МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Допустити до захисту

В.о. завідувача кафедри

_____ Мнацаканян М.С.

(підпис) *(ПІБ завідувача кафедри)*

«_____»_____20__ р.

WEB-ДОДАТОК УПРАВЛІННЯ ТОВАРООБІГОМ ДЛЯ МАЛОГО БІЗНЕСУ

Кваліфікаційна робота
здобувача вищої освіти
першого (бакалаврського) рівня вищої освіти
освітньо-професійної програми
«Комп'ютерні науки»
(назва освітньо-професійної програми)

Костенко В.О.
(прізвище, імя, по батькові здобувача вищої освіти)

Науковий керівник:
Козловський В.В., д.т.н., професор
(прізвище, ініціали, науковий ступінь, вчене звання)

Рецензент:
Охріменко Т.О., к.т.н., НАУ
(прізвище, ініціали, науковий ступінь, вчене звання, місце роботи)

Кваліфікаційна робота захищена
з оцінкою_____
Секретар ЕК_____
«____» _____20__ р.

Київ -2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти *бакалавр*
Шифр та назва спеціальності *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри к.т.н.
(науковий ступінь, вчене звання)
_____ Мнацаканян М.С.
(підпис) *(ПІБ завідувача кафедри)*
«__» _____ 20__ р.

ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Костенко Віталій Олексійович
(прізвище, ім'я, по батькові)

1. Тема роботи: «Web-додаток управління товарообігом для малого бізнесу»
керівник роботи_ Козловський В.В., д.т.н., професор_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Маріупольського державного університету від «___» _____20__ р. №__
2. Строк подання здобувачем роботи .
3. Вихідні дані до роботи (мета, об'єкт, предмет):

Об’єкт дослідження – Web-додаток .

Предмет дослідження – Web-додаток для управління товарообігом для малого бізнесу, який має можливість надавати всю необхідну для ведення бізнесу інформацію, написаний на платформі IntelliJ IDEA з використанням Java та PostgreSQL .

Мета кваліфікаційної роботи – на основі порівняльного аналізу різних видів web-додатків розробити додаток, який можна використовувати для управління товарообігом для малого бізнесу.

4. Зміст роботи

Розділ 1. Порівняльний аналіз web-додатків та можливостей їх використання.

Розділ 2. Теоретичні засади проектування web-додатку.

Розділ 3. Практична web-додатку.

5. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз предметної області, порівняння можливостей різних web-додатків. Написання першого розділу.		
2.	Опис необхідного програмного забезпечення та функцій серверної частини для створення web-додатку. Написання другого розділу		
3.	Розробка web-додатків та його тестування. Написання третього розділу		
4.	Редагування пояснювальної записки кваліфікаційної роботи.		
5.	Проходження перевірки на плагіат.		
6.	Підготовка презентації.		

Здобувач _____ _____
(підпис) *(прізвище та ініціали)*

Науковий керівник роботи _____ _____
(підпис) *(прізвище та ініціали)*

ЗМІСТ

ВСТУП..... 6
1. СТАН ДОСЛІДЖЕНОСТІ ПРОБЛЕМИ РОЗРОБКИ ВЕБ-ДОТАТКУ УПРАВЛІННЯ ТОВАРООБІГОМ ДЛЯ МАЛОГО БІЗНЕСУ 9
1.1. Аналіз термінологічного апарату дослідження 9
1.2. Специфіка веб-додатку 12
1.3. Аналіз ресурсів аналогічної тематики 14
2. МЕТОДИ ТА ЗАСОБИ ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ УПРАВЛІННЯ ТОВАРООБІГОМ ДЛЯ МАЛОГО БІЗНЕСУ 15
2.2. Основні методи та засоби проектування веб-додатку 15
2.3. Основні типи шаблонів проектування веб-додатку 16
3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ УПРАВЛІННЯ ТОВАРООБІГОМ..... 19
3.1. Розробка концепції веб-додатку 19
3.2. Програмна розробка 29
ВИСНОВОК 38
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 39
ЛІСТИНГ ПРОГРАМИ 40

ВСТУП

У сучасному світі малі бізнеси є важливим і активним гравцем у сфері товарообігу. Ефективне управління товарообігом є ключовим чинником для їх успіху та конкурентоспроможності на ринку. Однак, багато малих бізнесів стикаються з низкою викликів та проблем, пов'язаних з організацією та контролем процесів управління товарами. Традиційні методи та ручний підхід до управління можуть бути недостатньо ефективними та часомірними, що обмежує ріст та розвиток цих підприємств.

Отже, метою даної дипломної роботи є розробка Web-додатку для управління товарообігом малого бізнесу. Цей додаток пропонує нове рішення для оптимізації та автоматизації процесів управління товарами, спрощуючи рутинні завдання та покращуючи продуктивність.

Актуальність цієї теми полягає у тому, що сучасний розвиток електронної комерції та інтернет-технологій вимагає від малих бізнесів швидкого та ефективного реагування на змінні умови ринку. Web-додатки для управління товарообігом надають можливість підприємствам забезпечити точність та актуальність даних про продукти, контролювати запаси, оптимізувати процеси доставки та забезпечити більш ефективну взаємодію з клієнтами.

Наукова і практична значимість даної роботи полягає в тому, що розроблений Web-додаток буде мати практичне застосування для малих бізнесів у сфері товарообігу. Він дозволить підприємствам покращити ефективність управління товарами, знизити ризики помилок та зайвих витрат, а також забезпечить підтримку росту та розвитку бізнесу.

Для досягнення поставленої мети дипломна робота буде базуватися на аналізі потреб та вимог малого бізнесу щодо управління товарообігом, використанні сучасних методів та технологій розробки веб-додатків, а також розробці функціональних модулів та інтерфейсу додатку.

У подальших розділах цієї дипломної роботи будуть детально розглянуті процес розробки, вибрані технології, реалізація функцій та перевірка ефективності розробленого додатку. Заключній розділ міститиме висновки щодо досягнутих результатів, рекомендації для практичного використання додатку та перспективи подальшого його розвитку.

Ця дипломна робота важлива, оскільки вона спрямована на вирішення актуальної проблеми малого бізнесу щодо управління товарообігом та пропонує нове інструментальне рішення для покращення продуктивності та ефективності.

Приведемо декілька більш розповсюджених проблем та труднощів, зв’язаних з розробко веб-додатків:

1. Вибір технології: Існує безліч технологій, язиків програмування, фреймворків та інструментів, доступних для розробки веб-додатків. Вибір правильної комбінації може бути важким, здебільшого для команд, не маючих досвіду в даній сфері. Невірний вибір технології може призвести до проблеми з продуктивністю, маштабованістю та обслуговуванню програми.
2. Безпека: Веб-додатки є пріоритетними цілями для хакерів та зловмисників. Недоліки у захисті даних та небезпечні практики програмування можуть призвести до вразливостей, таким як ін’єкція SQL,

міжсайтовий скриптинг (XSS), підробка міжсайтових запитів (CSRF) ті іншим. Забезпечення безпеки веб-додатку є важливим аспектом розробки.

3. Маштабованість: Коли веб-додаток розвивається та збільшує чисто користувачів, виникають проблеми з маштабованістю. Неправильна архітектура чи недостатня оптимізація можуть призвести до зниження продуктивності програми інколи і до непрацездатності. Необхідно заздалегідь продумати можливість горизонтального та вертикального маштабуваня системи.
4. Управління станом: Веб-додаток може бути складним з точки зору управління станом. Програми, які мають велику кількість станів та взаємодій, можуть стати складними для налагодження та підтримки. Використання правильної архітектури управління станом, такої як Flux чи Redux, можуть допомогти спростити таку проблему.
5. Сумісність та браузерна підтримка: Різні браузери мають різні стандарти та функціональність, тому веб додаток має розроблятись з урахуванням сумісності та підтримки різних браузерів.

1. СТАН ДОСЛІДЖЕНОСТІ ПРОБЛЕМИ РОЗРОБКИ ВЕБ-ДОТАТКУ УПРАВЛІННЯ ТОВАРООБІГОМ ДЛЯ МАЛОГО БІЗНЕСУ

1.1. Аналіз термінологічного апарату дослідження

Аналіз термінологічного апарату дослідження допоможе нам розібратися в основних термінах та поняттях, пов'язаних із розробкою веб-додатку для управління товарообігом малого бізнесу. Таких як:

Розробка веб-застосунку - процес створення програмного забезпечення, спеціально призначеного для використання через інтернет, за допомогою веб-браузера. Містить у собі ряд єтапів:

Планування - цей етап включає визначення мети веб-додатку, його функціональності та макету. Треба створити макети і зберегти їх у вигляді дизайнів, щоб мати уявлення про те, як буде виглядати та працювати ваш застосунок.

Фронтенд-розробка - етап яий включає в себе створення користувацького інтерфейсу веб-додатку. Ви можете використовувати HTML, CSS та JavaScript для створення сторінок та взаємодії з користувачами.

Бекенд-розробка - на цьому етапі я розробляв логіку та функціональність веб-застосунку. Проаналізувавши різні технології, такі як Python, PHP, Ruby або Node.js, для розробки серверної частини застосунку я обрав Java.

База даних – мій веб-додаток вимагає зберігання даних, при проектуванні базя даних, обрав PostgreSQL, для збереження та керування цими даними.

Тестування - навіть після завершення розробки, важливо протестувати веб-додаток, щоб переконатися, що він працює належним чином і не містить помилок або проблем.

Розгортання - після успішного тестування веб-додатку треба розгорнути його на веб-сервері, щоб користувачі могли отримати до нього доступ. Вибір платформи для розгортання залежить від вашого веб-сервера та технологій, що ви використовуєте.

- Управління товарообігом: Контроль та регулювання потоку товарів чи послуг усередині бізнесу. Це включає управління поставками, закупівлями, інвентаризацією, продажами та іншими аспектами пов'язаними з товарами або послугами.

- Малий бізнес: Комерційне підприємство, що має обмежені ресурси і менші масштаби операцій порівняно з великими корпораціями. У різних країнах можуть бути різні критерії визначення малого бізнесу, такі як кількість співробітників або річний оборот.
- Інвентаризація: Процес обліку та контролю запасів товарів чи матеріалів у компанії. Включає відстеження кількості, вартості і розташування запасів.
- Постачальник: Організація чи компанія, яка надає товари чи послуги іншої компанії чи клієнту.
- Замовлення: Запит на придбання товару чи послуги. У контексті веб-додатку для управління товарообігом, замовлення зазвичай включає інформацію про товар, кількість, клієнта та інші деталі.
- Клієнт: Особа чи організація, яка купує товари чи послуги у компанії. У контексті малого бізнесу клієнти можуть бути як фізичними особами, так і іншими компаніями.

- Аналітика: Збір, аналіз та інтерпретація даних для прийняття рішень та оптимізації бізнес-процесів. Аналітика може включати в себе звітність про продаж, популярність товарів, переваги клієнтів та іншу інформацію.
- Інтеграція: Процес об'єднання різних систем або програм для обміну даними та забезпечення їх взаємодії. У контексті веб-додатку для керування товарообігом інтеграція може включати зв'язок із системами обліку, платіжними системами або іншими додатками.
- Веб-додаток: програмне забезпечення, розроблене для виконання певних завдань через веб-браузер. Воно може надавати доступ до бази даних, функцій керування та інших можливостей через Інтернет.
- Інтерфейс користувача: візуальне подання та взаємодія між користувачем та програмою. Це може включати елементи дизайну, кнопки, форми, таблиці та інші елементи, які дозволяють користувачеві взаємодіяти з додатком.
- Технічні вимоги: набір специфікацій та параметрів, що визначають функціональність та характеристики веб-програми. Вони можуть включати мови програмування, бази даних, вимоги до безпеки, масштабованості та інші технічні аспекти.
- Проектування бази даних: процес створення структури та організації бази даних, яка використовуватиметься у веб-додатку. Включає визначення таблиць, зв'язків між ними, типів даних та інших аспектів зберігання і обробки даних.

1.2. Специфіка веб-додатку

Веб-додаток для управління товарообігом малого бізнесу має бути спеціально розроблений для особливої уваги сегмента. Ось деякі особливості та функції, які можна врахувати при створенні таких додатків:

Управління інструментом

Включає додавання нових товарів, оновлення інформації про очікувані товари, відстеження рівня запасів і наближення до необхідності дозамовлення товарів.

Оформлення замовлень та продажів

Веб-додаток дозволяє легко отримувати замовлення та отримувати товари. Це може бути додано функції, такі як товари в кошик, встановлення цін, розрахунок вартості загальної заявки, генерація рахунків і можливість приймати онлайн-платежі.

Облік фінансів

Сталась подія для обліку фінансів малого бізнесу. Це може спостерігатись у функції з відстеження доходів, витрат, прибутку, податків та фінансових звітів.

Керування клієнтами

Важливим аспектом є управління інформацією про клієнтів, включаючи контактні дані, історію замовлень, переваги та іншу супутню інформацію. Потрібна була можливість створити та створити профілі клієнтів, а також проаналізувати дані для більш ефективного управління відносинами з клієнтами.

Аналітика та звітність

Веб-додаток може використовувати наявні інструменти та можливості генерації звітів. Це допоможе власникам бізнесу проаналізувати дані про продаж, запаси, прибутки та інші пропозиції для реалізації інформаційних рішень.

Інтеграція з іншими продуктами

Веб-додаток може бути об’єднаний з іншими функціями, реалізованими як система обліку, електронна комерція, платіжні системи тощо. Це дозволяє автоматизувати процеси та здійснювати більш ефективну роботу бізнесу.

Зручний інтерфейс та мобільна доступність

Програма повинна мати доступний та простий інтерфейс у колекції. Воно також може бути доступним на мобільних пристроях, щоб мати можливість керувати бізнесом у будь-який час і в будь-якому місці.

Безпека даних

Особлива увага має бути приділена безпеці даних. Додаток повинен забезпечувати захист конфіденційної інформації про клієнтів, замовлення, фінанси та інші ділові зустрічі.

Важливо відмітити, що характеристики та вимоги можуть бути дуже різними залежно від малого бізнесу. При розробці програми рекомендується використовувати потенційні користувачі та бізнес-аналітикам, щоб визначити основні функції та особливості, найбільш важливі для конкретного бізнесу.

1.3. Аналіз ресурсів аналогічної тематики

При аналізі ресурсів аналогічної тематики для розробки веб-додатку управління товарообігом для малого бізнесу, звернувся до наступних джерел:

Веб-додатки та платформи для управління запасами: Приклади таких ресурсів включають TradeGecko, Cin7, Zoho Inventory та Ordoro. Вивчіть їх функціонал, можливості та користувацький досвід.

Блоги та форуми про управління запасами: Пошукайте блоги та форуми, де обговорюються питання управління запасами для малого бізнесу. Наприклад, Inventory Management Blog, Small Business Bonfire або розділи, присвячені управлінню запасами на форумах типу Reddit або Quora.

Веб-ресурси про розробку веб-додатків: Для практичних порад та ресурсів щодо розробки веб-додатків, таких як фронтенд і бекенд розробка, використання фреймворків та інструментів, відвідайте ресурси, такі як MDN Web Docs, Stack Overflow, Dev.to, Medium та YouTube-канали, присвячені розробці програмного забезпечення.

Відкриті джерела проектів: Деякі проекти управління товарообігом для малого бізнесу можуть бути розроблені на основі відкритих джерел. Перегляньте популярні відкриті проекти, такі як Odoo, ERPNext або OpenBoxes, які можуть мати модулі або функціонал, пов’язаний з управлінням запасами та товарообігом.

Курси та навчальні матеріали: Вивчайте онлайн-курси та навчальні матеріали, присвячені розробці веб-додатків та управлінню запасами. Сервіси, такі як Udemy, Coursera або Codecademy, можуть пропонувати корисні курси з програмування та розробки

Ці ресурси допомогли мені зрозуміти поточний стан ринку, основні функціональні можливості та найкращі практики для розробки веб-додатку.

2. МЕТОДИ ТА ЗАСОБИ ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ УПРАВЛІННЯ ТОВАРООБІГОМ ДЛЯ МАЛОГО БІЗНЕСУ

2.2. Основні методи та засоби проектування веб-додатку

Ретельно проаналізувавши потреби та вимоги малого бізнесу до системи управління товарообігом. Зрозумів, які функції і можливості повинен мати додаток.

При розробці структуру бази даних, яка відображатиме потрібні елементи управління товарообігом. Обрав реляційну модель бази даних та розробив таблиці товари, замовлення, склад тощо.

Використовуючи інструменти для прототипування, наприклад, Figma, Sketch або Adobe XD створив прототипи веб-додатку, які демонструють його основні функціональні елементи та навігацію.

