МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри

_____ Литвиненко О.Є.

«___» _____2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: *Web*-додаток управління замовленнями в інтернет-магазині

Виконавець: Опанасенко В.К.

Керівник: Масловський Б.Г.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп’ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Литвиненко О.Є.
«_____»_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)
Опанасенка Валерія Кириловича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту):
Web-додаток управління замовленнями в інтернет-магазині
затверджена наказом ректора від «4» лютого 2021 р. №135/ст.
2. Термін виконання роботи (проєкту): з 17.05.2021 по 20.06.2021
3. Вихідні дані до роботи (проєкту):
1) Документація створення *web*-додатку;
2) Документація *Java;*
3) Документація *Spring*.
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) Аналіз предметної області;
2) Моделювання та проектування;
3) Програмна реалізація *web*-додатку управління замовленнями інтернет-магазину.
5. Перелік обов’язкового графічного матеріалу:
1) *UML*-діаграми;
2) Презентація.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомитись з постановкою задачі дипломного проектування	17.05.2021 – 19.05.2021	
2	Вивчити спеціальну літературу і технічну документацію	20.05.2021 – 21.05.2021	
3	Розглянути основні засоби і методи	22.05.2021 – 23.05.2021	
4	Проаналізувати методи і алгоритми програми	16.06.2021 – 18.06.2021	
5	Провести описання роботи програми	16.06.2021- 18.06.2021	
6	Оформити пояснювальну записку	19.06.2021	
7	Підготовити графічний демонстраційний матеріал	20.06.2021	

7. Дата видачі завдання**:** «17» травня 2021 р.

Керівник дипломної роботи (проєкту) _____ Масловський Б.Г.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання_____ ОпанасенкоВ.К.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи на тему «*Web*-додаток управління замовленнями в інтернет-магазині»: 46 с., 30 рис., 20 літературних джерел, 1 додаток.

Ключові слова: *ВЕБ-ДОДАТОК, УПРАВЛІННЯ ЗАМОВЛЕННЯМИ, JAVA, SPRING, МОДЕЛЮВАННЯ, ІНТЕРНЕТ-МАГАЗИН, БІЗНЕС-ПРОЦЕСИ, АРХІТЕКТУРА ДОДАТКУ, ІНФОРМАЦІЙНА СИСТЕМА*

Об'єкт дослідження – створення веб-додатку.

Предмет дослідження – веб-додаток управління замовленнями в інтернет-магазині.

Мета дипломного проекту – розробити веб-додаток управління замовленнями в інтернет-магазині.

Методи проектування – теоретичне ознайомлення з існуючими аналогами, використання здобутих знань для розробки додатку.

Результати дипломного проекту можна розділити на дві групи: теоретичні та практичні. Теоретичні результати можна використовувати при розробці програмного забезпечення чи ознайомлення з ним, у якому постає проблема вибору інструментів та концепцій, практичні – використовувати в якості необхідного додатку.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 6
ВСТУП..... 7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1. Огляд досліджень і публікацій 8
1.2. Аналіз програмних продуктів-аналогів 14
1.3. Постановка задачі 19
1.4. Вибір засобів реалізації 20
1.5. Висновки до розділу 21
РОЗДІЛ 2 WEB-ДОДАТОК УПРАВЛІННЯ
ЗАМОВЛЕННЯМИ В ІНТЕРНЕТ-МАГАЗИНІ 22
2.1. Моделювання процесу роботи інтернет-магазину 22
2.2. Проектування інформаційної системи 23
2.3. Проектування *web*-додатку 28
2.4. Висновки до розділу 29
РОЗДІЛ 3 ПРОГРАМА УПРАВЛІННЯ
ЗАМОВЛЕННЯМИ ІНТЕРНЕТ-МАГАЗИНУ 30
3.1. Структура веб-додатку 30
3.2. Програмна реалізація 33
3.3. Висновки до розділу 38
ВИСНОВКИ 40
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ
ВИКОРИСТАНИХ ДЖЕРЕЛ 42
ДОДАТОК А 44

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

DNS – Domain Name System

HTML – HyperText Markup Language

AWS – Amazon Web Services

IP – Internet Protocol

PHP – Hypertext Preprocessor

SQL – Structured Query Language

ID – Identifier

FIFO – First In, First Out

CDN – Content Delivery Network

CSS – Cascading Style Sheets

JDBC – Java DataBase Connectivity

ORM – Object-Relational Mapping

JMS – Java Message Service

ВСТУП

Під мережею Інтернет розуміють об’єднання комп’ютерів всього світу у єдине ціле. Завдяки цьому принципу життя в нашому світі стало простішим. Проте, не зважаючи на це, про минулі досягнення людства ніхто не забуває, тому інтернет-магазини наразі є доволі популярним явищем.

Чому інтернет-магазин? Якщо розглядати плюси й мінуси ведення бізнесу в мережі, то можна побачити просту закономірність, завдяки котрій ця сфера розвивається й є популярною на даний час.

По-перше, це економія часу – продавець й покупець за допомогою запитів в пошукових сервісах, наприклад, *Google,* можуть знайти один одного без зайвих зусиль. Варто лише ввести в пошуковий рядок свій запит, наприклад, «замовити одяг» чи «придбати ноутбук» й пошуковий сервіс за лічені секунди надасть не одну сотню рекомендацій, де це можна зробити, у кого, який продукт є в наявності, а на що потрібно буде зачекати.

По-друге, це економія фізичного простору – всі бізнес-процеси, окрім доставки замовлення, звісно, проходять в мережі. Завдяки чому можна зробити висновок, що простіше купити серверний простір, ніж домовлятись за аренду чи придбання приміщення – звісно, масштаб залежить від виду та сфери діяльності, проте плюси використання хмарного зберігання даних очевидні. Проте, це можна вважати й мінусом – адже питання безпеки данних постає в першу чергу.

По-третє, зберігається більшість налагоджених бізнес-процесів, як і в реальному житті, потрібно знайти постачальника, оформити відповіді папери й документи на ведення цієї діяльності. В кінці кінців власник бізнесу чи інтернет-магазину зіштовхнеться з питанням адміністрування заявок – а це тема цієї роботи. Адже одним з головних питань все ж залишається – кому і як отримати замовлення, в котрий час – тому можна зробити висновок, що завдання розробки web-додатку для управління замовленнями інтернет-магазину є актуальним на даний час.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В даному розділі будуть розглянуті останні дослідження та публікації на тему створення веб-додатків, а також сформульовано завдання побудови веб-додатку на основі розглянутих продуктів-аналогів та засобів для реалізації.

1.1 Огляд досліджень і публікацій

Для того, щоб розуміти як побудувати веб-додаток варто розібратись в тому, як працюють веб-додатки. На рисунку 1.1. зображена концептуальна схема роботи веб-додатків. [1]

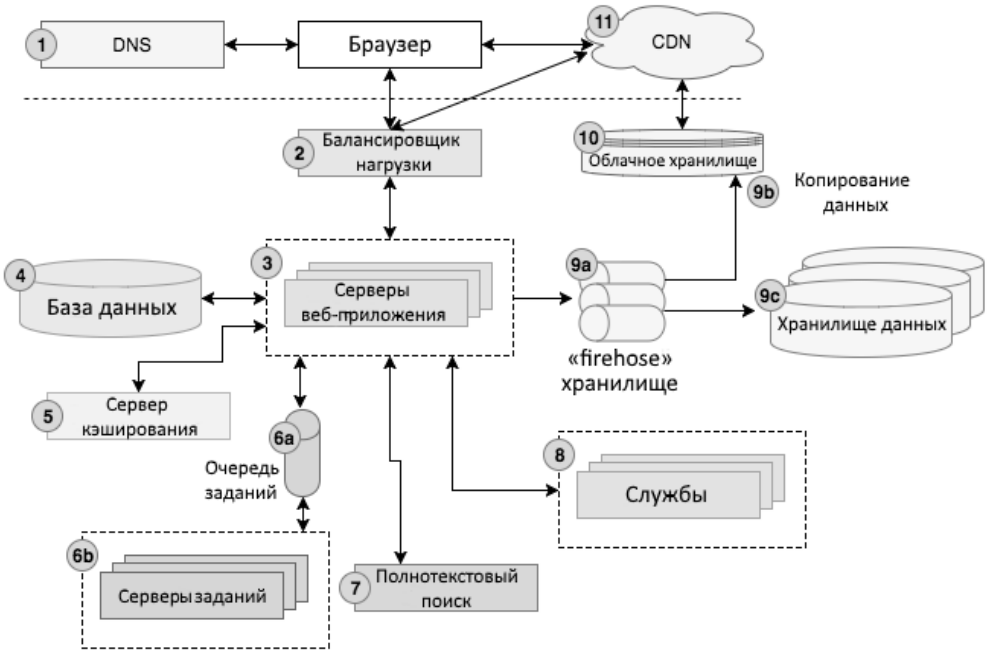

Рис. 1.1. Концептуальна схема роботи *web*-додатку

DNS [2]

Під цим терміном розуміють систему доменних імен, що являється базовою технологією для роботи інтернету. На цьому рівні відбувається забезпечення пошуку пари з доменного імені та *IP*-адреси, наприклад,

google.com та адреса 85.129.83.120, що дозволить відправити запит на сервер задля отримання результату – переходу на потрібну сторінку. Можна привести аналогію з телефонними номерами – різниця між *IP*-адресом да доменним іменем така ж як і між номером телефону та власником цього номеру телефону. Тобто, можна сказати, що доменне ім'я та *IP*-адреса – це як телефонний номер та абонентська книга. Таким чином, можна вважати *DNS* телефонною книгою всієї мережі інтернет.

Балансувальник навантаження [3]

Перед тим як описати балансувальник навантаження, варто сказати лекілька слів про горизонтальне та вертикальне масштабування проекту додатку. В чому ж полягає різниця між цими масштабуваннями? Перш за все, горизонтальне масштабування має на увазі збільшення кількості виконуваних пристроїв, тобто, додаванням пристроїв для збільшення загальної потужності, в той час як вертикальне масштабування має на увазі збільшення потужності конкретної машини й конкретних характеристик – наприклад, оперативної пам'яті. Зазвичай проект масштабують горизонтально, як мінімум, через те, що це не приводить до поломок. Сервери також можуть вийти з ладу з незрозумілих причин – а причину ще потрібно знайти – іншими словами: мережі деградують. Завдяки тому, що в наявності буде декілька серверів можливе збільшення відмовостійкості – це дозволяє, наприклад, серверам пережити відключення світла в мережі. З іншого боку, горизонтальне масштабування дозволяє мінімально пов'язати різні частини проекту – такі як веб-сервери, бази даних та інші, тому що кожна з таких частин працює на іншому сервері. Й в один момент можна прийти до висновку, що вертикальне масштабування стає неможливим, адже в світі відсутній комп'ютер такої потужності, що міг би виконувати всі обчислення додатку. Для прикладу розглянемо компанію *Google,* хоча й менші компанії також зіштовхуються з такою ж проблемою. Іншим прикладом є компанія *StoryBlock,* що використовує від 150 до 400 *AWS*-машин EC2 в будь-який момент часу роботи компанії. Завдяки вертикальному масштабуванню досить складно отримати цю необхідну потужність. Повертаючись до балансувальника навантаження, саме завдяки ним стає можливе горизонтальне масштабування – завдяки ним відбувається

розподілення запитів на сервери, що використовує сервіс чи додаток. Вони направляють вхідні запити на один з серверів та повертають назад відповідь до користувача. Будь-який сервер обчислює запити однаково, тому балансувальник займається розподіленням завдань задля того щоб не було перевантаження на серверах. Ось такий простий і зрозумілий принцип роботи, що лежить в основі.

Сервери веб-додатків [4]

На перший погляд сервери веб-додатків є простими – адже вони виконують функцію обробки запиту та повернення *HTML* коду браузеру. Зазвичай, для виконування своєї роботи ці сервери мають звертатись до різних інфраструктур, що знаходяться на сервері – бази даних, сервери кешування, черги завдань, служби пошуку та інші. Як було сказано вище, необхідно, як мінімум, пара серверів, бажано навіть більше, що підключені до регулювальника навантаження для обчислень запитів користувачів. Щоб створити сервер додатку перш за все необхідно обрати мову програмування, завдяки якій буде створено сервер, це може бути *Node.js, PHP, Java, C#, .NET* та інші мови, а також слід обрати *MVC*-фреймворк в залежності від мови.

