених даних відображаються контекстні підказки, зображені на рис.3.19.

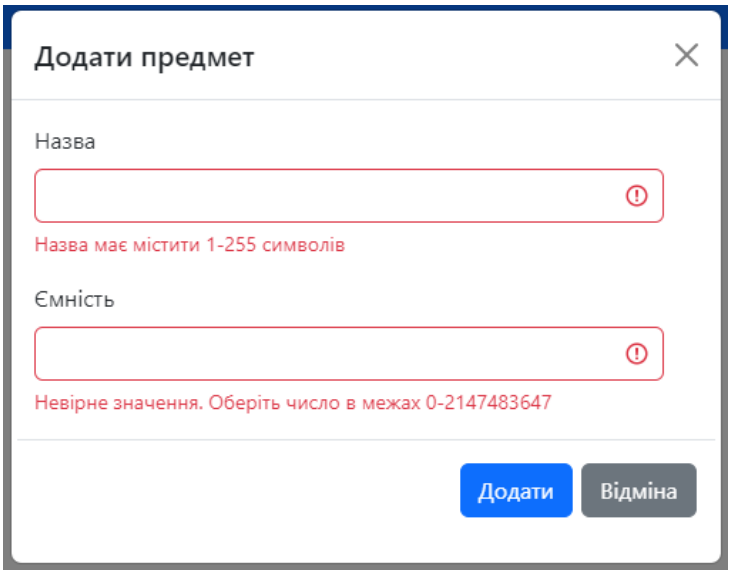

Рис.3.19 Вікно додавання предмету з підказками

Якщо ж валідація пройшла успішно як у клієнта так і на сервері, то за допомогою класу JdbcItemDAOImpl – DAO, що стосується предметів, який зображено на рис 3.20, контролер створює новий запис в базі даних.

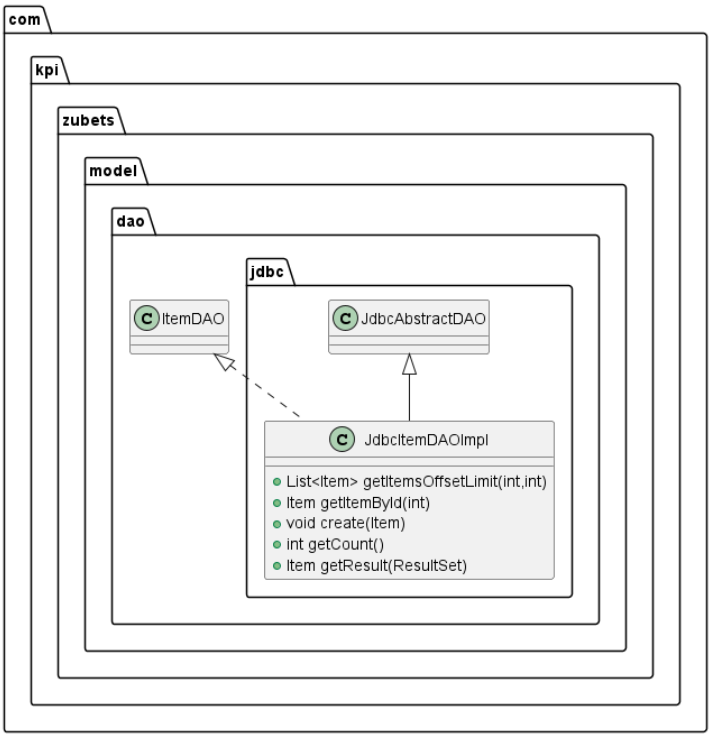

Рис.3.20 UML-діаграма класу JdbcItemDAOImpl

3.3.5. Сторінка складів

Оскільки інформації про склади більше ніж про предмети, то подача інформації про них у вигляді таблиці не є доцільною. Через це склади відображені у вигляді карток, що зображено на рис.3.21.

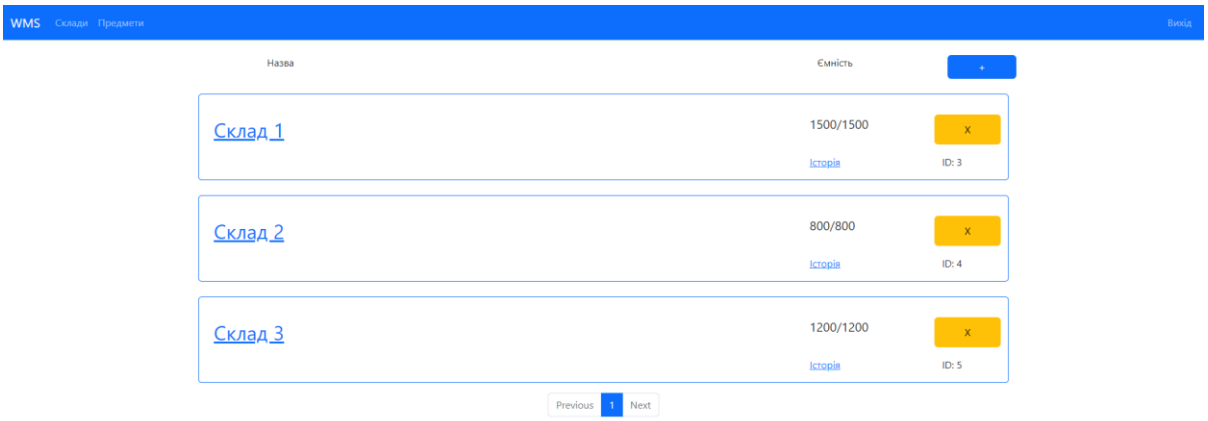

Рис.3.21 Сторінка складів

Аналогічно сторінці з предметам, кнопка додавання складу викликає модальне меню, зображене на рис.3.22.