Визначте технологічний стек для розробки веб-додатку. Виберіть фреймворки для фронтенду (наприклад, React, Angular, Vue.js) та бекенду (наприклад, Node.js, Django, Ruby on Rails). Також оберав базу даних (PostgreSQL) та інші необхідні інструменти та бібліотеки.

Реалізувавши функції, які задовольняють вимоги малого бізнесу. Функції управління запасами (додавання, видалення, редагування товарів), замовлення та обробка платежів, створення звітів тощо.

Протестував веб-додаток для перевірки його функціональності, стабільності та безпеки.

Розгорнув веб-додаток на веб-сервері та забезпечив його належну роботу. Після розгортання забезпечив підтримку для користувачів. У перскективі під час виявлення проблем планується впровадження нового функціоналу на основі зворотного зв’язку та потреб малого бізнесу.

2.3. Основні типи шаблонів проектування веб-додатку

При проектуванні веб-додатку можна використовувати різні типи шаблонів проектування, щоб забезпечити ефективну організацію коду та покращити перевикористання компонентів.

Модель-Вид-Контролер (Model-View-Controller, MVC)

Цей шаблон розділяє додаток на три основні компоненти – модель (представляє дані та бізнес-логіку), вид (відображає інтерфейс користувача) та контролер (управляє взаємодією між моделлю та видом). MVC дозволяє розділити логіку додатку та відображення даних.

Модель-Вид-Шаблон (Model-View-Template, MVT)

Цей шаблон, використовуваний у фреймворку Django, також розділяє додаток на модель (дані), вид (управляє відображенням) та шаблон (визначає, як дані відображаються в HTML). MVT спрощує розробку, розділяючи ролі різних компонентів.

Компонентно-орієнтований шаблон (Component-Based Pattern)

Цей шаблон базується на розбитті додатку на незалежні компоненти, які можуть бути перевикористані. Кожен компонент включає в себе як HTML-код, так і логіку. Реактивні фреймворки, такі як React або Vue.js, сприяють реалізації цього шаблону.

Репозиторій (Repository)

Цей шаблон орієнтований на організацію доступу до даних. Репозиторій забезпечує інтерфейс для отримання та збереження даних, приховуючи деталі

роботи з базою даних. Використання цього шаблону дозволяє легко змінювати джерело даних, не змінюючи код в інших частинах додатку.

Застосування/Композит (Application/Composite)

Цей шаблон використовується для організації складних ієрархій компонентів. Кожен компонент може мати підкомпоненти, утворюючи ієрархічну структуру. Це особливо корисно для додатків з багатьма рівнями навігації або динамічним створенням інтерфейсу.

Засіб відправлення повідомлень (Message Bus)

Цей шаблон використовує механізм передачі повідомлень між компонентами. Компоненти можуть публікувати та передавати повідомлення через шину повідомлень, спілкуючись один з одним без прямого зв’язку. Це забезпечує розробку локально зв’язаних компонентів.

Ці шаблони проектування допомагають забезпечити структуру та організацію веб-додатку, полегшують розробку, підтримку та розширення. Вибір шаблону залежить від конкретних потреб та вимог вашого проекту.

Для свого дипломного проекту я обрав Модель-Вид-Контролер (Model-View-Controller, MVC) так як вона є одним з найпопулярніших шаблонів проектування для розробки веб-додатків. Він допомагає відокремити логіку додатку, дані та інтерфейс користувача, що сприяє полегшенню розробки, тестування та підтримки.

Основні компоненти шаблону MVC:

Модель (Model): Модель відповідає за обробку даних та бізнес-логіку додатку. Вона представляє собою структуру даних, яка може бути отримана, змінена або збережена. Модель не залежить від інтерфейсу користувача або

контролера і може сповіщати про зміни даних за допомогою патерну «Спостерігач» (Observer).

Вид (View): Вид відповідає за представлення даних користувачеві та відображення інтерфейсу. Він отримує дані з моделі та відображає їх у відповідному форматі (наприклад, HTML). Вид також може сприймати взаємодію користувача та повідомляти про це контролер.

Контролер (Controller): Контролер відповідає за обробку взаємодії користувача та керування логікою додатку. Він отримує вхідні дані від користувача через вигляд, виконує необхідні дії та оновлює модель та вид. Контролер також може забезпечувати маршрутизацію запитів та відповідей.

Основний принцип роботи шаблону MVC полягає в тому, що модель і вид не взаємодіють безпосередньо, але залежать від контролера. Контролер служить посередником між моделлю та видом, керуючи потоком даних та взаємодією з користувачем.

Переваги шаблону MVC:

- Розділення логіки додатку та інтерфейсу користувача, що полегшує розробку та підтримку.

- Можливість повторного використання компонентів (наприклад, моделі) у різних частинах додатку.

- Забезпечення складнішої структури додатку, де кожен компонент має свою відповідальність.

- Покращена тестованість, оскільки окремі компоненти можуть бути тестовані незалежно один від одного.

Шаблон MVC широко застосовується в різних веб-фреймворках, таких як Ruby on Rails, Laravel, Spring MVC та багатьох інших, тому він є корисним для розробки мого веб-додатку.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ УПРАВЛІННЯ ТОВАРООБІГОМ

3.1. Розробка концепції веб-додатку

Створив веб-додатку, який демонструє основну функціональність та вигляд. Це простий додаток, завдяки чому користуватися їм доволі зручно.

Робота з веб-додатком починається з відкриття сайту та реєстрації. Треба ввести Ім’я, прізвище, пошту та пароль (рисунок3.1).

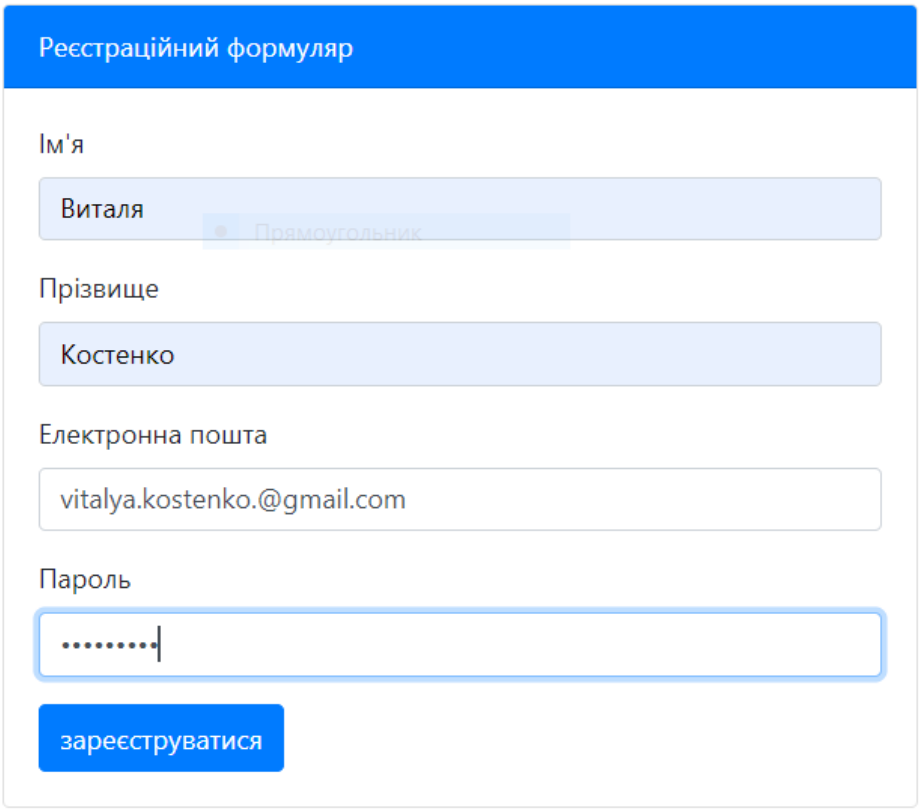


Рисунок 3.1 - Реєстрація

Форма авторизації відповідає за вхід на сайт. Необхідно обрати користувача і ввести пароль (рисунок 3.2).

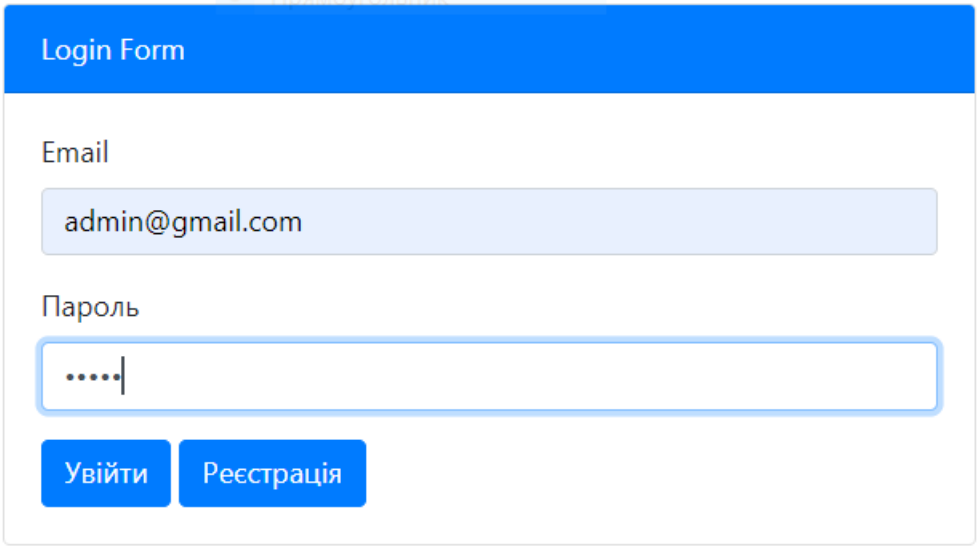


Рисунок 3.2 – авторизація користувача

Якщо логін або пароль введено невірно, з’являється попередження (рисунок 3.3).

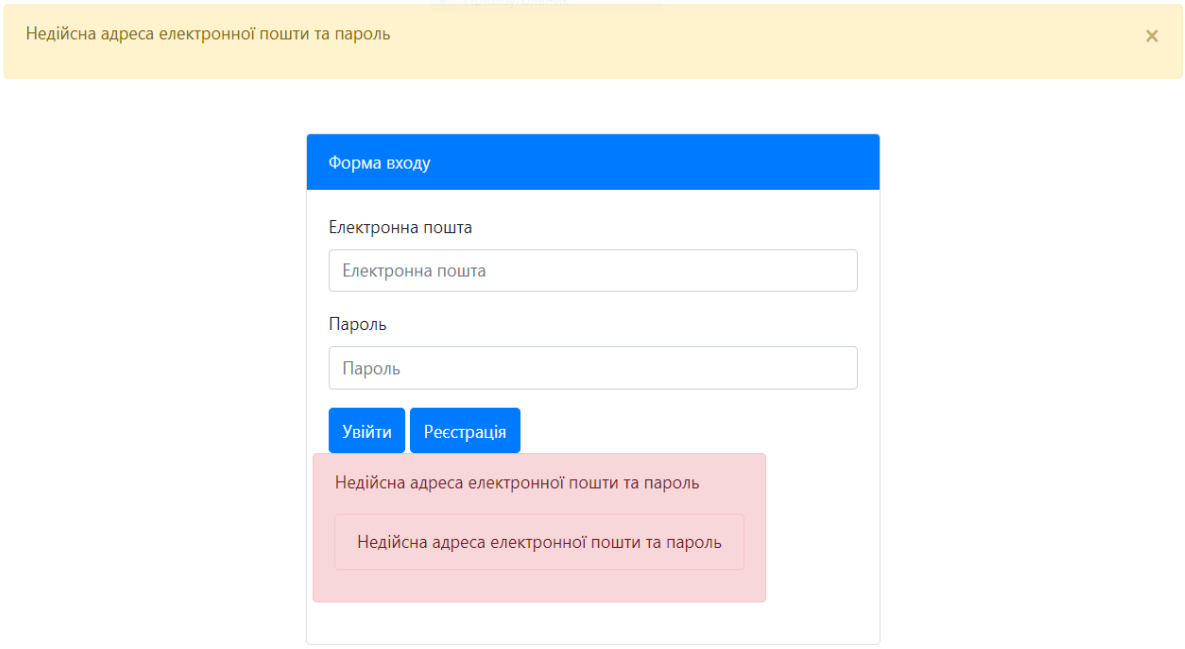


Рисунок 3.3 – Повідомлення про помилку авторизації

Після успішної відкривається головна сторінка сайту, яка використовується для звітності продажу. У шапці сайту знаходиться функції переходу на інші сторінки сайту а саме товари та документи. Також у правій частині сайту знаходиться кнопка щоб змінити користувача (рисунок 3.4).

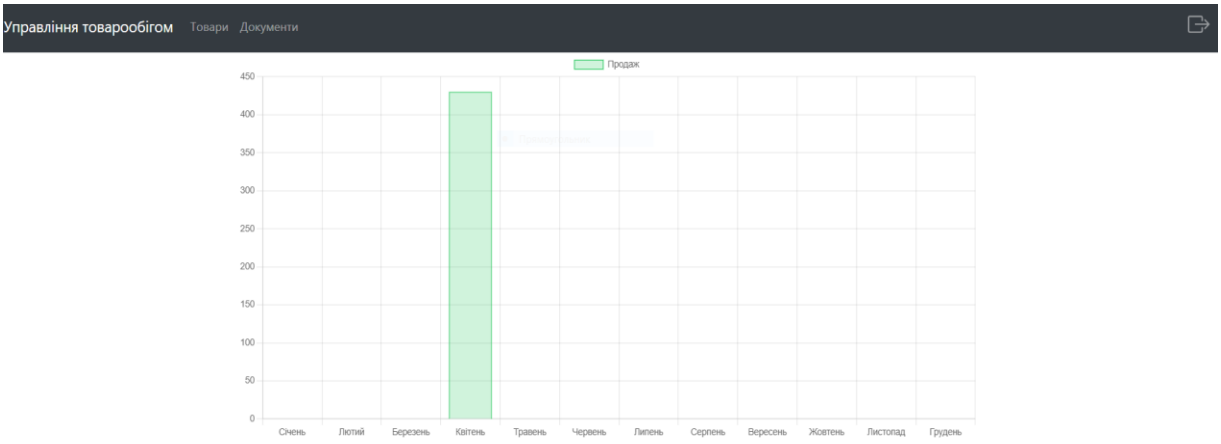


Рисунок 3.4 – Головна сторінка сайту

Сторінка товари містить у собі таблицю з товарами їх назву, ціну, категорію товару (одяг, будинок, книги та інше)(рисунок 3.5).

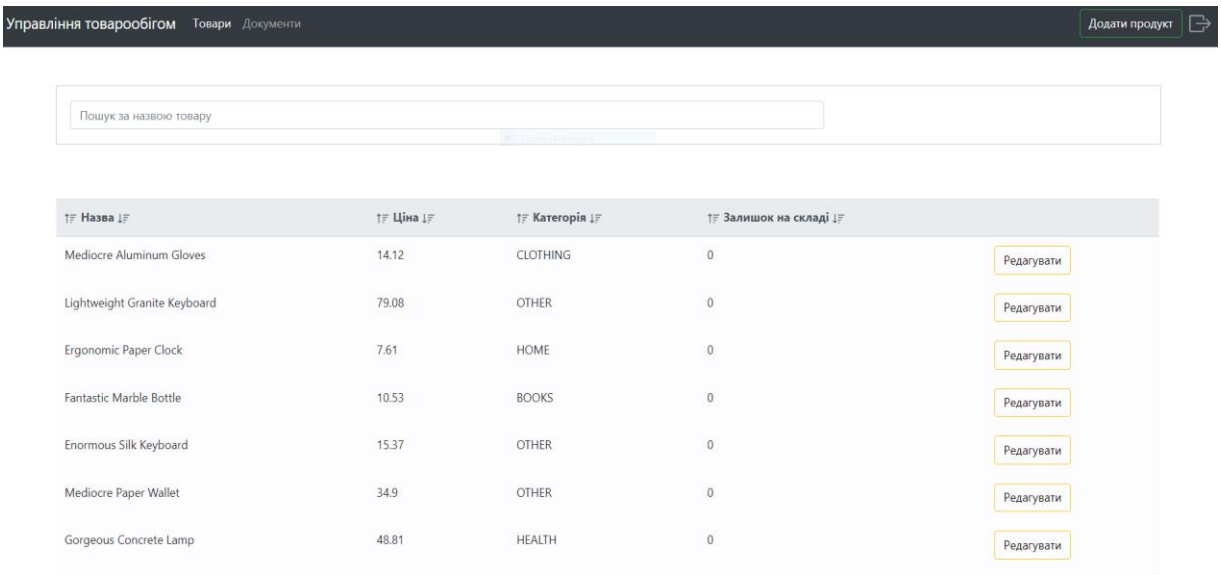


Рисунок 3.5 – Сторінка товарів

Кожен товар можна відредагувати. Змінити в ньому назву, ціну, котигорію та опис. Знизу форми є кнопка Редагувати, це буде значити що відредаговані данні збережуться. Також є кнопка Відміна – якщо ви обрали не той товар або ввели не той опис (рисунок 3.6).

