НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
Комп'ютерних наук

_____ Голуб Б.Л. к. техн. наук, доцент

(підпис) (ПІБ, вчене звання і ступінь)

«____»_____2025р

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна система з продажу текстильних виробів»

Спеціальність 122 «Комп'ютерні науки»

Гарант освітньої програми

_____д. ек. н, професор_____ _____ _____Руденський Р.А.__

(науковий ступень та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____д. пед. н., професор____ _____ _____Кириченко В.В.__

(науковий ступень та вчене звання) (підпис) (ПІБ)

Виконав _____ ____Ковальчук Р.О.

_____

(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук

_____ Голуб Б.Л. к. техн. наук,
(підпис) (ПІБ, вчене звання і ступінь)
«__»__2025р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи студенту

КОВАЛЬЧУКУ РОМАНУ ОЛЕКСАНДРОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема проєкту «Програмне забезпечення для інформаційної системи з продажу текстильних виробів»

затверджена наказом ректора НУБіП України від 16.12.2024р. №2247-С

2. Термін подання завершеної роботи на кафедру 2025.05.25
(рік, місяць, число)
3. Вихідні дані до проєкту: база даних MySQL, серверна частина, інтерфейсна частина.
4. Перелік питань що розглядаються:
    1. Аналіз предметної області та постановка завдання.
    2. Визначення основних функцій в електронній комерції в сфері продажу текстильних виробів.
    3. Розробка інформаційної системи текстильних виробів.
    4. Тестування системи.
    5. Впровадження і експлуатація системи.
    6. Висновки.
5. Перелік графічного матеріалу:
    1. Діаграма прецедентів;
    2. ER діаграми бази даних;
    3. Діаграма розгортання.

Дата видачі завдання " 16 " грудня 2024р.

Керівник бакалаврської кваліфікаційної роботи
_____ _____ Кириченко В.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Ковальчук Р.О.
(підпис) (ПІБ)

Зміст

ВСТУП

У наш час інформаційні технології все більше прогресують, і разом із ними відбувається трансформація в різних галузях, і комерційну сферу це не обійшло стороною. Електронна комерція набуває все більшого значення, надаючи компаніям нові можливості для взаємодії з клієнтами та розширення ринків збуту. Ця індустрія знаходиться в стані постійного розвитку, змінюючи традиційні підходи до торгівлі та забезпечуючи споживачам зручність покупок для позитивного досвіду. Індустрія електронної комерції почала з’являтись вже давно, але масове своє застосування почалося відносно нещодавно, проте вона встигла розширитись до таких масштабів, що сучасне ведення бізнесу вже неможливо уявити без неї. Такий стрімкий зліт популярності онлайн-торгівлі легко пояснити, адже все почало цифровізуватись, в результаті чого онлайн магазини стали доступнішими та зручнішими для споживача.

Сфера продажу текстильних виробів не є винятком та активно застосовує інформаційні технології в комерції. Однак, багато підприємств малого та середнього бізнесу досі використовують застарілі або неповні системи управління продажами, що призводить до втрати потенційних клієнтів, ускладнює облік товарів та замовлень, а також знижує загальну ефективність бізнес-процесів. Створення сучасної, зручної та функціональної інформаційної системи у форматі веб-додатку дозволить автоматизувати ключові операції, прискорити обслуговування клієнтів, забезпечити точний облік та аналіз даних, що є критично важливим для конкурентоспроможності на сучасному ринку.

Тому метою цієї роботи є комплексне дослідження основ електронної комерції, а також розробка і реалізація ефективного, функціонального та зручного у використанні веб-додатку для інформаційної системи з продажу текстильних виробів. Програмний додаток має забезпечувати автоматизацію таких процесів, як управління каталогом товарів, обробка та моніторинг

замовлень, реєстрація і ведення клієнтських даних. Також додаток повинен включати адміністративну панель, яка надає можливість централізовано керувати всіма зазначеними процесами. Це дозволяє підприємству ефективно контролювати роботу системи, своєчасно оновлювати інформацію та оперативно реагувати на зміни в попиті. Такий підхід сприяє підвищенню продуктивності, оптимізації внутрішніх процесів і покращенню якості взаємодії з клієнтами.