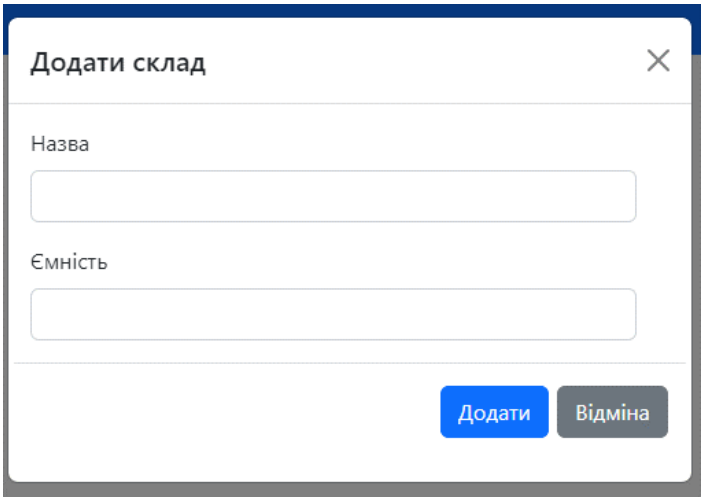

Рис.3.22 Вікно додавання складу

Оскільки видалення складу це незворотна дія, яка видаляє як склад так і його історію, то вона потребує додаткового підтвердження. Через це кнопка видалення викликає модальне меню, зображене на рис.3.23.

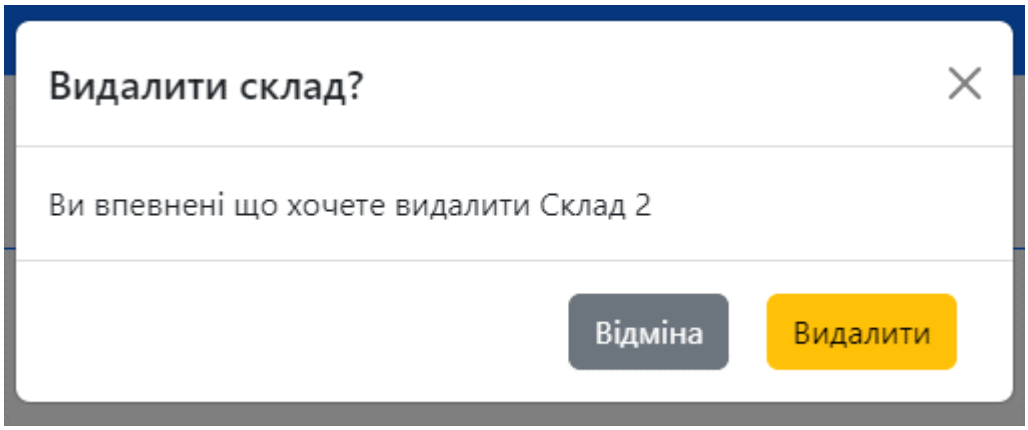

Рис.3.23 Вікно підтвердження видалення складу

Усі операції, пов’язані з моделлю складу знаходяться у відповідному класі, який реалізує інтерфейс WarehouseDAO.

3.3.6. Сторінка інформації про склад

Сторінка інформації про конкретний склад – це сторінка з найбільшим насиченням, яка потребує багатьох кнопок і дій пов’язаних з предметами, що знаходяться на складі, тому тут теж є логічним використання карток, що зображено на рис.3.24.