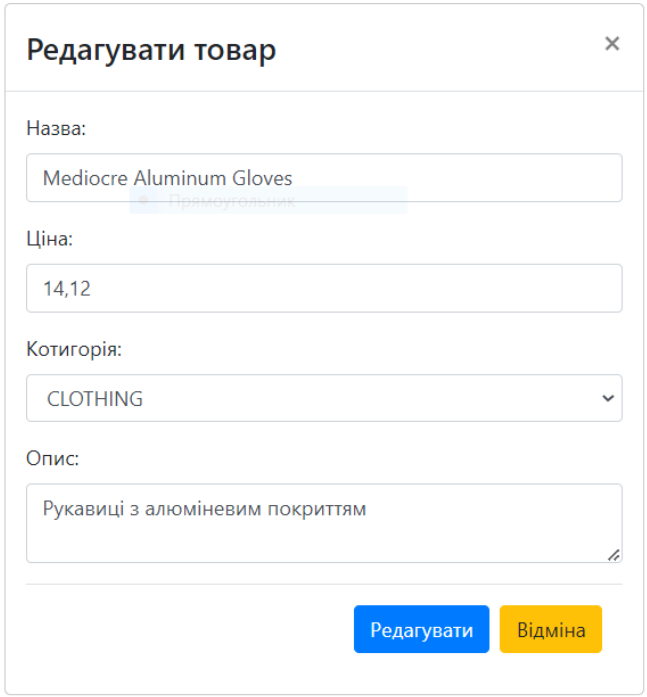


Рисунок 3.6 – Редагування товару

На сторінці Товари можна додавати нові товари. Натиснувши на кнопку Додати продукт з’явиться форма у якій ми зможемо обрати назву товару, його ціну. У полі категорії зможемо обрати із вже зазначених категорій те що більш підходить нашому товару. У полі опис додати інформацію про товар. У низу форми є кнопка зберегти товар а якщо товар не потрібен, натиснувши кнопку скинути скасуємо додавання (рис 3.7).

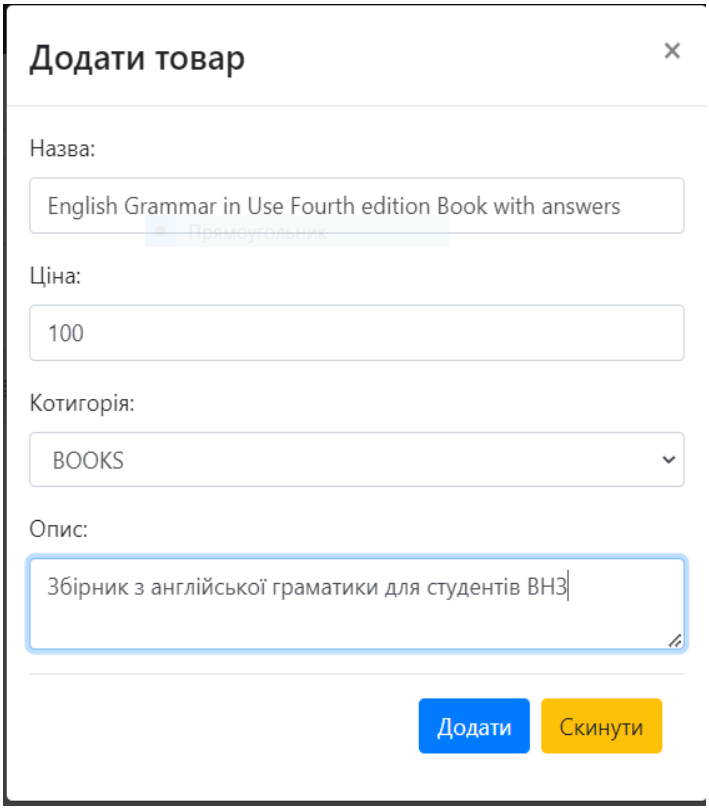


Рисунок 3.7 – Додавання товару.

Сторінка документи містить у собі таблицю з товарами їх датою, номером документа, сумою, типом документа прийом або продаж товару та кількість товару. Також кожну операцію з товарами можна роздрукувати чек. (рисунок 3.8).

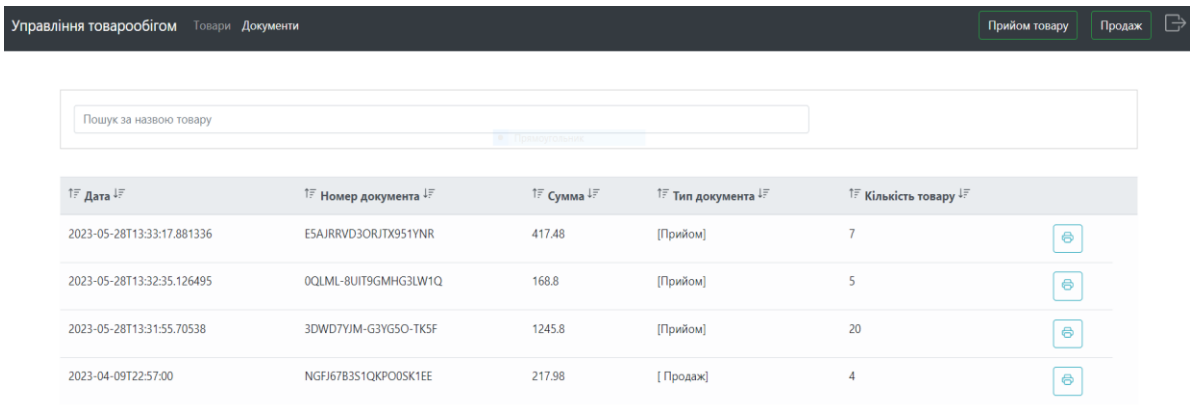


Рисунок 3.8 – Сторінка документи

При натисканні на кнопку прийому товару з’явиться форма додавання кількості вже існуючих позицій товарів. При натисканні кнопки «Створити» ми додаємо кількість товарів. При натисканні на «Close» ми відміняємо операцію(рисунок 3.9).

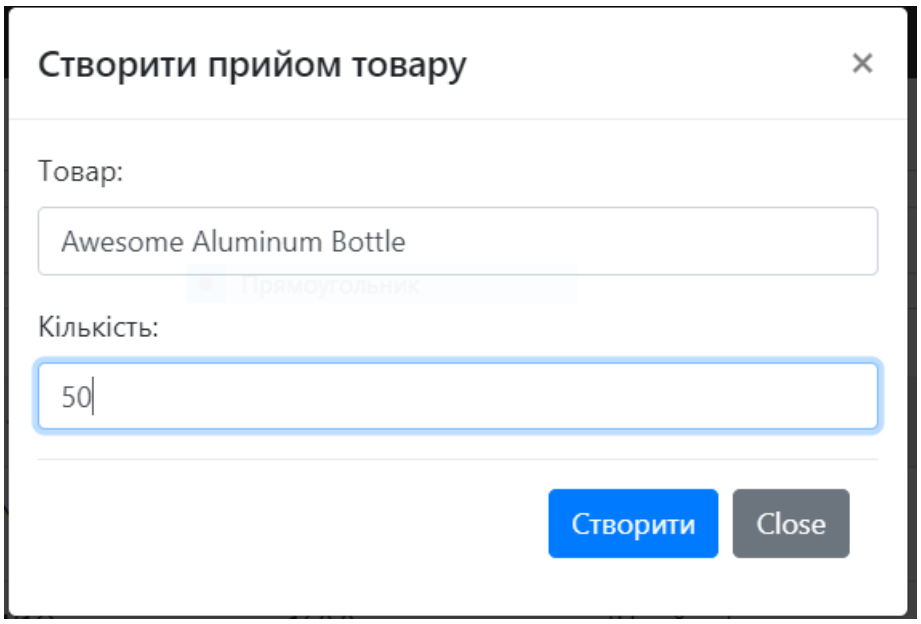


Рисунок 3.9 – Прийом товару

Продаж товару здійснюється за допомогою натискання на «Продаж». У формі є пошук за назвою товару. Для продажу будуть доступні тільки ті товари які є у наявності. Товари можна додавати до замовлення за допомогою кнопка «+». У замовленні є таблиця з полями назва товару, ціна, кількість товару, тип товару та ціна разом за кожен товар. У низу таблиці відображається фінальна вартість. Замовлення зберегается після натискання на «Збереження». Також можна оновити замовлення натиснувши на знак оновлення (Рисунок 3.10)

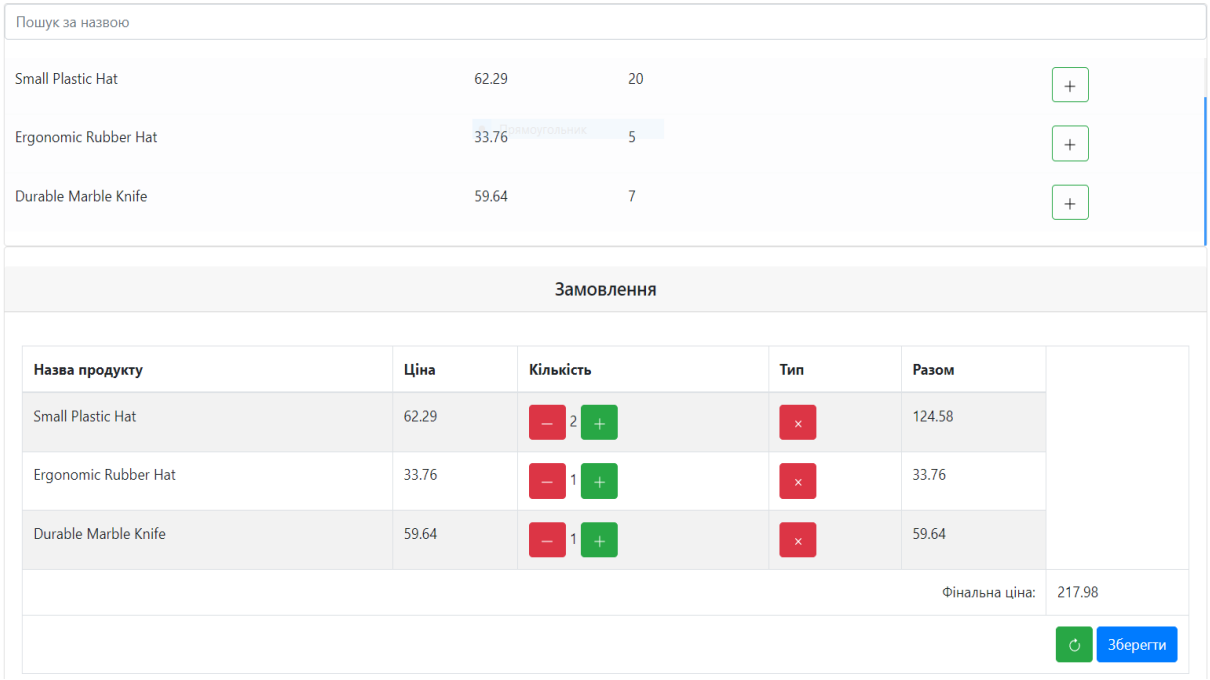


Рисунок 3.10 – Продаж

На сторінці «Товари» для роздрукування чеку у кінці кожної строки є значок друку. Інформація зберегається у PDF форматі (рисунок 3.11).

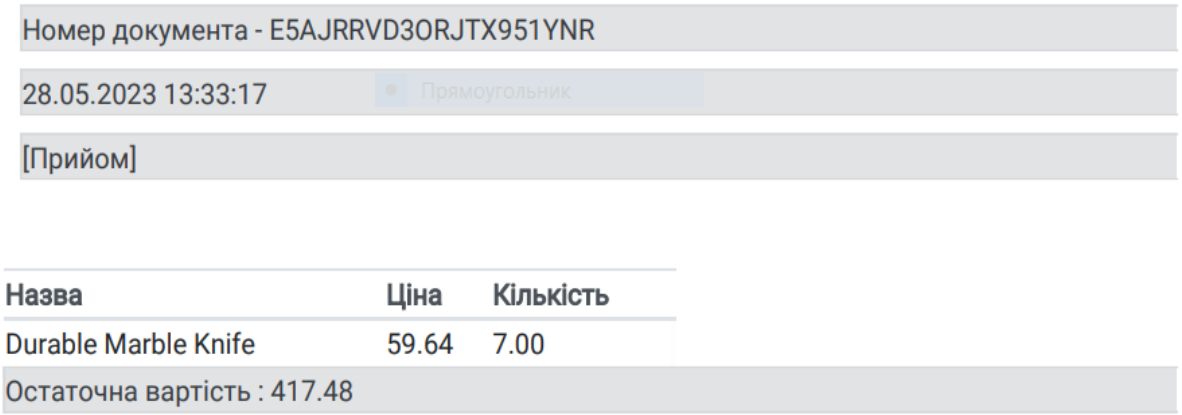
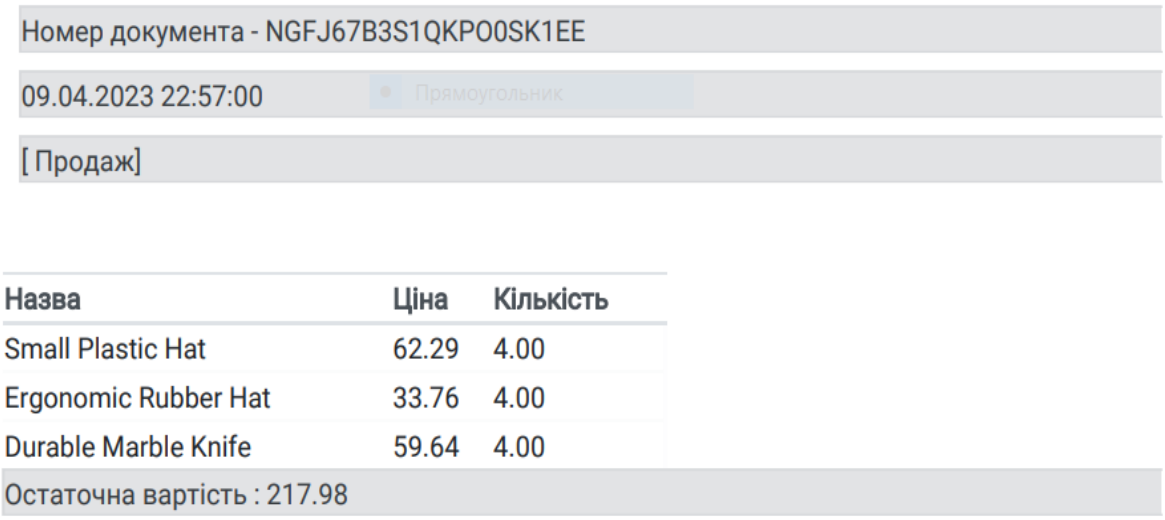


Рисунок 3.11 – Чек прийому товару

Для роздрукування чеку про продаж товару вибираємо строку з продажом, натискаємо на значок друку(рисунок 3.12).


Номер документа - NGFJ67B3S1QKPO0SK1EE

09.04.2023 22:57:00

[Продаж]

Назва	Ціна	Кількість
Small Plastic Hat	62.29	4.00
Ergonomic Rubber Hat	33.76	4.00
Durable Marble Knife	59.64	4.00

Остаточна вартість : 217.98

Рисунок 3.12 – Чек продажу товару

3.2. Програмна розробка

PostgreSQL — це система керування реляційними базами даних (RDBMS) із відкритим кодом, яка відома своєю надійністю, розширюваністю та дотриманням стандартів SQL. PostgreSQL пропонує широкий спектр функцій і можливостей, що робить його популярним вибором як для малих, так і для корпоративних програм.

Визначив структуру бази даних, розробив схему баз даних.
Яка складається з 7 таблиць: «Користувач», «Права користувача», «Документи», «Продукти», «Авторизація», «Чек прийому товару», «Чек продажу товару» (рисунок 3.13).