Сервер баз даних [5]

Більшість сучасних веб-додатків використовують, як мінімум, одну базу даних для зберігання тієї чи іншої інформації. Вони використовуються як для зберігання інформації, додання інформації, зберігання, пошуку, обновлення, а також видалення та здійснення обчислень й розрахунків над даними. До того ж, у кожної серверної служби може бути в наявності власна база даних, що відокремлена від всіх інших, крім цього в більшості випадках сервери веб-додатків здійснюють спілкування з сервером завдань напряму. Варто також згадати про *SQL* та *NoSQL*. *SQL* – це структурована мова запитів, що була створена в 1970-х роках для взаємодії із реляціонними наборами даних. Ця база даних зберігає в собі таблиці, що пов’язані між собою загальними ключами. Зазвичай ключем є ціле число, що є відображенням *ID*. Нижче буде розглянутий типовий приклад зберігання інформації про історії адресів користувачів. Зображено дві таблиці, перша – це користувач (*user*), а наступна – це адреса користувача (*user_addresses*). Вони пов’язані між собою ідентифікатором користувача – *ID(INT)*. Ці таблиці є

пов'язаними, адже ID в таблиці *users*, та *ID* в таблиці *user_addresses* – є відображенням одних і тих самих даних, тобто, можна сказати, що цей стовбець є зовнішнім ключем (рисунок 1.2.).

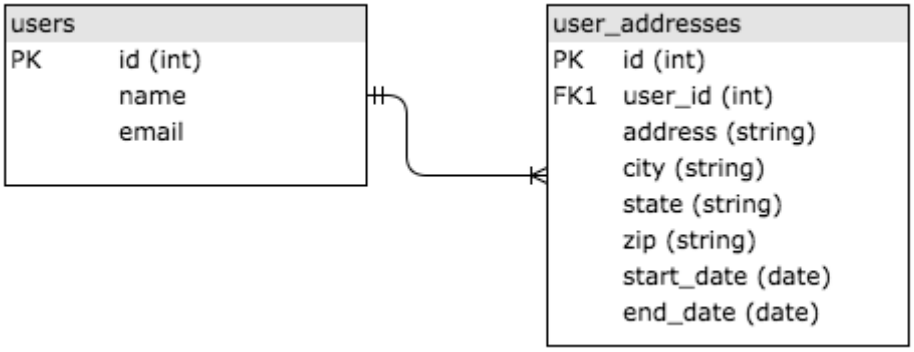

Рис. 1.2. Таблиця *SQL*

Кеш

Служба кешу представлена у вигляді простого місця, що зберігає дані. Форматом зберігання даних є формат ключ-значення, що дозволяє зберігати та віднайти інформацію за час, що близький до лінійного – O. Зазвичай ці служби використовуються задля збереження часу на обчислення складних задач, а пізніше скористуватись (відобразити їх) із вмісту кешу, а не обчислювати це значення ще раз. Додатки можуть використовувати кеш задля збереження запитів до баз даних чи зовнішніх служб, або для збереження *HTML* коду заданої сторінки, чи іншого.

Черги задач

Більша частина веб-додатків виконує дії, що напряму не пов'язані з відповіддю на запити користувачів, тим паче, це відбувається асинхронно, в фоновому режимі. Візьмемо для прикладу *Google,* ця пошукова система сканує та індексує простір інтернету для того щоб повернути релевантні результати пошуку. В той же час цей сервіс не робить це під час кожного запиту, а сканує мережу асинхронно, що дозволяє обновити пошукові індекси в результаті роботи сервісу. Виконувати асинхронну роботу дозволяють різні архітектури, проте поширений варіант – це черга задач, що складається з наступних компонентів: черги задач, котрі повинні бути виконані, та декількох робочих серверів, що оброблюють ці завдання з черги. В чергах зберігаються списки задач, що будуть виконані

асинхронно. Щоразу, коли додаток має виконати котрусь задачу по розкладу, наприклад, вона додається в чергу. Найбільш просте рішення черг – це *FIFO* – перший елемент був доданий, перший оброблений, проте більшість систем потребують балансувальника черг.

Повнотекстовий пошук

Багато, якщо не більшість, веб-додатків підтримують функцію пошуку за допомогою текстового вводу, в якому додаток повертає релевантні результати. Технологія, що використовується для цього, зазвичай носить назву «повнотекстовий пошук» – для цього використовується інвертований індекс для швидкого пошуку документів, що містять ключові слова запиту. (рисунок 1.3.).

Documents (Photo titles)	
id	title
1	Man running in the mountains
2	Mountains with snow
3	Man running marathon

Inverted Index	
keyword	photo_ids
man	1, 3
running	1, 3
mountains	1, 2
snow	2
marathon	3

Рис. 1.3. Повнотекстовий пошук

В цьому прикладі наявні три заголовки, що реформуються в інвертовані індекси – завдяки чому пошук стає більш простим через використання ключових слів.

Служби

Під час масштабування проекту зазвичай настає такий момент, коли проект потребує відокремлення певних дій – завдяки чому з’являються служби, що виконують ці дії. Можливий такий варіант, що ці служби є «прихованими», проте, інші сервіси чи служби взаємодіють з ними.

Сховище даних

Життєвий цикл компанії в більшості випадків залежить від того, як компанія оброблює та працює з даними. Більшість сучасних компаній під час досягнення певного масштабу переходить на один й той самий спосіб організації, зберігання та аналізу даних. Це все відбувається в три етапи: перший – додаток відправляє дані в потоковий інтерфейс задля збереження та поглинання даних – в так звані *firehose*-сховища, що як правило зберігають інформацію щодо дій користувача. Доволі часто відбувається так, що дані, котрі не були оброблені переміщуються до інших таких сховищ. Найбільш популярні технології для цього процесу – *AWS Kinesis* та *Kafka*. Вихідні, а також кінцеві оброблені дані та додаткові дані зберігаються зазвичай в хмарному сховищі. *AWS Kinesis* пропонує рішення, що має назву *Firehose*, що дозволяє зберігати необроблені дані в хмарному сховищі *S*3. Це рішення доволі просте в налаштуванні – оброблені та додаткові дані завантажуються в сховище для подальшого аналізу.

Хмарне сховище

Хмарне сховище – це простий та масштабований засіб зберігання та обміну інформацією через інтернет – визначення, що дає *AWS*. Це сховище використовується для зберігання будь-якого роду та виду даних, наприклад, кожен елемент локальної файлової системи можна зберегти в такому сховищі – завдяки перевагам *Restful API* через *HTTP*. Наприклад, рішення від Амазон *S*3 є одним з найпопулярніших хмарних сховищ, його часто використовують для зберігання як фото, відео, аудіо, так і *CSS/JavaScript* файлів, даних щодо дій користувачів та тому подібного.

CDN [6]

Рис. 1.4. Приклад використання *CDN*

Під *CDN* розуміють систему доставки контенту – ця технологія дозволяє відправляти статичні файли *HTML, CSS, JavaScript* набагато швидше ніж вихідний сервер. Завдяки поширенню контенту через *CDN* користувачі виграють в часі, адже це займає менше часу. Наприклад, на рисунку 1.4. зображено, як користувач з Іспанії робить запит до сайту, що знаходиться на іншому континенті, завдяки використанню *CDN* йому не потрібно чекати, поки сервер з Америки отримає запит й відправить відповідь назад – відповідь приходить з серверу *CDN* в Європі, що є набагато швидше й практичніше, адже не потрібно чекати відповіді від повільного крос-атлантичного *HTTP*-запиту.

1.2 Аналіз програмних продуктів-аналогів

Нижче будуть розглянуті продукти-аналоги, на основі яких буде побудований web-додаток для управління заявками інтернет-магазину.

Quickbooks Commerce [7]

Позиціонують себе як найкраще рішення для малого бізнесу. Раніше *TradeGecko, Quickbooks Commerce* – це централізована платформа для управління замовленнями для малого бізнесу (інтерфейс зображено на рисунку 1.5.). Зараз *Quickbooks Commerce* доступна лише для нових клієнтів *Quickbooks Online*. Але вони працюють над розширенням для решти користувачів. Якщо ви користувач

Quickbooks Online, *Quickbook Commerce* є доступним варіантом – цінові плани починаються з 20 доларів на місяць протягом перших 12 місяців. З його допомогою ви зможете перераховувати товари за різними продажами, оновлювати рівні запасів, налаштовувати автоматизовані робочі процеси доставки та інтегрувати свої фінансові дані з *Quickbooks Online*. Сказавши це, *Quickbooks Commerce* все ще найкраще підходить лише для малого бізнесу: не підтримують продавців, які продають понад 30 тис. замовлень на місяць, більше 20 тис. *SKUS*, або предмети інвентарю з індивідуальними серійними номерами.

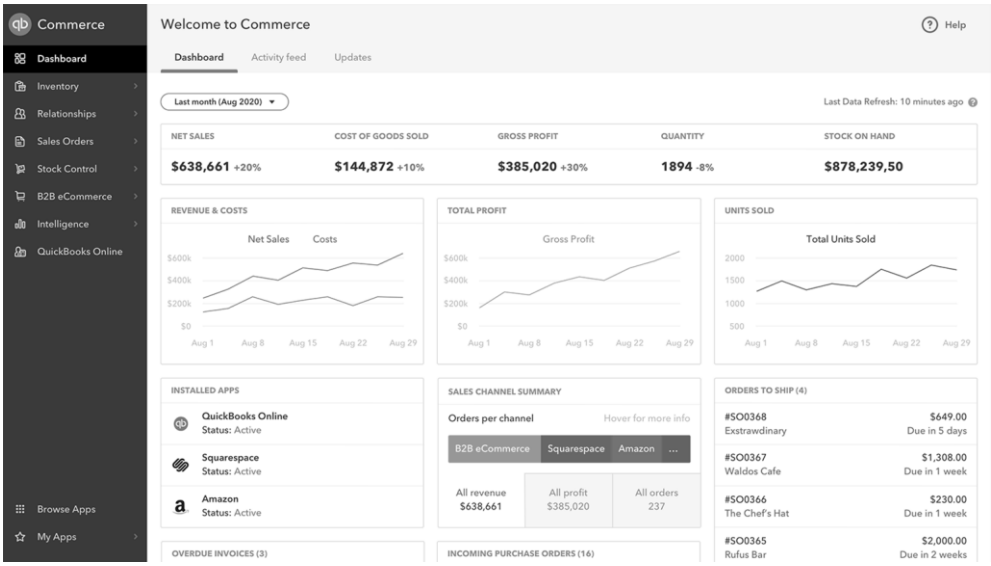

Рис. 1.5. *Quickbooks Online*

Brightpearl

Найкраще для підприємств електронної комерції середнього та великого бізнесу. *Brightpearl* – це універсальна операційна платформа (інтерфейс якої зображений на рисунку 1.6.). Він інтегрується з такими популярними каналами продажів, як *BigCommerce*, *Shopify*, *Amazon* та *eBay*. Вони також пропонують такі ключові функції, як: автоматичне виконання замовлення, управління поверненнями, кілька каналів продажів та інтеграція *POS* для Інтернет-покупки, отримання в магазині. Завдяки функціональності *ERP*-класу, *Brightpearl* – це система, яка масштабується відповідно до вашого бізнесу, коли ви переходите від 1000 замовлень на місяць до 30000. Це також означає, що *Brightpearl* може розмити лінії *ERP* та *OMS*, а також може краще обслуговувати середній та великий

бізнес завдяки їх функціоналу. Продавці повинні зв’язатися з компанією *Brightpearl*, щоб отримати інформацію про ціни.

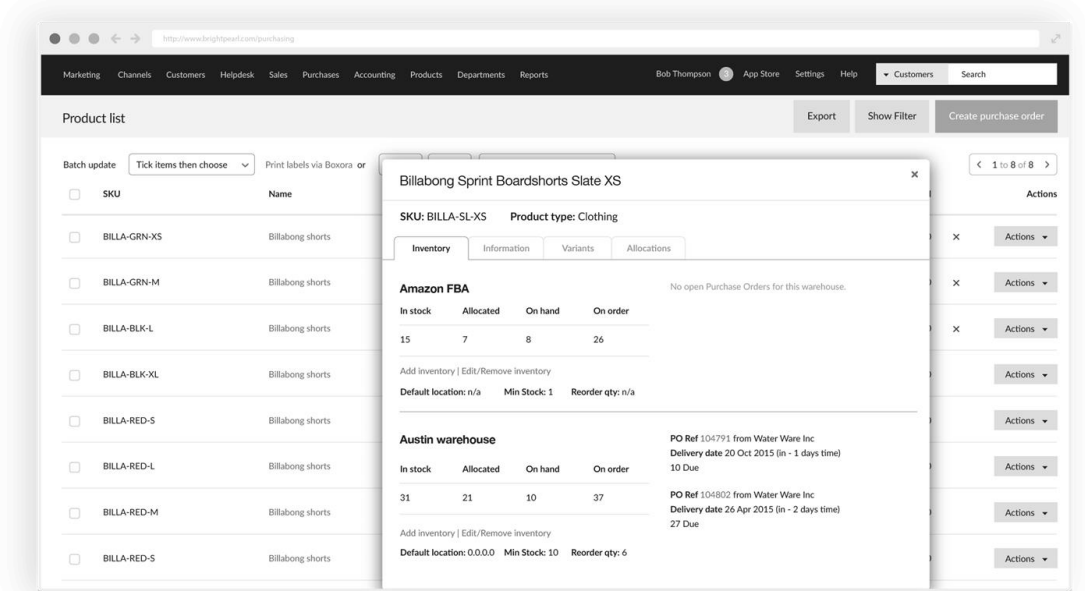

Рис. 1.6. *Brightpearl*

Skubana

Найкраще для малого та зростаючого бізнесу. *Skubana* – це також комплексне рішення, яке охоплює більше, ніж основні функціональні можливості *OMS*, включаючи управління ресурсами, інтеграцію каналів продажів, відстеження замовлень та комплексну звітність та аналітику (інтерфейс зображено на рисунку 1.7.). Часто порівняно з *Brightpearl*, *Skubana* може не досягти деяких функцій та опцій. Він також доступний лише як веб-платформа. З цих причин *Skubana* зазвичай краще обслуговує малий та зростаючий бізнес.