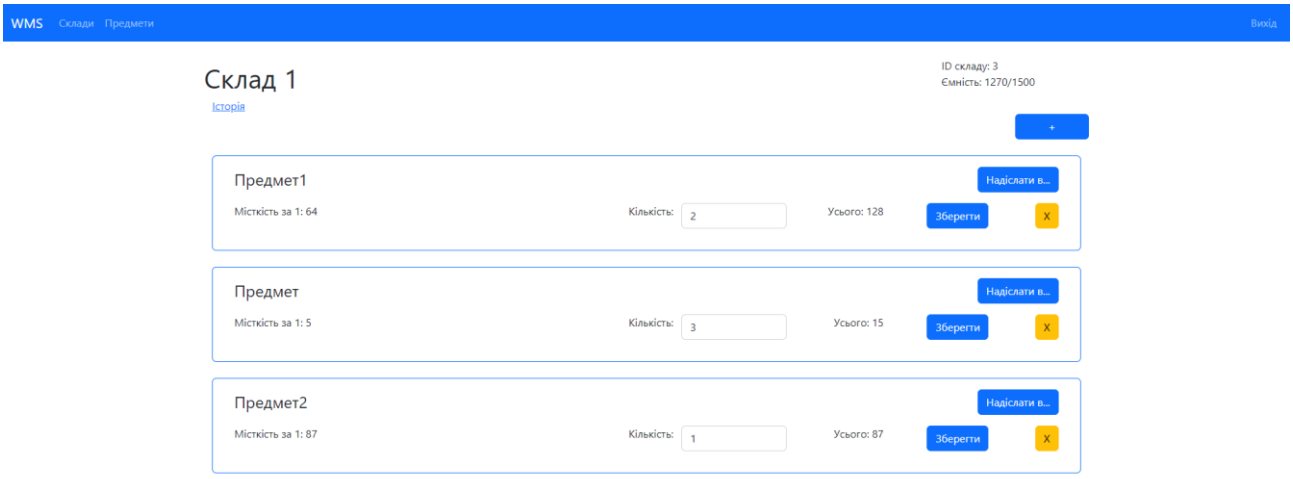

Рис.3.24 Сторінка інформації про склад

Додавання предметів на склад вирішено оформити не у вигляді модального вікна, а у вигляді меню, яке розгортається, зображеного на рис.3.25, щоб користувач мав змогу бачити вже наявні предмети на складі, при рішенні що саме додати до складу.

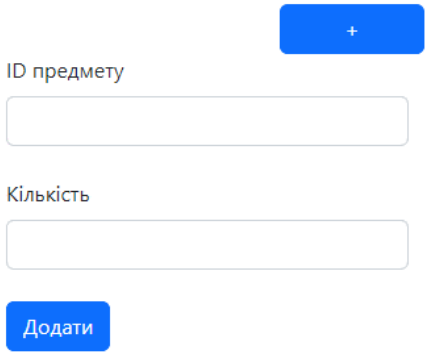

Рис.3.25 Меню додавання предметів на склад

Видалення предмету зі складу також потребує підтвердження, яке зображено на рис.3.26, через його незворотність.

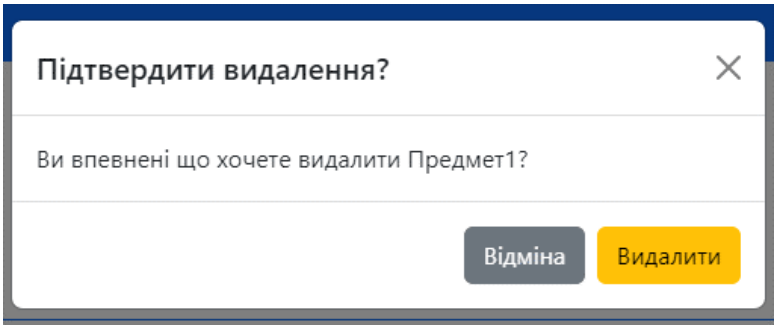

Рис.3.26 Підтвердження видалення предмету зі складу

Якщо розглянути картку предмету на складі, зображену на рис.3.27, більш детально, то кнопка «Надіслати в…» розгортає додаткове меню з випадаючим списком складів, де можна обрати ціль надсилання. Також важливим є рішення мати можливість змінити кількість уже доданих предметів на складі.

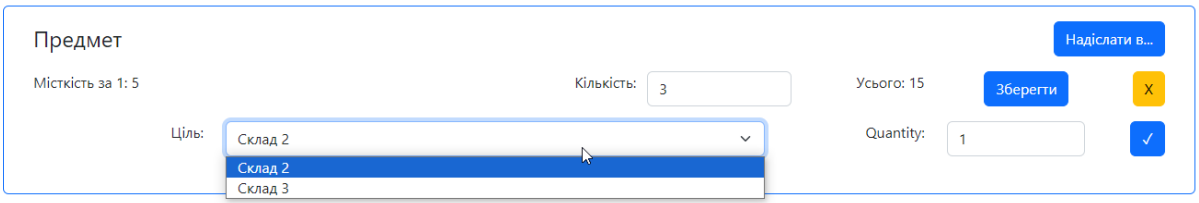

Рис.3.27 Меню пересилання предмету з одного складу на інший

3.3.7. Сторінка історії складу

Сторінка історії складу має відображати історію складу, а саме операцію, користувача, предмет, його кількість та час створення запису. Оскільки логічним є представлення цієї інформації відсортованої за часом створення запису, то найбільш оптимальним є рішення відображення історії у вигляді таблиці, зображеної на рис. 3.28.

