КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Програмна система керування складськими запасами»

Виконав _____
(Підпис)
Гусаківський Віктор Петрович
(прізвище, ім'я, по батькові)

Керівник Силантьєв Сергій Олексійович
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	Виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	Виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	Виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	Виконано
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	Виконано
9.	Подання роботи у першому варіанті	11.05.2021	Виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	Виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	24.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	2
Анотація (іноземною мовою-англійською)	2
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	10
2	14
3	19
Висновки	1
Перелік використаних джерел	2
Додатки	11

				ДП XXXX 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.				Відомість дипломної роботи	Лист	Листів
Керівн.						
Н/контр.						
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ (РЕФЕРАТ)

Дипломна робота Гусаківського Віктора Петровича на тему «Програмна система керування складськими запасами», спеціальність 122 «Комп'ютерні науки», освітня програма «Прикладне програмування», Київський національний університет імені Тараса Шевченка, 2021, Київ.

Дипломна робота складається із вступу, трьох розділів, висновку, списку використаних джерел і додатків.

Робота виконана в обсязі 69 сторінок, містить 24 рисунків, 3 таблиць, 23 джерел, 4 додатків.

Мета дипломної роботи зі створення системи керування складськими запасами є організація роботи і процесів на складах. При цьому використання сучасних інформаційних технологій вимагає перегляду принципів і механізмів управління підприємством.

Для досягнення мети дипломної роботи були поставлені наступні завдання:

- охарактеризувати організацію роботи складських систем;
- провести аналіз доступних архітектурних рішень та обрати оптимальні програмні засоби для реалізації програмної системи;
- спроектувати та розробити програмну реалізацію системи керування складськими запасами.

Об'єктом дослідження є процеси організації і управління складськими запасами.

Предметом дослідження дипломної роботи є розробка програмної системи управління складськими запасами.

При написанні дипломної роботи були використані наступні методи наукових досліджень: аналіз, синтез, моделювання, формалізація, сходження від абстрактного до конкретного.

Ключові слова: веб-застосунок, програмна система, сервер, браузер, фреймворк, склад, запаси.

ANOTATION (ABSTRACT)

The thesis of Viktor Husakivskyi on "Warehouse stock management software system", specialty 122 "Computer Science", the educational program "Applied Programming", Taras Shevchenko National University of Kyiv, 2021, Kyiv.

The thesis includes an introduction, three sections, conclusions, a list of sources and appendices.

The thesis consists of 69 pages, contains 24 figures, 3 tables, 23 sources, 4 appendices.

The purpose of the thesis on the creation of a warehouse management system is the organization of work and processes in warehouses. Hence, the use of modern information technologies requires a revision of the principles and mechanisms of enterprise management.

To achieve the goal of the thesis the following tasks were set:

- to describe the organization of warehouse systems workflow;
- to analyze the available architectural solutions and choose the optimal software instruments for the implementation of the software system;
- to design and develop software implementation of the warehouse stock management system.

The object of the thesis is the organization and management of warehouse stock.

The subject of the thesis is the development of the warehouse stock management software system.

The following research methods were used: analysis, synthesis, modeling, formalization, an ascent from the abstract to the concrete.

Keywords: web application, software system, server, browser, framework, warehouse, stock.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CSS (Cascading Style Sheets) – спеціальна мова стилю сторінок , що використовується для опису їхнього зовнішнього вигляду.

HTML (HyperText Markup Language) – мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет.

БД – база даних.

СУБД – система управління базою даних.

DOM – об'єктна модель документа, за допомогою якої утворюється доступ до елементів веб-сторінки.

WMS – система керування складськими запасами.

IDE – інтегроване середовище для розробки.

H2 – легка СУБД, повністю написана на java.

GIT – система керування версіями файлів.

JPA – Java Persistence API – стандартизований інтерфейс для фреймворків.

JSTL – бібліотека, що дозволяє додавати базові оператори прямо в HTML код.

ВСТУП

Актуальність теми. Розвиток промисловості та соціально-економічного благополуччя населення неминуче спричиняє зростання потреб споживачів. Попит позитивно впливає на розвиток ринку, значно зростає обіг товарів. Розгортання глобальних транспортних систем в поєднанні з вільним доступом до онлайн-торгівлі привело до необхідності стрімкого розгортання торгівельної інфраструктури.

Аналогічна ситуація спостерігається і в промисловості: науково-технічний прогрес забезпечив інтенсивний розвиток технологій та засобів виробництва, а також підходів до використання ресурсів та розміщення інфраструктури.

Станом на сьогодні в світі з’являється все більше і більше складських приміщень і зростає потреба у впровадженні систем керування складськими запасами для полегшення та пришвидшення роботи на них.

Це дає змогу як покращувати якість обслуговування товару, так і збільшувати свої прибутки, оскільки час – це гроші. Та і хороша система управління складськими запасами – один з важливих чинників на вузькоспеціалізованому ринку.

Тому актуальність проблеми автоматизації управління складською інфраструктурою є надзвичайно високою та продовжує зростати.

Мета дипломної роботи зі створення системи керування складськими запасами є організація роботи і процесів на складах. При цьому використання сучасних інформаційних технологій вимагає перегляду принципів і механізмів управління підприємством.

Для досягнення мети дипломної роботи були поставлені наступні **завдання**:

- охарактеризувати організацію роботи складських систем;
- провести аналіз доступних архітектурних рішень та обрати оптимальні програмні засоби для реалізації програмної системи;
- спроектувати та розробити програмну реалізацію системи керування складськими запасами.

Об’єктом дослідження дипломної роботи є процеси організації і управління складськими запасами.

Предметом дослідження дипломної роботи є розробка програмної системи управління складськими запасами.

При написанні дипломної роботи були використані наступні **методи наукових досліджень**: аналіз, синтез, моделювання, формалізація, сходження від абстрактного до конкретного.

У процесі виконання дипломної роботи було використано засоби розробки Intellij Idea, Java, H2 database, Bootstrap, JSP, JSTL, CSS, Spring, Spring Boot, Spring MVS.

Результати дипломної роботи можуть бути використані при створені системи керування складськими запасами, зокрема, як готове рішення, що покриває головні потреби базової моделі організації функціонування складських приміщень.

Дипломна робота складається із вступу, трьох розділів, висновку, списку використаних джерел і додатків.

Розділ 1. Теоретичні відомості про організацію роботи складських приміщень.

Теоретичні відомості про організацію роботи складських приміщень.

Система керування складськими запасами (англ. Warehouse management system, абревіатура WMS) – це інформаційна система, що забезпечує автоматизацію управління бізнес процесами складських приміщень підприємства[14].

WMS-системи діляться на дві принципово різні групи: без адресного зберігання і з адресним зберіганням. У першій групі система автоматизує тільки звітність зовнішнього надходження, відправлення товарів і також займається оформленням первинних документів[11]. Такі системи часто називають програмами для забезпечення обліку обліку складських запасів. Система, яка

враховує адресний сховище, необхідна на великих, багатооборотних складах, а також додатково має такі функції[15]:

- обробка складських перевезень з дотриманням параметрів і габаритних розмірів складу, а також топологічних властивостей складу(розташування різних зон на території складського приміщення);
- планування розташування продукту з урахуванням власностей, таких як місце зберігання, Fast moving категорії, тобто наскільки товар «ходовий», побажань власника продукту, що зберігається в приміщенні складу (якщо це склад типу 3PL);
- спеціальні підсистеми підбору товарів для відвантаження, включаючи можливість розміщення товарів (у залежності від попиту на вимогу) в проміжних зонах з відбором по автоматичному заповненню (Picking zone);
- планування відвантаження товарі без тимчасового розміщення їх на території складу: «Cross Docking»;
- підсистема обліку робочого часу працівників на основі виконуваних ними завдань, а також спеціальні бонусні системи для працівників, для їх заохочення;
- проведення планових інвентаризаційних робіт;

Розміри WMS можна поділити наступним чином[13]:

- WMS початкового рівня: склади невеликих або вузькоспеціалізованих підприємств, малого бізнесу, невеликих магазинів розбрібної торгівлі;
- «Коробочні» системи управління складом: склад 1000-10 000 м^2 з великою номенклатурою, але малим товарообігом;

- адаптивні системи: зазвичай системи такого рівня впроваджують великі логістичні компанії, розподільні центри зі складами 5000 м2;
- Налаштовувана система: склад від 5000 м² з великою номенклатурою і високим товарообігом.

Зазвичай площа складу розділена на зони за типом технічних операцій для автоматизації процедур: прийом, розміщення, зберігання, обробка і відвантаження товарів, що дозволяє спростити роботу персоналу в різних місцях і ефективно розділити обов'язки[1].

На етапі реалізації вводиться опис фізичних властивостей складу, вантажної техніки, параметрів всього використовуваного обладнання і правил роботи.

Усі типи пакунків, від штук (EACH) до палет (Pallet) маркуються штрихкодом. Операції технічного складування під управлінням системи виконуються на основі даних зі штрих-кодів, місця зберігання товару і завантажувального обладнання. Таке обладнання та складські робітники оснащені радіо терміналами введення / виведення даних, портативними комп'ютерами, які по радіо зв'язуються з головним сервером системи. Система може використовувати один з існуючих типів коду або друкувати етикетки з внутрішнім штрих-кодом.

При проведенні інвентаризації фахівці зчитують дані за допомогою спеціальних терміналів (TSI), штрих-коди, автоматично заносяться в базу даних приладу.

Система враховує всі вимоги до умов зберігання при виділенні складських площ для товарів, що надходять на склад. Наприклад, можна враховувати вологість, температуру, термін придатності, виробника, час виконання замовлення, постачальника, правила сумісності та інші параметри. WMS автоматично вибирає місце зберігання отриманих вантажів і створює робочі

завдання для складських робітників. Завдання з'являються на екрані радіотерміналу у вигляді основних покрокових інструкцій, що індивідуально формуються для кожного співробітника залежно від його поточного розташування на території складу[2].

Під час об’єднання робітників в команду система обирає оптимальний спосіб переміщення техніки всередині складського комплексу, що дозволяє скоротити пробіг не завантаженої техніки. Для виконання операцій система розподіляє вантажне устаткування, використання якого найкраще підходить для даної задачі. Завдання підтверджується скануванням штрих-коду. Таким чином, система контролює дії всіх співробітників і дозволяє практично повністю виключити можливість неправильного розміщення вантажу або неправильного виконання замовлення. Система миттєво оновлює всю інформацію про місцезнаходження товарів, наявність товарів на складі, дії працівників і виконані операції. Для простоти в деяких системах керування складськими запасами можна відстежувати дії працівників в режимі двовимірного графічного відображення. За результатами роботи або поточним станом товару, що знаходиться на території складського приміщення, система дозволяє формувати звіти, які можна роздрукувати і відправити в операційну систему компанії.

Аналіз існуючих систем керування складськими запасами.

Зростаючий попит на системи керування складськими запасами став причиною появи цілого ряду програмних продуктів, кожен з яких у свій спосіб намагається забезпечити реалізацію потреб користувачів.

За даними інформаційного проєкту Logist.FM, український ринок систем керування складськими запасами представлений наступними програмними продуктами: G.O.L.D Stock, Instock WMS, Manhattan WMS, Qguar WMS, Blue Yonder WMS, Solvo WMS, SAP WMS, Oracle WMS. Їхній функціонал є достатньо

схожим, однак, має й ряд ключових відмінностей, серед яких: виконання всіх операцій з використанням RF-терміналів, інтеграція з іншими програмами та модулями, а також наявність функціоналу, що не пов’язаний на пряму з керуванням складськими запасами, але спрощує таку діяльність.

Варто наголосити, що перераховані є системами класу «А», тобто повнофункціональними системами з масштабними можливостями. Їх використовують складські комплекси зі складними системами зберігання (значна кількість палетомісць, мезоніни і т.д.); широкою номенклатурою (від 1000 SKU); необхідністю тонкого налаштування бізнес-процесів і т.д. Крім цього, існують також рішення, котрі автоматизують основні або окремі задачі, однак, їх не можна вважати повноцінними системами керування складськими запасами (WMS MobileWarehouse, ABM Cloud).

Існує ряд факторів, які необхідно враховувати при оцінці систем керування складськими запасами, наприклад:

1. Можливості і специфіка системи.
   Для керування невеликим складом з простими товарами тривалого зберігання достатньо WMS-систем базового рівня, котрі здатні забезпечити прийом, облік та розміщенням товарів, інвентаризацію та фіксацію операцій з товарами. А для розподільчого центру роздрібної мережі необхідна значно складніша система, яка зможе консолідувати дані по кількох складах, дозволить вести партійний облік, контроль за терміном придатності та інші завдання.
2. Ресурси, необхідні для впровадження системи.
   Для швидкого старту ефективної роботи з системою важливе чітке розуміння щодо матеріальних та людських ресурсів, які необхідно залучити.
3. Ергономічність системи.

Навіть найбільш високотехнологічна система не принесе достатньої ефективності, якщо буде незручна у використанні.

4. Можливості інтеграції з сучасними технологіями відбору та автоматизації.

   Не всі WMS-системи мають первинну інтеграцію з такими технологіями як Pick-by-Voice, Pick-to-Light, динамічним зважуванням, машинним зором, роботизацією та ін. Тому ще на початку роботи з системою необхідно оцінювати свої очікування щодо впровадження різноманітних «допрацювань».