1. Системний аналіз предметної області

1.1 Стан індустрії електронної комерції

Індустрія електронної комерції є однією з найбільш динамічних та швидкозростаючих галузей світової економіки. За останні десятиліття вона змінила традиційні способи торгівлі, надавши споживачам зручність, а бізнесу – зробила можливість виходу на глобальні ринки простішою. Стрімкий розвиток інтернету, широке розповсюдження мобільних пристроїв та вдосконалення логістики створили сприятливі умови для стрімкого росту онлайн-продажів. Ця галузь безперервно розвивається, пропонуючи нові функціональні можливості, персоналізовані сервіси та покращений користувацький досвід.

Фахівці у сфері розробки веб-додатків для електронної комерції займаються не лише програмуванням, але й системним аналізом бізнес-процесів, дизайном архітектури інформаційних систем, забезпеченням безпеки даних, інтеграцією з платіжними системами та впровадженням маркетингових інструментів. Значна частина світових роздрібних продажів вже відбувається онлайн, і ця частка продовжує зростати, що підкреслює актуальність та перспективність розробки ефективних інформаційних систем для онлайн-торгівлі. Сегмент продажу текстильних виробів в електронній комерції також демонструє стабільне зростання, оскільки споживачі все частіше обирають онлайн-платформи для придбання одягу, взуття та аксесуарів завдяки широкому асортименту, зручності вибору та доставці.

Отже, електронна комерція зараз займає ключовий вектор сучасної економіки, що стрімко змінюється під впливом технологічного прогресу та трансформації споживацьких звичок. Створення веб-додатків для представлення каталогу та продажу в сфері текстильних виробів, не втрачає актуальності й залишається надзвичайно перспективним, що потребує комплексного підходу та застосування ефективних інформаційних рішень. [1]

1.2 Особливості функціонування онлайн-магазинів текстильних виробів та їхні механіки

Магазин текстильних виробів — це спеціалізований торговий заклад, що пропонує широкий вибір асортименту тканинних виробів, від предметів одягу, таких як сукні, сорочки та штани, до домашнього текстилю, включаючи постільну білизну, рушники, скатертини, крісла мішки та пледи. Ці магазини успішно функціонують як у фізичному вигляді, так і в онлайн-форматі, адаптуючись до різні потреб сучасних покупців. Текстильна промисловість, маючи глибоке історичне коріння, є невід'ємною частиною повсякденного життя, тісно переплітаючись з модою, дизайном інтер'єру та культурними традиціями.

Магазини можуть мати різний асортимент, задовольняючи різні інтереси покупців. Основні різновиди виробів включають наступні:

- Магазин одягу: Це заклади, які спеціалізуються виключно на текстильних виробах для носіння, охоплюючи одяг для чоловіків, жінок та дітей.
- Магазин домашнього текстилю: Цей тип зосереджений на текстильних виробах, призначених для створення затишку та функціональності у житловому просторі. До їхнього асортименту входять постільна білизна, рушники, пледи, покривала, а також декоративні подушки та інше.
- Магазин тканин та фурнітури: Орієнтовані на тих хто займається шиттям, рукоділлям або дизайном. Вони пропонують широкий вибір тканин, нитки, блискавки та інші вироби для створення або відновлення текстилю.
- Магазин змішаного асортименту: Це найбільш поширений і універсальний тип, який поєднує кілька категорій текстильних виробів. Вони можуть пропонувати одночасно предмети одягу та домашнього текстилю.

Сучасний магазин текстилю виходить за рамки простого місця продажу, перетворюючись на інтерактивне середовище, де покупець може легко підібрати товар завдяки зручній навігації та деталізованій фільтрації за кольором, розміром, матеріалом, брендом, ціною тощо. Завдяки стрімкому розвитку електронної комерції, онлайн-магазини активно впроваджують деталізовані картки товарів, які включають високоякісні фото з різних ракурсів, відео огляди, докладні інструкції з догляду та таблиці розмірів, надаючи клієнтові можливість повноцінно ознайомитися з товаром ще до покупки. Окрім цього, в таких магазинах реалізовано ефективну систему управління запасами, яка дозволяє в режимі реального часу відстежувати наявність товарів, забезпечуючи точність інформації. Зручний особистий кабінет покупця полегшує процес повторних покупок, дає змогу відстежувати замовлення, змінювати особисті дані та зберігати обрані товари, тоді як адміністративна панель керування дозволяє ефективно працювати з каталогом, замовленнями, користувачами та аналітикою продажів.