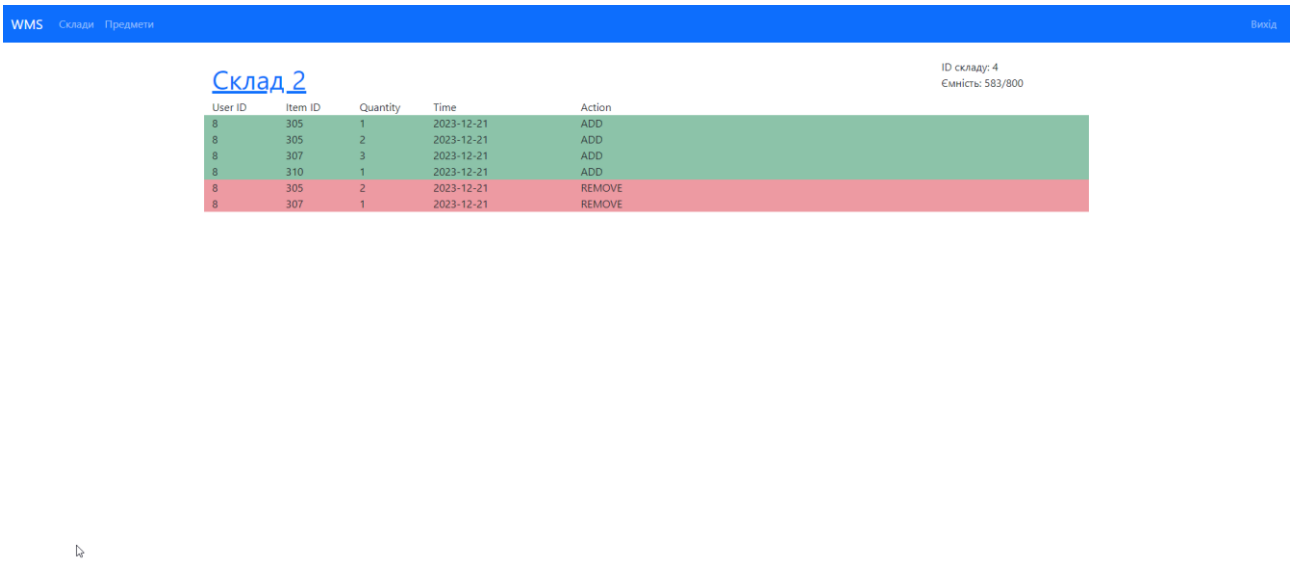

Рис.3.28 Сторінка історії складу

Кожна зміна кількості предметів на складі включає в себе запис в таблицю історії в базі даних, який в подальшому відображається на цій сторінці.

3.3.8. Сторінка помилок

При помилках, як внутрішніх, так і з доступом, користувача має бути перенаправлено на сторінку з помилками, зображеної на рис.3.29. На цій сторінці має бути відображено код помилки, сторінку, яка цю помилку створила, та повідомлення до користувача.

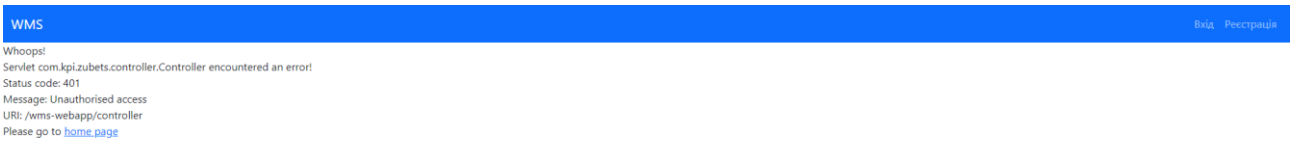

Рис.3.29 Сторінка помилок

Без цієї сторінки користувач бачитиме стандартне повідомлення серверу, яке містить слід помилки, що неприпустимо.

Висновки до розділу 3

У цьому розділі було детально розглянуто аспекти розробки програмного продукту, обґрунтовано технологію для написання програми та шаблони програмування і підходи, використані під час її розробки. Також було вирішено наступні задачі:

1. Було встановлено всі вимоги до програмного продукту, вказано необхідні технічні характеристики для комп'ютера, на якому планується запуск системи. Крім цього докладно описано основні завдання, які ця програма має вирішувати.

2. Було обрано технології, використані під час розробки системи, та надано їх короткий опис.

3. Детально описано структуру системи а також аргументовано використання шаблонів програмування. Власне описано використані шаблони, а саме DAO, Abstract Factory, Command.

4. Детально описано процес розробки системи та обґрунтовано рішення, що стосуються інтерфейсу користувача і внутрішньої архітектури системи.

5. Паралельно з процесом розробки наведено приклад використання системи з її крайніми випадками.

Загалом виконано завдання роботи зі створення моделі розробленої системи для демонстрації її роботи.

РОЗДІЛ 4

РОЗРОБКА СТАРТАП ПРОЕКТУ

4.1. Опис ідеї проекту

Бізнес-системи WMS уже не є новиною. В п.1.3. були розглянуті корпоративні успішні рішення WMS. Розроблена модель призначена для демонстрації архітектури централізованої системи для віддаленого управління складськими приміщеннями. На поточному етапі розробки модель не є повноцінним продуктом. Але якщо ми розглянемо можливість розвитку в майбутньому, то можна заздалегідь визначити основні концепції та положення, а також розробити стратегію для створення ефективної, конкурентоздатної системи

В цьому розділі буде описано розробку стартап-проекту централізованої системи для віддаленого управління складськими приміщеннями. У таблиці 4.1 представлено суть пропозиції ідеї, її потенційні сфери застосування, основні переваги для користувачів, що можна отримати від цієї ідеї, і відмінність цієї ідеї від існуючих аналогів.

Таблиця 4.1.

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Централізована система для віддаленого управління складськими приміщеннями	Торгівля, виробництво, логістика	Можливість контролю предметів на складі та взаємодії цих складів

Далі проводиться аналіз техніко-економічних переваг ідеї. Основними аналогами для порівняння обрано Sortly, Netsuite WMS, Fishbowl Inventory.