5. Підхід до документації.

   Важливо звернути увагу на технічну документацію системи, адже без належно описаного функціоналу не можна гарантувати правильну експлуатацію системи, особливо на етапах зміни осіб, відповідальних за її роботу.

Іншими критеріями, на які звертають увагу є також приклади успішно реалізованих проєктів та наявність локальної команди впровадження та підтримки користувачів.

Logist.FM оцінює представлені в Україні програмні системи керування складськими запасами наступним чином:

WMS система	Підхід до документації	Ергономічність	Відповідність сучасним технологіям	Локальна команда впровадження і підтримки	Реалізовані проекти	Загальна оцінка
G.O.L.D Stock	10	10	10	10	9	57
Instock WMS	9	9	10	8	7	53
Manhattan WMS	10	10	10	5	10	51
Qguar WMS	9	8	7	10	10	49
Blue Yonder WMS	10	10	10	2	9	46
Solvo WMS	10	7	9	4	10	45

SAP WMS	10	8	9	3	9	43
Oracle WMS	8	8	7	7	6	40

Таблиця 1.1. Порівняльна характеристика систем управління складськими запасами.

Розглянемо детальніше програмні системи керування складськими запасами на прикладі компанії Blue Yonder.

ЇЇ основними продуктами є: Blue Yonder Warehouse Management, Blue Yonder Dispatcher WMS, Blue Yonder Supply Chain Planner та Luminate.

Blue Yonder Warehouse Management володіє широким функціоналом та потенціалом для масштабування, що дозволяє автоматизувати повний комплекс складських операцій (від прийому товару до його відвантаження).

З цією метою впроваджені наступні рішення:

- розрахунок ефективності роботи персоналу;
- використання таких технологій як: голосові системи управління, RFID, інтеграція з устаткуванням;
- оптимізація розміщення товарів на складі;
- управління завданнями для працівників та контроль навантаження складського персоналу;
- моніторинг операцій на складі;
- впорядкування документообігу;
- виконання всіх операцій з використанням RF-терміналів.

Хоча на ринку України є багато систем керування складськими запасами, не рідко – це відносно «старі» програмні рішення, які працюють вже багато років. Через свій вік вони пройшли вже багато етапів модернізації, що помітно відзначилося на архітектурі застосунку і організації бази даних. Структура останньої постійно змінювалася, додавалися нові рішення для зберігання інформації. Також постійно за потреби клієнта додається новий специфічний вузько направлений функціонал, який нерідко може стати корнем проблеми у

вже існуючому функціоналі. Тому постає гостра потреба в постійному мануальному і автоматизованому тестуванні як нововведених рішень, так і вже існуючих.

Підводні камені ховаються саме в частині регресійного тестування, оскільки для побудови тестових сценаріїв необхідно:

1.Тісно співпрацювати з великою кількістю веб елементів застосунку;

2.Мати постійний, стабільний і легкий доступ до бази даних в разі необхідності швидкого отримання даних.

Отже, як бачимо, програмні системи найвищого рівня автоматизації, які призначені для великогабаритних складів та компаній, дозволяють достатньо ефективно здійснювати управління складськими запасами. Їхні функціональні можливості та інтегровані бізнес-рішення є достатньо схожими, хоча мають і відмінності. При цьому їм характерні певні недоліки, які необхідно взяти до уваги під час розробки системи керування складськими запасами.

Постановка задачі на розробку програмної системи управління складськими запасами.

В загальному понятті постановка задачі – це опис продукту з зазначеним запланованим функціоналом, мовою програмування, додатковими технологіями, що використовуються для розробки даного застосунку, середовища для розробки, можливими помилками і недоліками системи, які необхідно врахувати. Також до пунктів постановки задачі можуть належати опис вхідних і вихідних параметрів системи, які такі є в наявності.

Отже, для початку слід визначити назву і дати короткий опис продукту:

Назва програмної системи: «Складська система».

Опис: Дана система слугує програмним забезпеченням для автоматизації складських процесів, а саме: завантаження товару, відвантаження товару та зберігання товару на території складських приміщень. (Додаток Г)

Про кожен процес окремо:

- Зберігання товару на складі:

Має бути реалізована можливість перегляду товару, який в даний момент знаходиться на складі, також треба виводити базову необхідну інформацію про кожну палету товару.

- Завантаження та відвантаження товару:

Треба реалізувати можливість створення спеціальних замовлень на Завантаження чи Відвантаження та можливість додавання ліній товару до поточного створеного замовлення.

При створенні замовлення необхідно ввести спеціальну назву даного замовлення.

При створенні лінії замовлення необхідно Артикул товару, який необхідно відвантажити.

Архітектура програмного продукту:

Для створення програмної системи прийняти рішення використати архітектуру MVC.

Мова програмування додаткові технології та середовище для розробки, що будуть використовуватися для розробки програмної системи:

Для реалізації структури MVC було обрано об’єктно-орієнтовану мову програмування джава та фреймворки Spring, Spring Boot, Spring MVC.

Для побудови користувацького інтерфейсу застосовуватимуться такі технології, як: JSP, JSTL, CSS, Bootstrap.

Для реалізації бази даних будуть використовуватися фреймворк Hibernate та СУБД H2.

В якості середовища для розробки програмного продукту буде використовуватися Intellij Idea від компанії JetBrains.

Детальна інформація про дані технології в наступному розділі.

Особливості системи:

Необхідно реалізувати різні типи доступу для користувачів, а саме:

 Звичайний користувач; Адміністратор.

Лише користувач з роллю «Адміністратор» має право доступу до таких функцій системи, як:

- Додати користувача;
- Змінити інформацію про користувача;
- Змінити роль користувача;

Функція реєстрації нового користувача має бути відсутньою.

Висновки:

Під час виконання першого розділу дипломної роботи було досліджено теоретичну інформацію про системи керування складськими запасами.

Наведено порівняльну характеристику провідних систем управління складськими запасами.

Визначено основні проблеми та недоліки використовуваних в Україні систем керування складськими запасами.

Сформовано і поставлено задачі для розробки програмної системи керування складськими запасами.

Розділ 2. Аналіз архітектурних рішень і вибір програмних засобів для реалізації програмної системи

Архітектурні рішення для розробки.

Основне завдання дипломної роботи є створення програмної системи керування складськими запасами. Для виконання поставленої задачі необхідно послідовно вирішити наступні задачі:

1. Визначити стек технологій, що будуть використовуватись під час розробки програмного продукту.
2. Визначити функціонал сервісу.
3. Розробити та протестувати WEB застосунок.
4. Розробити інструкцію користувача.

Системи керування складськими запасами часто є досить великими і далеко не найпростішими в обслуговуванні. Для розробки таких систем потрібна хороша команда вмілих розробників, але, якщо трохи подумати – можна вирішити, яку структуру обрати для системи такого рівня.

Для реалізації WEB-застосунку було використано стек технологій для створення програмних продуктів: JSP, JSTL CSS, Java, пакетний менеджер Maven, фреймворк Spring (MVC, Boot), Hibernate, база даних H2.

Jsp (Java server pages) – технологія, що дозволяє розробнику генерувати HTML код динамічно, а в зв’язці з JSTL вставляти Java код на сторінки HTML. В контексті курсової роботи використовується для виводу даних і наповнення таблиць на сторінках. Приклад використання:

Метод для передачі даних на HTML форму за допомогою DOM моделі:

```
static void userOutput(User user, Model model) {
```

```
        model.addAttribute("name", user.getName());
        model.addAttribute("surname", user.getSurname());

         }
```

Позначення елементів в які за допомогою моделі передаються дані:

```
${name}
${surname}
```

Для стилізації Jsp документів використовуються каскадні таблиці стилів (**CSS**) − це технологія описування зовнішнього вигляду документа, який написаний мовою гіпертекстової розмітки HTML[6].

CSS використовується авторами веб−сторінок для вказування кольорів, шрифтів, розташування блоків та інших аспектів представлення документа.

Основною метою CSS є розділення вмісту (написаного на HTML) та подання документа (написаного на CSS). Цей поділ документа збільшує доступність документа, надає велику гнучкість та можливість виконувати управління його показу, а також зменшити складність та повторюваність в структурі самого документа.

Крім того, CSS надає можливість показувати один і той же документ в різних стилях або методах виведення.

Java (вимовляється Джава; інколи — Ява) — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

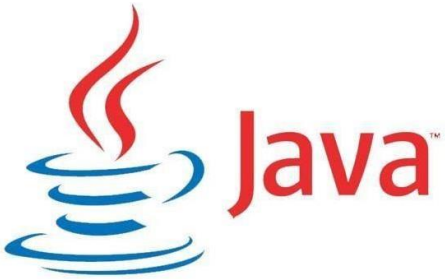

Рисунок 2.1. Логотип мови програмування Java

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Розробники Java створили офіційний технічний опис, у якому пояснюють цілі та переваги нової мови. Усього їх одинадцять. Ця мова:

1. проста;
2. об'єктно-орієнтована;
3. розподілена;
4. інтерпретована;
5. безпечна;
6. надійна;
7. архітектурно нейтральна;
8. портативна;
9. високопродуктивна;
10. багатопоточна;
11. динамічна.

Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За

необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Основна перевага використання байт-коду — це портативність. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть майже завжди працювати повільніше, ніж скомпільовані у машинний код, і саме тому Java одержала репутацію «повільної» мови. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Швидкість офіційної віртуальної машини Java значно покращилася з моменту випуску ранніх версій, до того ж, деякі випробування показали, що продуктивність JIT-компіляторів у порівнянні зі звичайними компіляторами у машинний код майже однакова. Проте ефективність компіляторів не завжди свідчить про швидкість виконання скомпільованого коду, тільки ретельне тестування може виявити справжню ефективність у даній системі.

У Java всі об'єкти є похідними від головного об'єкта (він називається просто Object), з якого вони успадковують базову поведінку і властивості.

У Java можливе тільки одинарне успадкування, завдяки чому виключається можливість конфліктів між членами класу (методи і змінні), які успадковуються від базових класів.

В Java існує система винятків або ситуацій, коли програма зустрічається з неочікуваними труднощами, наприклад:

- операції над елементом масиву поза його межами або над порожнім елементом
- читання з недоступного каталогу або неправильної адреси URL
- ввід недопустимих даних користувачем

Одна з особливостей концепції віртуальної машини полягає в тому, що помилки (виключення) не призводять до повного краху системи. Крім того, існують інструменти, які «приєднуються» до середовища періоду виконання і

кожен раз, коли сталося певне виключення, записують інформацію з пам'яті для відлагодження програми. Ці інструменти автоматизованої обробки виключень надають основну інформацію щодо виключень в програмах на Java.

Проте мову програмування Java не рекомендується використовувати в системах, збій в роботі яких може призвести до смерті, травм чи значних фізичних ушкоджень (наприклад, програмне забезпечення для керування атомними електростанціями, польотами, систем життєзабезпечення чи систем озброєння) через ненадійність програм, написаних на мові програмування Java, пункт ліцензії Microsoft 7.7.h.

Java використовує автоматичний збирач сміття для керування пам'яттю під час життєвого циклу об'єкта. Збирання сміття дозволене у будь-який час. В ідеалі воно відбувається під час бездіяльності програми. Збірка сміття автоматично форсується при нестачі вільної пам'яті в купі для розміщення нового об'єкта, що може призводити до кілька секундного зависання. Тому існують реалізації віртуальної машини Java з прибиральником сміття, спеціально створеним для програмування систем реального часу.

Java не має підтримки вказівників у стилі C/C++. Це зроблено задля безпеки й надійності, аби дозволити збирачу сміття переміщувати вказівникові об'єкти.

Maven – пакетний менеджер, що автоматично завантажує необхідні блоки і плагіни в проект, за умови коректного опису цих модулів в POM-файлі, що є конфігурацією для maven.

Як вже було сказано вище – система досить велика, тому краще розділити її на декілька частин. Для таких потреб чудово підходить модель MVC.

Spring – фрейморк, що є «обгорткою» над базовими java класами і цим полегшує роботу робробника[16]. Spring має певну кількість модулів, в контексті

курсової роботи використовувались Spring MVC – технологія, що являє собою механізм умовного поділу застосунку на 3 частини[18]:

Model - в дані частині описується бізнес логіка застосунку.

View - відповідає за вивід інформації і передачу вхідної інформації на контроллер.

Controller – відповідає за обробку вхідної інформації на основі бізнес логіки застосунку і передачі даних на View для її відображення користувачеві[7].