Сфера текстильної торгівлі постійно еволюціонує, адаптуючись до нових технологій, сезонних змін попиту, екологічних трендів та загальних потреб ринку. Саме тому онлайн-магазини текстилю, на відміну від традиційних торгових точок, вимагають особливого підходу до візуалізації продукції, компенсуючи відсутність фізичного контакту з товаром за допомогою високоякісних зображень, 360-градусних фотографій, відео оглядів та, в окремих випадках, технологій доповненої реальності.

Для всебічного ознайомлення покупця з асортиментом кожен виріб має детальний опис, який охоплює склад тканини, точні розміри, інструкції з догляду, рекомендації щодо використання, а також можливі варіанти поєднання з іншими товарами, щоб уникнути непорозумінь.

Зручність вибору забезпечується розширеними системами фільтрації та пошуку, які дозволяють сортувати продукцію за різноманітними критеріями — типом тканини, кольором, візерунком, розміром, брендом, ціновим діапазоном і призначенням. Додаткову зручність у користуванні системою

забезпечує інтуїтивно зрозумілий процес замовлення: зручний кошик, можливість створення особистого кабінету для відстеження статусу замовлень і збереження даних. [2] Не менш важливою є система управління запасами, яка в режимі реального часу оновлює інформацію про наявність товарів, гарантуючи точність даних для покупців та ефективне функціонування складу.

Маркетингова стратегія онлайн-магазинів ґрунтується на широкому спектрі цифрових інструментів: SEO-оптимізації, контекстній і таргетованій рекламі, email-маркетингу та активній присутності в соціальних мережах.

Постійна оптимізація діяльності магазину забезпечується завдяки збору та аналізу даних про поведінку користувачів, обсяги продажів і ефективність маркетингових кампаній. Системи відгуків і прямі канали зв’язку з клієнтами є ключовими для безперервного вдосконалення асортименту та сервісу, що разом робить текстильний магазин важливою ланкою між виробником і споживачем, поєднуючи функціональність, естетику та технології у зручному для клієнта форматі.

Отже, сучасний магазин текстильних виробів це не лише місце продажу, а повноцінна сервісна платформа, яка поєднує широкий асортимент, зручність вибору, високі стандарти обслуговування та інноваційні технології. Завдяки онлайн-формату такі магазини ефективно адаптуються до потреб споживачів, забезпечуючи доступність, інформативність та комфорт на всіх етапах покупки.

Цю тематику для інформаційної системи я обрав, адже маю базове розуміння функціонування магазину, зокрема процесів приймання замовлень та ведення обліку товарів і тканин. Моєю основною метою було спробувати реалізувати ці процеси в онлайн-форматі, враховуючи та усуваючи потенційні незручності, притаманні традиційним фізичним магазинам. Вибір змішаного типу асортименту дозволив уникнути обмежень одним видом товару, що розширило функціональні можливості системи та її потенційну цінність.

1.3 Огляд існуючих рішень

1.3.1 Інтернет-магазин «Sanchobag»

Це онлайн-магазин, що спеціалізується на продажу безкаркасних меблів, таких як крісла-мішки, пуфи, подушки тощо. Хоча це не суто текстильні вироби в розумінні одягу чи постільної білизни, магазин активно використовує текстиль у своїй продукції та демонструє типовий функціонал електронної комерції. Сайт пропонує каталог товарів, інформацію для клієнтів, послуги брендування та оренди.