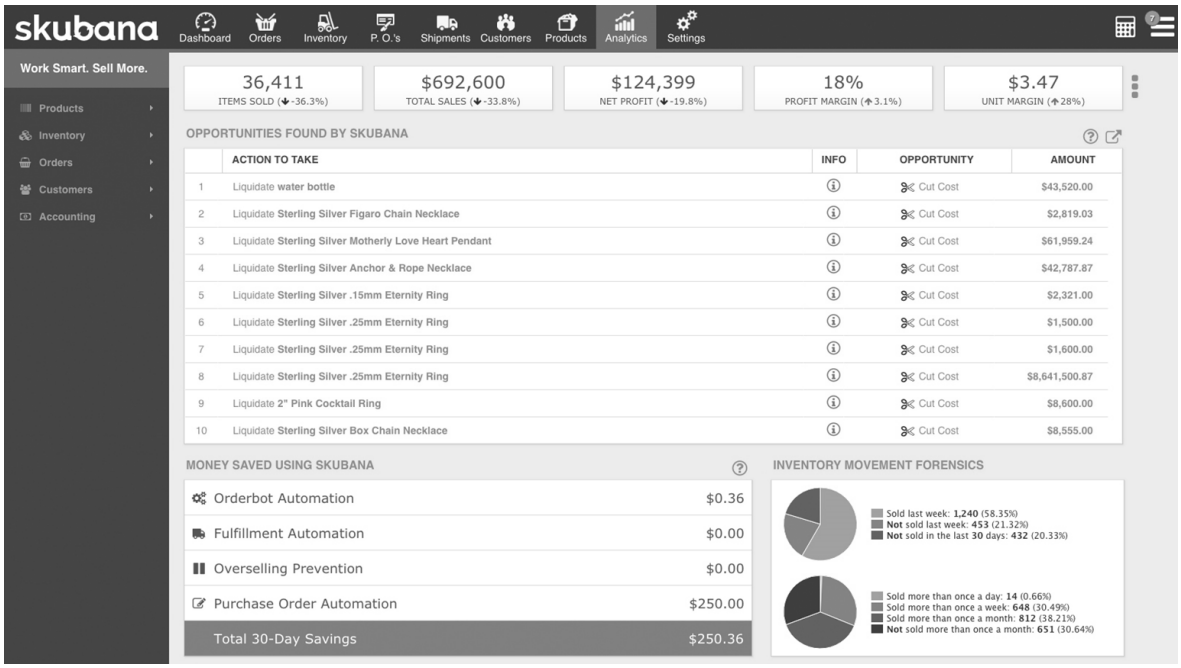

Рис. 1.7. *Skubana*

Freestyle Solution Software

Це рішення позиціонується як найкраще для малого та середнього бізнесу (інтерфейс зображений на рисунку 1.8.). Раніше *Dydacomp* у 2015 році, *Freestyle Solutions* допомагав зростаючому бізнесу централізувати та впорядкувати обробку своїх замовлень. Завдяки ньому можна синхронізувати свої рівні ресурсів через різні канали, інтегрувати процес доставки, наприклад, пряму доставку, керувати кількома складами та приймати обґрунтовані рішення за допомогою вбудованої звітності. Користувачі можуть скористатися перевагами системи *Freestyle Solutions*, не отримуючи плати за кількість користувачів, канали або *SKU*.

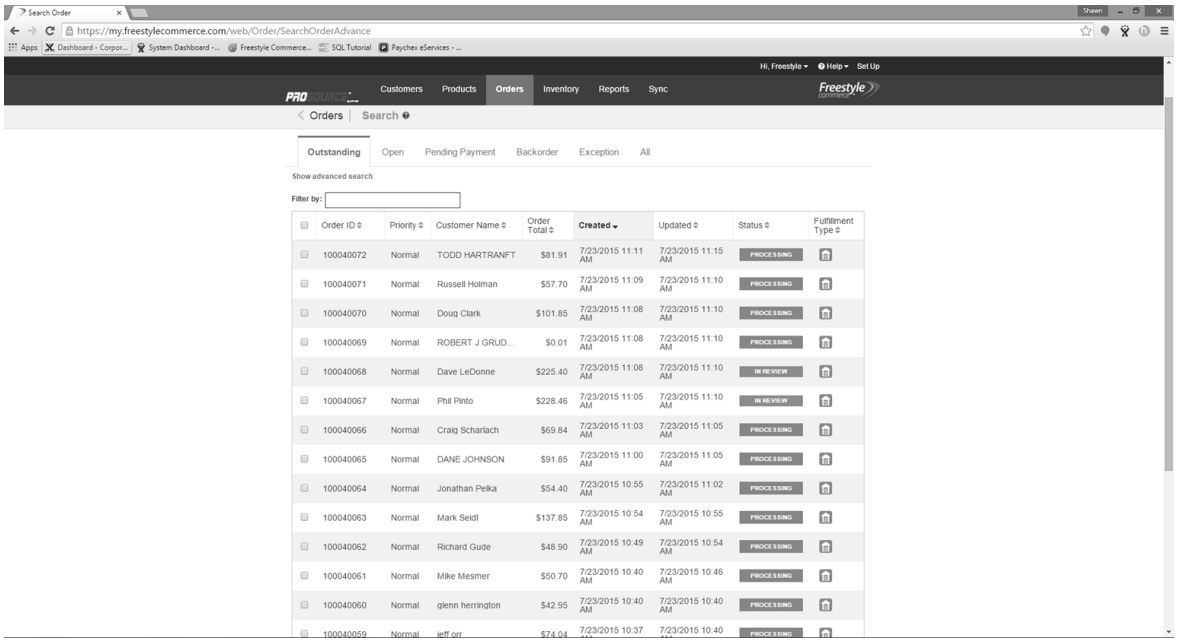

Рис. 1.8. *Freestyle Solution Software*

Odoo [8]

Маючи понад 5 мільйонів користувачів, починаючи від стартапів і закінчуючи підприємствами, *Odoo* – це платформа з відкритим кодом, яка пропонує набір бізнес-додатків, що включає все, від електронної комерції до *CRM* та управління ресурсами (інтерфейс зображений на рисунку 1.9.). Вони пропонують 30 основних програм, створених *Odoo*, та ще 1600+ програм, розроблених їх власною активною спільнотою. *Odoo* – це інше рішення в, оскільки воно є відкритим, тобто кінцеві користувачі можуть завантажувати та налаштовувати код платформи. Іншими словами, ви маєте повний контроль над налаштуванням програмного забезпечення відповідно до ваших унікальних потреб. Це також означає, що ви можете легко інтегрувати *Odoo* в інші необхідні системи, такі як електронна комерція, *ERP* або програмне забезпечення для доставки. Незважаючи на те, що вони пропонують деякі базові інвентаризаційні та складські функції заздалегідь побудовані, ви можете легко змінювати та масштабувати програмне забезпечення в міру розвитку ваших процесів. У деяких випадках програмне забезпечення з відкритим кодом є одночасно економічно вигідним та гнучким для функціонування *OMS*.

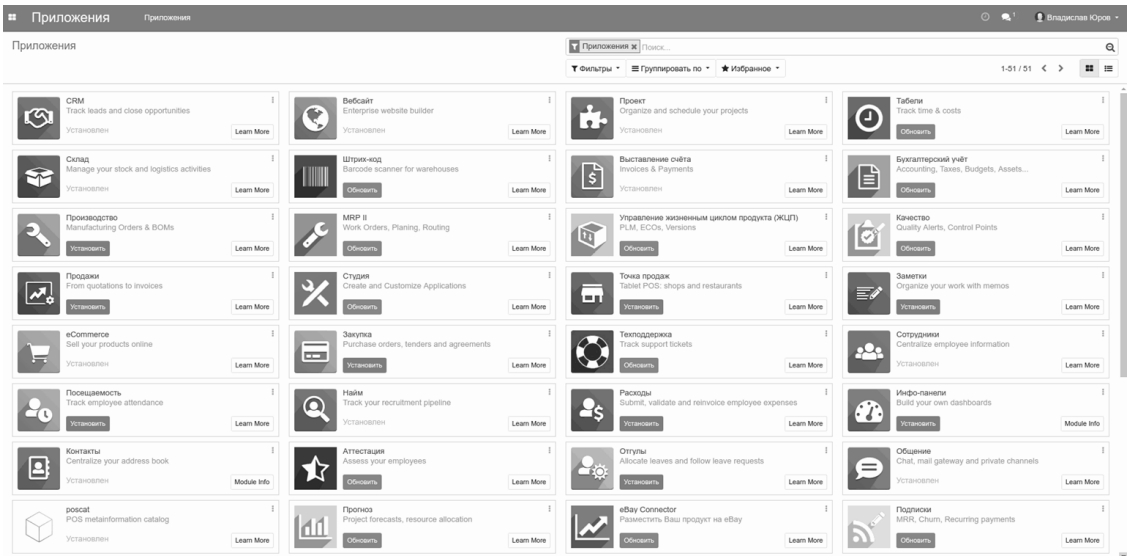

Рис. 1.9. *Odoo*

Як можемо бачити, більшість сервісів мають свої переваги та недоліки, до основних можна віднести високу вартість програмного забезпечення – що створює ряд проблем для початкових та малих підприємств. Крім цього – проблеми з інтеграцією та налаштуванням. Варто ознайомитись з базовими функціями та можливостями програмного забезпечення, а потім налаштувати під власні потреби. Тому було прийнято рішення створити свій web-додаток для забезпечення управління замовленнями інтернет-магазину продажу авто-комплектуючих на прикладі аналогів, що були розглянуті вище.

1.3 Постановка задачі

Отже, судячи з аналогів, для того, щоб створити сервіс для управління замовленнями необхідно наступне:

- Обрати необхідні інструменти для розробки клієнтської та серверної частини додатку.
- Обрати базу даних для оброблення інформації.
- Обрати інструменти для управління базою даних.
- Реалізувати логіку додатку, що включає в себе: замовлення, нотатки до замовлення, клієнтів, постачальників, номер замовлення, час створення замовлення, категорію замовлення, тип замовлення, ім’я на котре було

створене замовлення, ціна замовлення, знижка (якщо наявна), статистику замовлень та налаштування сервісу.

– А також наявні продукти для продажу.

1.4 Вибір засобів реалізації

Розглянувши поставлене завдання, було прийнято рішення реалізовувати веб-додаток за допомогою наступних інструментів:

– *Java* – технологія *Java* дозволяє працювати в захищеному обчислювальному середовищі. *Java* дозволяє створювати онлайн-ігри, спілкуватися з людьми по всьому світу, створювати веб-сервери та переглядати зображення у 3*D* – та багато іншого [9];

– *PrimeFaces* – це популярний фреймворк з відкритим кодом для JavaServer Faces, що включає понад 100 компонентів, оптимізований сенсорний мобільний набір, перевірку на стороні клієнта, механізм тем тощо;

– *Spring Security* – це потужна та налаштовувана система автентифікації та контролю доступу. Це фактичний стандарт захисту програм на основі *Spring* [10];

– *Spring Boot* – дозволяє легко створювати окремі, виробничі додатки на основі *Spring*;

– *Spring Data* – місія *Spring Data* полягає у забезпеченні знайомої та послідовної моделі програмування на основі *Spring*, що забезпечує доступ до даних, зберігаючи при цьому особливі риси основного сховища даних;

– *Spring Data JPA* – частина великого сімейства *Spring Data*, дозволяє легко реалізувати сховища на основі *JPA*. Цей модуль займається розширеною підтримкою рівнів доступу до даних на основі *JPA*. Це спрощує створення *Spring* програм, що використовують технології доступу до даних.

– *MySQL* – найпопулярніша у світі база даних з відкритим кодом. Незалежно від того, чи є ви швидко зростаючою веб-власністю, технологією *ISV* чи великим підприємством, *MySQL* може економічно ефективно допомогти вам забезпечити високопродуктивні, масштабовані програми баз даних [11].

– *Maven* – *Apache Maven* – це програмний засіб управління та розуміння проектів. На основі концепції об'єктної моделі проекту (*POM*), *Maven* може управляти збіркою проекту, звітуванням та документацією з центральної інформації.