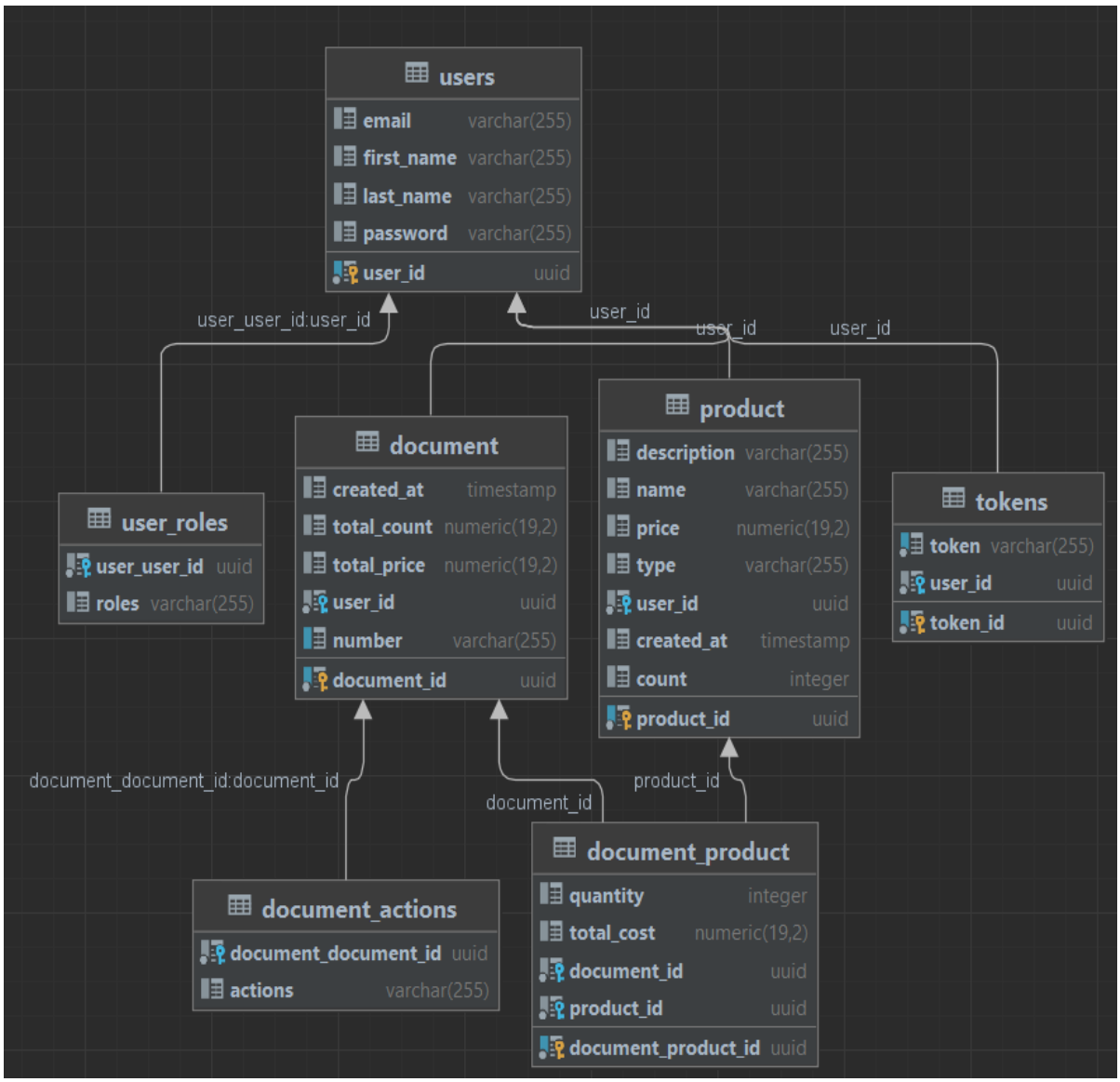
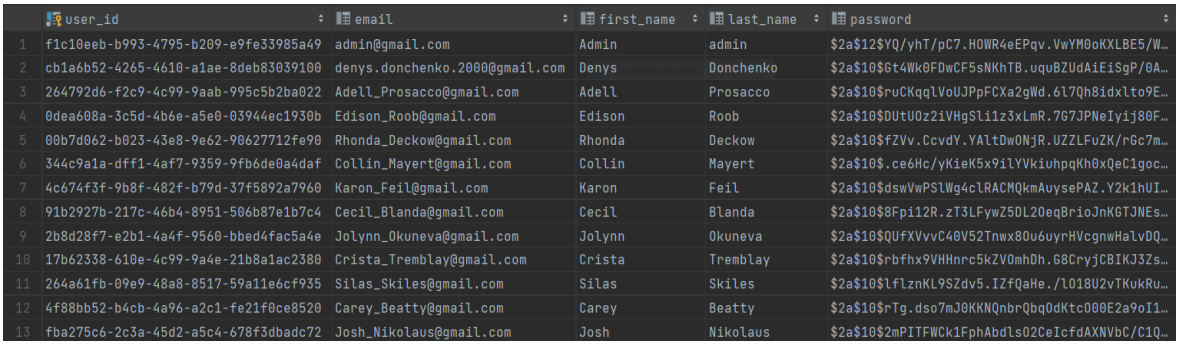


Рисунок 3.13 – Схема баз данних

Таблиця «Користувач» (users) складається з полів Користувач (первинний унікальний ключ), Пошта, Ім’я, Прізвище, Пароль (рисунок 3.14).



	user_id	email	first_name	last_name	password
1	f1c10eeb-b993-4795-b209-e9fe33985a49	admin@gmail.com	Admin	admin	$2a$12$YQ/yhT/pC7.HOWR4eEPqv.VwYM0oKXLBE5/W…
2	cb1a6b52-4265-4610-a1ae-8deb83039100	denys.donchenko.2000@gmail.com	Denys	Donchenko	$2a$10$Gt4WkOFDwCF5sNKhTB.uquBZUdAiEiSgP/0A…
3	264792d6-f2c9-4c99-9aab-995c5b2ba022	Adell_Prosacco@gmail.com	Adell	Prosacco	$2a$10$ruCKqqlVoUJPpFCXa2gWd.6l7Qh8idxlto9E…
4	0dea608a-3c5d-4b6e-a5e0-03944ec1930b	Edison_Roob@gmail.com	Edison	Roob	$2a$10$DUtUOz2iVHgSli1z3xLmR.7G7JPNeIyij80F…
5	00b7d062-b023-43e8-9e62-90627712fe90	Rhonda_Deckow@gmail.com	Rhonda	Deckow	$2a$10$fZVv.CcvdY.YAltDwONjR.UZZLFuZK/rGc7m…
6	344c9a1a-dff1-4af7-9359-9fb6de0a4daf	Collin_Mayert@gmail.com	Collin	Mayert	$2a$10$.ce6Hc/yKieK5x9ilYVkiuhpqKh0xQeC1goc…
7	4c674f3f-9b8f-482f-b79d-37f5892a7960	Karon_Feil@gmail.com	Karon	Feil	$2a$10$dswVwPSlWg4clRACMQkmAuysePAZ.Y2k1hUI…
8	91b2927b-217c-46b4-8951-506b87e1b7c4	Cecil_Blanda@gmail.com	Cecil	Blanda	$2a$10$8Fpi12R.zT3LFywZ5DL2OeqBrioJnKGTJNEs…
9	2b8d28f7-e2b1-4a4f-9560-bbed4fac5a4e	Jolynn_Okuneva@gmail.com	Jolynn	Okuneva	$2a$10$QUfXVvvC40V52Tnwx8Ou6uyrHVcgnwHalvDQ…
10	17b62338-610e-4c99-9a4e-21b8a1ac2380	Crista_Tremblay@gmail.com	Crista	Tremblay	$2a$10$rbfhx9VHHnrc5kZVOmhDh.G8CryjCBIKJ3Zs…
11	264a61fb-09e9-48a8-8517-59a11e6cf935	Silas_Skiles@gmail.com	Silas	Skiles	$2a$10$lflznKL9SZdv5.IZfQaHe./lO18U2vTKukRu…
12	4f88bb52-b4cb-4a96-a2c1-fe21f0ce8520	Carey_Beatty@gmail.com	Carey	Beatty	$2a$10$rTg.dso7mJ0KKNQnbrQbqOdKtcO00E2a9oI1…
13	fba275c6-2c3a-45d2-a5c4-678f3dbadc72	Josh_Nikolaus@gmail.com	Josh	Nikolaus	$2a$10$2mPITFWCk1FphAbdlsO2CeIcfdAXNVbC/C1Q…

Рисунок 3.14 – Таблиця БД Користувач

Таблиця «Права користувача» (users_roles) складається з полів Користувач (первинний унікальний ключ), Права (Рисунок 3.15).

	user_user_id	roles
1	cb1a6b52-4265-4610-a1ae-8deb83039100	ADMIN
2	264792d6-f2c9-4c99-9aab-995c5b2ba022	ADMIN
3	0dea608a-3c5d-4b6e-a5e0-03944ec1930b	ADMIN
4	00b7d062-b023-43e8-9e62-90627712fe90	ADMIN
5	344c9a1a-dff1-4af7-9359-9fb6de0a4daf	ADMIN
6	4c674f3f-9b8f-482f-b79d-37f5892a7960	ADMIN
7	91b2927b-217c-46b4-8951-506b87e1b7c4	ADMIN
8	2b8d28f7-e2b1-4a4f-9560-bbed4fac5a4e	ADMIN
9	17b62338-610e-4c99-9a4e-21b8a1ac2380	ADMIN
10	264a61fb-09e9-48a8-8517-59a11e6cf935	ADMIN
11	4f88bb52-b4cb-4a96-a2c1-fe21f0ce8520	ADMIN
12	fba275c6-2c3a-45d2-a5c4-678f3dbadc72	ADMIN
13	3b668b82-c38d-463f-9179-359c6994e519	ADMIN

Рисунок 3.15 - Таблиця БД Права користувача

Таблиця «Документи» (documents) складається з полів Документ (первинний унікальний ключ), Дата, Ціна товару, Ціна товарів, Користувач (первинний унікальний ключ), Номер документу(рисунок 3.16).

	document_id	created_at	total_count	total_price	user_id	number
1	538bc2b3-24f9-4264-bada-648040cf53ff	2023-03-26 18:25:00.000000	17.00	1207.68	cb1a6b52-4265-4610-a1ae-8deb83039100	HB-M01LXDPFQMZO-YPXN
2	865f6d3f-1733-43c3-90ba-0b428035423c	2023-05-04 10:56:00.000000	28.00	780.36	cb1a6b52-4265-4610-a1ae-8deb83039100	EO5ZMOY82UPMOSD8X80N
3	74e83b9b-4454-4ae2-8d46-7595fe31c6e0	2023-01-29 09:11:00.000000	23.00	1138.04	cb1a6b52-4265-4610-a1ae-8deb83039100	IFCDYFKBM20-OM63IOKF
4	d5c7c6e1-ff00-4d29-8ae1-1fa5acf290ae	2023-05-21 07:15:00.000000	23.00	63.25	cb1a6b52-4265-4610-a1ae-8deb83039100	3F1U35CN4RT0PSB7A4HA
5	a5b882cc-0b48-44c4-9082-6330490a1e28	2023-05-09 17:22:00.000000	1.00	77.32	cb1a6b52-4265-4610-a1ae-8deb83039100	EFMBW62VG-D-Z9L34OPK
6	b276bacb-e181-4655-b96f-59bcc1fae23e	2023-04-11 07:55:00.000000	21.00	431.76	cb1a6b52-4265-4610-a1ae-8deb83039100	OVSK2W1SNSWGJSSQKTTP
7	d1c2abba-58e0-443b-a543-d46fc9383298	2023-03-03 18:07:00.000000	5.00	355.20	cb1a6b52-4265-4610-a1ae-8deb83039100	DD3V2XDLPFXLRK0OQ0H9
8	fc39df04-da20-4827-8919-0b2d9b7c6120	2023-04-26 17:08:00.000000	1.00	24.76	cb1a6b52-4265-4610-a1ae-8deb83039100	W3RZ29PFQ8QFV5NEUMJ
9	aa69f9d1-e80f-4320-ae8e-b49d2db1fa1c	2023-03-05 07:46:00.000000	8.00	140.40	cb1a6b52-4265-4610-a1ae-8deb83039100	P5VSGLI1TN5SH44Z30QU
10	34087a19-aa39-4f57-a19f-aad1a12eef3d	2023-01-03 13:38:00.000000	25.00	1895.00	cb1a6b52-4265-4610-a1ae-8deb83039100	HL6ZKVF9I1S2G2KDA-U2
11	4cfeb59f-ac71-4146-a7cc-1de8e3c5cdeb	2023-05-30 08:19:00.000000	16.00	323.84	cb1a6b52-4265-4610-a1ae-8deb83039100	HA6-7R5W8DISQX2LH-DG
12	6239870f-7fe7-495a-9e53-a37b74d97090	2023-02-06 07:16:00.000000	27.00	838.35	cb1a6b52-4265-4610-a1ae-8deb83039100	PQUR05-CTIZ99N8SR48L
13	359e08ee-fbd7-4a33-8d00-bdeefe66ce39	2023-03-30 16:41:00.000000	0.00	0.00	cb1a6b52-4265-4610-a1ae-8deb83039100	G7Z4XURM25ZLF78UAJWV

Рисунок 3.16 - Таблиця БД Документи

Таблиця «Продукти» (product) складається з полів Продукт (первинний унікальний ключ), Кількість, Назва, Ціна, Тип, Користувач (первинний унікальний ключ), Дата, Розрахунок(рисунок 3.17).

	product_id	description	name	price	type	user_id	created_at	count
1	092c9230-cfec-4803-ac10-8f965e792289	<null>	Mediocre Plastic Chair	72.68	OTHER	5e353874-56a6-49ae-b514-2bd248674f12	2023-05-10 00:44:25.233922	0
2	92b42761-88ed-4c64-b897-41e9f74f2359	<null>	Practical Leather Bag	97.99	OTHER	440886d3-6cb7-4a94-9fad-d561f5bfa231	2023-05-10 00:44:25.242653	0
3	96d1f16f-d5d3-4f2c-9ef6-d8b254ce26f8	<null>	Heavy Duty Marble Shoes	57.17	JEWELRY	91d9a10a-d23f-43c6-b316-798eb599d4d8	2023-05-10 00:44:25.251978	0
4	56e237d5-e50e-4dd1-8b44-4669a2d12447	<null>	Heavy Duty Rubber Shoes	23.47	OTHER	655a093c-f0d9-4826-a2ee-02222c98db9d	2023-05-10 00:44:25.260960	0
5	e26722be-f9f5-4c97-a4f8-aff0638f498e	<null>	Intelligent Paper Keyboard	9.53	HEALTH	9888c89b-2c01-4474-9028-a43ccf815ad7	2023-05-10 00:44:25.269438	0
6	c33acadd-ad2e-4125-b435-c98fca42e1cd	<null>	Sleek Marble Lamp	45.87	OTHER	5823e683-1fda-4ebf-8ca7-66f8e6502af2	2023-05-10 00:44:25.277825	0
7	707af0fb-e9cb-4c5c-9c50-a525ba460717	<null>	Rustic Leather Bottle	66.82	BABY	78a14f9a-9d60-44ed-8e56-157d5499f82c	2023-05-10 00:44:25.285887	0
8	c6a19e5d-587e-4b96-a0cd-b80e0c720b3a	<null>	Sleek Concrete Bottle	47.35	OTHER	2840e703-b2dc-4363-a65e-115646c6d815	2023-05-10 00:44:25.294477	0
9	a6f865ed-25cb-4a82-a9b2-c638a99fe9b8	<null>	Practical Steel Bench	86.71	COMPUTERS	8d2cb2e3-dcdf-4ca1-9b84-e9cbe21d0f7a	2023-05-10 00:44:25.302497	0
10	ba25bf07-1319-47d6-b9d9-9e62072a81eb	<null>	Sleek Cotton Coat	0.96	BABY	2840e703-b2dc-4363-a65e-115646c6d815	2023-05-10 00:44:25.310735	0
11	f35eba51-8a85-46fd-a802-d194a438372c	<null>	Practical Linen Bench	84.93	INDUSTRIAL	237884f3-81fc-46a2-83d5-7df75e2e877f	2023-05-10 00:44:25.320333	0
12	08e5bc33-b38d-46ac-add0-095ea170d534	<null>	Lightweight Bronze Car	99.34	BEAUTY	35c2d596-0209-4673-9386-1e7913529f8a	2023-05-10 00:44:25.329530	0
13	b87d91bd-eebc-45d9-a7aa-12df06e08688	<null>	Synergistic Leather Table	99.73	SPORTS	c156a70d-2db9-480b-8d3b-bc18a09ac8f0	2023-05-10 00:44:25.338350	0