Переваги:

- Зручна навігація: Каталог добре структурований, що дозволяє легко знаходити різні види продукції.
- Детальні картки товарів: Містять інформацію про розміри, матеріали, кольори, а також можливості кастомізації (вибір наповнювача, чохла). Це критично важливо для текстильних виробів, де матеріал та зовнішній вигляд відіграють ключову роль.
- Додаткові послуги: Наявність послуг брендування та оренди розширює бізнес-модель, що є перевагою для потенційного розвитку власної системи.
- Адаптивний дизайн: Сайт, ймовірно, добре відображається на різних пристроях.
- Інформативність: Присутні розділи про компанію, виробництво, гарантії, клієнтів, магазини, а також блог та інформація про ремонт та догляд за виробами.

Недоліки:

- Відсутність розширених фільтрів: Для широкого асортименту текстильних виробів можуть знадобитися більш детальні фільтри за матеріалом, розміром, кольором тощо, що не завжди очевидно на цьому сайті.

1.3.2 Інтернет-магазин «Malva Style»

Це виробник постільної білизни та продавець пледів. Сайт орієнтований на домашній текстиль. Він пропонує каталог продукції, інформацію про компанію, умови доставки та оплати, а також розділи з відгуками та бонусами.

Переваги:

- Чітка спеціалізація: Сфокусованість на домашньому текстилі дозволяє реалізувати відповідний функціонал.
- Вигідні ціни та акції: Наголос на ціновій привабливості та безкоштовній доставці для певних умов.
- Відгуки та бонуси: Наявність розділів для відгуків та бонусної програми сприяє залученню та утриманню клієнтів.
- Простий та зрозумілий інтерфейс: Сайт виглядає чистим і легким у використанні, що важливо для споживача.

Недоліки:

Групування товарів: Кожен товар відбражається окремо навіть якщо це один і той самив вид товару але в нього інший колір він показується на окремій вкладці, що робить каталог перенавантаженим

1.3.3 Інтернет-магазин «Malva Style»

Цей веб-сайт є онлайн-магазином, що продає тканини для домашнього текстилю, а також готову продукцію, таку як постільна білизна, напірники та пледи. Він орієнтований як на роздрібних, так і на оптових покупців.

Переваги:

- Широкий асортимент текстилю: Включає як сировину (тканини), так і готову продукцію, що охоплює різні потреби клієнтів.
- Фокус на тканинах: Наявність фільтрів за типом тканини, складом, шириною рулону тощо є дуже важливою для магазину, що продає тканини.
- Інформація про контактні дані: Чітко вказані фізична адреса та номери телефонів, що підвищує довіру.

- Наявність оптових продажів: Це додає складність, але й розширює потенційну бізнес-модель.

Недоліки:

- Дизайн та користувацький досвід: Інтерфейс може виглядати дещо застарілим порівняно з сучасними веб-додатками, що може впливати на зручність використання та візуальну привабливість.
- Складність навігації: Для великої кількості типів тканин та готової продукції навігація та фільтрація можуть бути не такими інтуїтивними, як хотілося б.
- Обмежений функціонал для клієнтів: Не завжди очевидні можливості для відстеження замовлень, особистого кабінету, крім основних функцій покупки.

Проаналізовані веб-магазини для продажу текстильних та суміжних виробів підтверджують високий попит у цій галузі. Успішні платформи вирізняються інтуїтивним інтерфейсом, деталізованими картками товарів та ефективними пошуковими фільтрами.

Головним висновком є те, що розроблена система повинна бути максимально зручною та простою у використанні для кінцевого споживача, забезпечуючи легку навігацію та інтуїтивне оформлення замовлень.

2. ВИБІР МЕТОДІВ ТА ЗАСОБІВ ДЛЯ РОЗРОБКИ ПРОЕКТУ

2.1 Постановка завдання

Перед початком розробки інформаційної системи я сформулював чіткі завдання, які вона повинна виконувати. Ці завдання містять детальний опис даних, що використовуватимуться для зберігання ключової інформації. Дотримання цих вимог є обов’язковим для створення системи, яка забезпечуватиме стабільні та безпомилкові маніпуляції з інформацією і відповідатиме сучасним стандартам побудови та зв’язків даних. [3]

2.1.1 Опис інформації яку потрібно обробляти та зберігати

Почав я з визначення ключових учасників та об'єктів. Перш за все, це, звичайно, користувачі. Мені було важливо зберігати про них таку інформацію, як ім'я, мобільний номер, електронну пошту, адресу та поштовий індекс, а також, зрозуміло, пароль для безпечного доступу до системи. Крім того, для розширення функціоналу, я додав поле роль, що дозволяє розрізняти звичайних покупців від адміністраторів, які можуть мати доступ до додаткових функцій системи. Ці дані дозволяють ідентифікувати кожного клієнта та забезпечувати персоналізований досвід, а також керувати доступом.