Таблиця 4.2.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Мій проект	Конкуренти			W	N	S
			1	2	3			
1	Віддалене розміщення	+	-	+	+		+	
2	Централізація	+	+	-	-			+
3	Журналювання	+	+	+	+		+	
4	Інтеграція QR-кодів	-	+	+	+		+	

Установлений набір сильних, слабких та нейтральних характеристик та властивостей ідеї потенційного товару становить основу для формування його конкурентоздатності. Як видно з таблиці 4.2, основною перевагою розроблюваного продукту є його централізація, тобто можливість працювати з декількома складами одночасно без інтеграцій, на це варто звернути увагу при створенні стартап проекту. Основним недоліком запропонованої системи є відсутність інтеграції QR-кодів, що варто врахувати при вдосконаленні системи.

4.2. Технологічний аудит ідеї проекту

Проведено аудит технологічних засобів, за допомогою яких реалізовано проект, результати якого можна побачити в табл.4.3.

Таблиця 4.3.

Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
1	Централізована система для віддаленого управління складськими приміщеннями	Java 11	Є в наявності	Доступні безкоштовно
2		Apache Tomcat	Є в наявності	Доступні безкоштовно
3		Bootstrap 5	Є в наявності	Доступні безкоштовно
4		Log4j	Є в наявності	Доступні безкоштовно
5		Шаблони програмування та архітектури	Є в наявності	Доступні безкоштовно

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Наступним кроком є аналіз ринку, що наведений у табл. 4.4.

Таблиця 4.4.

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	5-10
2	Загальний обсяг продажу, грн/ум.од	Підписка на сервіс щомісячно 8000, або готове рішення - 40000
2	Динаміка ринку (якісна оцінка)	Постійний ріст
3	Наявність обмежень для входу	Значна кількість конкурентів, і перевага вибору компаній з репутацією
4	Специфічні вимоги до стандартизації та сертифікації	Відсутні, проте варто враховувати закон про захист персональних даних

У таблиці 4.5 наводяться можливі користувацькі групи, їх вимоги та потреби від додатку.

Таблиця 4.5.

Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в обліку	Будь-які організації, компанії зі сховищем	Персональні вимоги до продукту	Зручність, доступність використання

Після визначення можливих цільових груп користувачів ринок було проаналізовано на наявність факторів загроз (табл. 4.6) та можливостей (табл. 4.7).

Таблиця 4.6.

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока конкуренція	На даний момент існує велика кількість аналогів, які в основному повторюють функціонал одне одного	Розширення функціоналу стартап-проекту, готове рішення на персональне замовлення
2	Вузький спектр функціоналу і можливостей	Основних функцій може бути не досить для виходу на ринок	Розширення функціоналу

Таблиця 4.7.

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Залучення інвестицій	Можливість залучення інвестицій з готовим бізнес-планом	Збільшення штату задля підтримки та розробки нового функціоналу системи
2	Розширення клієнтської бази	За умови проведення хорошої PR-компанії існує можливість переманити клієнтів інших компаній	Спланувати рекламну кампанію

Загальні фактори конкуренції наведені у таблиці 4.8.

Таблиця 4.8.

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентос-проможною)
1. Тип конкуренції: чиста	Наявність незалежної конкуренції на ринку	Націлення на функціональність та стрімкий розвиток
2. Рівень конкуренції: міжнаціональна	Конкуренція з міжнародними компаніями	Вихід на міжнародний ринок
3. За галузевою ознакою: внутрішньогалузева	ІТ-сектор	Необхідно зосередити зусилля на пошуку конкурентних переваг, які дозволять компанії займати стійкі конкурентні позиції на даному ринку.
4. Конкуренція товарним видом: товарно-видова	Наявність відомих конкурентів у сфері	Отримання авторитету влаштовуючи спеціалістів сфери поточного проекту
5. За інтенсивністю: марочна	Наявність відомих конкурентних брендів	Створення власного бренду

Виконаємо аналіз конкурентних умов по моделі М. Портера (табл. 4.9).

Таблиця 4.9.

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти в галузі	Постачальники	Клієнти	Товари-замінники
	Sortly, Oracle Netsuite, Fishbowl	Sortly, Oracle Netsuite, Fishbowl	Sortly, Oracle Netsuite, Fishbowl	Компанії від великих до малих	Немає
Висновки:	Велика інтенсивність конкуренції	Є багато додатків, що можуть виконувати такі ж операції	Постачальниками є світові лідери у цій сфері	Високі вимоги усієї клієнтської бази	Безкоштовних замінників немає

Узагальнення конкурентоспроможних аспектів показано у табл. 4.10

Таблиця 4.10.

Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Зручний інтерфейс	Маючи доступ до конкурентів можна врахувати переваги та недоліки існуючих рішень перед виходом на ринок збуту
2	Централізованість системи	У порівнянні з конкурентами можна реалізувати зручну систему, без необхідності додаткових інтеграцій
3	Забезпечення кросплатформеності	Більшість з конкурентів розробляють додаток під конкретну операційну систему. Створення веб додатку розширить клієнтську базу
4	Ціна системи	Встановлення рентабельної ціни у порівняння з конкурентами

Наступним кроком проведемо зіставлення сильних та слабких сторін проекту порівняно із конкуруючими розробками, його наведено в табл. 4.11.