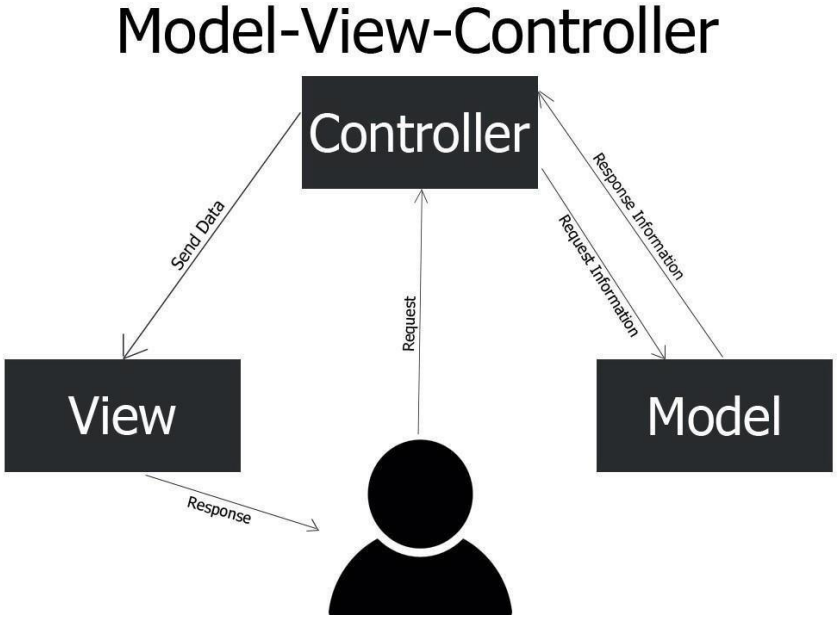

Рисунок 2.2. Структура MVC

Та Spring Boot – технологія, за допомогою якої не потрібно розгортати сервер, і лише потім на цьому сервер запускати застосунок на даному сервері. Завдяки embed tomcat, що вбудований в Spring Boot, потрібно лише запустити наш застосунок, а він в свою чергу сам розгорне вбудований в себе сервер[17].

Hibernate – фреймворк, що полегшує роботу з базою даних, повністю бере на себе відповідальність за створення бази даних та sql запитів. Але потребує чітких налаштувань, для того, щоб правильно згенерувати всю необхідну інформацію. Має в арсеналі досить великий спектр SQL діалектів, тому його можна використовувати мало не з кожною сучасною відомою СУБД[12].

H2 – кросплатформна легка база даних, що повністю написана на Java. Дану БД можна конфігурувати по різному, відповідно – дані будуть зберігатися або в пам’яті, або записуватися у файли, як в будь якій базі даних, або часто в пам’ять, частково у файл.

Середовище для розробки

В якості середовища для розробки я обрав Intellij Idea. Це інтегроване середовище для розробки від компанії JetBrains, яке на мою думку чудово підходить як для початківців, так і для просунутих спеціалістів, що займаються програмуванням на мові Java і похідних від неї: Scala, Kotlin і так далі.

Інтегроване середовище розробки (IDE) являє собою комплекс програмних засобів, які можна використовувати для різних аспектів розробки програмного забезпечення. Майже «з коробки» доступні такі програмні рішення, як компілятори, засоби автоматичного завершення виконання коду, автоматичний запуск тестів, графічні конструктори та інше.

IDE чудово себе проявляє як під час розробки велико масштабних програм, так і під час створення власних невеликих пет-проектів. Компанія JetBrains інтегрувала в середовище розробки багато сторонніх плагінів без яких в теперішньому світі важко уявити сучасне програмування на мові Java.
Наприклад:

- GIT;
- Maven;
- Terminal;
- Database;

Засоби для роботи з GIT навіть виведені на основну верхню панель середовища для розробки, що банально зберігає розробнику дорогоцінний час.

Єдине, що необхідно зробити, щоб даний функціонал працюв – перевірити, щоб в папці з проектом був ініціалізований GIT проект,

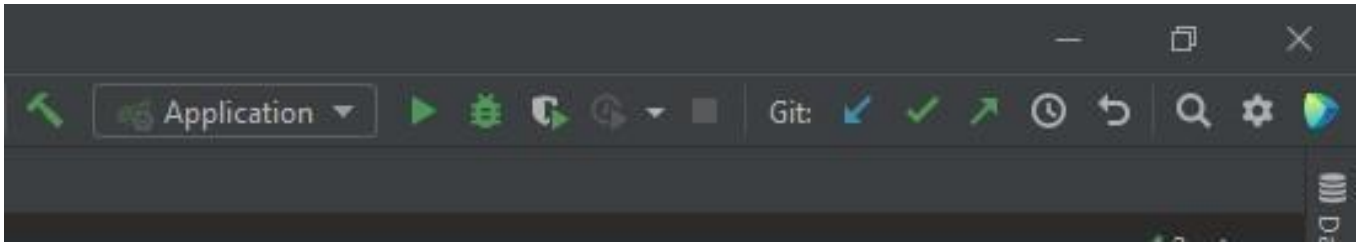

Рисунок 2.3. Верхня панель контекстного меню Intellij Idea

На бічній лівій панелі (Рисунок 2.4.) можна проглянути список змін у всіх файлах і директоріях, які знаходяться всередині проекту перед тим, як внести (commit) зміни в код, і власне, зберегти їх на сервері, або ж одразу завантажити (push) їх на сервер. А також файли в директоріях будуть підсвічуватися різними кольорами, залежно від того, який на даний момент стан файлу. На рисунку 2.3. видно, що присутній мінімальний достатній функціонал, що необхідний для адекватної роботи з GIT навіть не згортаючи IDE присутній в Intellij Idea.

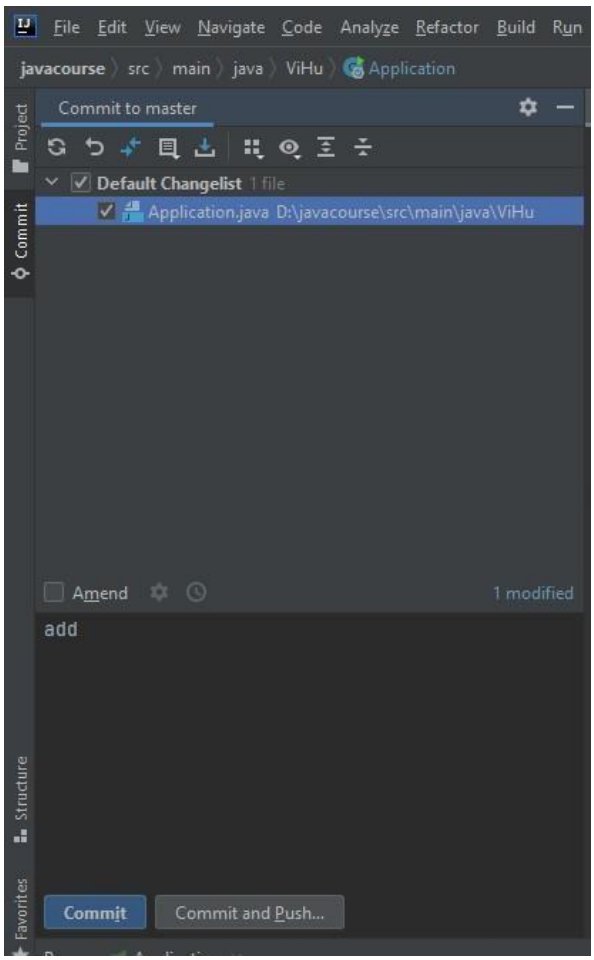

Рисунок 2.4. Бічна панель зі змінами файлів для GIT

Також є можливість працювати з Maven – це менеджер збирання пакетів. Він простий і зручний у використанні, щоправда запускати його потрібно через консоль, а не в кожного починаючого спеціаліста є такі можливості. В JetBrains є вихід і в таких ситуаціях. В будь який момент можна перейти на вкладку Maven прямо в середовищі розробки і зі списку вибрати ті дії, які необхідно зробити[9].

Рисунок 2.5. Функціонал Maven доступний з середовища розробки

В процесі написання програмної системи керування складськими запасами я використовував виключно Intellij Idea. Бо навіть HTML код можна писати в даному середовищі розробки. Все, що потрібно зробити – лише додати потрібні плагіни для роботи з HTML та CSS.

Вибір програмно-технічних рішень для реалізації web-інтерфейсу з урахуванням недоліків існуючих програмних систем.

Як було згадано в параграфі 1.2., одним з недоліків існуючих систем керування складськими запасами є складність в регресійному тестуванні. Вона виникає за рахунок великої кількості веб елементів. Насичена структура сторінок обов'язкова в застосунках такого типу і уникнути її неможливо, проте є можливість зробити її зрозумілою і прозорою для полегшення процесу тестування в майбутньому.

З швидким розвитком такої мови програмування як JavaScript з'явилися і далі продовжує з'являтися ряд фреймворків, що полегшують процес створення веб сторінок[19]. За допомогою коду можна одразу формувати веб сторінку і додавати до неї всі необхідні статичні і динамічні об'єкти. Дані технології мають місце бути, проте в окремих випадках і саме наш випадок не є таким, оскільки ці генеровані об'єкти створюються як набір клонів одного елементу, і їх важко, або взагалі неможливо виокремити серед десятків ідентичних[20].

Моїм рішенням у створенні веб частини програмної системи було використання таких технологій, як JSP та JSTL, що вже були згадані в попередньому параграфі[21].

JSP – це аналог HTML веб сторінок, що є по суті Java кодом, що при запуску передає браузеру по черзі рядок за рядком весь файл. Зручним є те, що синтаксис, що використовується в JSP, нічим не відрізняється від HTML коду. Це означає, що навіть розробники-початківці, що ще не до кінця знайомі з даною технологією, можуть з легкістю перейти на неї і в процесі використання вивчити її тонкощі.

Ще однією перевагою у використанні технології JSP є те, що вона, повністю сумісна з Java застосунками. Завдяки даному фактору краще забезпечується стабільна робота програмної системи.

Під час розробки веб сторінок інколи виникає потреба у створенні певного набору об'єктів, або груп об'єктів, кількість яких невідома, до моменту витягнення цих об'єктів з бази даних. На основі вище сказаного можна зробити логічний висновок, що з цією роботою добре впоралися б JavaScript фреймворки. Проте для підтримання стабільності роботи і в цілях збереження зручності процесу тестування системи керування складськими запасами я вирішив, що необхідним моментом, якого слід дотримуватися в процесі розробки є використання програмних рішень, що написані мовою Java, тому для описаних вище випадків обрав технологію JSTL.

Дана технологія дозволяє за допомогою спеціальних позначень вставляти в HTML код сторінки прості логічні вирази (IF ELSE) і оператори циклів (FOR, FOREACH, WHILE), що повністю задовольняє потребу в створенні невідомої до моменту вивантаження з БД кількості об'єктів.

```
<table class="table table-striped">
    <thead>
    <tr>
        <td><b>Номер палети</b></td>
        <td><b>Артикул</b></td>
        <td><b>Дата загрузки на склад</b></td>
    </tr>
    </thead>
    <c:forEach items="${pallets}" var="pallet">
        <tr>
            <td>${pallet.palletName}</td>
            <td>${pallet.articleName}</td>
            <td>${pallet.date}</td>
        </tr>
    </c:forEach>
</table>
```

Рисунок 2.6. Приклад використання JSTL в коді для створення і заповнення таблиці

Вибір програмно-технічних рішень для введення бази даних з урахуванням недоліків існуючих програмних систем.

СУБД (система управління базами даних) – це програмно-технічне рішення, яке діє як посередник між базою даних і безпосередніми користувачами цієї БД. Використання системи програмних та мовних засобів, дозволяє створювати, вести та спільно використовувати БД необмеженому колу користувачів[22].

Обов’язковими елементами сучасної СУБД є ядро, процесор мови БД, підсистема підтримки часу виконання, а також сервісні програми, котрі визначають функціонал, спрямований на підтримку інформаційної системи.

Вибір СУБД для роботи з базою даних є одною з первинних проблем при проектуванні web-застосунку.

СУБД поділяють на окремі типи, виходячи з моделі даних, методів надання доступу до БД і рівня розподілення[23].

В залежності від моделі даних СУБД бувають:

- мережевими;
- ієрархічними;
- реляційними; об’єктно-реляційними;
- об’єктно-орієнтованими.

Відповідно до методу надання доступу до БД СУБД поділяються на:

- інтегровані; «клієнт-сервер»; «файл-сервер».

За рівнем розподілення СУБД бувають:

- розподіленими (складові елементи одної СУБД можуть бути розподілені на різних машинах);
- локальними (всі елементи СУБД розміщені на одній машині).

Найбільш поширеними СУБД, які активно застосовуються при розробці web-застосунків є: Derby, HSQLDB, MySQL, PostgreSQL та H2.

Як вказувалося раніше, при розробці програмної системи керування складськими запасами мною була обрана СУБД H2. Далі наведена таблиця взята з офіційного сайту H2. Вона містить порівняльну характеристику різних популярних СУБД та яскраво ілюструє переваги H2 над аналогічними рішеннями[10]:

	H2	**Derby**	**HSQLDB**	**MySQL**	**PostgreSQL**

Pure Java	Yes	Yes	Yes	No	No
Memory Mode	Yes	Yes	Yes	No	No
Encrypted Database	Yes	Yes	Yes	No	No
ODBC Driver	Yes	No	No	Yes	Yes
Fulltext Search	Yes	No	No	Yes	Yes
Multi Version Concurrency	Yes	No	Yes	Yes	Yes
Footprint (embedded)	~2 MB	~3 MB	~1.5 MB	—	—
Footprint (client)	~500 KB	~600 KB	~1.5 MB	~1 MB	~700 KB

Таблиця 2.1. Порівняльна таблиця різних СУБД

Окрім переваг СУБД H2, поданих в таблиці(Таблиця 2.1.) є ще одна, про яку не згадується. Це те, що дана БД повністю написана мовою програмування Java, завдяки чому добре співпрацює з самим Java кодом, що в свою чергу забезпечує кращий і більш стабільний доступ до інформації.