Рисунок 3.17 - Таблиця БД Продукти

Таблиця «Авторизація» (tokens) складається з полів Авторизація (первинний унікальний ключ), Авторизація, Користувач (первинний унікальний ключ)(рисунок 3.18).

	token_id	token	user_id
1	986c6627-fbe7-44d3-bf35-e0b0a666e97d	4b97c1c6-e3c4-499b-8fb3-c30be598f47d	cb1a6b52-4265-4610-a1ae-8deb83039100
2	5dade84c-5ef7-4c05-a274-944e5321c514	d0ac97d8-d60f-409b-96ff-4fa411b349d3	76709fab-c0e7-4b30-b2e5-83a49794ac6b
3	4c48abc5-062e-41e5-89b7-e944d37938a5	d8f472c8-f168-4157-8a3a-d9c7216714b8	9e05bb3a-c189-4ef8-8d6c-98aee75e05bf
4	0c68aea0-8ea0-4f72-b60f-66bb7fe3e58e	eb70ed44-ee92-464a-9c6e-aa7e1bddcc1b	70afa977-a641-47e4-845a-94807bdc4c6e
5	837595cf-5054-4cf6-9f3e-f4598fed85cb	9eea4bbd-54a5-4ddf-9681-58f85401877e	c652f65e-85cc-4b3e-9bf8-cbbf9549b9bb
6	7ec653bc-44f3-463d-a8c7-b7e5059a25ca	e34b3a7a-f4cf-492b-bd6f-b6aef88b6f5a	f1c10eeb-b993-4795-b209-e9fe33985a49

Рисунок 3.18 - Таблиця БД Авторизація

Таблиця «Чек прийому товару» (document_action) складається з полів Документ (первинний унікальний ключ), Накладна (рисунок 3.19).

	document_document_id	actions
1	538bc2b3-24f9-4264-bada-648040cf53ff	RECEPTION
2	865f6d3f-1733-43c3-90ba-0b428035423c	RECEPTION
3	74e83b9b-4454-4ae2-8d46-7595fe31c6e0	RECEPTION
4	d5c7c6e1-ff00-4d29-8ae1-1fa5acf290ae	RECEPTION
5	a5b882cc-0b48-44c4-9082-6330490a1e28	RECEPTION
6	b276bacb-e181-4655-b96f-59bcc1fae23e	RECEPTION
7	d1c2abba-58e0-443b-a543-d46fc9383298	RECEPTION
8	fc39df04-da20-4827-8919-0b2d9b7c6120	RECEPTION
9	aa69f9d1-e80f-4320-ae8e-b49d2db1fa1c	RECEPTION
10	34087a19-aa39-4f57-a19f-aad1a12eef3d	RECEPTION
11	4cfeb59f-ac71-4146-a7cc-1de8e3c5cdeb	RECEPTION
12	6239870f-7fe7-495a-9e53-a37b74d97090	RECEPTION
13	359e08ee-fbd7-4a33-8d00-bdeefe66ce39	RECEPTION

Рисунок 3.19 - Таблиця БД Чек прийому товару

Таблиця «Чек продажу товару» (document_product) складається з полів Документ товару (первинний унікальний ключ), Якість, Фінальна ціна, Документ (первинний унікальний ключ), Продукт (первинний унікальний ключ) (рисунок 3.20).

	document_product_id	quantity	total_cost	document_id	product_id
1	a0bb23a9-77f4-4503-8735-e2439e1f7d5e	<null>	1207.68	538bc2b3-24f9-4264-bada-648040cf53ff	e45498a7-545e-449d-9394-55fa07bdc094
2	91001fcc-18c8-43de-8490-536e1b3b4d36	<null>	780.36	865f6d3f-1733-43c3-90ba-0b428035423c	cf2dc994-b749-4a9e-b63b-4e7341306c02
3	e3e0b5af-0c06-4f93-9fa8-c684db00c18c	<null>	1138.04	74e83b9b-4454-4ae2-8d46-7595fe31c6e0	e99cc983-228f-444c-8f62-d8edd7d04e22
4	5ae4e2d6-b3a1-45c7-917b-0a25d92134df	<null>	63.25	d5c7c6e1-ff00-4d29-8ae1-1fa5acf290ae	62dc7a8a-4995-47b9-91f5-97f42700c155
5	fb163f52-1250-438a-b1b1-31b32b808837	<null>	77.32	a5b882cc-0b48-44c4-9082-6330490a1e28	9bb7c042-1dc0-4d7a-b71f-6d7a2fb0742f
6	7cd07b2f-3640-4255-8dc3-570529e8900f	<null>	431.76	b276bacb-e181-4655-b96f-59bcc1fae23e	1000d9d0-3528-46ea-8062-42273945675d
7	4a2c5ded-d9b7-4c2c-90d1-004ced78296d	<null>	355.20	d1c2abba-58e0-443b-a543-d46fc9383298	e45498a7-545e-449d-9394-55fa07bdc094
8	3bf6e2f7-1b56-433c-afe7-c64ec0b7f867	<null>	24.76	fc39df04-da20-4827-8919-0b2d9b7c6120	4898f59d-7a2b-49f5-97c7-9081d98d1f23
9	9b9fb60e-4cf7-4666-a57d-2248784c5067	<null>	140.40	aa69f9d1-e80f-4320-ae8e-b49d2db1fa1c	c9a492cc-60cb-4bd9-9420-69ea5539864c
10	b5a71cc6-5ec0-461e-a904-1b1e43781e5b	<null>	1895.00	34087a19-aa39-4f57-a19f-aad1a12eef3d	96f7b843-d833-4adc-99e1-dbee126a1ebc
11	69170bdf-20ff-4643-bdc0-e86394db096d	<null>	323.84	4cfeb59f-ac71-4146-a7cc-1de8e3c5cdeb	3df452bb-00a0-43e1-9aa0-f06f0d86ac94
12	9f9e2ad2-e44a-4f29-991c-0edd4ae4a780	<null>	838.35	6239870f-7fe7-495a-9e53-a37b74d97090	7e2c20a9-8e44-4e01-8598-db69e136f80f
13	86196943-b5f1-4d26-b4fa-e394a2f29042	<null>	0.00	359e08ee-fbd7-4a33-8d00-bdeefe66ce39	aa514a5d-7be6-427b-8f6d-cf27b8dcc745

Рисунок 3.20 - Таблиця БД Чек продажу товару

При розробці програми я використав такі фреймворки:

Spring Boot — це фреймворк Java з відкритим вихідним кодом, який забезпечує спрощений спосіб створення автономних додатків на основі Spring робочого рівня. Він побудований на основі Spring Framework і дотримується принципу «угода над конфігурацією», що означає, що він спрямований на мінімізацію шаблонного коду та надання розумних стандартних значень, що дозволяє розробникам швидко налаштовувати та запускати програми.

Spring Security — це потужна структура безпеки з можливістю налаштування для програм Java, зокрема веб-програм. Він забезпечує автентифікацію, авторизацію та інші функції безпеки, щоб захистити вашу програму та її ресурси.

Spring Data JPA є частиною проекту Spring Data і забезпечує структуру для спрощення доступу до бази даних і роботи з Java Persistence API (JPA) у програмах Spring.
Spring Data JPA пропонує кілька функцій, які оптимізують операції з базою даних і зменшують кількість необхідного шаблонного коду.

Також використав біблеотеку Flying Saucer PDF

Це бібліотека Java з відкритим вихідним кодом, яка дозволяє конвертувати документи HTML або файли XHTML у формат PDF. Він побудований на основі бібліотеки iText, яка забезпечує низькорівневі можливості роботи з PDF-файлами.

Flying Saucer PDF використовує W3C Document Object Model (DOM) для аналізу та відтворення вмісту HTML або XHTML. Він підтримує підмножину специфікацій HTML і CSS, дозволяючи створювати PDF-документи, які дуже нагадують оригінальний макет і стиль HTML.

Програму розробляв на мові програмування Java 16.

Це версія мови програмування Java, яка була випущена 16 березня 2021 року. Вона також відома як Java 16 JDK (Java Development Kit). Java 16 представляє кілька нових функцій і вдосконалень мови та віртуальної машини Java (JVM).

Програма складається з класів контролерів які відповідають за зв’язок серверної та веб частинами.

DTO метою якого є інкапсуляція та передача даних у структурованому форматі між різними частинами системи. Це допомагає відокремити подання даних від базових деталей реалізації та забезпечує стандартизований спосіб обміну даними.

Фасад — це структурний патерн проектування, який надає простий інтерфейс до складної системи класів, бібліотеки або фреймворку.

У розробці програмного забезпечення репозиторій — це шаблон проектування, який забезпечує рівень абстракції для доступу до даних і керування ними. Він діє як посередник між джерелом даних (наприклад, базою даних) і рештою програми, дозволяючи відокремити проблеми та узгодити спосіб роботи з даними.

«клас обслуговування» відноситься до класу, який інкапсулює бізнес-логіку та виконує певні операції або послуги в програмі. Класи обслуговування часто є частиною рівня обслуговування, який знаходиться між рівнем представлення (UI) і рівнем доступу до даних (репозиторій або DAO).

«Клас утиліти» або «клас утиліти» — це клас у розробці програмного забезпечення, який містить статичні методи або константи, які забезпечують утиліти загального призначення або допоміжні функції. Ці класи часто складаються зі статичних методів і не вимагають створення екземплярів, оскільки їх основна мета полягає в тому, щоб забезпечити багаторазове використання функціональності в різних частинах програми.

На рисунку 3.21 перелічені класи програми.

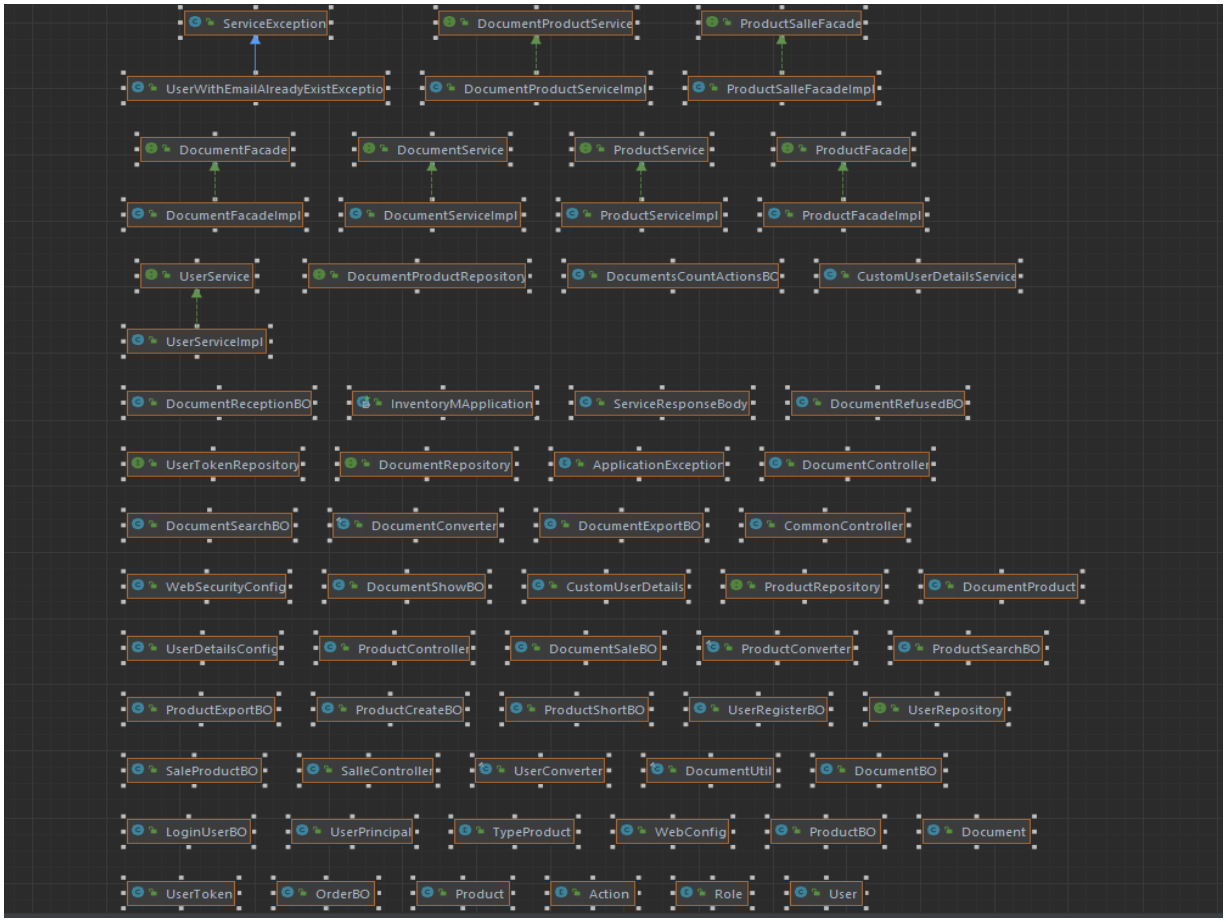


Рисунок 3.21 – Класи програми

ВИСНОВОК

Основною метою дипломної роботи була розробка Web-додаток управління товарообігом для малого бізнесу

Поставлена мета досягнута.

Веб-додато простий в експлуатації, не вимагає спеціальної кваліфікації персоналу, і розрахований на постійне використання.

Користувачами веб-додатку є працівники малого бізнесу. Архітектура реалізації веб-додатку заснована на технологічній базі в області створення додаткі подібного роду. Розробка бази даних велася під СКБД PostgreSQL, що підтримує багатокористувацький режим доступу та побудова клієнт-серверних механізмів доступу до даних.

Методами вирішення поставленого завдання були:

технології розробки програмних продуктів на основі візуального об'єктно-орієнтованого програмування в середовищі програмування IntelliJ IDEA;

проектування інформаційної системи для роботи з базою даних з використанням технології PostgreSQL.

макет сайту розроблявся у Figma та дороблявся у IntelliJ IDEA

Веб-додаток був виконаний відповідно до поставленого завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Брнакевич І.Є., Вагін П.П. Програмування мовою Java: використання фундаментальних класів: Тексти лекцій. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2002. – 75 с.

2. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – К.: Держстандарт України, 1995.– 38 с.

3. Копитко М.Ф., Іванків К.С. Основи програмування мовою Java: Тексти лекцій. – Львів: Видавничий центр ЛНУ ім. Івана Франка, 2002. – 83 с.

4. Лавріщева К.М. Програмна інженерія. Підручник – К.: Видавничий дім «Академперіодика» НАН України, 2008. – 319с.

5. Табунщик Г.В., Каплієнко Т.І., Петрова О.А. Проєктування та моделювання програмного забезпечення сучасних інформаційних систем. Навчальний посібник – Запоріжжя.: Дике поле, 2016. - 250с.

6. Шилдт Герберт. Java 8. Повне керівництво; 9 видавництво. : Пер. з англ. - М.: ТОВ «І.Д. Віл’ямс», 2015. - 1376с.

7. Штаєр Л. О. Технології розробки програмного забезпечення : конспект лекцій / Л. О. Штаєр. - Івано-Франківськ: ІФНТУНГ, 2017. – 139с.