Таблиця 4.11.

Порівняльний аналіз сильних та слабких сторін

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним продуктом						
			-3	-2	-1	0	1	2	3
1	Віддалене розміщення	15				✓			
2	Централізованість	17			✓				
3	Журналювання	10						✓	

Фінальним кроком є SWOT-аналіз (табл. 4.12).

Таблиця 4.12.

SWOT- аналіз стартап-проекту

Сильні сторони: Можливість врахування досвіду конкурентів Швидке реагування на побажання клієнтів	**Слабкі сторони:** Велика кількість конкурентів Недовіра великих клієнтів
Можливості: Отримати контракти на встановлення системи управління	**Загрози:** Програти конкуренцію Не переманити клієнтів

Альтернативні варіанти дій на ринку та заходи їхньої інтеграції наведені в таблиці. 4.13.

Таблиця 4.13.

Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Перетворення розробки на систему з відкритим кодом	Фінансових – низька, спеціалістів - висока	Декілька місяців
2	Зацікавленість компаній у придбанні розробки	Висока - продаж розробки	До року
3	Участь в тендерах і укладення договорів з державними установами	Середня, наявна конкуренція, треба вигравати тендери проти відомих гравців	Декілька місяців

4.4. Розроблення ринкової стратегії проекту

В табл. 4.14 наведено опис цільових груп потенційних споживачів продукту.

Таблиця 4.14.

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Малий бізнес	Готові	Високий	Середня	Простий
2	Середній бізнес	Готові	Високий	Середня	Середній
3	Великий бізнес	Не готові	Високий, але значно більші вимоги	Середня	Складний
Які цільові групи обрано: Потрібно працювати з малим та середнім бізнесом і при розвитку технології залучати також і великий бізнес.					

Для функціонування в обраних ринкових сегментах слід обрати основну стратегію розвитку (табл. 4.15).

Проаналізувавши отримані результати було прийняте рішення про те, що альтернативним шляхом буде продаж своєї реалізації.

Таблиця 4.15.

Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Продаж своєї реалізації	Встановлення конкурентноспроможної ціни, маркетингова кампанія	Фокус на потреби фокус групи, зниження собівартості розробки	Оперативне оновлення продукту відповідно до побажань клієнтів

У таблиці 4.16 сформовано базову стратегію конкурентної поведінки розроблюваного проекту

Таблиця 4.16.

Визначення базової стратегії конкурентної поведінки

№	Чи є проект «пер-шопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні, проект не є «першопрохідцем»	Важливий пошук як і нових споживачів так і вихід на споживачів існуючих конкурентів з цільової групи	Через схожість подібних систем в цілому багато функціоналу є однаковим, наприклад інтеграція QR-кодів	Пропозиція схожого, але більш розширеного функціоналу на вигідних умовах

У таблиці 4.17 визначено основну стратегію позиціювання компанії на ринку.

Таблиця 4.17.

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап- проекту	Вибір асоціацій, які мають сфо-рмувати комплексну позицію власного проекту
1	Висока частота оновлення, широкий функціонал, Кросплатфор-меність	Розширення штату, впровадження нового функціоналу	Оперативна реакція на побажання клієнтів	Автоматизація складу, облік інвентарю

4.5. Розроблення маркетингової програми стартап-проекту

На старті слід визначити ключові переваги концепції розробки, отримуваної користувачем. Отримані результати аналізу наведені в таблиці 4.18.

Таблиця 4.18.

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Управління складським приміщенням	Легкість та простота управління складом, облік наявних позицій на складі	Широкий спектр інструментів для управління складськими приміщеннями
2	Оновлення програми	Врахування побажань клієнтів	Створення відділу підтримки для реагування на відгуки клієнтів
3	Формування звітів	Можливість формувати звіти по роботі програми а також звіти з використання складських приміщень	Модуль для формування звітів

В таблиці 4.19 побудовано трьохрівневу модель товару.

Таблиця 4.19.

Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="2">Централізована система для віддаленого управління складськими приміщеннями</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Розмір</td></tr>
<tr><td>1. Власне система</td><td>500МБ</td></tr>
<tr><td>2. База даних</td><td>500МБ</td></tr>
<tr><td colspan="2">Якість: внутрішнє тестування програми, логування збоїв та система зворотнього зв’язку для повідомлення про неналежну роботу програми</td></tr>
<tr><td colspan="2">Пакування: відсутнє</td></tr>
<tr><td colspan="3">Марка: назва організації-розробника + назва товару</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="2">До продажу: програмний код</td></tr>
<tr><td colspan="2">Після продажу: права на володіння</td></tr>
<tr><td colspan="3">За рахунок чого потенційний товар буде захищено від копіювання: наявна користувацька ліцензійна домовленість та прямі договори, що забороняють копіювання розробки та плагіат</td></tr>
</table>

У таблиці 4.20 наведено межі встановлення цін при ціноутворенні продукту.

Таблиця 4.20.

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі становлення ціни на товар/послугу
1	0$ - 300$	20-2000	>10000$/місяць	200-1000

Система збуту сформована в таблиці 4.21.

Таблиця 4.21.