– *Liquibase* – це інструмент управління базами даних.

1.5 Висновки до розділу

У результаті проведеного аналізу було визначено предметну область – було розглянуто дослідження та публікації на тему веб-додатків, проаналізовано концепцію їх створення. Наступним кроком було проведення аналізу програмних продуктів-аналогів – були розглянуті основні продукти зі сфери управління заявками – проаналізовано їх плюси й мінуси. Після чого було обрано засоби для створення веб-додатку, а саме: мова програмування *Java* та фреймворк для неї.

Для розробки теми дипломної роботи необхідно наступне:

– Змоделювати процес роботи інтернет-магазину;

– Зпроектувати інформаційну систему;

– Зпроектувати веб-додаток;

РОЗДІЛ 2
WEB-ДОДАТОК УПРАВЛІННЯ ЗАМОВЛЕННЯМИ В ІНТЕРНЕТ-МАГАЗИНІ

2.1 Моделювання процесу роботи інтернет-магазину

В основі будь-якого інтернет-магазину лежать бізнес-процеси, завдяки котрим відбувається робота магазину. В нашому випадку магазин займається продажем автомобільних запчастин та необхідних для повноцінного функціонування ресурсів (мастило, хімія для збереження зовнішнього вигляду й тому подібне).

Перш за все, як відбувається робота інтернет-магазину? [12]

В основі лежить сайт, завдяки якому здійснюються основні процеси, головна мета якого – отримати замовлення.

Співробітники виконують рядові операції, менеджери інтернет-магазину формують замовлення покупця, комірник відповідає за переміщення між складами, товарознавець відповідає за поповнення необхідних для продажу товарів. Документи автоматично реєструються задля замовлення. Логіст відповідальний за поділ замовлень для доставки транспортними засобами – будь то автомобіль, кур’єр чи пошта.

Після доставки замовлення цикл вважають завершеним – інтернет-магазин отримує прибуток, а далі все по новому колу, коли буде отримане замовлення.

Схематично це можна показати наступним чином (рисунок 2.1.):

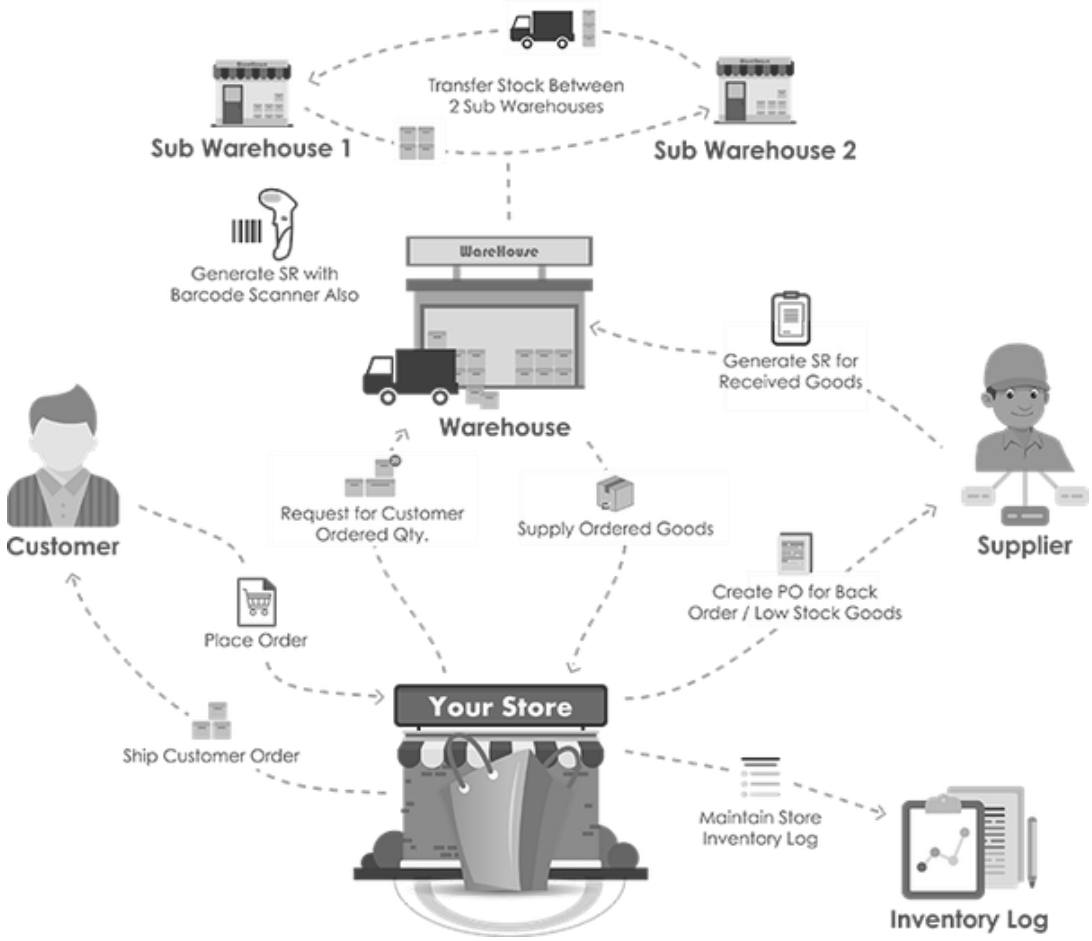

Рис. 2.1. Принцип роботи інтернет-магазину

Як можемо бачити, в основі лежить планування та реалізація доставки замовлення. Тому наявна необхідність розробки додатку для керування замовленнями інтернет-магазину.

2.2 Проектування інформаційної системи

В основі майбутнього додатку лежить процес оперування заявкою клієнта [13].

Цей процес можна описати наступним чином: перш за все відбувається формування замовлення в інтернет-магазині клієнтом. Менеджер спілкується з клієнтом та формує остаточне замовлення, котре далі передається(рисунок 2.2.). У разі якщо товару немає на складі, відбувається його замовлення у постачальника (рисунок 2.3.-2.5.)