Розділ 3. Програмна реалізація системи керування складськими запасами

Опис структури та ієрархія класів програмного продукту.

Структура програмної системи є класичною для застосунків написаних в середовищі для розробки Intellij Idea, тобто вона складається з таких директорій:

- .idea – Системна папка, що створюється одразу при створенні проекту в середовищі для розробки Intellij Idea;
- src – source directory – папка, що створюється спеціально для коду програми;
- target – папка, в яку поміщується остання скомпільована за допомогою пакетного менеджера версія застосунку;
- iml файл – системний файл в якому вказується кодування продукту, версія XML та версія java модулів.

Проте – є певні відмінності. Оскільки програмна система була створена за допомогою фреймворків Spring, Spring Boot, Hibernate це вносить певні зміни в структуру проекту і додається новий файл під назвою pom.xml. Це системний файл фреймворка, в якому вказується перелік модулів і плагінів з доступних в Spring, що будуть підключені до застосунку.

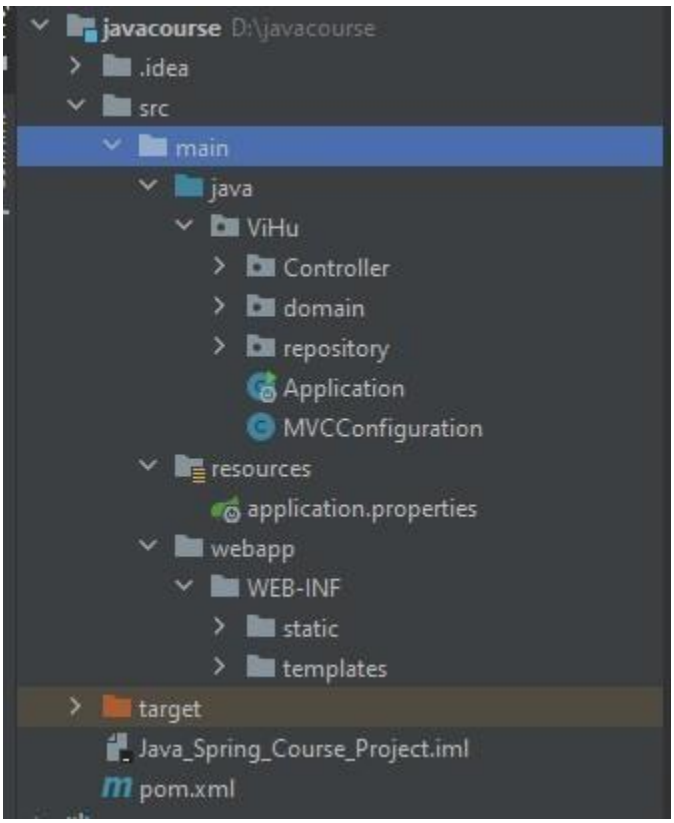

Рисунок 3.1. Структура програмної системи

Для зручності, одразу під час подальшого аналізу структури та ієрархії класів програмної системи будемо розділювати файли за структурою MVC, тобто на модель(Model), вигляд(View) і контролер(Controller).

Заглибившись в директорію src бачимо, що вона поділяється на 3 папки:

- java – папка, в яку слід помістити безпосередньо всі класи програми;
- resources – папка з додатковими налаштуваннями для Spring; - webapp – папка для веб складової програми.

Тепер більш детально про кожну з них. **resources**. Як вже було вказано вище – це папка, в якій розміщується файл загальних налаштувань фреймворка. Таких як:

- шлях до файлів сторінок веб застосунку;

- розширення веб сторінок(.jsp, .html і тд.);
- шлях до картинок, музичних файлів, відео, що потрібні для ініціалізації веб сторінок і не інформацією, яка зберігається в базі даних системи;
- налаштування бази даних.

```
spring.mvc.view.prefix=/WEB-INF/templates/
spring.mvc.view.suffix=.jsp
spring.resources.static-locations=/WEB-INF/static/
spring.mvc.static-path-pattern=/static/**

spring.datasource.url=jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

Рисунок 3.2. Файл налаштувань application.properties

Важливо при створенні даного файлу правильно назвати його application.properties. В противному випадку ці налаштування будуть просто проігноровані фреймворком.

webapp. Як було сказано вище – це папка в якій розміщуються файли сторінок. Для розмежування основної і другорядної інформації я розділив файли jsp на 2 папки:

- templates – папка, в якій знаходяться виключно необхідні jsp файли;
- static – папка, в яку я помістив лого для своєї сторінки а також javaScript файли.

Файли, що знаходяться в папці webapp належать до вигляду(View за моделлю MVC), оскільки за допомогою них і відбувається комунікація користувача з системою.

java. Папка, в якій розміщується основний код програми. В Intellij Idea ця папка за замовчуванням має підсвічуватися блакитним кольором, що свідчить про те, що вона розпізнана як папка для вихідного коду програми. Якщо з значок директорії не підсвічується – слід визначити його за допомогою внутрішніх інструментів інтегрованого середовища розробки Intellij Idea. В противному випадку можуть виникати конфлікти при запуску програми.

Ще однією особливістю Intellij Idea є те, що клас Main.java, в нашому випадку за вимог фреймворку Spring названий Application.java, не може бути розташований в директорії java, тому для зручності і уникнення проблем всі пакети і класи поміщені в проміжну директорію ViHu, скорочено від Viktor Husakivskji(Рисунок 3.3.).

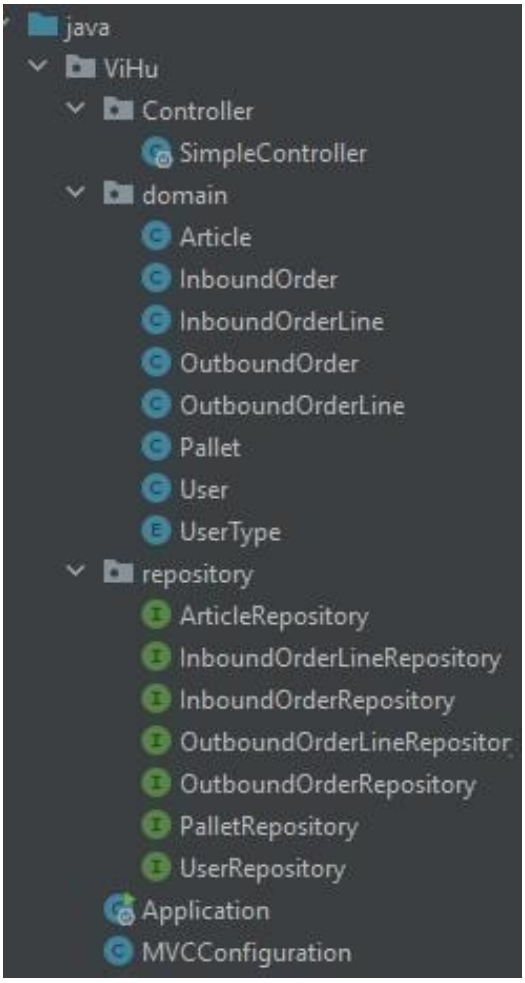

Рисунок 3.3. Структура класів програмної системи

Розпочнемо зі згаданого вище класу Application.java. Для того, щоб фреймворк розумів, що цей клас використовується для запуску застосунку, до нього додається анотація @SpringBootApplication(Рисунок 3.4.).

Відмінністю Spring і Spring Boot є те, що по суті при запуску метода main в даному класі ми наче запускаємо цей же клас(Рисунок 3.4.).

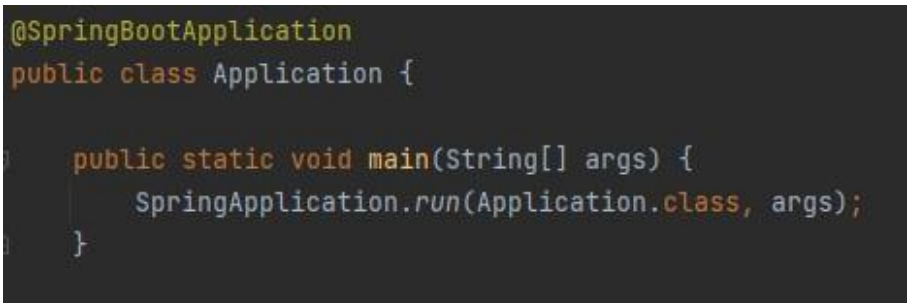

Рисунок 3.4. Головний метод для запуску застосунку на Spring Boot

Чому це відбувається саме так? Тому що Spring Boot на відміну від Spring після старту спочатку створює і запускає сервер, в моєму випадку Apache Tomcat embed, а згодом сам проект. Даний процес проходить автоматично, тому зникає потреба в додаткових налаштуваннях локального сервера, які фреймворк бере на себе.

Також, якщо потрібно при запуску програми ініціалізувати певні тестові дані – в цьому класі додається спеціальний метод, в якому вони створюються напряму запитами до бази даних (Додаток А).

Клас MVCConfiguration.java. Його назва говорить сама за себе, він потрібен для додаткового налаштування модулю MVC в Spring.

Наступний пункт – директорія controller в які розміщений клас SimpleController.java. Даний клас містить в собі всю реалізацію програмної системи і якраз слугує контролером (Controller з моделі MVC), який є посередником між користувачем і базою даних (Додаток Б).

Директорії domain і repository слід розглядати разом, оскільки кожному класу з domain відповідає свій інтерфейс з repository. Всі класи цих папок по суті

і є моделлю (Model з моделі MVC). Їх структура побудована так саме через особливості фреймворку Hibernate, для якого класи domain є прикладами сутностей, на основі яких створюється база даних, а інтерфейси repository містять в собі базові методи, такі як:

- додати елемент;
- видалити елемент за ідентифікатором;
- отримати елемент; - змінити елемент.

Та всі методи, які можна додати власноруч у відповідні інтерфейси (Додаток В).

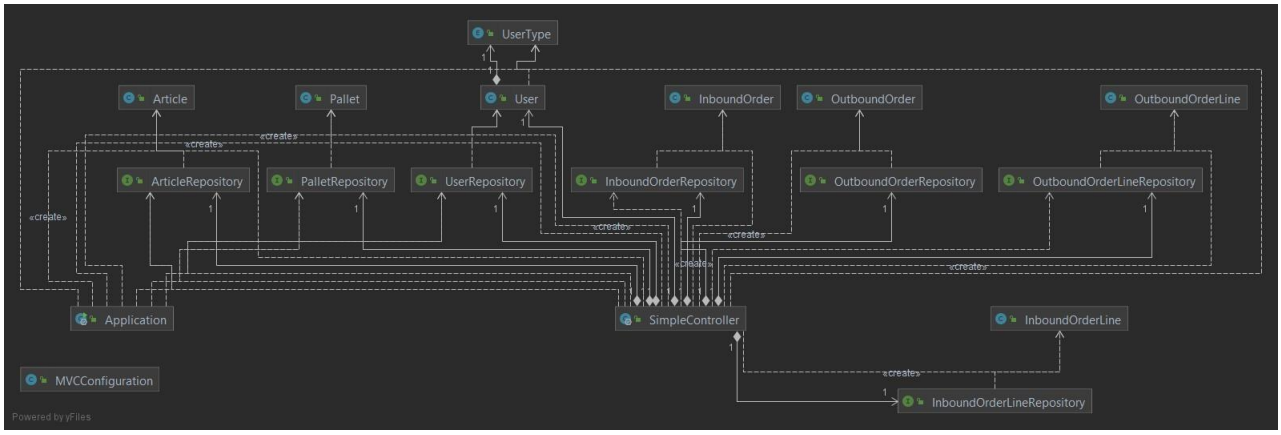

Рисунок 3.5. UML діаграма класів програмної системи

З UML діаграмі класів(Рисунок 3.5.) може здаватися, що структура програми досить складна, проте якщо придивитися – можна побачити головну перевагу патерну проектування MVC, а саме – те, що користувач не «спілкується» з базою даних напряму, будь які процеси проходять лише через контролер. Що і є головною задачею оскільки дані мають бути закритими для доступу ззовні.

Реалізація бази даних.

Як вже згадувалося вище – для введення бази даних в програмну систему я обрав Java H2. Оскільки для розробки я використовував фреймворки JPA і

Hibernate, підхід до створення її був трохи специфічним. Завдяки вище згаданим утилітам мені не прийшлося налаштовувати базу даних самому, бо це робить Hibernate, якщо надати список інструкцій.

До цих інструкцій входять:

- сутності та анотації;
- налаштування у файлі application.properties.

```
spring.datasource.url=jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

Рисунок 3.6. Налаштування для бази даних

На рисунку(Рисунок 3.6.) можна побачити, що налаштування у файлі application.properties займає всього 5 рядків.