Наступним критично важливим блоком є товари. Тут я передбачив зберігання унікального ідентифікатора товару, його назви, типу, детального опису з усіма характеристиками, ціни та поточної кількості на складі. Важливою особливістю є можливість зберігання зображення товару. Щоб забезпечити гнучкість в управлінні асортиментом, я додав атрибут приховати, який дозволяє тимчасово приховувати товари від відображення покупцям. Найголовніше, враховуючи специфіку товарів, я доповнив структуру товару полями розмір та тип тканини, що дозволяє прив'язати кожен товар до конкретного розміру та типу тканини.

Для деталізації інформації про розміри я створив окрему сутність. Вона включає ідентифікатор розміру, його назву, тип продукту, до якого цей розмір відноситься, а також довжину та ширину. Це дозволяє чітко визначати габарити виробів та забезпечувати точний вибір для покупця.

Аналогічно, для тканин я розробив дві сутності. Перша – це типи тканини, де зберігається назва типу. Друга – це власне інформація про саму тканину, яка містить унікальний ідентифікатор тканини, назву її типу, колір та зображення тканини. Такий поділ дозволяє гнучко управляти характеристиками тканин та асоціювати їх з різними товарами.

Логічним продовженням функціоналу електронної комерції є кошик покупця. Я реалізував його таким чином, щоб кожен запис у кошику чітко посилався на ідентифікатор користувача та ідентифікатор товару, фіксував бажану кількість цього товару, а тепер і ідентифікатор тканини. Це дозволяє користувачам додавати товари, враховуючи обрану тканину, змінювати їх кількість і переглядати свій вибір перед оформленням замовлення.

Після заповнення кошика настає етап транзакцій. Кожна транзакція є унікальною подією, і я фіксую її унікальним ідентифікатором, ім'ям користувача, що її здійснив, точною датою і часом та загальною сумою. Це основа для відстеження фінансових операцій у системі.

Сама транзакція складається з однієї або кількох позицій замовлення. Для кожної такої позиції я передбачив її зв'язок з відповідною транзакцією, сама транзакція має такі поля індефікаційний номер, пошту користувача, час коли була виконана транзакція та сума. Це є критично важливим для логістики та управління виконанням замовлень.

Саме замовлення включає в себе номер транзакції, індефікаційний номер продукту, кількість цього товару, тип тканини та статус замволення. Це дозволяє зручно формувати повну та вичерпну історію замовлень для користувача та адміністратора.

Усі ці сутності тісно взаємопов'язані, що дає змогу будувати складніші запити та отримувати повну картину про діяльність системи, починаючи від

реєстрації користувача і закінчуючи аналізом попиту, з урахуванням всіх специфічних характеристик текстильних виробів. Такий підхід забезпечує гнучкість у роботі з даними та дозволяє легко розширювати функціонал системи в майбутньому, адаптуючись до нових вимог ринку.

2.1.2 Операції, які необхідно автоматизувати

Для кінцевих користувачів автоматизація починається вже на етапі реєстрації та авторизації, де система автоматично обробляє дані, забезпечуючи швидкий і безпечний доступ до персоналізованих функцій. Коли користувач переглядає каталог товарів, система миттєво реагує на запити, автоматично фільтруючи та відображаючи релевантні результати пошуку.

Процес формування замовлення також значною мірою автоматизований: при додаванні/видаленні товарів до та з кошика та зміні кількості, система динамічно перераховує загальну суму та оновлює стан кошика в реальному часі. Це усуває потребу в ручних підрахунках і забезпечує актуальність інформації. Після оформлення замовлення, система автоматично генерує підтвердження, реєструє замовлення у базі даних і починає процес його обробки. Користувачам надається можливість переглядати історію своїх замовлень та їхній статус, що також автоматично оновлюється системою в міру просування замовлення по ланцюжку виконання. Це забезпечує повну прозорість без необхідності ручних запитів або дзвінків.