ЛІСТИНГ ПРОГРАМИ

```
package com.example.inventorym.config;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.User;

import java.util.Collection;
import java.util.UUID;

public class CustomUserDetails extends User {

    private UUID id;

    public CustomUserDetails(com.example.inventorym.entity.User user,
Collection<? extends GrantedAuthority> authorities) {
        super(user.getEmail(), user.getPassword(), true, true, true, true,
authorities);
        this.id = user.getId();
    }

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
```

```
            this.id = id;
        }
    }


    package com.example.inventorym.config;

    import com.example.inventorym.entity.User;
    import com.example.inventorym.entity.UserToken;
    import com.example.inventorym.entity.enums.Role;
    import com.example.inventorym.repository.UserRepository;
    import com.example.inventorym.repository.UserTokenRepository;
    import org.springframework.security.core.GrantedAuthority;
    import org.springframework.security.core.authority.AuthorityUtils;
    import org.springframework.security.core.userdetails.UserDetails;
    import org.springframework.security.core.userdetails.UserDetailsService;
    import
org.springframework.security.core.userdetails.UsernameNotFoundException;
    import org.springframework.stereotype.Service;
    import org.springframework.transaction.annotation.Transactional;

    import java.util.Collection;
    import java.util.UUID;

    @Service
    @Transactional
```

```
public class CustomUserDetailsService implements UserDetailsService {

    private final UserRepository userRepository;
    private final UserTokenRepository userTokenRepository;

    private UUID id;

    // other fields, getters and setters

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
    }

    public CustomUserDetailsService(UserRepository userRepository,
UserTokenRepository userTokenRepository) {
        this.userRepository = userRepository;
        this.userTokenRepository = userTokenRepository;
    }



/*      @Override
```

```
        public UserDetails loadUserByUsername(String userName) throws
UsernameNotFoundException {
            User user = userRepository.findByEmail(userName)
                    .orElseThrow(() -> new UsernameNotFoundException("Email "
+ userName + " not found"));



            UserToken userToken =
userTokenRepository.findByUserId(user.getId());
            if (userToken == null) {
                userToken = new UserToken();
            }
            userToken.setToken(generateToken());
            userToken.setUser(new User(user.getId()));
            userTokenRepository.save(userToken);
            return new CustomUserDetails(user, getAuthorities(user));
        }*/

        @Override
        public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
            User user = userRepository.findByEmail(username)
                    .orElseThrow(() -> new UsernameNotFoundException("Email "
+ username + " not found"));
```

```
            UserToken userToken =
userTokenRepository.findByUserId(user.getId());
            if (userToken == null) {
                userToken = new UserToken();
            }
            userToken.setToken(generateToken());
            userToken.setUser(new User(user.getId()));
            userTokenRepository.save(userToken);

            return new UserPrincipal(user);
        }
        private static Collection<? extends GrantedAuthority> getAuthorities(User
user) {
            String[] userRoles =
user.getRoles().stream().map(Role::name).toArray(String[]::new);
            return AuthorityUtils.createAuthorityList(userRoles);
        }

        private String generateToken() {
            UUID uuid = UUID.randomUUID();
            String token = uuid.toString();
            UserToken userToken = userTokenRepository.findByToken(token);
            if (userToken != null) {
                return generateToken();
            }
            return token;
```

```
    }
}
```

```
package com.example.inventorym.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
public class UserDetailsConfig {

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

```
package com.example.inventorym.config;

import com.example.inventorym.entity.User;
import com.example.inventorym.entity.enums.Role;
import org.springframework.security.core.GrantedAuthority;
```

```
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.Collections;

public class UserPrincipal implements UserDetails {

    private User user;

    public UserPrincipal(User user) {
        this.user = user;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        String[] userRoles =
user.getRoles().stream().map(Role::name).toArray(String[]::new);
        return AuthorityUtils.createAuthorityList(userRoles);

    }

    @Override
    public String getPassword() {
        return user.getPassword();
```

```
}

@Override
public String getUsername() {
    return user.getEmail();
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return true;
}
```

```
    }


    package com.example.inventorym.config;

    import org.springframework.context.MessageSource;
    import org.springframework.context.annotation.Configuration;
    import org.springframework.validation.Validator;
    import
org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
    import
org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
    import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

    @Configuration
    public class WebConfig implements WebMvcConfigurer {

        private final MessageSource messageSource;

        public WebConfig(MessageSource messageSource) {
            this.messageSource = messageSource;
        }

        @Override
        public void addViewControllers(ViewControllerRegistry registry) {
            registry.addViewController("/").setViewName("index");
```

```
                registry.addViewController("/login").setViewName("login");
                registry.addViewController("/register").setViewName("register");
                registry.addViewController("/products").setViewName("products");

registry.addViewController("/products/addProduct").setViewName("products");
                registry.addViewController("/sale").setViewName("salle");
                registry.addViewController("/documents").setViewName("docs");

registry.addViewController("documents/createRefused").setViewName("documentRet
urn");
        }

        @Override
        public Validator getValidator() {
            LocalValidatorFactoryBean factory = new LocalValidatorFactoryBean();
            factory.setValidationMessageSource(messageSource);
            return factory;
        }

    }


    package com.example.inventorym.config;

    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.context.annotation.Bean;
```

```
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.web.authentication.AuthenticationSuccessHandler;
import
org.springframework.security.web.authentication.rememberme.JdbcTokenRepositoryImpl;
import
org.springframework.security.web.authentication.rememberme.PersistentTokenRepository;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;
import java.io.IOException;
import java.nio.file.attribute.UserPrincipal;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true, proxyTargetClass =
true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    private final PasswordEncoder passwordEncoder;
    private final UserDetailsService customUserDetailsService;
    private final DataSource dataSource;

    public WebSecurityConfig(PasswordEncoder passwordEncoder,
UserDetailsService customUserDetailsService, DataSource dataSource) {
        this.passwordEncoder = passwordEncoder;
        this.customUserDetailsService = customUserDetailsService;
        this.dataSource = dataSource;
    }

    @Autowired
```

```
        public void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {
            auth
                    .userDetailsService(customUserDetailsService)
                    .passwordEncoder(passwordEncoder);
        }

        @Override
        protected void configure(HttpSecurity http) throws Exception {
            http
    //                      .csrf().disable()
                    .headers()
                    .frameOptions().sameOrigin()
                    .and()
                    .authorizeRequests()
                    .antMatchers("/resources/**", "/webjars/**",
"/assets/**").permitAll()
                    .antMatchers("/register").permitAll()
                    .antMatchers("/products").authenticated()
                    .antMatchers("/products/**").authenticated()
                    .antMatchers("/products/all").authenticated()
                    .antMatchers("/admin/**").hasRole("ADMIN")
                    .antMatchers("/api/reports/sale").permitAll()
                    .anyRequest().authenticated()
                    .and()
                    .formLogin()
```

```
                .loginPage("/login")
                .defaultSuccessUrl("/home")
                .failureUrl("/login?error")
                .permitAll()
                .and()
                .logout()
                .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                .logoutSuccessUrl("/login?logout")
                .deleteCookies("my-remember-me-cookie")
                .permitAll()
                .and()
                .rememberMe()
                //.key("my-secure-key")
                .rememberMeCookieName("my-remember-me-cookie")
                .tokenRepository(persistentTokenRepository())
                .tokenValiditySeconds(24 * 60 * 60)
                .and()
                .exceptionHandling();
    }

    PersistentTokenRepository persistentTokenRepository() {
        JdbcTokenRepositoryImpl tokenRepositoryImpl = new
JdbcTokenRepositoryImpl();
        tokenRepositoryImpl.setDataSource(dataSource);
        return tokenRepositoryImpl;
    }
```

```
        @Bean
        public AuthenticationSuccessHandler successHandler() {
            return new AuthenticationSuccessHandler() {
                @Override
                public void onAuthenticationSuccess(HttpServletRequest request,
HttpServletResponse response,
                                                    Authentication authentication)
throws IOException {
                    UserPrincipal userPrincipal = (UserPrincipal)
authentication.getPrincipal();
                    request.getSession().setAttribute("userName",
userPrincipal.getName());
                    response.sendRedirect("/home");
                }
            };
        }

    }


    package com.example.inventorym.controller;


    import com.example.inventorym.dto.DocumentsCountActionsBO;
    import com.example.inventorym.dto.UserRegisterBO;
    import com.example.inventorym.entity.enums.Action;
```

```
import com.example.inventorym.facade.DocumentFacade;
import com.example.inventorym.services.UserService;
import
com.example.inventorym.util.exception.UserWithEmailAlreadyExistException;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

@Controller
public class CommonController {
    private final UserService userService;
    private final DocumentFacade documentFacade;


    public CommonController(UserService userService, DocumentFacade
documentFacade) {
        this.userService = userService;
        this.documentFacade = documentFacade;
    }
```

```
    @GetMapping("/")
    public String index(Model model, Authentication authentication) {
        var reportSale = documentFacade.findReport(Action.SALE,
authentication.getName());
        var reportReturn = documentFacade.findReport(Action.RETURN,
authentication.getName());

        model.addAttribute("reportSale", reportSale);
        model.addAttribute("reportReturn", reportReturn);

        return "index";
    }

    @GetMapping("/home")
    public String home(Model model, Authentication authentication) {
        var reportSale = documentFacade.findReport(Action.SALE,
authentication.getName());
        var reportReturn = documentFacade.findReport(Action.RETURN,
authentication.getName());
        model.addAttribute("reportSale", reportSale);
        model.addAttribute("reportReturn", reportReturn);

        return "index";
    }
```

```
@GetMapping("/register")
public String showRegister() {
    return "register";
}


@PostMapping("/register")
public String registerUser(@ModelAttribute("user") UserRegisterBO
userDto, BindingResult result) throws UserWithEmailAlreadyExistException {
    if (result.hasErrors()) {
        return "register";
    }
    userService.createNewUser(userDto);
    return "redirect:/login";
}

@GetMapping("/redirProducts")
public String redirectProducts() {
    return "redirect:/products";
}


@GetMapping("/redirDocuments")
public String redirectDocument() {
    return "redirect:/documents";
}
```

```
}


package com.example.inventorym.controller;

import com.example.inventorym.dto.*;
import com.example.inventorym.facade.DocumentFacade;
import com.example.inventorym.facade.ProductFacade;
import com.example.inventorym.facade.ProductSalleFacade;
import org.springframework.data.domain.Sort;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.thymeleaf.TemplateEngine;
import org.thymeleaf.context.Context;
import org.xhtmlrenderer.pdf.ITextRenderer;

import java.io.ByteArrayOutputStream;
import java.util.UUID;

@Controller
```

```
@RequestMapping("/documents")
public class DocumentController {

    private final DocumentFacade documentFacade;
    private final ProductFacade productFacade;
    private final TemplateEngine templateEngine;

    private final ProductSalleFacade productSalleFacade;

    public DocumentController(DocumentFacade documentFacade,
ProductFacade productFacade, TemplateEngine templateEngine, ProductSalleFacade
productSalleFacade) {
        this.documentFacade = documentFacade;
        this.productFacade = productFacade;
        this.templateEngine = templateEngine;
        this.productSalleFacade = productSalleFacade;
    }

    @GetMapping("")
    public String getDocs(@RequestParam(defaultValue = "createdAt") String
sort,
                          @RequestParam(defaultValue = "desc") String dir,
                          Model model, Authentication authentication) {

        var sortOrder = dir.equals("asc") ? Sort.by(sort).ascending() :
Sort.by(sort).descending();
```

```
            var documents =
documentFacade.findAllByUser(authentication.getName(), sortOrder);
            var products = productFacade.findAllByUser(authentication.getName(),
Sort.by("name").descending());

            model.addAttribute("docs", documents);
            model.addAttribute("productsForModal", products);
            model.addAttribute("documentRefusedBO", new
DocumentRefusedBO());
            model.addAttribute("documentReceptionBO", new
DocumentReceptionBO());

            return "docs";
        }

        @GetMapping("/refusend/{id}")
        public String refusedProduct(@PathVariable("id") UUID id, Model model) {
            var products = documentFacade.findProductByDocId(id);
            model.addAttribute("productsForModal", products);
            model.addAttribute("documentRefusedBO", new
DocumentRefusedBO());
            return "documentReturn";
        }

        @PostMapping("/createRefused")
```

```
    public String refusedDocument(DocumentRefusedBO documentRefusedBO,
Authentication authentication) {
        documentFacade.createRefused(documentRefusedBO,
authentication.getName());
        return "redirect:/documents";
    }

    @PostMapping("/createReception")
    public String receptionDocument(DocumentReceptionBO
documentReceptionBO, Authentication authentication) {
        documentFacade.createReception(documentReceptionBO,
authentication.getName());
        return "redirect:/documents";
    }

    @GetMapping("/generate/{id}")
    @ResponseBody
    public ResponseEntity<byte[]> exportDoc(@PathVariable("id") UUID id)
throws Exception {
        var doc = documentFacade.findDocByIdForExport(id);

        Context context = new Context();

        context.setVariable("document", doc);
        context.setVariable("products", doc.getProductExportBOS());
```

```
            // Render the Thymeleaf template into an HTML string
            String htmlContent = templateEngine.process("ExportDoc", context);
            // Generate PDF from HTML content
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            ITextRenderer renderer = new ITextRenderer();
            renderer.setDocumentFromString(htmlContent);
            renderer.getFontResolver().addFont("/static/Roboto-Regular.ttf",
com.lowagie.text.pdf.BaseFont.IDENTITY_H,
com.lowagie.text.pdf.BaseFont.EMBEDDED);
            renderer.layout();
            renderer.createPDF(outputStream, false);
            renderer.finishPDF();

            HttpHeaders headers = new HttpHeaders();
            headers.setContentType(MediaType.APPLICATION_PDF);
            headers.setContentDispositionFormData("attachment", "document.pdf");

            return new ResponseEntity<>(outputStream.toByteArray(), headers,
HttpStatus.OK);

        }

    }
```

```
package com.example.inventorym.controller;

import com.example.inventorym.dto.ProductBO;
import com.example.inventorym.dto.ProductCreateBO;
import com.example.inventorym.dto.ProductSearchBO;
import com.example.inventorym.entity.Product;
import com.example.inventorym.entity.User;
import com.example.inventorym.entity.enums.TypeProduct;
import com.example.inventorym.facade.ProductFacade;
import com.github.javafaker.Faker;
import org.springframework.data.domain.Sort;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.UUID;

@Controller()
@RequestMapping("/products")
public class ProductController {

    private final ProductFacade productFacade;

    public ProductController(ProductFacade productFacade) {
        this.productFacade = productFacade;
```

```
    }

    @GetMapping("")
    public String getUsers(@RequestParam(defaultValue = "createdAt") String
sort,
                           @RequestParam(defaultValue = "desc") String dir,
                           Model model, Authentication authentication) {

        var sortOrder = dir.equals("asc") ? Sort.by(sort).ascending() :
Sort.by(sort).descending();
        var products = productFacade.findAllByUser(authentication.getName(),
sortOrder);

        model.addAttribute("newProduct", new ProductCreateBO());
        model.addAttribute("searchProduct", new ProductSearchBO());
        model.addAttribute("editProduct", new ProductBO());
        model.addAttribute("productsList", products);

        return "products";
    }

    @PostMapping("/create")
    public String addProduct(ProductCreateBO productCreateBO,
Authentication authentication) {
        productCreateBO.setEmail(authentication.getName());
        productFacade.createProduct(productCreateBO);
```

```
        return "redirect:/products";
    }

    @GetMapping("/products/edit/{id}")
    public String edit(@PathVariable("id") UUID id,Model model) {
        model.addAttribute("editProduct", productFacade.findById(id));
        return "productEdit";
    }

    @PostMapping("/edit")
    public String editSave(ProductBO productBO) {
        productFacade.updateProduct(productBO);
        return "redirect:/products";
    }

    @PostMapping("/search")
    public String searchProduct(ProductSearchBO productBO,
                                @RequestParam(defaultValue = "createdAt")
String sort,
                                @RequestParam(defaultValue = "desc") String
dir,
                                Model model, Authentication authentication) {
        var sortOrder = dir.equals("asc") ? Sort.by(sort).ascending() :
Sort.by(sort).descending();
        var products = productFacade.searchProduct(productBO,
authentication.getName(), sortOrder);
```

```
            model.addAttribute("newProduct", new ProductCreateBO());
            model.addAttribute("editProduct", new ProductBO());
            model.addAttribute("searchProduct", new ProductSearchBO());
            model.addAttribute("productsList", products);
            return "products";
        }

}


package com.example.inventorym.controller;

import com.example.inventorym.dto.SaleProductBO;
import com.example.inventorym.facade.DocumentFacade;
import com.example.inventorym.facade.ProductFacade;
import com.example.inventorym.facade.ProductSalleFacade;
import org.springframework.data.domain.Sort;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.thymeleaf.TemplateEngine;

import java.time.LocalDateTime;
import java.util.UUID;
```

```
import java.util.stream.Collectors;

import static com.example.inventorym.util.DocumentUtil.getDateTime;

@Controller
@RequestMapping("/sale")
public class SalleController {

    private final DocumentFacade documentFacade;
    private final ProductFacade productFacade;
    private final ProductSalleFacade productSalleFacade;
    private final TemplateEngine templateEngine;

    public SalleController(DocumentFacade documentFacade, ProductFacade
productFacade, ProductSalleFacade productSalleFacade, TemplateEngine
templateEngine) {
        this.documentFacade = documentFacade;
        this.productFacade = productFacade;
        this.productSalleFacade = productSalleFacade;
        this.templateEngine = templateEngine;
    }

    @GetMapping("")
    public String makeSale(@RequestParam(defaultValue = "createdAt") String
sort,
                           @RequestParam(defaultValue = "desc") String dir,
```