Тепер до самої структури бази даних. Умовно її можна поділити на 5 груп:

- Користувач: User;
- Замовлення: InboundOrder, OutboundOrder;
- Лінії замовлення: InboundOrderLine, OutboundOrderLine;
- Палета: Pallet;
- Артикул: Article;

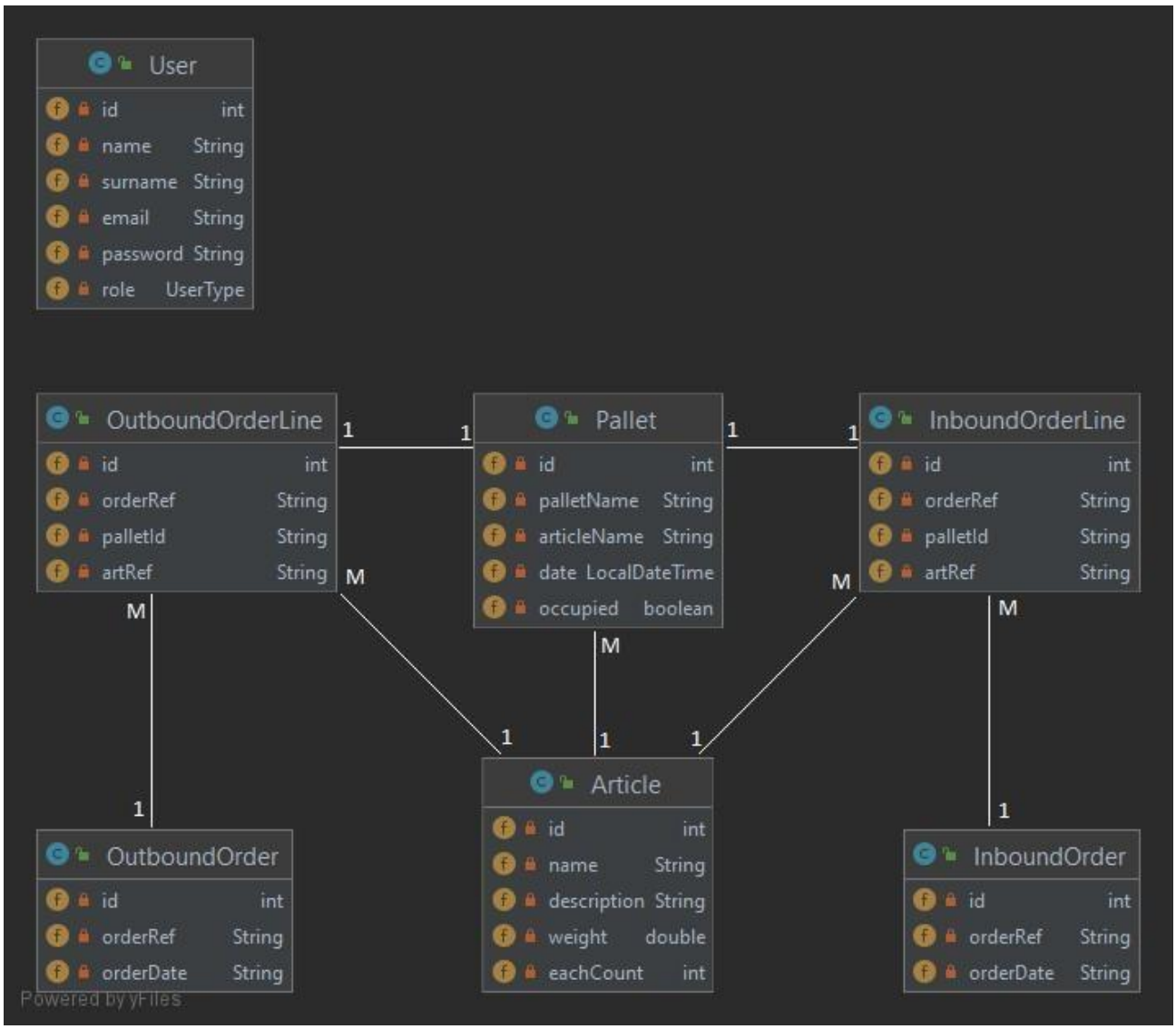

Рисунок 3.7. UML діаграма бази даних

З UML діаграми(Рисунок 3.7.), що подана вище можемо зробити такі висновки:

User – єдиний аркуш бази даних, який не пов’язаний з іншими, оскільки він слугує виключно сховищем інформації про користувача, а саме:

- Ім’я;
- Прізвище;
- Адресу електронної пошти;
- Пароль;
- Роль користувача(«Адміністратор», «Користувач»)

Article – аркуш для зберігання інформації про доступні для зберігання на складі артикули товару. Містить в собі інформацію про:

- Назву Артикулу;
- Стислу інформацію про артикул;
- Вагу однієї палети товару, оскільки це важливий критерій для визначення, чи можна взагалі зберігати цей товар на території складського приміщення;
- Кількість одиниці товару що поміщаються на палеті не виходячи за габаритні рамки.

Pallet – аркуш для зберігання інформації про палети, які в даний момент часу розміщені на території складського приміщення, містить інформацію про:

- Артикул товару;
- Номер даної палети;
- Дату прибуття на склад (важливо для відслідковування терміну придатності);
- Показник того, чи не зарезервована дана палета для лінії замовлення на відвантаження зі складу.

OutboundOrder/InboundOrder – аркуші для зберігання інформації про замовлення на відвантаження/завантаження відповідно. В даних аркушах зберігається наступна інформація:

- Назва замовлення;
- Дата створення замовлення (Потрібно для подальшої звітності).

OutboundOrderLine/InboundOrderLine – аркуші для зберігання інформації про лінії замовлення, що створені для замовлень на відвантаження/завантаження відповідно. Містять в собі таку інформацію:

- Назву замовлення, для якого вони власне і були створені;
- Номер палети, яка була зарезервована для лінії замовлення;
- Назва артикула товару даної палети.

На UML діаграмі, що подана вище лініями показано, як зв’язані між собою листи бази даних, та який саме тип зв’язку використовується під час їхньої зв’язки, де;

- 1-1: одному елементу з першої таблиці відповідає один елемент з другої таблиці і навпаки – одному елементу з другої таблиці відповідає лише один елемент з першої таблиці, наприклад зв’язок між аркушами

OutboundOrderLine – Pallet;

- 1-M: одному елементу з першої таблиці може відповідати декілька елементів з другої таблиці, але одному елементу з другої таблиці відповідає лише один елемент з першої, наприклад OutboundOrder – OutboundOrderLine;
- M-1: декільком елементам з першої таблиці може відповідати один елемент з другої таблиці, але одному елементу з другої таблиці відповідає лише один елемент з першої, наприклад Pallet – Article.

Інструкція користувача для роботи з програмною системою керування складськими запасами.

Оскільки система керування складськими запасами не є відкритою системою, то і функція реєстрації в ній відсутня. Для кращої захищеності

системи нового користувача можна створити або системно, прямим запитом в базу даних, або ж це робить «Адміністратор» увійшовши в систему.

Користувач з роллю «Адміністратор» може виконувати всі ті ж самі функції, що і звичайний користувач, серед яких:

- Формування замовлення на завантаження/відвантаження

товарів на склад та зі складу відповідно;

- Виконання замовлень на завантаження та відвантаження;
- Доступ до інвентарю, загальний огляд всіх палет, що

знаходяться на складі;

- Доступ до загальної статистики;
- Додавання нового користувача, про що було вказано раніше;
- Зміна даних користувача;

Користувач, в якого відсутні права «Адміністратора», має доступ до всіх описаних вище функцій, окрім останніх двох.

	Користувач	**Адміністратор**
Виконати замовлення	+	+
Формувати замовлення на завантаження	+	+
Формувати замовлення на відвантаження	+	+
Перегляд інвентарю	+	+
Загальна статистика	+	+
Додати користувача	-	+
Змінити інформацію користувача	-	+

Таблиця 3.1. Порівняльна характеристика різних типів користувачів системи

При запуску сайту користувач одразу бачить перед собою форму для входу в систему, оскільки дана сторінка сайту не несе якогось смислового навантаження, а слугує лише «дверима» в систему. Дана сторінка є максимально простою, форма розташована в центрі, тому одразу потрапляє в поле зору.

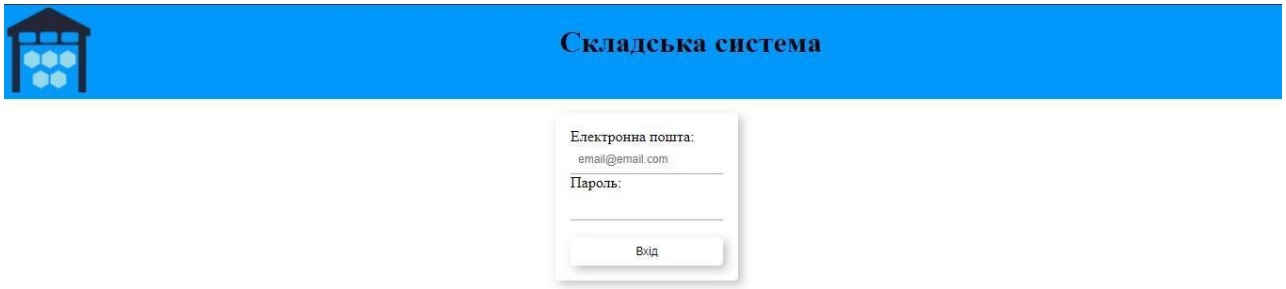

Рисунок 3.8. Сторінка для входу в систему

Після введення коректних даних, користувач потрапляє на сторінку «Статистика», на які можемо побачити основну стислу інформацію стосовно стану складу:

- Поточна кількість замовлень на завантаження/відвантаження;
- Поточна кількість палет;

Як можемо побачити з рисунків, що подані нижче – користувачу з правами «Адміністратора» надано доступ до більшої кількості функцій, ніж користувачу без цих прав, як і описано вище.

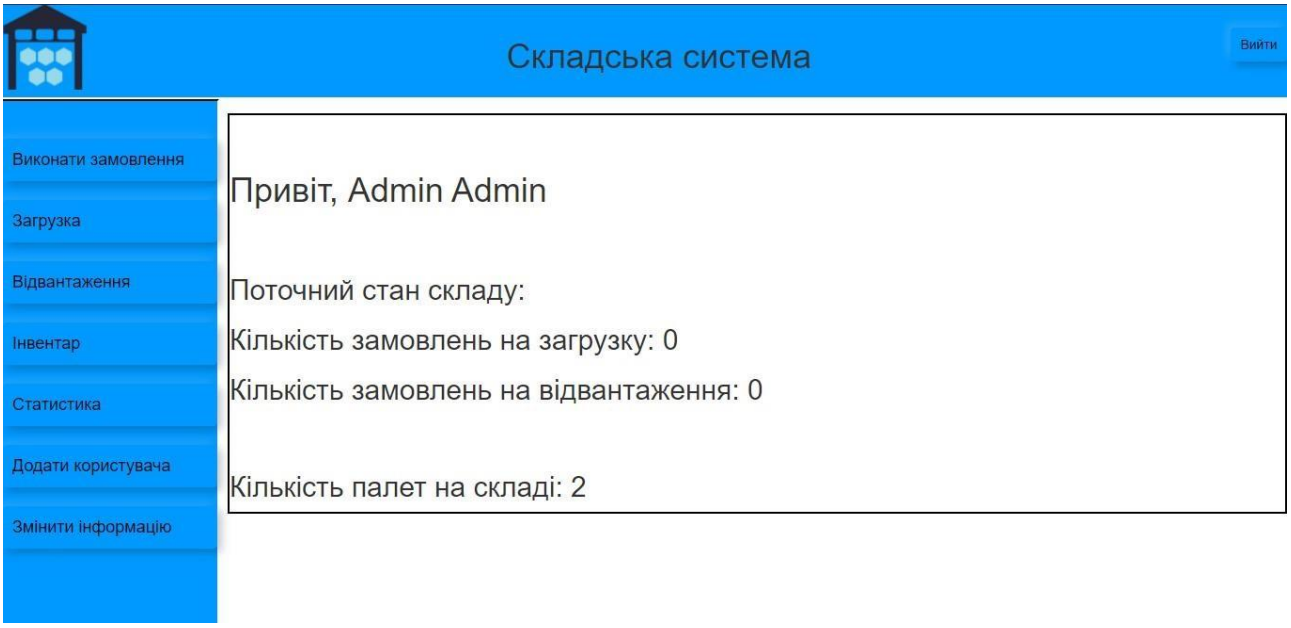

Рисунок 3.9. Основний вигляд програми після входу в систему «Адміністратора»