На адміністративному рівні система забезпечує всеосяжну автоматизацію для ефективного управління комерційними операціями. Управління товарами значно спрощене: адміністратори можуть автоматично додавати, редагувати або видаляти інформацію про товари через зручний інтерфейс, і ці зміни миттєво відображаються у каталозі для користувачів. Це виключає ручне оновлення кожного окремого елемента.

Перегляд та зміна статусу замовлень також є автоматизованим процесом, що дозволяє адміністраторам швидко оновлювати стан замовлення з мінімальними зусиллями, що в свою чергу може запускати автоматичні сповіщення для клієнтів. Управління тканинами та управління розмірами

також автоматизовані, дозволяючи адміністраторам легко додавати нові категорії, редагувати існуючі або видаляти неактуальні записи. Ці автоматизовані функції значно скорочують час на рутинні операції, мінімізують ризик людських помилок і дозволяють зосередитися на стратегічних завданнях та покращенні сервісу.

Для забезпечення належного користувацького досвіду та уникнення спроб замовлення недоступних товарів, в системі були реалізовані механізми автоматичного зміни візуального відображення та обмеження функціональної доступності товарів яких не можна замовити да даний час. Цей комплекс операцій дозволяє користувачам інтуїтивно розуміти статус товару без необхідності додаткових перевірок.

Система також автоматизує групування товарів з однаковою назвою та типом тканини. Це дозволяє подавати асортимент більш структуровано та інтуїтивно зрозуміло для користувача, полегшуючи пошук та вибір необхідної продукції, а також оптимізує управління товарними запасами для адміністраторів.

Для забезпечення ефективної комунікації з користувачами, реалізовано автоматизоване відправлення повідомлень на електронну пошту на ключових етапах взаємодії. Це включає сповіщення про успішну реєстрацію в системі, підтвердження оформлення замовлення з деталізацією його складу, а також інформування про відправлення замовлення, що надає користувачам актуальну інформацію про статус їхніх покупок.

2.1.3 Класифікація системи

Розроблювана система належить до інформаційних систем у сфері електронної комерції, що функціонує за моделлю клієнтською та серверною частиною. За характером використання інформації система належить до класу системи обробки даних та інформаційно-пошукові системи. За масштабом вона орієнтована на малий та середній бізнес, що займається продажем текстильних виробів. Система підтримує стандарти HTTP/HTTPS для комунікації між клієнтом і сервером, а також JDBC для взаємодії з базою

даних. Забезпечує високий ступінь автоматизації процесів замовлення, управління каталогом та базовими адміністративними операціями.

2.1.4 Функціональні вимоги

- Реєстрація: Можливість створення нового облікового запису в системі.
- Авторизація: Можливість входу в існуючий обліковий запис.
- Перегляд товарів: Доступ до каталогу з усіма доступними текстильними виробами.
- Додавання до кошика: Функціонал для додавання обраних товарів у кошик.
- Оформлення замовлення: Можливість завершити процес покупки, вказавши дані для доставки та оплати.
- Управління товарами: Функції додавання нових товарів, редагування інформації про існуючі та видалення товарів.
- Управління тканинами: Можливість створювати, редагувати та видаляти тканин.
- Управління розмірами: Можливість створювати, редагувати та видаляти розміри товарів.
- Управління замовленнями: Доступ до всіх замовлень, можливість перегляду деталей та зміни їхнього статусу.
- Відправлення повідомлень: Реалізувати автоматичну відправку електронного повідомлення корситувач, якщо виконується дія реєстрації, створення замовлення та зміна статусу замовлення.

2.1.5 Нефункціональні вимоги

- Продуктивність: Швидка завантаження сторінок та обробка запитів.
- Надійність: Стійкість до збоїв, збереження даних.
- Безпека: Захист даних користувачів, запобігання SQL-ін'єкціям та XSS-атакам, безпечна авторизація.