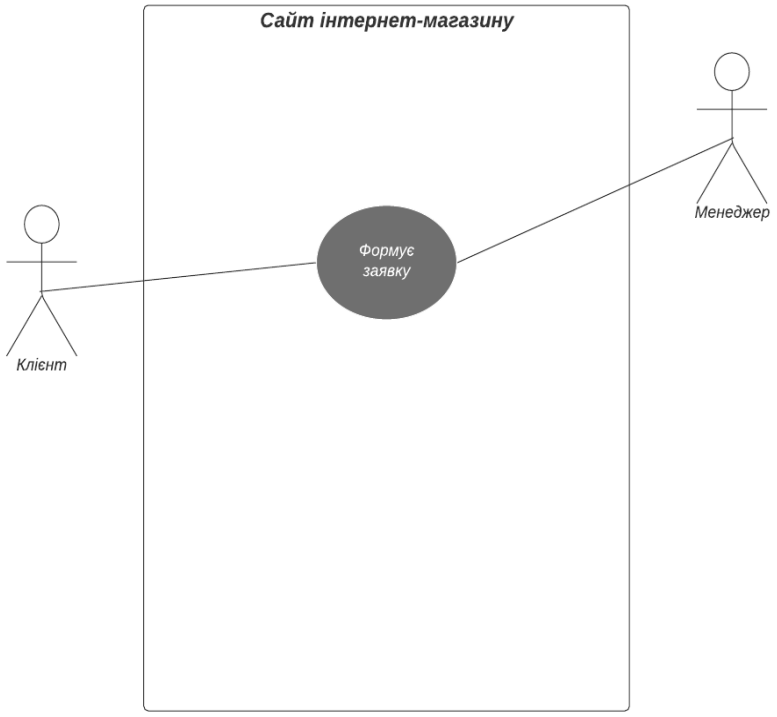

Рис. 2.2. Формування замовлення

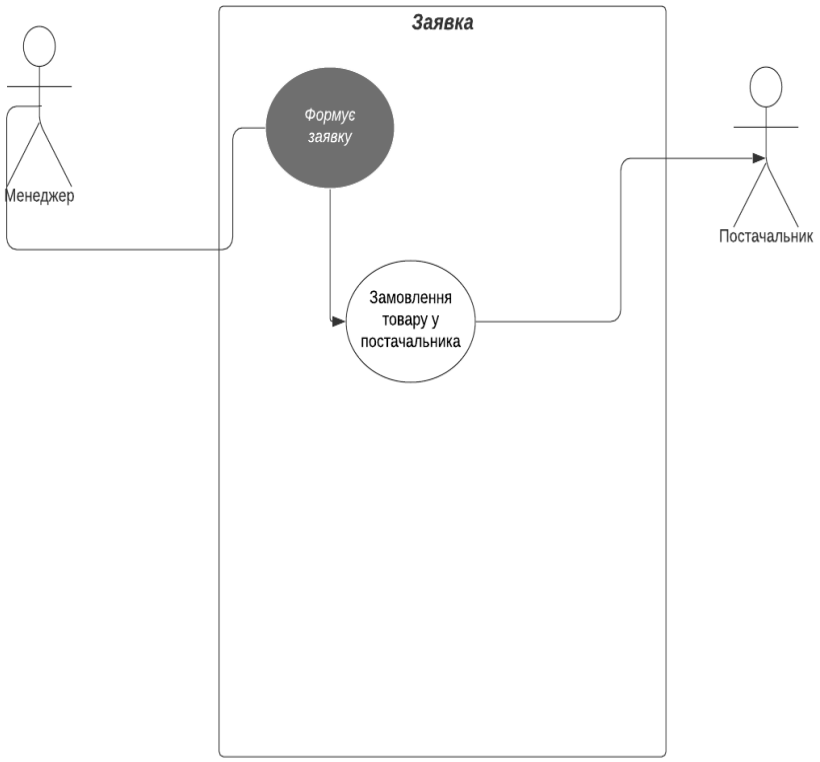

Рис. 2.3. Замовлення товару

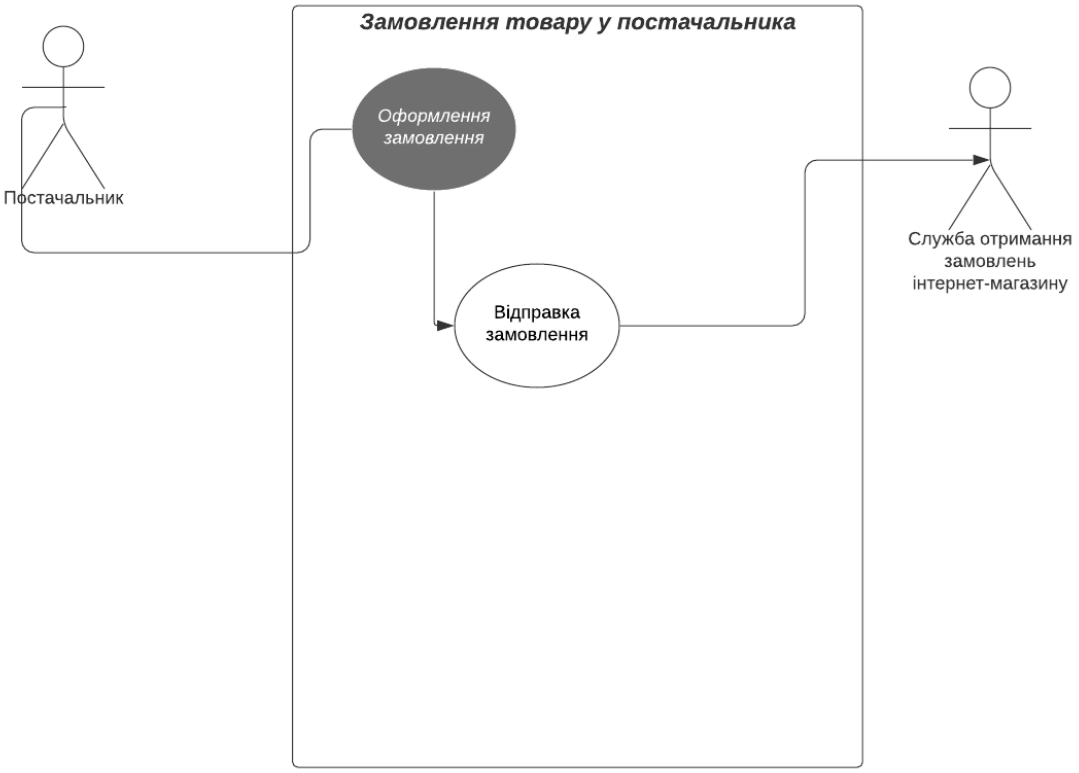

Рис. 2.4. Замовлення товару

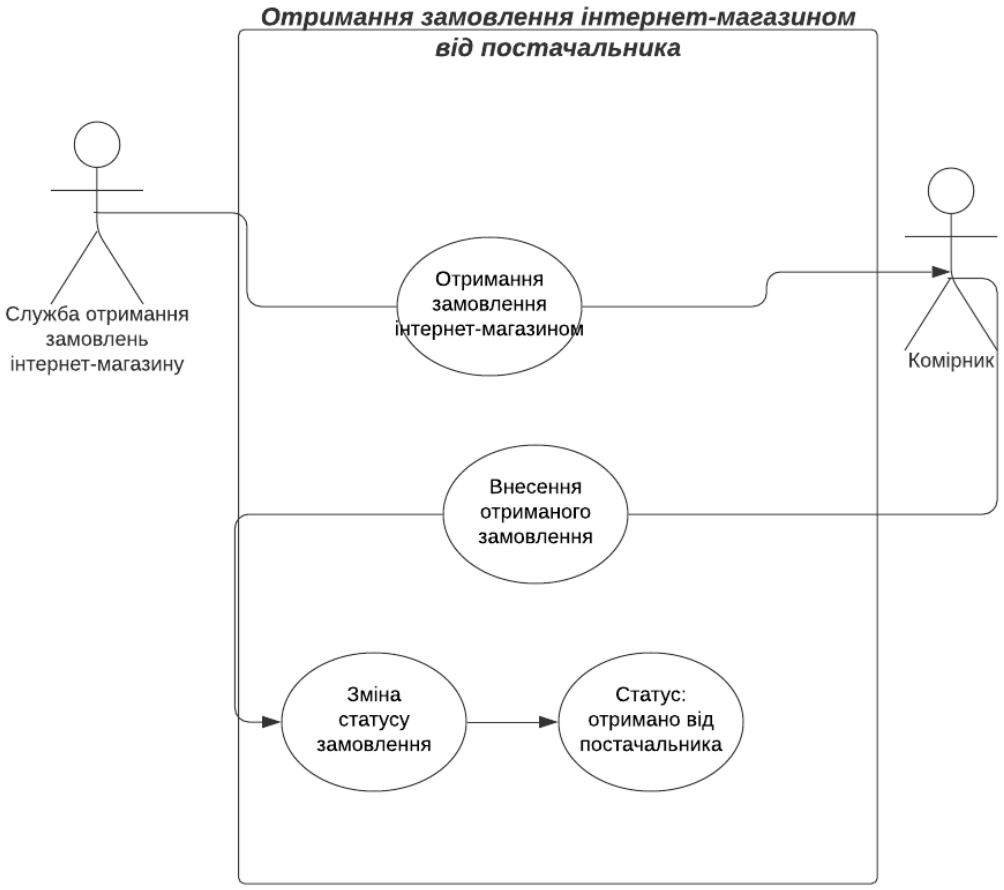

Рис. 2.5. Отримання замовлення від постачальника

Після того, як замовлення від постачальника було отримано (рисунок 2.6.) – комірник відповідає за комплектацію замовлення, друкується накладна та відбувається пошук необхідного (рисунок 2.7.). Після того, як все необхідне було зібрано та укомплектовано, відбувається відправка замовлення до клієнта –що зображено на рисунку 2.8., після чого відбувається комплектація наступного замовлення або його очікування.

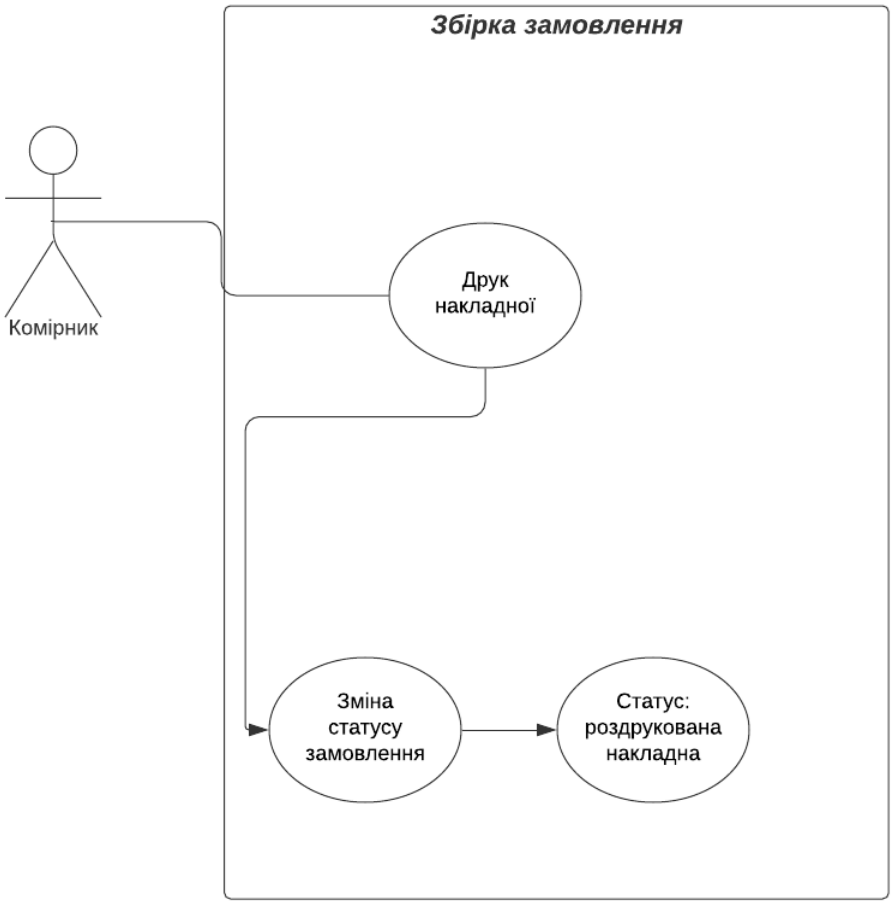

Рис. 2.6. Комплектація замовлення (1)

Рис. 2.7. Комплектація замовлення (2)

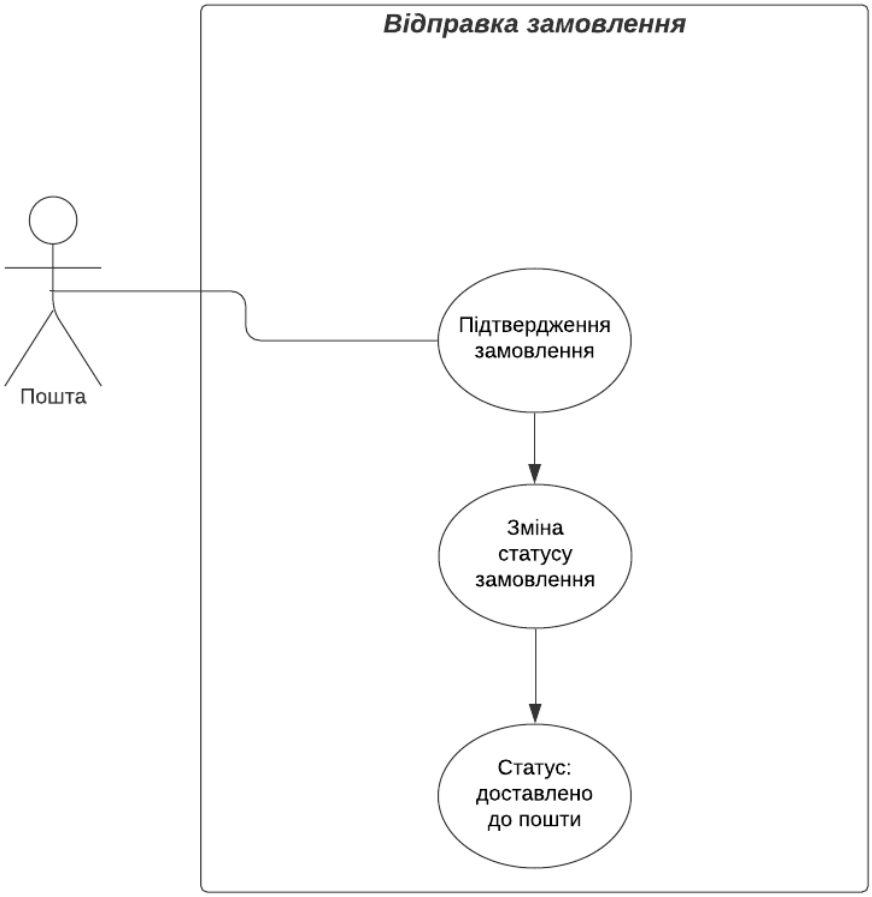

Рис. 2.8. Відправка замовлення

2.3 Проектування *web*-додатку

В основі веб-додатку лежить дві частини – клієнтська та серверна.

Те, що бачить користувач – це клієнтська частина, з котрою відбувається взаємодія. Перш за все, сюди входить інтерфейс. Під інтерфейсом в нашому випадку будемо розуміти основну робочу область веб-додатку. Вона складається з наступного:

- Вікно авторизації;
- Меню додатку після авторизації користувача;
- Меню додатку включає в себе підпункти, а саме: *Sales*, *Articles*, *Units*, *Inventories*, *Statistic*, *Settings*.

Будуть додані елементи до підпунктів, а саме:

- *Item* – що включає в себе *ID*, *Name*, *Notes*, *Created date*, *Valid to date* – та кнопки *Cancel*, *Save*;
- *Article – ID, Type, Category, Provider, Unit Of Measure, Logical ID, Name, Quantity, Warning quantity, Provider price, Sales price, Notes, Created date, Valid to date* та кнопки – *Cancel, Save;*
- *Article Category – ID, Name, Parent Name, Notes, Created date, Valid to date* та кнопки *Cancel*, *Save*;
- *Inventory – ID, Type, Name, Notes, Status, Provider, Quantity, Warning quantity, Created date, Valid to date* та кнопки *Cancel*, *Save*;
- *Order – ID, Logical ID, Type, Status, Client, Item list, Total sum, Total units, Delivery address, Name;*
- *Person – ID, First Name, Last name, Phone, Email, Fax, Notes, Created date, Valid to date* та кнопки *Cancel, Save;*
- *Unit – ID, Type, Unit Industry, Name, Address, Email, Phone, Fax, Contact person (First name, Last name, Phone, Email, Fax, Options* та кнопка *Add), Additional information, Notes, Created date, Valid to date* та кнопки *Cancel*, *Save*.

– *Unit Of Measure – ID, Name, Notes, Created date, Valid to date* та кнопки *Cancel, Save.*
– *Unit Type – ID, Name, Notes, Created date, Valid to date* та кнопки *Cancel, Save*.

Серверна частина буде реалізована за допомогою можливостей мови програмування *Java* – як було описано вище. В якості бази даних було обрано *MySQL*, для управління базою даних обрано інструмент *Liquibase* [14].

2.4 Висновки до розділу

В даному розділі було створено й описано модель роботи інтернет-магазину – в основі котрого лежить отримання замовлення від клієнта.

Наступним кроком було створення інформаційної системи – був описаний алгоритм роботи інтернет-магазину й процес обробки замовлення – перш за все замовлення має бути отримане за допомогою сайту інтернет-магазину, далі менеджери оброблюють замовлення, уточнюють деталі та передають до відповідного відділу. Далі – перевірка замовлення, комплектація, замовлення продуктів, якщо їх немає на складі, а після – отримання та відправка, все це відповідно до регламенту інтернет-магазину. Для спрощення роботи з замовленнями й буде створено веб-додаток управління замовленнями інтернет-магазину.

Останнім кроком в цьому розділі було проектування веб-додатку. Веб-додаток складається з двох частин – клієнтської та серверної. Було описано компоненти клієнтської частини, а в кінці – обрано інструменти для створення серверної – *Java*, *MySQL*.

Для досягнення мети цієї роботи ключовим моментом є третій розділ, в якому будуть вирішені наступні питання:

– Розглянути архітектуру веб-додатку;
– Розглянути програмну реалізацію.

РОЗДІЛ 3
ПРОГРАМА УПРАВЛІННЯ ЗАМОВЛЕННЯМИ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Структура веб-додатку

В основі веб-додатку [15] лежить фреймворк *Java – Spring*. Тому основні процеси веб-додатку базуються на основі цього фреймворку. На рисунку 3.1. зображено доступні модулі для кожної функції, а також їх назви артефактів та теми, які вони охоплюють. Назви артефактів співвідносяться з ідентифікаторами артефактів, що використовуються в засобах управління залежностями.