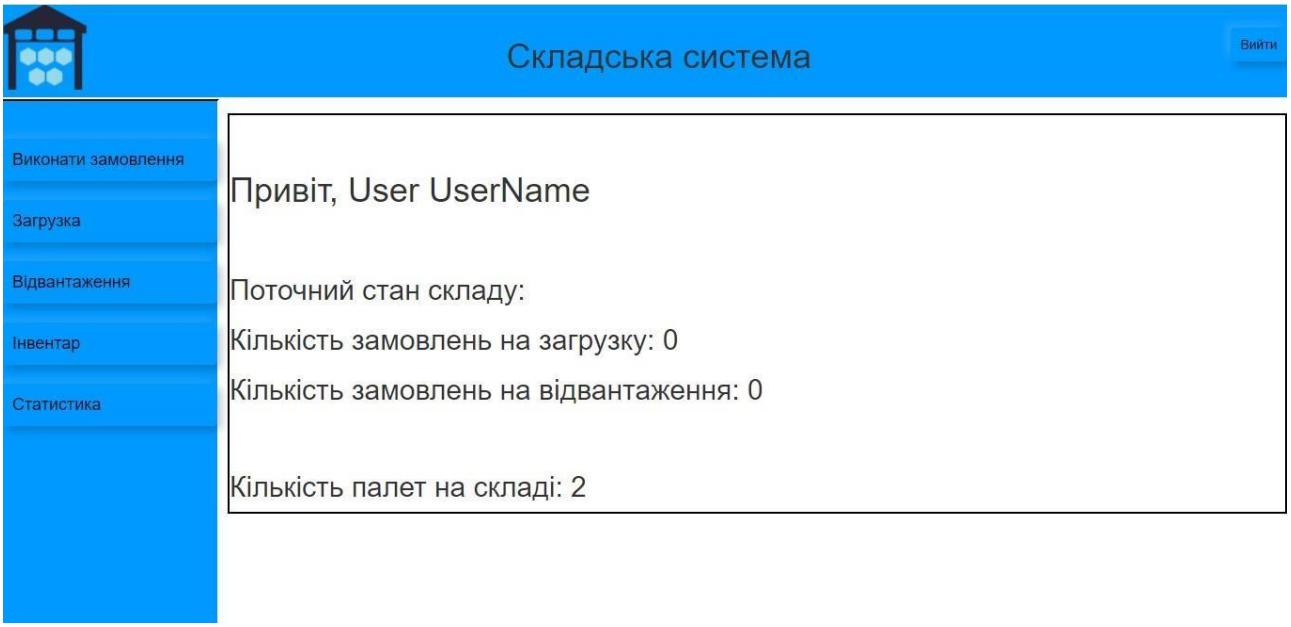

Рисунок 3.10. Основний вигляд програми після входу в систему звичайного користувача

При переході на вкладку «Інвентар» можемо побачити таблицю з основною необхідною інформацією стосовно палет, що в даний момент зберігаються на території складу, а саме:

- Унікальний номер палети;
- Артикул товару, що перебуває на палеті;

- Дата завантаження палети на склад;

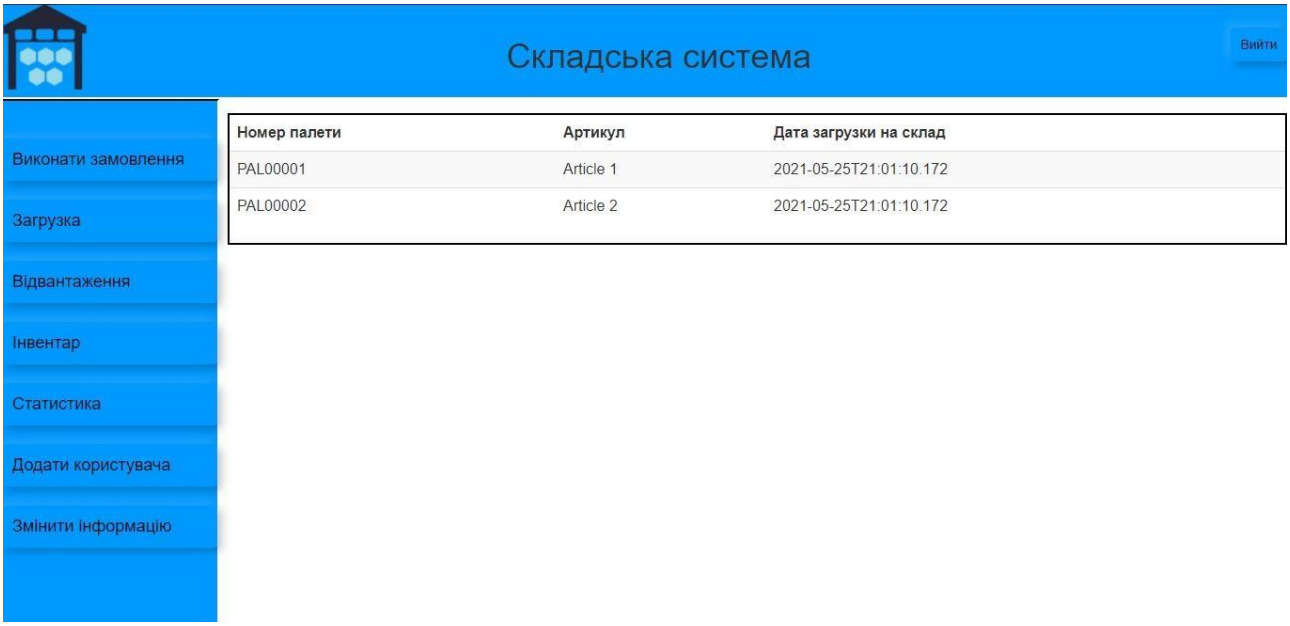

Рисунок 3.11. Сторінка «Інвентар»

Вкладки «Відвантаження» та «Загрузка» є досить схожими, оскільки в обох випадках ми створюємо певного роду замовлення товару: на відвантаження і завантаження товару відповідно. Також не важко помітити помітити, що користувацький інтерфейс на цих вкладках стає умовно поділеним на дві частини. Ліва частина містить в собі форму для створення замовлення, а права – таблицю, в якій відображаються повністю сформовані замовлення. Замовлення рахується сформованим, якщо в нього додана принаймні одна лінія товару, тобто мінімум одна палета.

Хоча самі вкладки і схожі, проте відповідають за абсолютно різні процеси, власне тому і було прийнято рішення розділити їх.

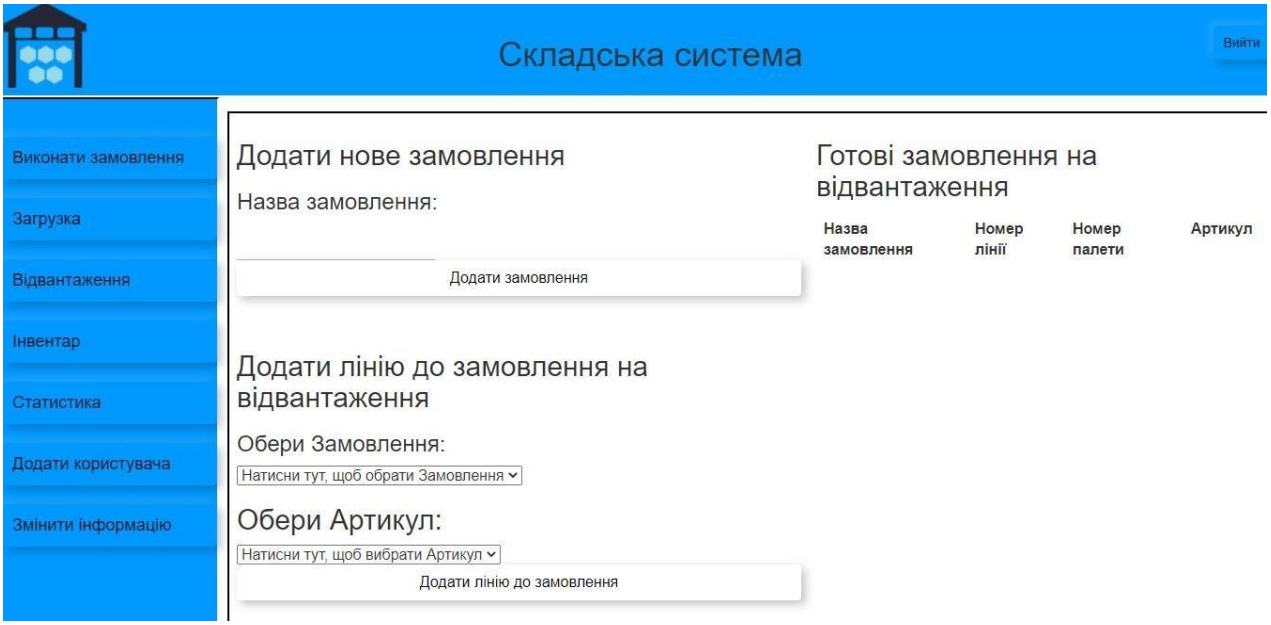

Рисунок 3.12. Сторінка для додавання замовлень на відвантаження

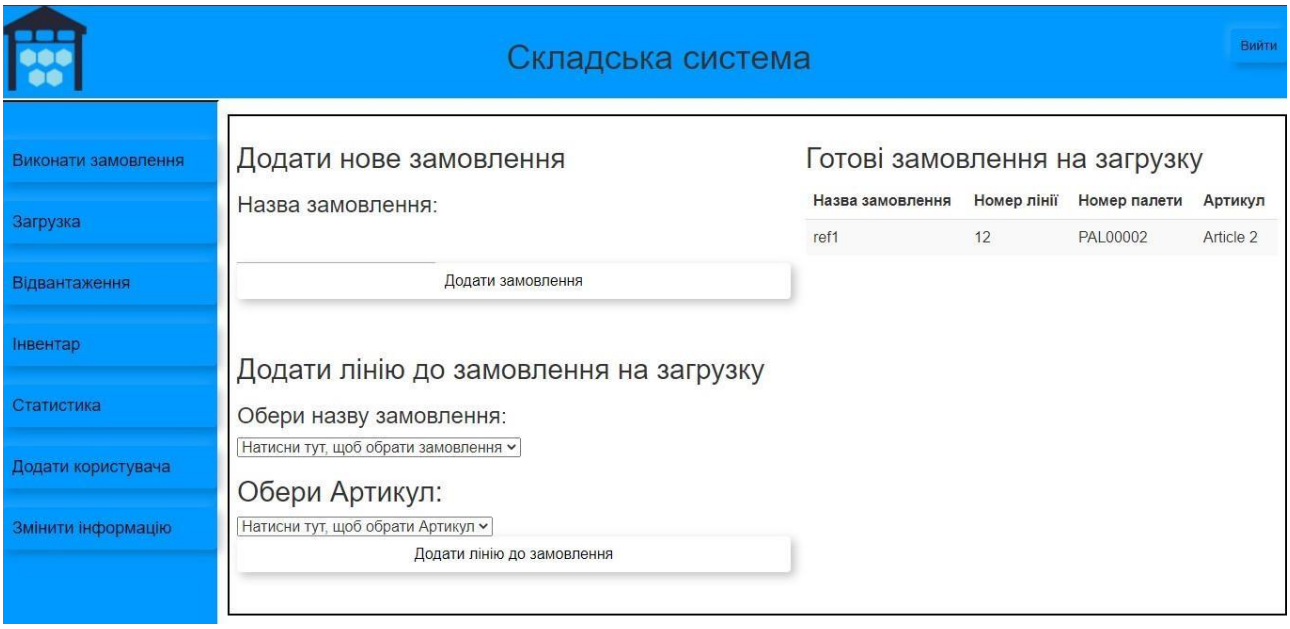

Рисунок 3.13. Сторінка для додавання замовлень на завантаження

Для створення замовлення користувач повинен вигадати назву замовлення і натиснути кнопку «Додати замовлення». Після цього, щоб додати лінію необхідно з випадаючого списку обрати створене замовлення і з другого випадаючого списку обрати необхідний артикул товару.

Після виконання цих дій в разі правильно обраних даних, можемо побачити, що замовлення з’явилося в таблиці в правій частині екрану.