```
                                    Model model, Authentication authentication) {

            var numberDoc = documentFacade.findDocNumber();
            var sortOrder = dir.equals("asc") ? Sort.by(sort).ascending() :
Sort.by(sort).descending();
            var products = productFacade.findAllByUser(authentication.getName(),
sortOrder)
                    .stream().filter(productBO -> productBO.getCount() !=
0).collect(Collectors.toList());

            model.addAttribute("numberDoc", numberDoc);
            model.addAttribute("saleProductBO", new
SaleProductBO(numberDoc));
            model.addAttribute("orderDate", getDateTime(LocalDateTime.now()));
            model.addAttribute("productsList", products);
            model.addAttribute("order", productSalleFacade.findProductInOrder());
            return "salle";
        }

        @GetMapping("/addToOrder/{id}")
        public String addToOrder(@PathVariable("id") UUID id) {
            productSalleFacade.addToOrder(id);
            return "redirect:/sale";
        }

        @GetMapping("/remove/{id}")
```

```
    public String removeProducts(@PathVariable("id") UUID id) {
        productSalleFacade.removeProducts(id);
        return "redirect:/sale";
    }

    @GetMapping("/editCount/{id}/{action}")
    public String addOneProd(@PathVariable("id") UUID id,
                             @PathVariable("action") String action) {
        productSalleFacade.editCount(action, id);
        return "redirect:/sale";
    }

    @PostMapping("/make")
    public String createSale(SaleProductBO saleProductBO, Model model,
Authentication authentication) {
        productSalleFacade.createSale(saleProductBO,
authentication.getName());
        return "redirect:/documents";
    }

}


package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.Action;
```

```
import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.List;
import java.util.Set;
import java.util.UUID;

public class DocumentBO {

    private UUID documentId;
    private LocalDateTime createdAt;
    private BigDecimal totalPrice;
    private Set<Action> typeDocument;
    private BigDecimal totalCount;
    private String docNumber;
    private boolean isSale;

    public LocalDateTime getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(LocalDateTime createdAt) {
        this.createdAt = createdAt;
    }

    public BigDecimal getTotalPrice() {
```

```
        return totalPrice;
    }

    public void setTotalPrice(BigDecimal totalPrice) {
        this.totalPrice = totalPrice;
    }

    public Set<Action> getTypeDocument() {
        return typeDocument;
    }

    public void setTypeDocument(Set<Action> typeDocument) {
        this.typeDocument = typeDocument;
    }

    public BigDecimal getTotalCount() {
        return totalCount;
    }

    public void setTotalCount(BigDecimal totalCount) {
        this.totalCount = totalCount;
    }

    public UUID getDocumentId() {
        return documentId;
    }
```

```
    public void setDocumentId(UUID documentId) {
        this.documentId = documentId;
    }

    public String getDocNumber() {
        return docNumber;
    }

    public void setDocNumber(String docNumber) {
        this.docNumber = docNumber;
    }

    public boolean getSale() {
        return isSale;
    }

    public void setSale(boolean sale) {
        isSale = sale;
    }
}


package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.Action;
```

```
import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.List;
import java.util.Set;

public class DocumentExportBO {

    private LocalDateTime createdAt;
    private BigDecimal totalPrice;
    private BigDecimal totalCount;
    private String typeDocument;
    private List<ProductExportBO> productExportBOS;
    private String docNumber;

    public LocalDateTime getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(LocalDateTime createdAt) {
        this.createdAt = createdAt;
    }

    public BigDecimal getTotalPrice() {
        return totalPrice;
    }
```

```
    public void setTotalPrice(BigDecimal totalPrice) {
        this.totalPrice = totalPrice;
    }

    public String getTypeDocument() {
        return typeDocument;
    }

    public void setTypeDocument(String typeDocument) {
        this.typeDocument = typeDocument;
    }

    public List<ProductExportBO> getProductExportBOS() {
        return productExportBOS;
    }

    public void setProductExportBOS(List<ProductExportBO>
productExportBOS) {
        this.productExportBOS = productExportBOS;
    }

    public String getDocNumber() {
        return docNumber;
    }
```

```
    public void setDocNumber(String docNumber) {
        this.docNumber = docNumber;
    }

    public BigDecimal getTotalCount() {
        return totalCount;
    }

    public void setTotalCount(BigDecimal totalCount) {
        this.totalCount = totalCount;
    }
}
```

```
package com.example.inventorym.dto;

public class DocumentReceptionBO {

    private String nameProduct;
    private int count;

    public DocumentReceptionBO(String nameProduct, int count) {
        this.nameProduct = nameProduct;
        this.count = count;
    }
```

```
    public DocumentReceptionBO() {
    }

    public String getNameProduct() {
        return nameProduct;
    }

    public void setNameProduct(String nameProduct) {
        this.nameProduct = nameProduct;
    }

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }
}


package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.Action;

import java.math.BigDecimal;
```

```
import java.time.LocalDateTime;
import java.util.List;
import java.util.Set;
import java.util.UUID;

public class DocumentRefusedBO {

    private String nameProduct;
    private String cause;

    public String getNameProduct() {
        return nameProduct;
    }

    public void setNameProduct(String nameProduct) {
        this.nameProduct = nameProduct;
    }

    public String getCause() {
        return cause;
    }

    public void setCause(String cause) {
        this.cause = cause;
    }
}
```

```
package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.Action;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.Set;
import java.util.UUID;

public class DocumentSaleBO {

    private LocalDateTime createdAt;
    private BigDecimal totalPrice;
    private Set<Action> typeDocument;
    private BigDecimal totalCount;
    private String docNumber;

    public LocalDateTime getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(LocalDateTime createdAt) {
        this.createdAt = createdAt;
    }
```

```
public BigDecimal getTotalPrice() {
    return totalPrice;
}

public void setTotalPrice(BigDecimal totalPrice) {
    this.totalPrice = totalPrice;
}

public Set<Action> getTypeDocument() {
    return typeDocument;
}

public void setTypeDocument(Set<Action> typeDocument) {
    this.typeDocument = typeDocument;
}

public BigDecimal getTotalCount() {
    return totalCount;
}

public void setTotalCount(BigDecimal totalCount) {
    this.totalCount = totalCount;
}

public String getDocNumber() {
```

```
        return docNumber;
    }

    public void setDocNumber(String docNumber) {
        this.docNumber = docNumber;
    }
}
```

```
package com.example.inventorym.dto;

import java.math.BigDecimal;

public class DocumentsCountActionsBO {
    private BigDecimal totalSum;
    private int countMonth;

    public DocumentsCountActionsBO(BigDecimal totalSum, int countMonth)
{
        this.totalSum = totalSum;
        this.countMonth = countMonth;
    }

    public BigDecimal getTotalSum() {
        return totalSum;
    }
```

```
    public void setTotalSum(BigDecimal totalSum) {
        this.totalSum = totalSum;
    }

    public int getCountMonth() {
        return countMonth;
    }

    public void setCountMonth(int countMonth) {
        this.countMonth = countMonth;
    }

    @Override
    public String toString() {
        return "DocumentsCountActionsBO{" +
                "totalSum=" + totalSum +
                ", countMonth=" + countMonth +
                '}';
    }
}
```

```
package com.example.inventorym.dto;

public class DocumentSearchBO {
```

```
    private String name;

    public String getName() {
        return name;
    }

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

```
package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.Action;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.Set;
import java.util.UUID;

public class DocumentShowBO {

    private UUID documentId;
    private LocalDateTime createdAt;
    private BigDecimal totalPrice;
```

```
private String typeDocument;
private BigDecimal totalCount;
private String docNumber;
private boolean isSale;

public LocalDateTime getCreatedAt() {
    return createdAt;
}

public void setCreatedAt(LocalDateTime createdAt) {
    this.createdAt = createdAt;
}

public BigDecimal getTotalPrice() {
    return totalPrice;
}

public void setTotalPrice(BigDecimal totalPrice) {
    this.totalPrice = totalPrice;
}

public String getTypeDocument() {
    return typeDocument;
}

public void setTypeDocument(String typeDocument) {
```

```
        this.typeDocument = typeDocument;
    }

    public boolean isSale() {
        return isSale;
    }

    public BigDecimal getTotalCount() {
        return totalCount;
    }

    public void setTotalCount(BigDecimal totalCount) {
        this.totalCount = totalCount;
    }

    public UUID getDocumentId() {
        return documentId;
    }

    public void setDocumentId(UUID documentId) {
        this.documentId = documentId;
    }

    public String getDocNumber() {
        return docNumber;
    }
```

```
    public void setDocNumber(String docNumber) {
        this.docNumber = docNumber;
    }

    public boolean getSale() {
        return isSale;
    }

    public void setSale(boolean sale) {
        isSale = sale;
    }
}
```

```
package com.example.inventorym.dto;

import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;

public class LoginUserBO {

    @NotBlank(message = "Email is required")
    @Email(message = "Invalid email format")
    private String email;

    @NotBlank(message = "Password is required")
```

```
    private String password;

    public String getEmail() {
        return email;
    }

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

    public String getPassword() {
        return password;
    }

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

```
package com.example.inventorym.dto;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;
```

```
public class OrderBO {

    private String docNumber;
    private int count;
    private List<ProductShortBO> productShortBOS = new ArrayList<>();
    private BigDecimal totalPrice;

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }

    public List<ProductShortBO> getProductShortBOS() {
        return productShortBOS;
    }

    public void setProductShortBOS(List<ProductShortBO> productShortBOS)
{
        this.productShortBOS = productShortBOS;
    }

    public BigDecimal getTotalPrice() {
        return totalPrice;
```

```
    }

    public void setTotalPrice(BigDecimal totalPrice) {
        this.totalPrice = totalPrice;
    }

    public String getDocNumber() {
        return docNumber;
    }

    public void setDocNumber(String docNumber) {
        this.docNumber = docNumber;
    }
}
```

```
package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.TypeProduct;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.UUID;

public class ProductBO {
```

```
private UUID productId;
private String name;
private String description;
private BigDecimal price;
private LocalDateTime createdAt;
private TypeProduct type;
private int count;

public String getName() {
    return name;
}

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

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public BigDecimal getPrice() {
    return price;
```

```
}

public void setPrice(BigDecimal price) {
    this.price = price;
}

public TypeProduct getType() {
    return type;
}

public void setType(TypeProduct type) {
    this.type = type;
}

public UUID getProductId() {
    return productId;
}

public void setProductId(UUID productId) {
    this.productId = productId;
}

public LocalDateTime getCreatedAt() {
    return createdAt;
}
```

```
    public void setCreatedAt(LocalDateTime createdAt) {
        this.createdAt = createdAt;
    }

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }
}
```

```
package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.TypeProduct;

import java.math.BigDecimal;
import java.util.UUID;

public class ProductCreateBO {

    private String name;
    private String description;
    private BigDecimal price;
```

```
private String type;
private String email;

public String getName() {
    return name;
}

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

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public BigDecimal getPrice() {
    return price;
}

public void setPrice(BigDecimal price) {
    this.price = price;
}
```

```
public String getType() {
    return type;
}

public void setType(String type) {
    this.type = type;
}

public String getEmail() {
    return email;
}

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

@Override
public String toString() {
    return "ProductCreateBO{" +
            "name='" + name + '\'' +
            ", description='" + description + '\'' +
            ", price=" + price +
            ", type=" + type +
            ", email='" + email + '\'' +
            '}';
```

```
    }
}
```

```
package com.example.inventorym.dto;

import java.math.BigDecimal;

public class ProductExportBO {

    private String name;
    private BigDecimal price;

    public ProductExportBO() {
    }

    public ProductExportBO(String name, BigDecimal price) {
        this.name = name;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
```

```
        this.name = name;
    }

    public BigDecimal getPrice() {
        return price;
    }

    public void setPrice(BigDecimal price) {
        this.price = price;
    }
}
```

```
package com.example.inventorym.dto;

public class ProductSearchBO {
    private String name;

    public String getName() {
        return name;
    }

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

```
package com.example.inventorym.dto;

import com.example.inventorym.entity.enums.TypeProduct;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.UUID;

public class ProductShortBO {

    private UUID productId;
    private String name;
    private BigDecimal price;
    private BigDecimal totalPrice;
    private int count;

    public ProductShortBO(UUID productId, String name, BigDecimal price, int
count) {
        this.productId = productId;
        this.name = name;
        this.price = price;
        this.totalPrice = totalPrice;
        this.count = count;
    }
```

```
public ProductShortBO() {
}

public UUID getProductId() {
    return productId;
}

public void setProductId(UUID productId) {
    this.productId = productId;
}

public String getName() {
    return name;
}

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

public BigDecimal getPrice() {
    return price;
}

public void setPrice(BigDecimal price) {
    this.price = price;
```

```
}

public int getCount() {
    return count;
}

public void setCount(int count) {
    this.count = count;
}

public BigDecimal getTotalPrice() {
    return totalPrice;
}

public void setTotalPrice(BigDecimal totalPrice) {
    this.totalPrice = totalPrice;
}

@Override
public String toString() {
    return "ProductShortBO{" +
            "productId=" + productId +
            ", name='" + name + '\'' +
            ", price=" + price +
            ", totalPrice=" + totalPrice +
            ", count=" + count +
```

```
                '}';
    }
}
```

```
package com.example.inventorym.dto;

import java.time.LocalDateTime;

public class SaleProductBO {
    private String numberDoc;

    public SaleProductBO(String numberDoc) {
        this.numberDoc = numberDoc;
    }

    public String getNumberDoc() {
        return numberDoc;
    }

    public void setNumberDoc(String numberDoc) {
        this.numberDoc = numberDoc;
    }
}
```

```
package com.example.inventorym.dto;

import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;

public class UserRegisterBO {

    @NotBlank(message = "Name is required")
    private String firstName;

    @NotBlank(message = "Name is required")
    private String lastName;

    @NotBlank(message = "Email is required")
    @Email(message = "Invalid email format")
    private String email;

    @NotBlank(message = "Password is required")
    private String password;

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
```

```
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

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

    public String getPassword() {
        return password;
    }

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

```
package com.example.inventorym.entity;

import com.example.inventorym.entity.enums.Action;
import com.example.inventorym.entity.enums.Role;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.Set;
import java.util.UUID;

@Entity
@Table(name = "document")
public class Document {

    @Id
    @GeneratedValue(generator = "uuid2")
    @GenericGenerator(name = "uuid2", strategy = "uuid2")
    @Column(name = "document_id")
    private UUID id;

    @Column(name = "created_at")
    private LocalDateTime createdAt;
```

```
    @Column(name = "total_price")
    private BigDecimal totalPrice;

    @ElementCollection(fetch = FetchType.EAGER)
    @Enumerated(EnumType.STRING)
    private Set<Action> actions;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", referencedColumnName = "user_id",
updatable = false, nullable = false)
    private User user;

    @Column(name = "total_count")
    private BigDecimal totalCount;

    @Column(name = "number", unique = true)
    private String documentNumber;

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
    }
```

```
public Set<Action> getActions() {
    return actions;
}

public void setActions(Set<Action> actions) {
    this.actions = actions;
}

public LocalDateTime getCreatedAt() {
    return createdAt;
}

public void setCreatedAt(LocalDateTime createdAt) {
    this.createdAt = createdAt;
}

public BigDecimal getTotalPrice() {
    return totalPrice;
}

public void setTotalPrice(BigDecimal totalPrice) {
    this.totalPrice = totalPrice;
}

public BigDecimal getTotalCount() {
```

```
        return totalCount;
    }

    public void setTotalCount(BigDecimal totalCount) {
        this.totalCount = totalCount;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String getDocumentNumber() {
        return documentNumber;
    }

    public void setDocumentNumber(String documentNumber) {
        this.documentNumber = documentNumber;
    }

    @Override
    public String toString() {
        return "Document{" +
```

```
                "id=" + id +
                ", createdAt=" + createdAt +
                ", totalPrice=" + totalPrice +
                ", actions=" + actions +
                ", user=" + user +
                ", totalCount=" + totalCount +
                ", documentNumber='" + documentNumber + '\'' +
                '}';
    }
}


package com.example.inventorym.entity;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.math.BigDecimal;
import java.util.UUID;

@Entity
@Table(name = "document_product")
public class DocumentProduct {

    @Id
    @GeneratedValue(generator = "uuid2")
```

```
    @GenericGenerator(name = "uuid2", strategy = "uuid2")
    @Column(name = "document_product_id")
    private UUID id;

    @Column(name = "quantity")
    private Integer quantity;

    @Column(name = "total_cost")
    private BigDecimal total_cost;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "product_id", referencedColumnName =
"product_id", updatable = false, nullable = false)
    private Product product;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "document_id", referencedColumnName =
"document_id", updatable = false, nullable = false)
    private Document document;