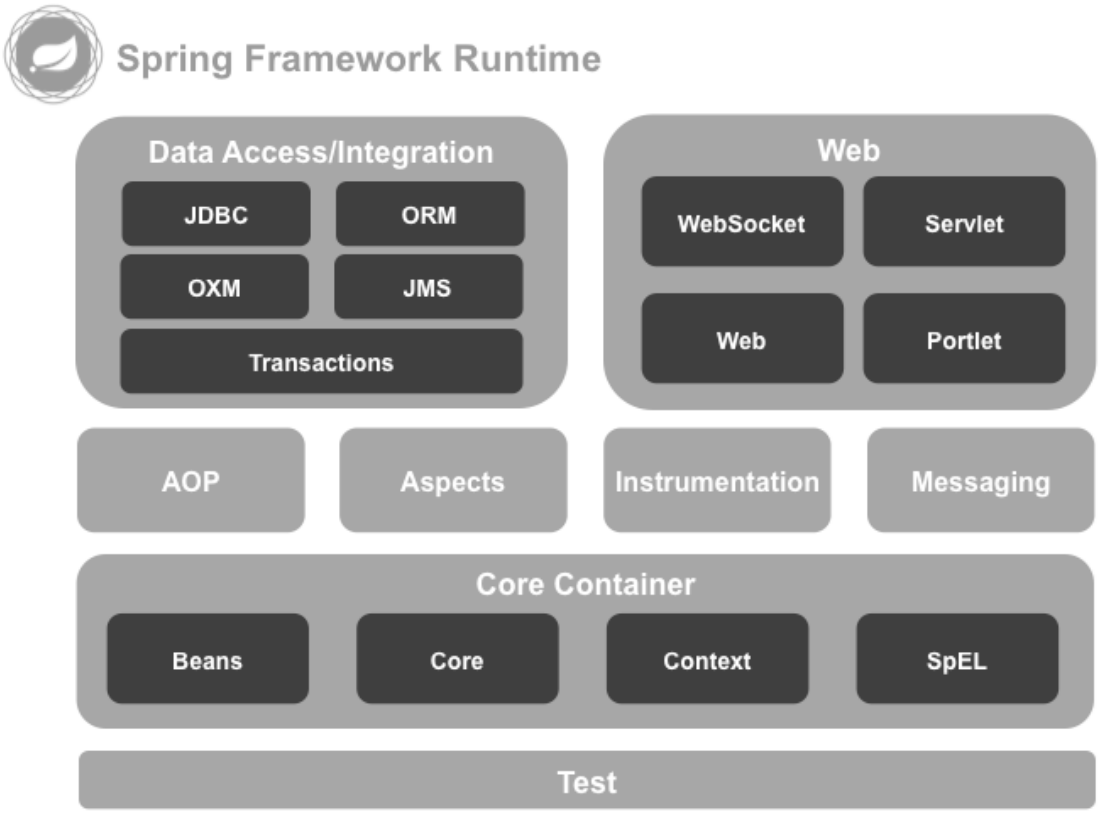

Рис. 3.1. Модулі *Spring Framework*

Основний контейнер складається з модулів *spring-core*, *spring-beans*, *spring-context*, *spring-context-support* та *spring-expression* (*Spring Expression Language*).

Модулі *spring-core* та *spring-beans* надають основні частини фреймворку, включаючи функції *IoC* та інжекції залежності. *BeanFactory* – це вишукана реалізація заводського зразка. Це позбавляє від необхідності програмних одиночних програм і дозволяє відокремити конфігурацію та специфікацію залежностей від фактичної логіки програми.

Модуль *Context* (*spring-context*) базується на надійній основі, що забезпечується модулями *Core* і *Beans*: це засіб доступу до об'єктів у стилі фреймворку, подібне до реєстру *JNDI*. Модуль *Context* успадковує свої функції від модуля *Beans* і додає підтримку інтернаціоналізації (використовуючи, наприклад, пакети ресурсів), розповсюдження подій, завантаження ресурсів та прозоре створення контекстів, наприклад, за допомогою контейнера *Servlet*. Модуль *Context* також підтримує такі функції *Java EE*, як *EJB*, *JMX* та базове віддалення. Інтерфейс *ApplicationContext* є фокусною точкою модуля *Context*. *spring-context-support* забезпечує підтримку інтеграції спільних сторонніх бібліотек у контекст Spring програми для кешування (*EhCache*, *Guava*, *JCache*), розсилки (*JavaMail*), планування (*CommonJ*, *Quartz*) та механізмів шаблонів (*FreeMarker*, *JasperReports*, *Velocity*).

Модуль *spring-expression* надає потужну мову виразів для запитів та маніпулювання графіком об’єктів під час виконання. Це розширення уніфікованої мови виразів (уніфікованої *EL*), як зазначено в специфікації *JSP* 2.1. Мова підтримує встановлення та отримання значень властивостей, присвоєння властивостей, виклик методу, доступ до вмісту масивів, колекцій та індексаторів, логічних та арифметичних операторів, іменованих змінних та отримання об’єктів за іменем із контейнера *IoC Spring*. Він також підтримує проектування та виділення списків, а також загальні агрегації списків.

Модуль *spring-aop* забезпечує реалізацію аспектно-орієнтованого програмування, сумісну з *AOP Alliance*, що дозволяє визначити, наприклад, перехоплювачі методів і точки для чистого роз'єднання коду, який реалізує функціональні можливості, які слід розділити. Використовуючи функціональність

метаданих на рівні джерела, ви також можете включити інформацію про поведінку у свій код, подібним чином до атрибутів *.NET*. Окремий модуль spring-аспекти забезпечує інтеграцію з *AspectJ*. Модуль *spring-instrument* забезпечує підтримку інструментарію класу та реалізації навантажувача класів, які будуть використовуватися на певних серверах додатків. Модуль *spring-instrument-tomcat* містить інструментальний агент Spring для *Tomcat*.

Spring Framework 4 включає модуль *Spring-Messaging* із ключовими абстракціями проекту Spring Integration, такими як *Message*, *MessageChannel*, *MessageHandler* та інші, що слугують основою для програм на основі обміну повідомленнями. Модуль також включає набір анотацій для зіставлення повідомлень із методами, подібними до моделі програмування на основі анотацій *Spring MVC*.

Рівень доступу / інтеграції даних складається з модулів *JDBC*, *ORM*, *OXM*, *JMS* та транзакцій. Модуль *spring-jdbc* забезпечує рівень абстракції *JDBC*, який усуває необхідність виконувати втомливе кодування *JDBC* та синтаксичний аналіз кодів помилок конкретного постачальника даних. Модуль *spring-tx* підтримує програмне та декларативне управління транзакціями для класів, що реалізують спеціальні інтерфейси, та для всіх ваших *POJO* (звичайні старі об'єкти *Java*). Модуль *spring-orm* забезпечує інтеграційні рівні для популярних *API*-об'єктів реляційного відображення, включаючи *JPA*, *JDO* та *Hibernate*. Використовуючи модуль spring-orm, ви можете використовувати всі ці каркаси *O* / *R*-відображення у поєднанні з усіма іншими функціями, які пропонує *Spring*, такими як проста декларативна функція управління транзакціями, згадана раніше. Модуль *spring-oxm* надає рівень абстракції, який підтримує реалізації об'єктів / *XML*, таких як *JAXB*, *Castor*, *XMLBeans*, *JiBX* та *XStream*. Модуль *spring-jms* (*Java Messaging Service*) містить функції для створення та споживання повідомлень. Починаючи з *Spring Framework* 4.1, він забезпечує інтеграцію з модулем обміну повідомленнями *spring*.

Веб-рівень складається з модулів *spring-web*, *spring-webmvc*, *spring-websocket* та *spring-webmvc-portlet*. Модуль *spring-web* надає основні веб-орієнтовані функції інтеграції, такі як функціональність завантаження файлів із

декількох частин та ініціалізація контейнера *IoC* за допомогою прослуховувачів *Servlet* та контексту веб-орієнтованого додатка. Він також містить клієнт *HTTP* та пов'язані з Інтернетом частини віддаленої підтримки *Spring*. Модуль *spring-webmvc* (також відомий як модуль *Web-Servlet*) містить контролер перегляду моделі (*MVC*) та реалізацію веб-служб *REST* для веб-додатків. Структура *MVC Spring* забезпечує чітке розділення між кодом доменної моделі та веб-формами та інтегрується з усіма іншими функціями *Spring Framework*. Модуль *spring-webmvc-portlet* (також відомий як модуль *Web-Portlet*) забезпечує реалізацію *MVC* для використання в середовищі портлету та відображає функціональність модуля *spring-webmvc* на основі сервлета.

Також даний фреймворк підтримує тестування створеного програмного забезпечення.

3.2 Програмна реалізація

Наступним кроком є огляд створеного веб-додатку. Нижче будуть наведені основні екрани, що були створені за допомогою фреймворку *Spring*. Інтерфейс користувача був розроблений за допомогою мови *HTML*.

Перше, що бачить користувач – це вікно авторизації (рисунок 3.2.) – в ньому, як можна зрозуміти, відбувається авторизація користувача – завдяки логіну та паролю здійснюється перевірка користувача у системі – у разі відповідності надається доступ до ресурсу.

Рис. 3.2. Вікно авторизації

Після авторизації користувач бачить головний екран веб-додатку, в котрому наявні наведені на рисунку 3.3. елементи.

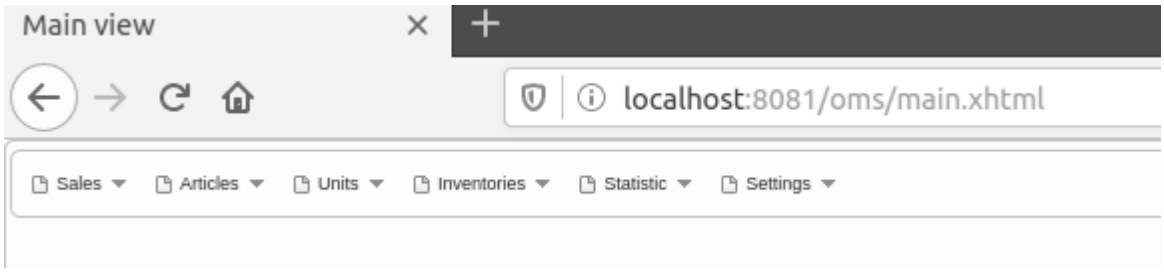

Рис. 3.3. Головне меню

Користувачу доступний список елементів (рисунок 3.4.-3.5.):

Рис. 3.4. Список доданих елементів

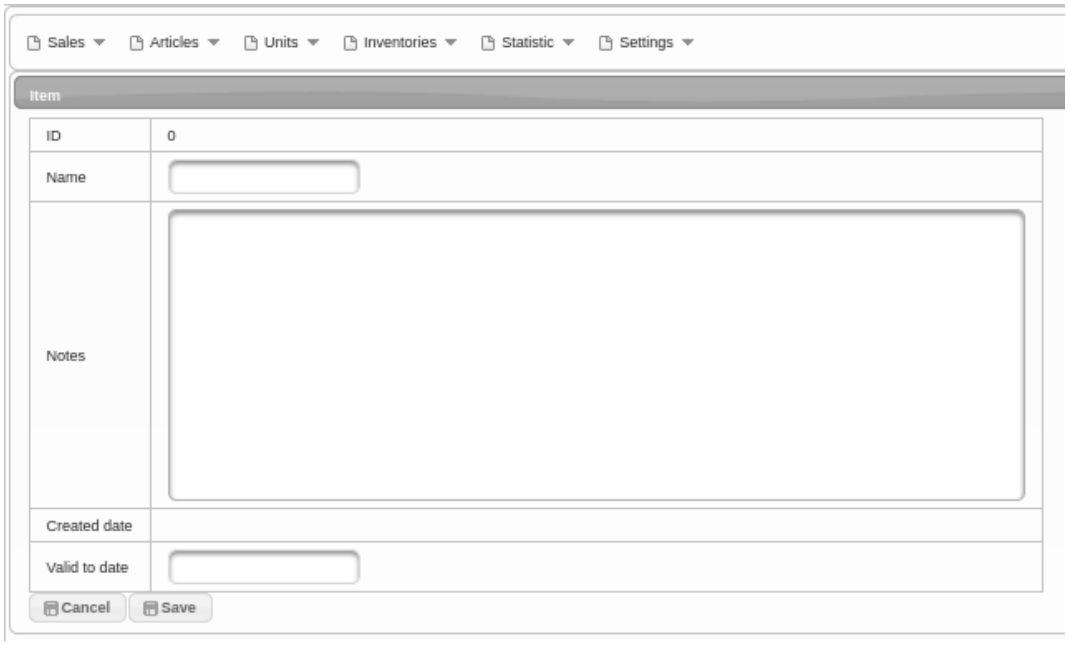

Рис. 3.5. Вкладка *Item*

Наступним елементом є – *Article* (рисунок 3.6.):