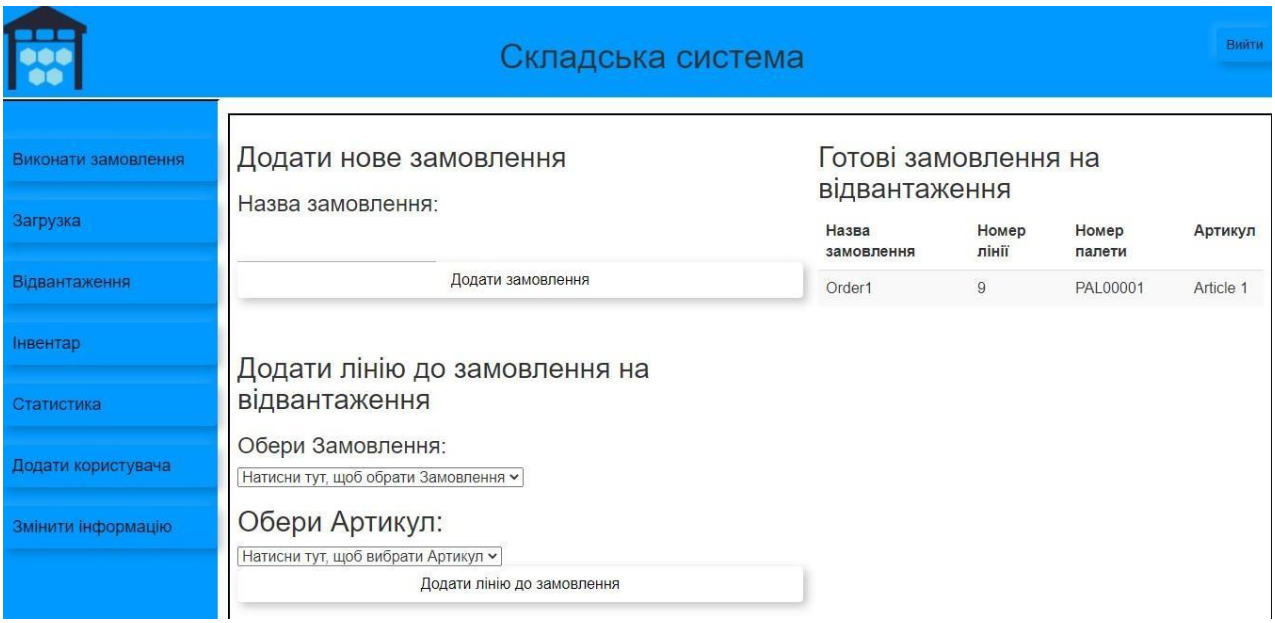

Рисунок 3.14. Сторінка «Відвантаження» з доданим готовим замовленням

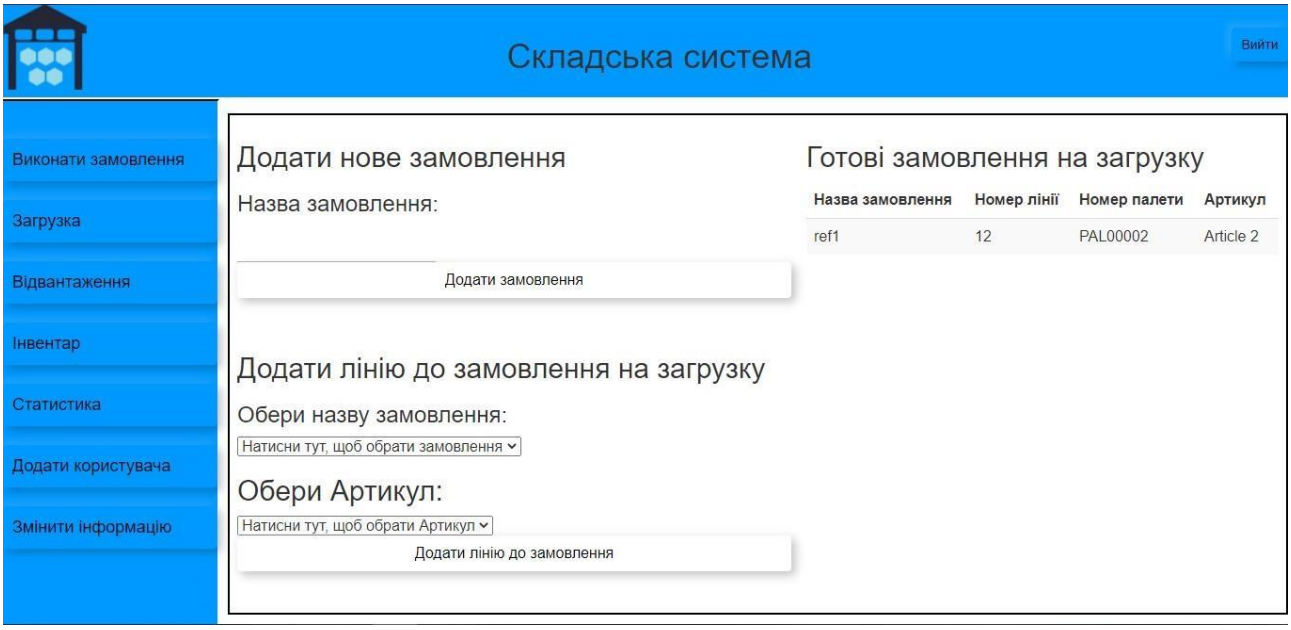

Рисунок 3.15. Сторінка «Загрузка» з доданим готовим замовленням

Перейшовши на сторінку «Виконати замовлення» бачимо що тут теж екран поділений на 2 частини. В лівій частині – таблиця з замовленнями на відвантаження, в правій – на завантаження. Для завершення замовлення користувачу необхідно з відповідного списку обрати замовлення, з яким необхідно працювати і натиснути кнопку «Завершити замовлення на відвантаження/завантаження».

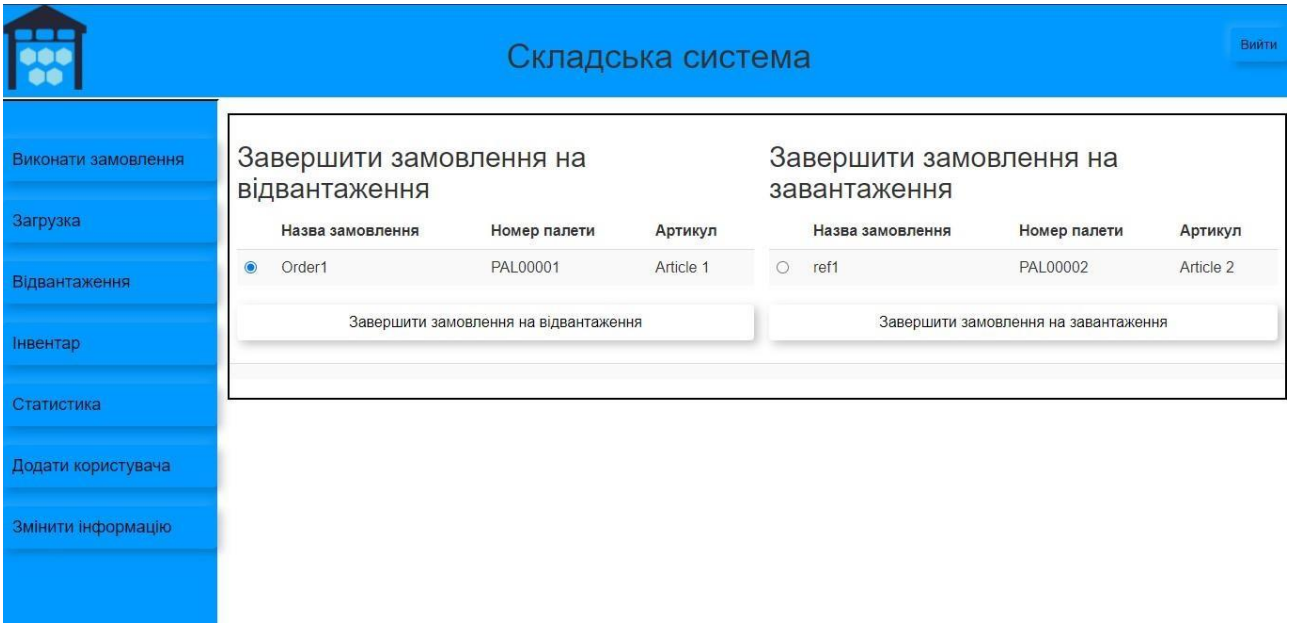

Рисунок 3.16. Сторінка для завершення замовлень

Як вказувалося вище – якщо в користувача є права «Адміністратора» - йому доступний функціонал для роботи з даними користувачів. Перейшовши на вкладку «Змінити інформацію», адміністратор може оновити певні дані, змінити роль користувача, чи взагалі повністю замінити всю інформацію про нього. Для цього в списку зліва необхідно обрати користувача, по відношенню до якого будуть проводитися дані маніпуляції, а в формі в правій частині ввести дані лише в ті поля, які потрібно оновити, а решту залишити без змін. Переконавшись в тому, що всі поля, які необхідно змінити заповнені (вони підсвічуватимуться іншим кольором), слід натиснути кнопку «Змінити вибрані поля».

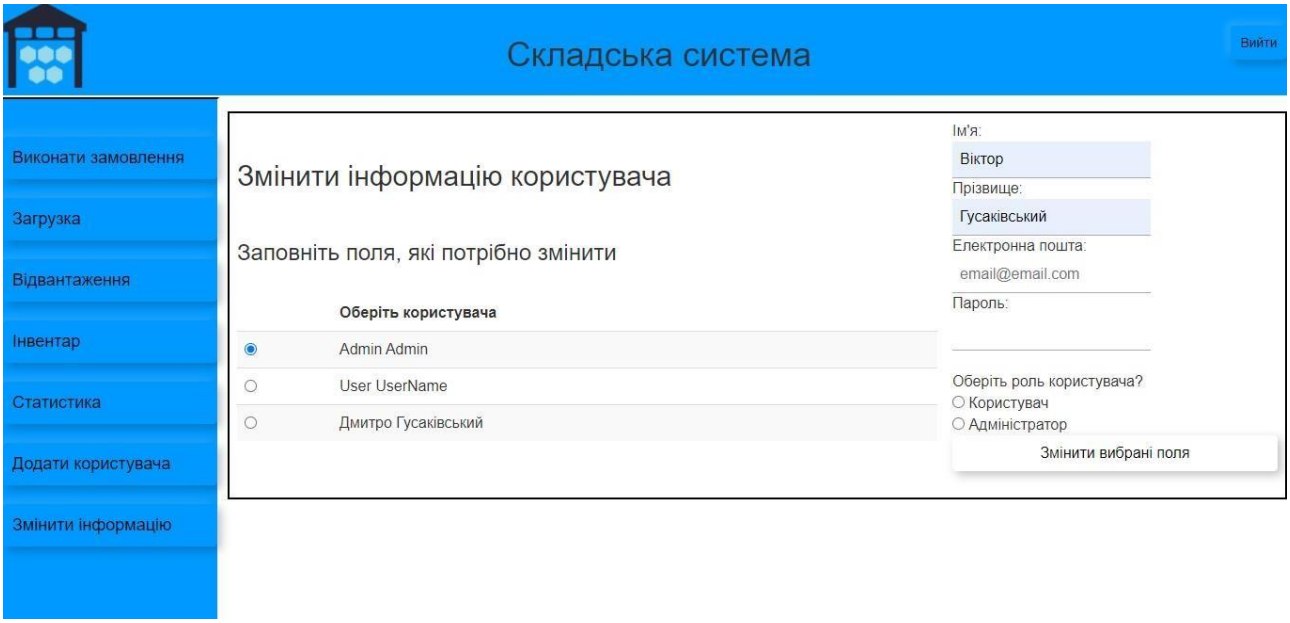

Рисунок 3.17. Сторінка для внесення змін в дані користувачів

Перейшовши на вкладку «Додати користувача» бачимо спеціальну форму для створення нового оператора в системі. Всі поля на даній формі повинні бути заповненими. В разі необхідності адміністратор може швидко очистити всі заповнені ним поля. Переконавшись, що дані введено вірно, натискаємо кнопку «Додати користувача».

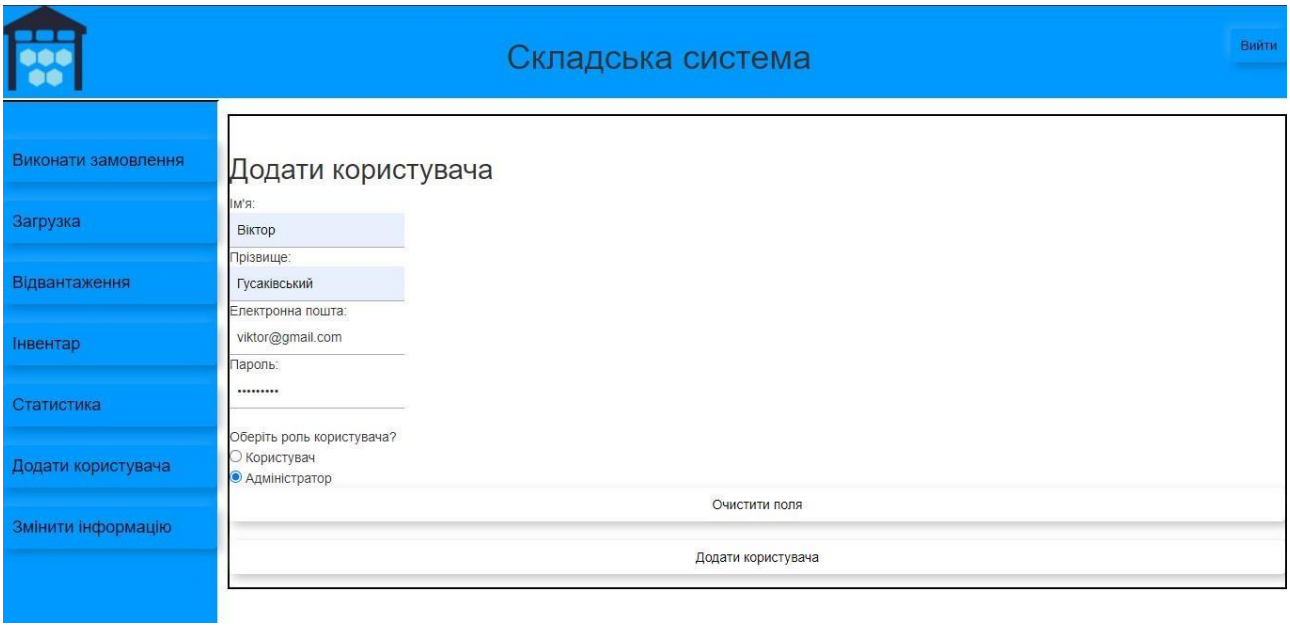

Рисунок 3.18. Сторінка для додавання нового користувача в систему

Висновки:

Під час виконання третього розділу дипломної роботи було успішно реалізовано програмну систему керування складськими запасами. Наведені UML-діаграми класів та бази даних, детальний опис структури пакетів застосунку, та опис кожного класу, або набору класів.

У розділі наведений загальний вигляд програмної системи та додана інструкція користувача з поясненнями для ознайомлення з системою.

ВИСНОВКИ

Результатом виконання дипломної роботи є створення програмної системи керування складськими запасами. Для цього було обрано мову програмування Java з допоміжними фреймворками, які вирізняються зручністю та простотою розробки веб-застосунків, наявністю великої кількості бібліотек для вирішення поставлених задач проектування серверу програмної системи, та інтерфейсу клієнтської частини.