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
```

```
}

public Integer getQuantity() {
    return quantity;
}

public void setQuantity(Integer quantity) {
    this.quantity = quantity;
}

public BigDecimal getTotal_cost() {
    return total_cost;
}

public void setTotal_cost(BigDecimal total_cost) {
    this.total_cost = total_cost;
}

public Product getProduct() {
    return product;
}

public void setProduct(Product product) {
    this.product = product;
}
```

```
    public Document getDocument() {
        return document;
    }

    public void setDocument(Document document) {
        this.document = document;
    }
}
```

```
package com.example.inventorym.entity;


import com.example.inventorym.entity.enums.TypeProduct;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.UUID;

@Entity
public class Product {

    @Id
    @GeneratedValue(generator = "uuid2")
```

```
    @GenericGenerator(name = "uuid2", strategy = "uuid2")
    @Column(name = "product_id")
    private UUID id;
    private String name;
    private String description;
    private int count;
    @Enumerated(EnumType.STRING)
    private TypeProduct type;

    @Column(name = "created_at")
    private LocalDateTime createdAt;
    @Column(name = "price")
    private BigDecimal price;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", referencedColumnName = "user_id",
updatable = false, nullable = false)
    private User user;

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
    }
```

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}
public String getDescription() {
    return description;
}
public void setDescription(String description) {
    this.description = description;
}

public TypeProduct getType() {
    return type;
}

public void setType(TypeProduct type) {
    this.type = type;
}
public BigDecimal getPrice() {
    return price;
}
public void setPrice(BigDecimal price) {
```

```
        this.price = price;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public LocalDateTime getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(LocalDateTime createdAt) {
        this.createdAt = createdAt;
    }

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }
```

```
}
```

```
package com.example.inventorym.entity;

import com.example.inventorym.entity.enums.Role;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.UUID;

@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(generator = "uuid2")
    @GenericGenerator(name = "uuid2", strategy = "uuid2")
    @Column(name = "user_id")
    private UUID id;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;
```

```
    private String email;
    private String password;

    @ElementCollection(fetch = FetchType.EAGER)
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;

    @OneToMany(mappedBy = "user", cascade = CascadeType.REMOVE)
    private List<Product> saleProducts = new ArrayList<>();

    public User(UUID id) {
        this.id = id;
    }

    public User(UUID id, String firstName, String lastName, String email,
String password, Set<Role> roles, List<Product> saleProducts) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.roles = roles;
        this.saleProducts = saleProducts;
    }
```

```
    public User(String firstName, String lastName, String email, String
password, Set<Role> roles) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.roles = roles;
    }

    public User() {
    }

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
```

```
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getEmail() {
    return email;
}

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

public String getPassword() {
    return password;
}

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

```
    public List<Product> getSaleProducts() {
        return saleProducts;
    }

    public void setSaleProducts(List<Product> saleProducts) {
        this.saleProducts = saleProducts;
    }

    public Set<Role> getRoles() {
        return roles;
    }

    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }
}
```

```
package com.example.inventorym.entity;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.util.UUID;

@Entity
```

```
@Table(name = "tokens")
public class UserToken{

    @Id
    @GeneratedValue(generator = "uuid2")
    @GenericGenerator(name = "uuid2", strategy = "uuid2")
    @Column(name = "token_id")
    private UUID id;

    @Column(name = "token", nullable = false, unique = true)
    private String token;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", referencedColumnName = "user_id",
updatable = false, nullable = false)
    private User user;


    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
    }
```

```
    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

```
package com.example.inventorym.facade;

import com.example.inventorym.dto.*;
import com.example.inventorym.entity.enums.Action;
import org.springframework.data.domain.Sort;

import java.util.List;
import java.util.UUID;
```

```
public interface DocumentFacade {

    List<DocumentShowBO> findAllByUser(String userEmail, Sort sortOrder);

    DocumentExportBO findDocByIdForExport(UUID documentId);

    List<ProductBO> findProductByDocId(UUID documentId);

    void createReception(DocumentReceptionBO documentReceptionBO,
String userEmail);

    void createRefused(DocumentRefusedBO documentRefusedBO, String
userEmail);

    List<DocumentBO> searchDocument(String name, Sort sortOrder,
DocumentSearchBO documentSearchBO);

    List<DocumentsCountActionsBO> findReport(Action action, String
userEmail);

    String findDocNumber();


}
```

```
package com.example.inventorym.facade;

import com.example.inventorym.dto.ProductBO;
import com.example.inventorym.dto.ProductCreateBO;
import com.example.inventorym.dto.ProductSearchBO;
import com.example.inventorym.entity.Product;
import org.springframework.data.domain.Sort;

import java.util.List;
import java.util.UUID;

public interface ProductFacade {

    List<ProductBO> findAllByUser(String userEmail, Sort sortOrder);

    void createProduct(ProductCreateBO productBO);

    Product findProductByName(String nameProduct, UUID userId);

    ProductBO findById(UUID id);

    void updateProduct(ProductBO productBO);

    List<ProductBO> searchProduct(ProductSearchBO productBO, String
userEmail, Sort sortOrder);
```

```
}
```

```
package com.example.inventorym.facade;

import com.example.inventorym.dto.OrderBO;
import com.example.inventorym.dto.SaleProductBO;

import java.util.UUID;

public interface ProductSalleFacade {

    OrderBO findProductInOrder();

    void addToOrder(UUID id);

    void editCount(String action, UUID id);

    void removeProducts(UUID id);

    void createSale(SaleProductBO saleProductBO,String userEmail);
}
```

```
package com.example.inventorym.repository;
```

```
import com.example.inventorym.entity.DocumentProduct;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository
public interface DocumentProductRepository extends
JpaRepository<DocumentProduct, UUID> {

    @Query("SELECT dp FROM DocumentProduct dp where dp.document.id =
:idDoc")
    List<DocumentProduct>
findDocumentProductByDocumentId(@Param("idDoc") UUID docId);

    @Query("SELECT dp FROM DocumentProduct dp where dp.document.id =
:id ")
    List<DocumentProduct> findDocumentProductById(@Param("id") UUID
id);
}

package com.example.inventorym.repository;
```

```
import com.example.inventorym.dto.DocumentsCountActionsBO;
import com.example.inventorym.entity.Document;
import com.example.inventorym.entity.enums.Action;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository
public interface DocumentRepository extends JpaRepository<Document, UUID> {
    @Query("SELECT d from Document d where d.user.id = :userId")
    List<Document> findAllByUser(@Param("userId") UUID userId, Sort sortOrder);

    boolean existsByDocumentNumber(String documentNumber);

    @Query("SELECT d FROM Document d where d.user.id = :id AND d.documentNumber LIKE %:name%")
    List<Document> searchDocumentByNumber(@Param("name") String name, @Param("id") UUID id, Sort sortOrder);
```

```
        @Query("SELECT new
com.example.inventorym.dto.DocumentsCountActionsBO( SUM(d.totalPrice), " +
                "EXTRACT(MONTH FROM d.createdAt)) " +
                "FROM Document d " +
                "WHERE :action MEMBER OF d.actions   " +
                "AND d.user.id =:userId " +
                "GROUP BY EXTRACT(MONTH FROM d.createdAt) " +
                "ORDER BY EXTRACT(MONTH FROM d.createdAt) ASC ")
        List<DocumentsCountActionsBO> findAnnualReport(@Param("action")
Action action,
                                                       @Param("userId") UUID
userId);


    }


    package com.example.inventorym.repository;

    import com.example.inventorym.dto.ProductBO;
    import com.example.inventorym.entity.Product;
    import org.springframework.data.domain.Sort;
    import org.springframework.data.jpa.repository.JpaRepository;
    import org.springframework.data.jpa.repository.Query;
    import org.springframework.data.repository.query.Param;
```

```
import org.springframework.stereotype.Repository;

import java.util.Collection;
import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Repository
public interface ProductRepository extends JpaRepository<Product, UUID> {

    @Query("SELECT p from Product p where p.user.id =:userId")
    List<Product> findAllByUserId(@Param("userId") UUID userId, Sort sort);

    @Query("SELECT p from Product p where p.user.id =:userId")
    List<Product> findAllByUserId(@Param("userId") UUID userId);

    @Query("SELECT pr FROM Product pr where pr.name = :name AND
pr.user.id = :userId")
    Optional<Product> findByName(@Param("name") String name,
@Param("userId") UUID userId);

    @Query("SELECT pr FROM Product pr WHERE pr.user.id = :userId " +
            "AND pr.name LIKE %:name%")
    List<Product> searchProductsByName(@Param("name") String name,
@Param("userId") UUID idUser, Sort sort);
```

```
        @Query("SELECT p FROM Product   p where p.user.id = :userId and
p.name = :name")
        List<Product> isExist(@Param("userId") UUID userId, @Param("name")
String name);
    }


    package com.example.inventorym.repository;

    import com.example.inventorym.entity.User;
    import org.springframework.data.jpa.repository.JpaRepository;
    import org.springframework.stereotype.Repository;

    import java.util.Optional;
    import java.util.UUID;

    @Repository
    public interface UserRepository extends JpaRepository<User, UUID> {
        Optional<User> findByEmail(String email);

        boolean existsByEmail(String email);


    }
```

```
package com.example.inventorym.repository;

import com.example.inventorym.entity.UserToken;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.UUID;

@Repository
public interface UserTokenRepository extends JpaRepository<UserToken,
UUID> {

    UserToken findByToken(String token);
    UserToken findByUserId(UUID userId);
}


package com.example.inventorym.repository;

import com.example.inventorym.entity.UserToken;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.UUID;
```

```
@Repository
public interface UserTokenRepository extends JpaRepository<UserToken,
UUID> {

    UserToken findByToken(String token);
    UserToken findByUserId(UUID userId);
}


package com.example.inventorym.services;

import com.example.inventorym.dto.DocumentsCountActionsBO;
import com.example.inventorym.dto.OrderBO;
import com.example.inventorym.dto.SaleProductBO;
import com.example.inventorym.entity.Document;
import com.example.inventorym.entity.User;
import com.example.inventorym.entity.enums.Action;
import org.springframework.data.domain.Sort;

import java.util.List;
import java.util.UUID;

public interface DocumentService {

    Document findById(UUID documentId);
```

```
    boolean isExistsDocument(String docNumber);

    List<Document> findByDocNumber(String docNumber, UUID userId, Sort
sortOrder);

    List<Document> findAllByUser(UUID userID, Sort sort);

    Document save(Document document);

    List<DocumentsCountActionsBO> findReport(Action action, UUID userId);

    Document save(String docNumber,OrderBO orderBO, User user);

}


package com.example.inventorym.services;

import com.example.inventorym.entity.Product;
import org.springframework.data.domain.Sort;

import java.util.List;
import java.util.UUID;

public interface ProductService {
```

```
        List<Product> findAllByUser(UUID userId, Sort sortProduct);

        List<Product> findAllByUser(UUID userId);

        void save(Product product);

        Product findByName(String nameProduct, UUID userId);

        Product findById(UUID productId);

        List<Product> findByNameAndUser(String nameProduct, UUID userId,
Sort sortOrder);

    }


    package com.example.inventorym.services;

    import com.example.inventorym.dto.UserRegisterBO;
    import com.example.inventorym.entity.User;
    import
com.example.inventorym.util.exception.UserWithEmailAlreadyExistException;

    import java.util.List;

    public interface UserService {
```

```
        void createNewUser(UserRegisterBO userDto) throws
UserWithEmailAlreadyExistException;

        User findUserByEmail(String userEmail);

        List<User> findAll();

        void saveUser(User userG);
    }


    package com.example.inventorym.util.converter;

    import com.example.inventorym.dto.*;
    import com.example.inventorym.entity.Document;
    import com.example.inventorym.entity.DocumentProduct;
    import com.example.inventorym.entity.User;
    import com.example.inventorym.entity.enums.Action;

    import java.math.BigDecimal;
    import java.time.LocalDateTime;
    import java.util.ArrayList;
    import java.util.Collections;
    import java.util.List;
    import java.util.stream.Collectors;
```

```
public final class DocumentConverter {

    private DocumentConverter() {
    }

    public static List<DocumentBO> toDocumentsBO(List<Document> documents) {
        return documents.stream().map(DocumentConverter::toDocumentBO).collect(Collectors.toList());
    }

    public static List<DocumentShowBO> toDocumentsShowBO(List<Document> documents) {
        return documents.stream().map(DocumentConverter::toDocumentShowBO).collect(Collectors.toList());
    }

    public static DocumentExportBO toDocumentExportBO(List<DocumentProduct> documentProducts, Document document) {
        var doc = new DocumentExportBO();
        doc.setTotalPrice(document.getTotalPrice());
        doc.setDocNumber(document.getDocumentNumber());
```

```
            doc.setCreatedAt(document.getCreatedAt());

doc.setTypeDocument(document.getActions().stream().map(Action::getAction).toList
().toString());
            doc.setTotalCount(document.getTotalCount());

doc.setProductExportBOS(documentProducts.stream().map(DocumentConverter::toP
roductExportBO).collect(Collectors.toList()));
            return doc;
        }

        private static ProductExportBO toProductExportBO(DocumentProduct
documentProduct) {
            return new ProductExportBO(documentProduct.getProduct().getName(),
documentProduct.getProduct().getPrice());
        }

        private static DocumentShowBO toDocumentShowBO(Document
document) {
            var documentBO = new DocumentShowBO();
            documentBO.setDocumentId(document.getId());

documentBO.setTypeDocument(document.getActions().stream().map(Action::getActi
on).toList().toString());
            documentBO.setCreatedAt(document.getCreatedAt());
            documentBO.setTotalCount(document.getTotalCount());
```

```
            documentBO.setTotalPrice(document.getTotalPrice());
            documentBO.setDocNumber(document.getDocumentNumber());
            documentBO.setSale(document.getActions().contains(Action.SALE));
            return documentBO;
        }

        private static DocumentBO toDocumentBO(Document document) {
            var documentBO = new DocumentBO();
            documentBO.setDocumentId(document.getId());
            documentBO.setTypeDocument(document.getActions());
            documentBO.setCreatedAt(document.getCreatedAt());
            documentBO.setTotalCount(document.getTotalCount());
            documentBO.setTotalPrice(document.getTotalPrice());
            documentBO.setDocNumber(document.getDocumentNumber());
            documentBO.setSale(document.getActions().contains(Action.SALE));
            return documentBO;
        }

        public static Document toDocument(String docNumber,OrderBO orderBO,
User user) {
            var doc = new Document();
            doc.setCreatedAt(generateDateTime());
            doc.setTotalPrice(orderBO.getTotalPrice());
            doc.setActions(Collections.singleton(Action.SALE));
            doc.setUser(user);
            doc.setTotalCount(BigDecimal.valueOf(orderBO.getCount()));
```

```
            doc.setDocumentNumber(docNumber);
            return doc;
        }


        private static LocalDateTime generateDateTime() {
            var numberDate = generateNumberDate();
            return LocalDateTime.of(2023, numberDate.get(0), numberDate.get(1),
generateNumber(7, 22), generateNumber(0, 59));
        }

        private static int generateNumber(int min, int max) {
            return min + (int) (Math.random() * ((max - min) + 1));
        }

        private static List<Integer> generateNumberDate() {
            var listNumber = new ArrayList<Integer>(2);
            var month = generateNumber(1, 5);
            var dayOfMonth = month == 2 ? generateNumber(1, 27) :
generateNumber(1, 30);
            listNumber.add(month);
            listNumber.add(dayOfMonth);
            return listNumber;
        }
    /*      public static DocumentProduct toDocumentProduct(OrderBO orderBO,
Document document) {
```

```
            var products = orderBO.getProductShortBOS();
            var docProd = new DocumentProduct();

        }*/
    }


    package com.example.inventorym.util.exception;

    import org.springframework.http.HttpStatus;

    import java.util.Map;

    public class ServiceException extends Exception {

        private HttpStatus responseCode;
        private ServiceResponseBody responseBody;

        public ServiceException(HttpStatus responseCode, ServiceResponseBody
responseBody) {
            super(responseBody.getMessage());
            this.responseCode = responseCode;
            this.responseBody = responseBody;
        }
```

```
        public ServiceException(HttpStatus responseCode, String errorCode, String
message) {
            super(message);
            this.responseCode = responseCode;
            this.responseBody = new ServiceResponseBody(message, errorCode);
        }

        public ServiceException(HttpStatus responseCode, String errorCode, String
message, Map<String, String> data) {
            super(message);
            this.responseCode = responseCode;
            this.responseBody = new ServiceResponseBody(message, errorCode,
data);
        }

        public HttpStatus getCode() {
            return responseCode;
        }

        public ServiceResponseBody getBody() {
            return responseBody;
        }
    }
```