- Зручність використання: Інтуїтивно зрозумілий інтерфейс для всіх категорій користувачів, легкість навігації.
- Адаптивність: Коректне відображення на різних пристроях та розмірах екрана.

2.2 Формування комплексу моделей предметної області

Діаграма прецедентів є одним з інструментів моделювання в UML (Unified Modeling Language), який візуалізує функціональні вимоги до системи. Вона показує взаємодію між акторами та системою, відображаючи перелік усіх основних функціональних можливостей прецедентів, які система надає. Кожен прецедент описує певну послугу або результат, який система надає актору. Ця діаграма допомагає чітко визначити межі системи та її основні функції з погляду зовнішнього користувача.

В моїй системі вона демонструє, що незареєстрований користувач може виконувати базові дії, що не потребують аутентифікації, як-от перегляд асортименту та пошук товарів. Для зареєстрованого клієнта розширений функціонал, що включає управління замовленнями, особистим профілем та здійснення покупок. Адміністратора має повний контроль над контентом магазину та управлінням замовленнями, що є критично важливим для функціонування бізнесу. (Рис. 2.1.)

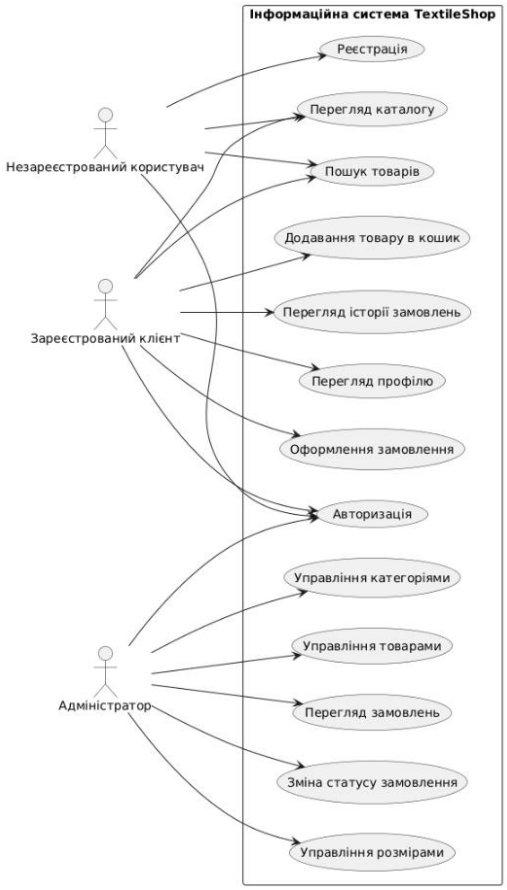

Рис. 2.1. Діаграма перецедентів

Опис діаграми:

Незареєстрований користувач - основний тип користувача системи, який взаємодіє з публічною частиною веб-додатку без проходження авторизації. Його дії обмежені переглядом доступного асортименту та базовими пошуковими функціями.

Зареєстрований клієнт - користувач, який пройшов процедуру реєстрації та авторизації у системі. Має розширені можливості, включаючи персоналізований функціонал та доступ до історії взаємодії з магазином.

Адміністратор - спеціальний тип користувача з розширеними правами доступу, який відповідає за управління контентом, замовленнями та іншими адміністративними аспектами роботи системи.

Перегляд каталогу - дозволяє користувачу переглядати асортимент доступних товарів. Цей прецедент включає відображення списку категорій, товарів та їх короткого опису.

Пошук товарів - дозволяє користувачу знаходити конкретні товари за заданими критеріями. Цей прецедент може включати функції фільтрації за ціною, розміром, кольором тощо.

Реєстрація - дозволяє незареєстрованому користувачу створити новий обліковий запис у системі, надавши необхідні дані. Цей прецедент є точкою входу для переходу до функціоналу зареєстрованого клієнта.

Авторизація - дозволяє користувачу увійти до свого облікового запису, надавши облікові дані. Успішна авторизація відкриває доступ до відповідного функціоналу системи.

Перегляд історії замовлень - дозволяє зареєстрованому клієнту переглядати список усіх раніше зроблених замовлень, їх статуси та деталі. Цей прецедент забезпечує доступ до персональної історії покупок.