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових кліє-нтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Покупка ліцензії на продукт або договір про надання послуг без передачі ПЗ до замовника	Підписання контрактів на збут	Однорівневий	Вертикальна

Останнім етапом є визначення концепції маркетингових комунікацій на ринку, що наведено у таблиці 4.22.

Таблиця 4.22.

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Єдиноразове придбання	Рекламна кампанія, конференції, презентації тощо	Віддалена централізована система	Демонстрація основних функцій розробленого продукту	Виділення основних переваг над конкурентами, демонстрація легкості роботи в системі

Висновки до розділу 4

Четвертий розділ присвячений розробці стартап-проекту для створеної системи, зроблено опис ідеї проекту, проведено детальний аналіз конкурентів та можливостей, що доступні на даному ринку. Для дослідження в рамках розробки стартап-проекту було виконано наступні завдання:

1. Наведено загальний опис ідеї проекту для виходу на ринок. Описаний функціонал проекту, визначено сильні, слабкі та нейтральні сторони проекту.

2. Проведено технічний аудит проекту, а саме визначено технологічну здійсненність ідеї проекту,

3. Проаналізовано ринкові можливості для запуску стартап-проекту. Наведено характеристику потенційного ринку стартап-проекту. Визначено характеристику потенційних клієнтів стартап-проекту. Розглянуто фактори загроз і можливостей проекту. Проведено ступеневий аналіз конкуренції на ринку і аналіз конкуренції в галузі за М. Портером. Розглянуто основні фактори конкурентоспроможності та проведено аналіз сильних і слабких сторін проекту в порівнянні з конкурентами. На основі цієї інформації проведено SWOT-аналіз.

4. Розроблено маркетингову програму для просування продукту на ринку. Визначено ключові переваги потенційного товару. Побудовано трьохрівневу модель товару. Визначено межі встановлення ціни, форму збуту та на їх основі визначено концепцію маркетингових комунікацій.

В підсумку, було розроблено стартап-проект для запуску розробленого програмного продукту на ринок збуту, отримано навички створення стартап-проектів, побудови маркетингової стратегії та аналізу обраного ринку.

ВИСНОВКИ

Магістерська робота присвячена створенню централізованої системи для віддаленого управління складськими приміщеннями

Під час виконання роботи було вирішено наступні завдання:

1. Розроблено архітектуру системи централізованого управління складськими приміщеннями для ефективної організації віддаленого управління розподіленими складськими приміщеннями.

2. Розроблено централізовану систему для віддаленого управління складськими приміщеннями, що має на меті вирішити проблему керування розподіленими складами.

Базуючись на проведеному дослідженні систем управління складськими приміщеннями було виконано поетапну розробку моделі централізованої системи для віддаленого управління складськими приміщеннями. Було розглянуто можливі для реалізації архітектури системи, та на основі розглянутих переваг і недоліків обґрунтовано вибір клієнт-серверної MVC архітектури, що відрізняє систему від більшості їй подібних. Під час аналізу можливих баз даних, були детально розглянуті реляційні та NoSQL бази даних. Враховуючи їх сильні та слабкі сторони обрано базу даних для реалізації моделі в архітектурі MVC. Розроблено схему бази даних для моделі централізованої системи для віддаленого управління складськими приміщеннями. Обґрунтовано необхідність використання системи шифрування паролів користувачів шляхом хешування з додаванням солі.

Також було детально розглянуто аспекти розробки програмного продукту, обґрунтовано технологію для написання програми та шаблони програмування і підходи, використані під час її розробки. Було встановлено всі вимоги до програмного продукту, вказано необхідні технічні характеристики для комп'ютера, на якому планується запуск системи. Крім цього докладно описано основні завдання, які ця програма має вирішувати. Детально описано структуру системи а також аргументовано використання шаблонів програмування. Описано процес розробки системи та обґрунтовано рішення, що стосуються

інтерфейсу користувача і внутрішньої архітектури системи. Тестування створеної системи продемонструвало виконання всіх вимог до системи і підтвердило виконання поставлених завдань.

Виконано аналіз стартап-проекту шляхом опису та документування бізнес плану. Виявлені потенційно слабкі та сильні сторони ідеї. Також визначено технічну можливість розробки і стратегії збуту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Bartholdi, J. J., & Hackman, S. T. (2006). Warehouse & distribution science. The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

2) Emmett, S. (2005). Excellence in Warehouse Management: How to minimise costs and Maximise value.

3) Tompkins, J. A., White, J. A., Bozer, Y. A., & Tanchoco, J. M. A. (2010). Facilities Planning (4th ed.). Wiley.

4) Jain, H. (2023, March 28). D365 Supply Chain Inbound Inventory Quality Check. [Електронний ресурс] https://www.linkedin.com/pulse/d365-supply-chain-inbound-inventory-quality-check-harsh-jain

5) Keller, S., & Keller, B. (2013). Definitive Guide to Warehousing, The: Managing the Storage and Handling of Materials and Products in the Supply Chain (Council of Supply Chain Management Professionals) (1st ed.). Pearson FT Press.

6) Helo Petri, Szekely Bulcsu, (2005) Logistics information systems: An analysis of software solutions for supply chain co-ordination, Industrial Management and Data Systems

7) Kamalruzaman, N. N. N., & Omar, N. A. (2023). Superdough Inventory Web-based System. Applied Information Technology And Computer Science, 4(2), 1889–1906.