Рис. 3.6. *Article*

На рисунку 3.7. зображено *Article Category:*

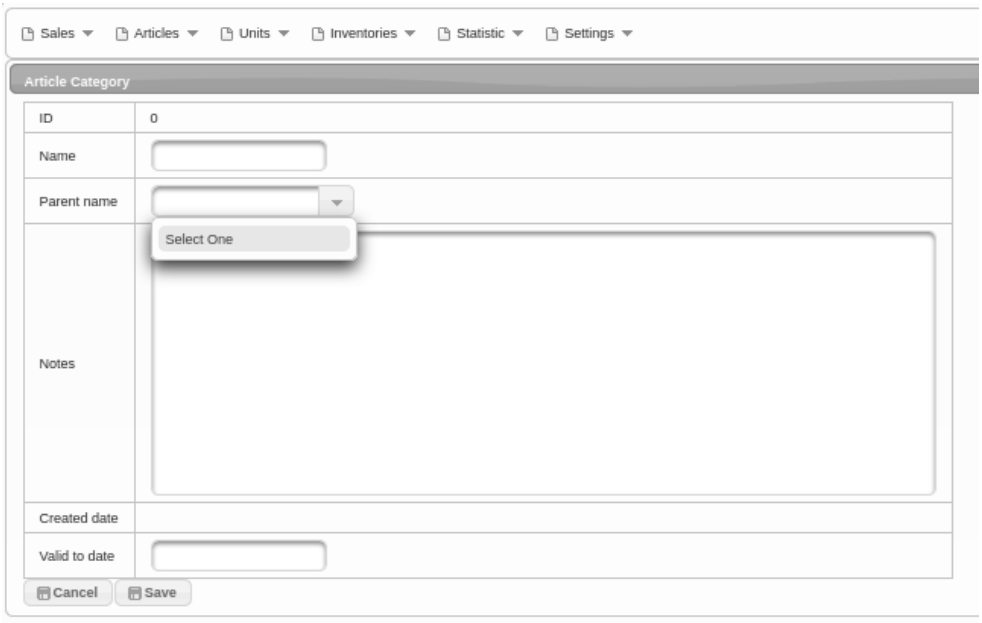

Рис. 3.7. *Article Category*

На рисунку 3.8. зображено *Inventory*:

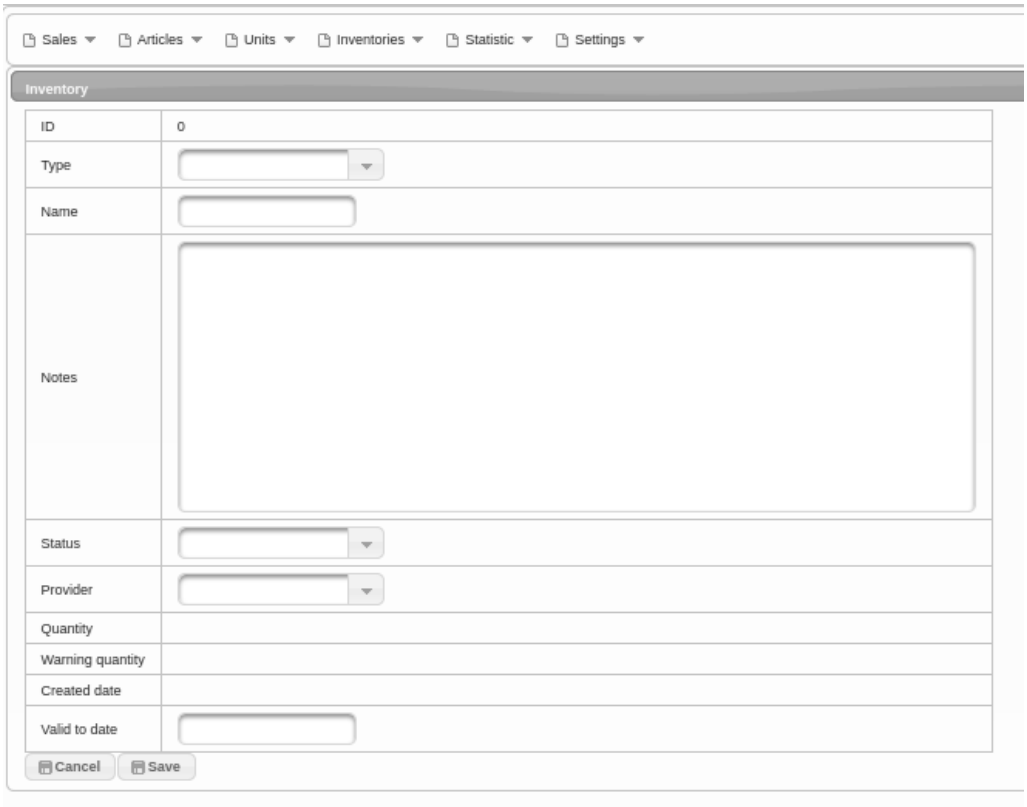

Рис. 3.8. *Inventory*

На рисунку 3.9. зображено *Order*:

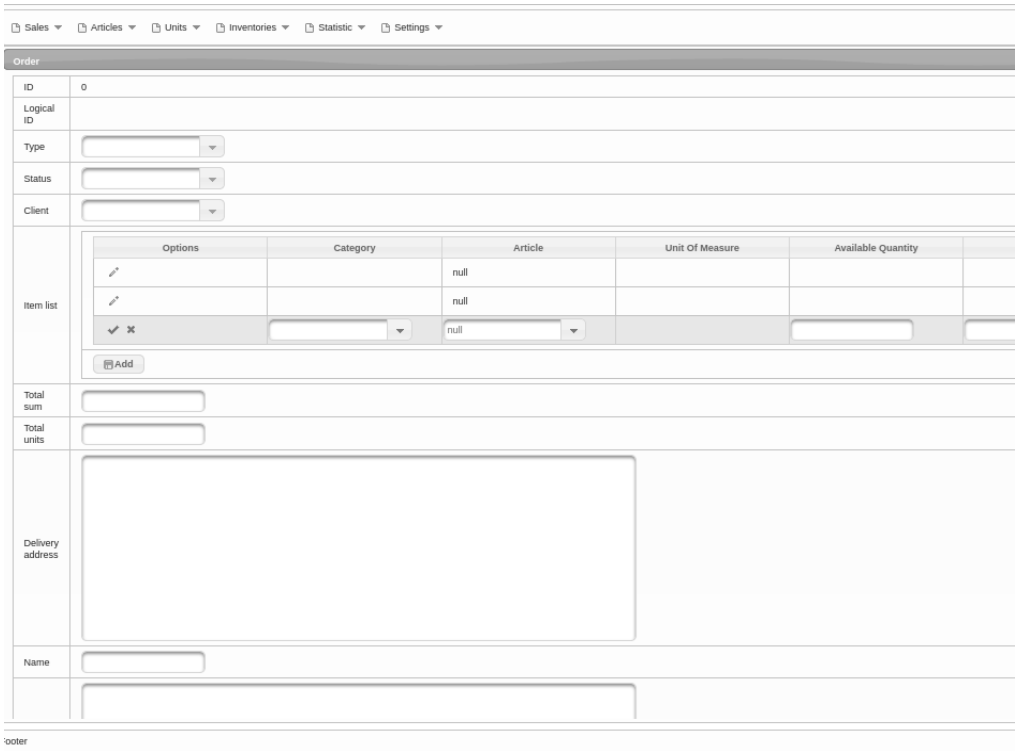

Рис. 3.9. *Order*

На рисунку 3.10. зображено вкладку *Person*:

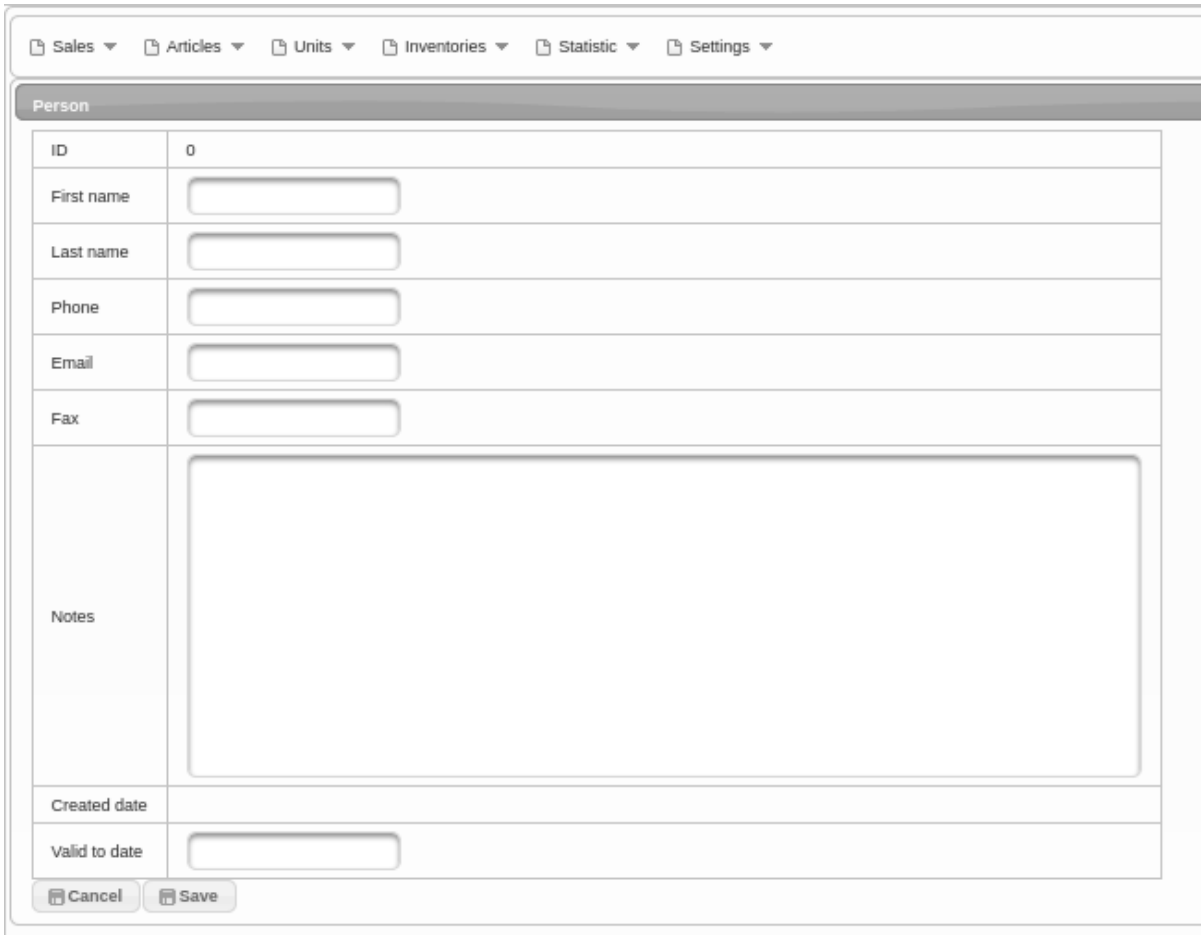

Рис. 3.10. Вкладка *Person*

На рисунку 3.11. зображено вкладку *Unit*:

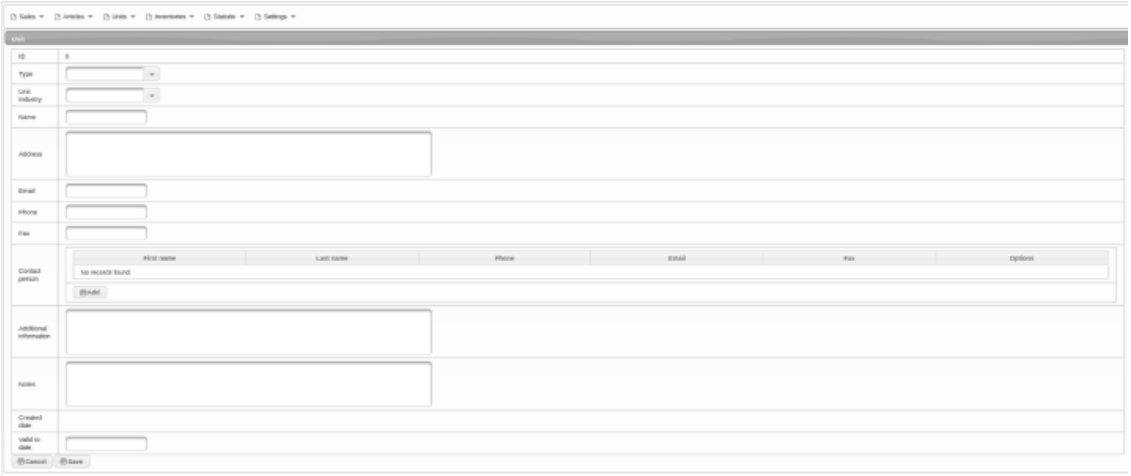

Рис. 3.11. *Unit*

На рисунку 3.12. зображено вкладку *Unit of Measure:*

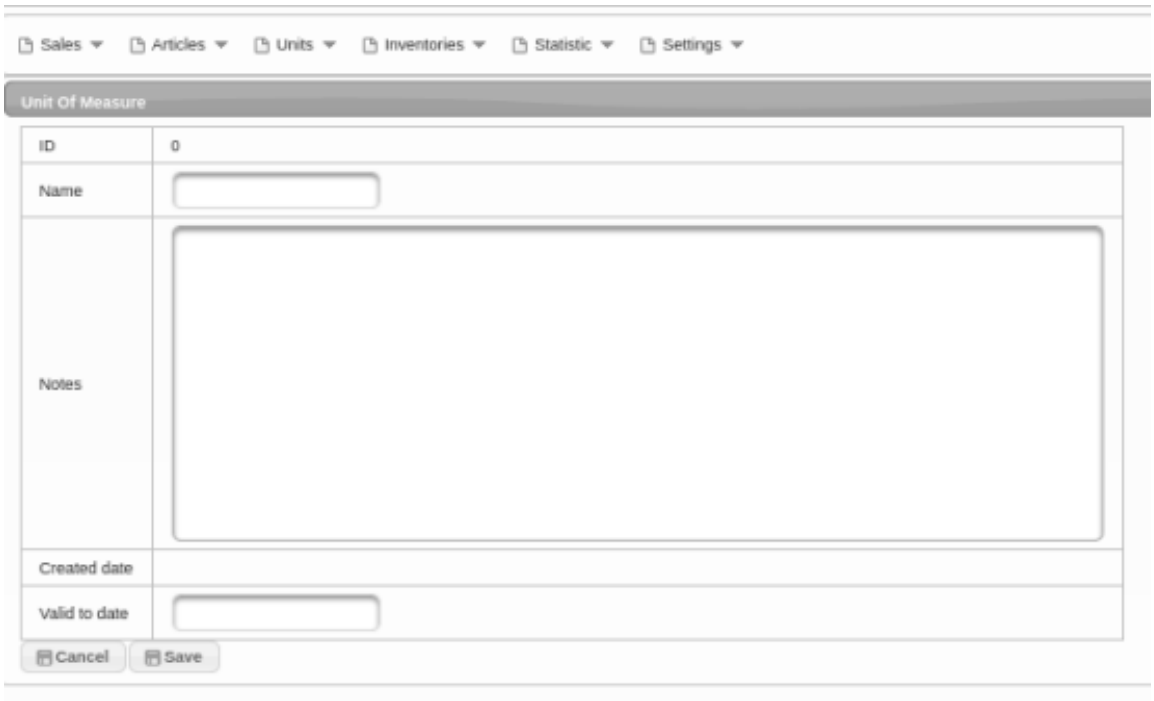

Рис. 3.12. *Unit Of Measure*

На рисунку 3.13. зображено *Unit Type*:

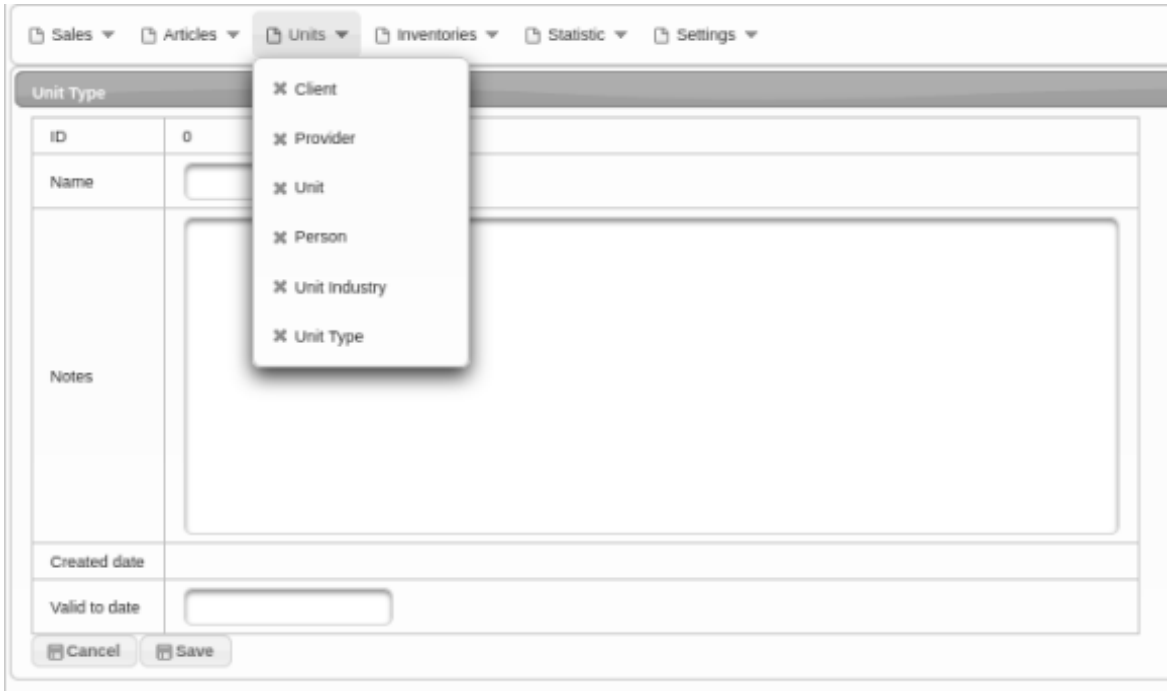

Рис. 3.13. *Unit Type*

3.3 Висновки до розділу

В даному розділі було описано архітектуру веб-додатку, що був створений, в основі цього додатку фреймворк для мови програмування *Java – Spring*. Цей

фреймворк базується на модулях, більшість з яких була використана під час розробки. А саме:

- Модуль *spring-core/spring-beans;*
- Модуль *Context (spring-context);*
- Модуль *expression;*
- Модуль *spring-aop;*
- Модуль *Spring-Messaging*;
- Модуль *JDBC, ORM, OXM, JMS,*
- Модуль *spring-jdbc*;

Веб-рівень складається з наступних модулів: *spring-web, spring-webmc, spring-websocket, spring-webmvc-portlet.*

Також було описано інтерфейс веб-додатку, що було створено. Були реалізовані екрани, що були поставлені в завданні.

Перш за все було реалізовано й продемонстровано вікно авторизації користувача, далі – головне вікно додатку та підпорядковані елементи – такі як *Person, Unit, Order, Article* та інші, що були поставлені в завданні для розробки.

А отже, судячи по виконаній роботі, можна сказати, що мета цієї дипломної роботи (проєкту) є досягнутою.

ВИСНОВКИ

В ході виконання дипломного проекту було виконано наступні завдання:

В першому розділі у результаті проведеного аналізу було визначено предметну область – було розглянуто дослідження та публікації на тему веб-додатків, проаналізовано концепцію їх створення. Наступним кроком було проведення аналізу програмних продуктів-аналогів – були розглянуті основні продукти зі сфери управління заявками – проаналізовано їх плюси й мінуси. Після чого було обрано засоби для створення веб-додатку, а саме: мова програмування *Java* та фреймворк для неї.

В другому розділі було створено й описано модель роботи інтернет-магазину – в основі котрого лежить отримання замовлення від клієнта. Наступним кроком було створення інформаційної системи – був описаний алгоритм роботи інтернет-магазину й процес обробки замовлення – перш за все замовлення має бути отримане за допомогою сайту інтернет-магазину, далі менеджери оброблюють замовлення, уточнюють деталі та передають до відповідного відділу. Далі – перевірка замовлення, комплектація, замовлення продуктів, якщо їх немає на складі, а після – отримання та відправка, все це відповідно до регламенту інтернет-магазину. Для спрощення роботи з замовленнями й буде створено веб-додаток управління замовленнями інтернет-магазину. Останнім кроком в цьому розділі було проектування веб-додатку. Веб-додаток складається з двох частин – клієнтської та серверної. Було описано компоненти клієнтської частини, а в кінці – обрано інструменти для створення серверної – *Java*, *MySQL*.

В третьому розділі було описано архітектуру веб-додатку, що був створений, в основі цього додатку фреймворк для мови програмування *Java* – *Spring*. Цей фреймворк базується на модулях, більшість з яких була використана під час розробки. А саме:

- Модуль *spring-core/spring-beans;*
- Модуль *Context (spring-context);*
- Модуль *expression;*
- Модуль *spring-aop;*

– Модуль *Spring-Messaging*;

– Модуль *JDBC, ORM, OXM, JMS,*

– Модуль *spring-jdbc*.

Веб-рівень складається з наступних модулів: *spring-web, spring-webmc, spring-websocket, spring-webmvc-portlet.*

Також було описано інтерфейс веб-додатку, що було створено. Були реалізовані екрани, що були поставлені в якості завдання.

Перш за все було реалізовано й продемонстровано вікно авторизації користувача, далі – головне вікно додатку та підпорядковані елементи – такі як *Person, Unit, Order, Article* та інші, що були в завданні для розробки.

А отже, судячи по виконаній роботі, можна сказати, що мета цієї дипломної роботи (проєкту) є досягнутою.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Архитектура веба [Електронний ресурс] – Режим доступу: *https://tproger.ru/translations/web-architecture-101* (дата звернення: 15.05.21р.)

2) *DNS* [Електронний ресурс] – Режим доступу: *https://aws.amazon.com/ru/route53/what-is-dns/* (дата звернення: 15.05.21р.)

3) Балансувальник навантаження [Електронний ресурс] – Режим доступу: *http://dspace.nbuv.gov.ua/bitstream/handle/123456789/14648/14%D0%A02%20%D0%A1%20193-201.pdf?sequence=1* (дата звернення: 15.05.21р.)

4) Что такое сервер веб-приложений [Електронний ресурс] – Режим доступу: *https://qna.habr.com/q/228398* (дата звернення: 15.05.21р.)

5) Сервер баз данных [Електронний ресурс] – Режим доступу: *https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85* (дата звернення: 15.05.21р.)

6) *CDN* [Електронний ресурс] – Режим доступу: *https://habr.com/ru/company/webzilla/blog/236511/* (дата звернення: 15.05.21р.)

7) *QuickBook* [Електронний ресурс] – Режим доступу: *https://quickbooks.intuit.com/app/apps/home* (дата звернення: 15.05.21р.)

8) *Odoo* [Електронний ресурс] – Режим доступу: *https://odoo.ru/* (дата звернення: 15.05.21р.)

9) *Java* [Електронний ресурс] – Режим доступу: *https://www.java.com/ru/* (дата звернення: 15.05.21р.)

10) *Spring Framework* [Електронний ресурс] – Режим доступу: *https://spring.io/projects/spring-framework* (дата звернення: 15.05.21р.)

11) *MySQL* [Електронний ресурс] – Режим доступу: *https://www.mysql.com/* (дата звернення: 15.05.21р.)

12) Интернет-магазин [Електронний ресурс] – Режим доступу: *https://www.expertplus.ru/information/stati/22/* (дата звернення: 15.05.21р.)

13) *OMS* [Електронний ресурс] – Режим доступу: *https://www.ibm.com/ru-*

ru/topics/order-management (дата звернення: 15.05.21р.)

14) *Liquibase* [Електронний ресурс] – Режим доступу: *https://habr.com/ru/post/436994/* (дата звернення: 15.05.21р.)

15) Веб-додаток [Електронний ресурс] – Режим доступу: *https://www.centum-d.com/uk/veb-dodatok-yogo-harakteristiki/* (дата звернення: 15.05.21р.)

16) Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного Авіаційного Університету. НАУ, 2017 р.

17) ГОСТ 19.701-90 ЕСКД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

18) ГОСТ 19.003-80 ЕСКД. Схемы алгоритмов и программ. Обозначения условные графические

19) ДСТУ 3582: 2013 «Бібліографічний опис скорочення слів і словосполучень в українській мові»

20) ДСТУ 8302-2015 «Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання».

ДОДАТОК А

XML код

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.oms</groupId>
  <artifactId>order-management-system</artifactId>
  <version>0.0.3-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>order-management-system</name>
  <description>Project for Spring Boot</description>
  <parent>
    <groupId>org.joinfaces</groupId>
    <artifactId>jsf-spring-boot-parent</artifactId>
    <version>2.4.1</version>
  </parent>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.liquibase</groupId>
      <artifactId>liquibase-core</artifactId>
    </dependency>

    <dependency>
      <groupId>org.joinfaces</groupId>
      <artifactId>jsf-spring-boot-starter</artifactId>
    </dependency>

    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
    </dependency>

    <dependency>
```

```
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!-- primefaces -->
<!-- https://mvnrepository.com/artifact/org.primefaces.themes/bootstrap -->
<dependency>
    <groupId>org.primefaces.themes</groupId>
    <artifactId>all-themes</artifactId>
    <version>1.0.10</version>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.4</version>
    <optional>true</optional>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>


<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>

<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.17</version>
</dependency>
```

```
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>3.10-FINAL</version>
        </dependency>
        <dependency>
            <groupId>com.lowagie</groupId>
            <artifactId>itext</artifactId>
            <version>2.1.7</version>
        </dependency>

        <dependency>
            <groupId>org.chartistjsf</groupId>
            <artifactId>ChartistJSF</artifactId>
            <version>2.1</version>
        </dependency>


    </dependencies>

    <repositories>
        <repository>
            <id>primefaces-maven-repository</id>
            <name>PrimeFaces Maven Repository</name>
            <url>http://repository.primefaces.org</url>
        </repository>
    </repositories>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>


</project>
```