Під час виконання дипломної роботи:

- дослідив теоретичні основи побудови програмної системи керування складськими запасами;
- проаналізував програмно-технічні рішення і види забезпечень типових веб-застосунків керування складом;
- спроектував і впровадив програмну систему керування складськими запасами.

Було проаналізовано існуючі рішення, виявлено їх переваги та недоліки, була досліджена загальна концепція веб-застосунку, проведено аналіз об’єкту програмування та визначено вимоги до інформаційного, програмного та технічного забезпечення.

Була надана інструкція користувача щодо використання програмної системи керування складськими запасами.

Основною перевагою розробленого проекту є простота та зрозумілість у використанні, незалежність від операційної системи, браузеру та технічних характеристик пристрою (смартфон, планшет або персональний комп’ютер), впровадження основного функціоналу на основі досліджених переваг та недоліків існуючих рішень.

Веб-застосунок має властивість масштабованості, що передбачає можливість впровадження нових функціональних можливостей та розширення існуючих.

Розроблена програмна система відповідає основними вимогам технічного завдання та може бути конкурентною серед аналогів завдяки впровадженим функціональним перевагам й простоті у використанні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Репнікова Н. Б. ТЕОРІЯ ЦИФРОВИХ СИСТЕМ УПРАВЛІННЯ / Н. Б. Репнікова, А. В. Писаренко. –Київ: НТУУ (КПІ), 2012. –87 с.
2. Офіційний сайт інформаційної системи управління складом WMS / Режим доступу: http://www.eme-wms.ru.
3. Кислий В.М.Логістика:Теорія та практика: навч. посіб. / Біловодська О.А., Олефіренко О.М., Соляник О.М К: Центр учбової літератури, 2010. 360с.
4. Черемних В. Технологія Клієнт-Сервер [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://it-black.ru/tekhnologiya-kliyent-server/.
5. Richardson L. RESTful Web APIs: Services for a Changing World / L. Richardson, M. Amundsen, S. Ruby., 2013. – 406 с.
6. Макфарланд Д. Нова велика книга CSS / Девід Макфарланд., Питер Пресс, 2020. – 720 с.
7. Структура і принципи WEB. WEB-сервери та принципи їх роботи з користувачем [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://dl.sumdu.edu.ua/textbooks/86975/413008/index.html.
8. Архітектура REST [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://habr.com/ru/post/38730/.
9. Опис продукту Intellij IDE [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://itpro.ua/product/visual-studio-2019enterprise/?tab=description.
10. H2 Database[Електронний ресурс] – 2015. – Режим доступу до ресурсу: http://h2database.com/html/main.html
11. Как выбрать систему управления складом [Електронний ресурс] – 2019. – Режим доступу до ресурсу: https://logist.fm/reshenia/kak-vybrat-sistemuupravleniya-skladom-reyting-wms-sistem-ukrainy-i-ih-primernaya-stoimost

12. СУБД (Система управления базами данных) складом [Електронний ресурс] – 2018. – Режим доступу до ресурсу: https://itglobal.com/ruru/company/glossary/subd-sistema-upravleniya-bazami-dannyh/
13. Анализ WMS-систем для автомаизации бизнес-процессов [Електронний ресурс] – 2020. – Режим доступу до ресурсу: https://habr.com/ru/post/528124/
14. WMS системы управления складом. WMS система – что это? [Електронний ресурс] – 2015. – Режим доступу до ресурсу: https://fb.ru/article/161588/wms-sistemyi-upravleniya-skladom-wms-sistema--chto-eto
15. Сравнительный анализ функциональных возможностей – система WMS и ERP [Електронний ресурс] – 2016. – Режим доступу до ресурсу: https://www.quantum-int.com/ru/news/sravnitelnyj-analiz-funkcionalnyxvozmozhnostej-wms-i-erp/
16. Системна документація Spring[Електронний ресурс] – 2021. – Режим доступу до ресурсу: https://spring.io/projects/spring-framework
17. Системна документація Spring Boot[Електронний ресурс] – 2021. – Режим доступу до ресурсу: https://spring.io/projects/spring-boot
18. Системна документація Spring MVC[Електронний ресурс] – 2021. – Режим доступу до ресурсу: https://spring.io/guides/gs/serving-web-content/
19. Advantages and disadvantages – web apps [Електронний ресурс] – 2015. – Режим доступу до ресурсу: https://objectiveit.com/blog/the-advantages-anddisadvantages-of-web-apps/
20. Desktop or web-application – what to develop [Електронний ресурс] – 2021. – Режим доступу до ресурсу: https://exoft.net/desktop-or-web-applicationwhat-to-develop/
21. Progressive web-apps [Електронний ресурс] – 2021. – Режим доступу до ресурсу: https://brainhub.eu/library/progressive-web-apps-advantagesdisadvantages/
22. Introduction of dbms [Електронний ресурс] – 2019. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/introduction-of-dbms-databasemanagement-system-set-1/

23. Types of database management systems [Електронний ресурс] – 2019. – Режим доступу до ресурсу: https://www.includehelp.com/dbms/types-ofdatabase-management-system.aspx

ДОДАТКИ

Додаток А Клас Application.java

Так як Spring працює з бінами (Bean), анотація @Bean необхідна для метода CommandLineRunner(), щоб запустити його першим одразу після запуску застосунку.

```
import ViHu.domain.*;

import ViHu.repository.*;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;



@SpringBootApplication

public class Application {


    public static void main(String[] args) {

        SpringApplication.run(Application.class, args);
    }



    @Bean

    public CommandLineRunner demo(final UserRepository userRepository,

                                  ArticleRepository articleRepository,

                                  PalletRepository palletRepository) {

        return strings -> {

            //Admin user

            User user = new User("Admin","Admin","admin@gmail.com","admin",
UserType.ADMIN);

            userRepository.save(user);
            //Simple user
            User user2 = new User("User", "UserName","user@gmail.com", "pass",
UserType.USER);

            userRepository.save(user2);
```

```
            //Articles

            Article artc1 = new Article("Article 1","Article 1
description",350.5,80);

            Article artc2 = new Article("Article 2","Article 2 description",400,100);

            articleRepository.save(artc1);

            articleRepository.save(artc2);



            //Pallets

            Pallet pallet1 = new Pallet("PAL00001", artc1.getName());

            Pallet pallet2 = new Pallet("PAL00002", artc2.getName());

            palletRepository.save(pallet1);

            palletRepository.save(pallet2);

        };

    }



}
```

Додаток Б Приклад метода з класу контролер

Клас SimpleController.java

Метод для додавання користувача addUser()

```
@RequestMapping("addUser")

public String addUser(@RequestParam String nameReg,

                      @RequestParam String surnameReg,
                      @RequestParam String emailReg,
                      @RequestParam String passwordReg,
                      @RequestParam String roleReg,
                      Model model) {




    if (!nameReg.equals("") || !surnameReg.equals("") || !emailReg.equals("") ||
!passwordReg.equals("")) {

        UserType role;

        if (roleReg.equals("USER")) {

            role = UserType.USER;

        } else {

            role = UserType.ADMIN;

        }
        User user = new User(nameReg, surnameReg, emailReg, passwordReg, role);

        userRepository.save(user);

        return "redirect:/personalCab";

    }
    model.addAttribute("errorMessage", "You must fill all the fields first!");

    return "progError";

}
```

Додаток В Приклад того, як в Hibernate реалізовуються налаштування для створення бази даних і доступу до неї

Клас User.java

```
import lombok.Data;



import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;
 import
javax.persistence.Id;




@Data

@Entity public
class User {

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)
     private int
id;
```

```
    private String name;

    private String surname;

    private String email;

    private String password;

    private UserType role;


    public User() {

    }


    public User(String name, String surname, String email, String password,
UserType role) {

        this.name = name;

        this.surname = surname;

        this.email = email;

        this.password = password;

        this.role = role;

    }


    public String getSurname() {

        return surname;

    }

      public void setName(String
name) {
         this.name =
name;

    }

      public void setEmail(String
email) {
         this.email =
email;

    }
```

```
    public void setPassword(String password) {
        this.password =
password;
    }
```

```
    public int getId() {
         return id;

    }


    public String getName() {
         return name;

    }


    public String getEmail() {
         return email;

    }


    public String getPassword() {

        return password;

    }

}
```

Інтерфейс UserRepository.java

```
import org.springframework.data.repository.CrudRepository;
 import
ViHu.domain.User;

  import
java.util.List;



public interface UserRepository extends CrudRepository<User, Integer> {
    User getUserById(int id);

    User getUserByEmail(String email);

    List<User> findAll();




}
```

Додаток Г Задачі, що вирішуються за допомогою систем керування складськими запасами

Прийом товарів:

Отримання товарів у реальному часі через радіотермінали абопаперових документів

Друк штрих-кодів;

Гнучка ідентифікація і з попередніми замовленнями на придбання відпостачальників і без них;

Обов’язок зі сторони складу на гідне зберігання товару;

1.5)Моніторинг відповідності та корекція даних за необхідності.

Зберігання:

Автоматичне розміщення на території складу або розміщення піднаглядом персоналу;

Спеціальні правила розташуванні товару на території складу дляоптимального використання потужностей та / або продуктивності для проведення інвентаризаційних операцій;

Комплексні критерії побудови комірок зберігання;

Створення завдань для зберігання;

Підготовка роздрібних товарів від різних постачальників дозберігання.

Автоматизація одночасного прийому та відправки товару:

Перегруз товарів на території складу, отриманих для доставкиклієнтам;

Транзит товару через склад.

Гнучка обробка замовлень та групові замовлення:

Комплексні групування замовлень.

Обробка та видача замовлень у групах з оптимізацією процесів та ресурсів;

Об’єднання малих та розподілення великих партій товарів;

Ідентифікація товару, яку можна налаштувати для упаковки під час доставки та повернення.

Функція поповнення запасів:

Настроювані параметри для потреб наповнення;

Додавання з неповних піддонів товару;

Спільне завантаження групи товарів на піддон;

Автоматичне генерування та надсилання даних для поповнення запасів в зонах для підбору;

Стратегії поповнення спеціальних зон товарами, які можна налаштувати під власні потреби;

Різні варіанти наповнення (штука, ящик, піддон).

Комплектація замовлень:

Автоматичне генерування та надсилання інформацію для замовленняробітникам, що займаються підбором товару;

Пряма комплектація на піддон з урахуванням ергономічних вимог, атакож розмірів, ваги та інших параметрів товару;

Комплектація для завантаження на конвеєрні стрічки;

Округлення товару залежно від партії;

Підтримка різного типу підбору товару: штуками, ящиками, повнимипалетами;

Комплектація з фіксуванням даних за допомогою радіотерміналів таспеціальних стікерів зі штрих кодами;

Комплектація за допомогою голосових команд;

Пакування:

Різні варіанти пакування товару;

Коригування замовлення під час збирання;

Генерування спеціальних транспортних накладних для контейнерів, що відвантажуються, та відстежування їх;

Завантаження товару:

Планування перевезення вантажів з урахуванням пріоритетів;

комбінування товару під час завантаження, залежно від замовлення на доставку;

Завантаження, перевірка та закриття автомобіля з фіксацією радіотерміналів;

Визначення (вибір) перевізника;

Відповідність маркування;

8.6)Складання додаткових документів.

Управління запасами:

Відстежування контейнерів;

Повна функціональність для роботи з важкими товарами(понад 500кг);

Гнучкість у переміщенні та регулюванні товарів;

Часткова інвентаризація(за потреби);

Повна фізична інвентаризація з підтвердженням ваги товару на вході та виході;

Відстежування стану і отримання інформації про товар в режимі реального часу;

Консолідація товару;

конфігурація площі складу та його зонування;

відстеження цілісності атрибутів товару (партія, код, серійний номер);

Відстеження дати та часу реалізації товарів;

Гнучка система відправки, поділу товару на партії, переміщення запасів.

Гнучкі методи підбору товару для відвантаження LIFO, FIFO, FPFO, FEFO, BBD

Управлінська робота для персоналу:

Автоматичне створення та надсилння завдання для:

- Прийому товару;
- Розміщення товару на складі;
- Переміщення товару;
- Підрахувати коштів за продаж товарів;
- Поповнення запасів;
- Комплектація замовлень;
- Погрузка товару;
- Відправка товару.

Планування роботи розподільчого центру:

Планування графіку відповідно до пріоритетності товару.

Постійна підтримка робітників за періодична зміна групи завдань для кожного з них;

Переміщення персоналу на інші зони складу.

Обробка контейнерів:

Нанесення спеціального маркування та інформації про ліцензію /патент;

Поміщення декількох різних предметів в один контейнер;

Ідентифікація товарів за упаковкою під час транспортування та повернення;

Встановлення обмежень на спільне зберігання різних товарів.

Управління складами та виробничими потужностями:

Визначає точне розташування комірки зберігання;

Оптимізація сховища;

Автоматичне заповнення та переведення товару на допоміжні склади в разу нестачі місця;

Переміщення товару всередині організації;

Контроль та правильне поводження з небезпечними матеріалами;

Контроль складського обладнання та планування дозаправок техніки, для того, щоб не заповільнювати роботу складу.

Облік людських ресурсів:

Реєстрація робочого часу;

Облік виконаних працівником завдань;

Звітність про людські ресурси;