8) Sortly. (2023, November 29). Sortly: Inventory simplified. [Електронний ресурс] https://www.sortly.com/

9) Medicherla, S. S., & Archana, M. (2022). Study on the ERP implementation Methodologies on SAP, Oracle NetSuite, and Microsoft Dynamics 365: A review. arXiv (Cornell University).

10) NetSuite, O. (2023, September 22). NetSuite Warehouse Management. Oracle NetSuite. [Електронний ресурс] https://www.netsuite.com/portal/products/erp/warehouse-fulfillment/wms.shtml

11) De Silva, S., & Ratnayake, G. S. (2022). Order Prioritization Prediction System for sample rooms in the garment industry. International Journal for Research in Applied Science and Engineering Technology, 10(3), 911–926.

12) Fishbowl Warehouse WMS - WMS pricing, demo & comparison tool. (n.d.). [Електронний ресурс] https://www.explorewms.com/fishbowl-warehouse-wms-software-profile.html

13) Bujak, A., Orzeł, A., & Smal, T. (2023). Increasing resilience of warehouse in logistic supply chain by use of the Manhattan WMOS on the example of the chosen enterprise. European Research Studies Journal, XXVI(Issue 1), 68–81.

14) Warehouse Management System (WMS) | Manhattan. (2023, January 5). Manhattan Associates. [Електронний ресурс] https://www.manh.com/en-sg/our-solutions/supply-chain-management-software/warehouse-management-system

15) Hui Nee Au Yong. (2009). Warehouse Management System and Business Performance: Case Study of a Regional Distribution Centre. Conference: International Conference on Computing and Informatics (ICOCI 2009). [Електронний ресурс] https://www.researchgate.net/publication/260058173_Warehouse_Management_System_and_Business_Performance_Case_Study_of_a_Regional_Distribution_Centre

16) Tompkins, J. A. (n.d.). The Distribution Management Handbook. McGraw-Hill Trade.

17) Mulcahy, D., & Sydow, J. (2008). A supply chain logistics program for warehouse management. In Auerbach Publications eBooks.

18) Sventek, J. (1992). The Distributed Application architecture. In The MIT Press eBooks.

19) Hanson, M. D. (2000). The client/server architecture. In Server Management (pp. 17-28). Auerbach Publications.

20) Fall, K. R., & Stevens, W. R. (2012). Tcp/ip illustrated (Vol. 1). Addison-Wesley Professional.

21) SOA Source Book - What Is SOA? (n.d.). [Електронний ресурс] https://collaboration.opengroup.org/projects/soa-book/pages.php?action=show&ggid=1314

22) Laskey, K. B., & Laskey, K. (2009). Service oriented architecture. Wiley Interdisciplinary Reviews: Computational Statistics, 1(1), 101-105.

23) Reenskaug, T., & O. Coplien, J. (2009, March 20). artima - The DCI Architecture: A New Vision of Object-Oriented Programming. [Електронний ресурс] https://www.artima.com/articles/the-dci-architecture-a-new-vision-of-object-oriented-programming

24) Krasner, G. E., & Pope, S. T. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. Journal of object oriented programming, 1(3), 26-49.

25) Hosting Data. (2023, September 27). NoSQL Databases List by hosting data. [Електронний ресурс] https://hostingdata.co.uk/nosql-database/

26) A Relational Database Overview (The JavaTM Tutorials >JDBC Database Access >JDBC Introduction). (n.d.). [Електронний ресурс] https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

27) Sumathi, S., & Esakkirajan, S. (2007). Fundamentals of Relational Database Management Systems. In Studies in computational intelligence.

28) Connolly, T., & Begg, C. (1998). Database Systems: A Practical approach to design, implementation and management.

29) Sumathi, S., & Esakkirajan, S. (2007). Fundamentals of relational database management systems (Vol. 47). Springer.

30) Grassi, P. A., Garcia, M., & Fenton, J. L. (2017). Digital identity guidelines: revision 3.

31) Zeeshan, A. A. (2014, November 20). Hashing Passwords using ASP.NET's Crypto Class. CodeProject. [Електронний ресурс] https://www.codeproject.com/Articles/844722/Hashing-Passwords-using-ASP-NETs-Crypto-Class

32) WebCite query result. (n.d.). [Електронний ресурс] https://webcitation.org/66n9tGLPk?url=http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

33) The GOF Design Patterns Memory - Learning Object-Oriented Design & Programming. (2017, October 1). w3sDesign. [Електронний ресурс] https://web.archive.org/web/20200923124858/http://w3sdesign.com/?gr=b02&ugr=proble

34) Dupire, B., & Fernandez, E. B. (2001, September). The command dispatcher pattern. In 8th Conference on Pattern Languages of Programs.

35) Freeman, E., Freeman, E., Bates, B., & Sierra, K. (2004). Head first design patterns.

36) Nahar, N., & Sakib, K. (2015). Automatic Recommendation of Software Design Patterns Using Anti-patterns in the Design Phase: A Case Study on Abstract Factory. In QuASoQ/WAWSE/CMCE@ APSEC (pp. 9-16)