Додавання товару в кошик - дозволяє зареєстрованому клієнту додати обраний товар з каталогу до свого віртуального кошика. Цей прецедент передбачає вибір кількості та, за необхідності, варіантів товару.

Оформлення замовлення - дозволяє зареєстрованому клієнту оформлює покупку, перетворивши вміст кошика на замовлення. Цей прецедент включає вибір адреси доставки, способу оплати та підтвердження замовлення.

Перегляд профілю - дозволяє зареєстрованому клієнту переглядати та редагувати свою особисту інформацію, таку як контактні дані та адреси доставки.

Управління товарами - дозволяє адміністратору виконувати операції з додавання, редагування та видалення товарів у каталозі магазину. Цей прецедент включає управління назвами, описами, цінами, зображеннями та наявністю товарів.

Управління категоріями - дозволяє адміністратору створювати, змінювати та видаляти категорії товарів, що використовуються для структурування каталогу.

Управління розмірами - дозволяє адміністратору додавати, редагувати та видаляти доступні розміри для текстильних виробів, забезпечуючи коректне відображення варіантів товарів.

Зміна статусу замовлення - дозволяє адміністратору оновлювати статус замовлень на статус «Відправлене» або «Скасоване».

Перегляд замовлень - дозволяє адміністратору переглядати список усіх замовлень у системі, застосовувати фільтри та отримувати детальну інформацію про кожне замовлення.

Діаграма наочно демонструє, що система забезпечує повний цикл взаємодії з клієнтом, від базового перегляду каталогу та пошуку товарів для незареєстрованих відвідувачів, до можливості реєстрації, авторизації, управління кошиком, оформлення та відстеження замовлень для зареєстрованих клієнтів. З боку адміністратора діаграма підкреслює критично важливий функціонал з управління всім контентом магазину та контролю за процесом виконання замовлень.

2.3 Етапи розробки системи

2.3.1 Концепція та планування

Планування є основним етапом розробки, де окреслюються ключові риси задуму та цілі проєкту. Стартуючи з вивчення ринку електронної комерції та окреслення цільової аудиторії, збирається інформація про онлайн продажах, перегляд конкурентні переваги наявних рішень. Необхідно окреслити основний функціонал системи та її цільове призначення, аби створити дієвий інструмент для продажу текстилю та виправдати сподівання бізнесу і користувачів.

Також на цьому етапі передбачається формування загального бачення веб-додатку, що демонструватиме його структуру та основних складових. Дизайнери й системні аналітики працюють над створенням концептуального бачення системи, яке ілюструватиме її унікальні можливості та взаємодію з користувачем. Планування цього етапу також включає визначення ключових

функціональних можливостей системи, а також формування нефункціональних вимог.

2.3.2 Проектування архітектури та даних

Етап проектування відіграє ключову роль у процесі розробки, оскільки дозволяє детально визначити структуру системи та її компонентів, перш ніж вкладати значні ресурси в повну реалізацію. Проектування починається з визначення основних функціональних блоків системи та їх взаємодії. На цьому етапі команда розробників створює докладні моделі, що відображають логіку роботи системи та організацію даних. Основна мета – забезпечити надійність, масштабованість та ефективність майбутнього програмного забезпечення.

Першим кроком у проектуванні є створення логічної моделі даних у вигляді ER-діаграми, що включає сутності та їхні взаємозв'язки. Ця модель деталізується до фізичної схеми бази даних MySQL, визначаючи таблиці, поля, типи даних та ключі, що забезпечує відповідність принципам реляційності та нормалізації.

Наступним важливим аспектом проектування є архітектура програмного забезпечення. Створюється діаграма компонентів або пакетів, що відображають організацію серверної частини, наприклад окремі пакети для моделей, DAO-рівня, сервлетів-контролерів. Детально розробляються алгоритми для ключових бізнес-процесів, які можуть бути представлені у вигляді блок-схем. Проектується взаємодія між інтерфейсною частиною та серверною, визначаються методи обробки запитів та передачі даних. Цей процес може включати багаторазове внесення змін до проектних рішень, щоб досягти максимальної функціональності та зручності для користувачів. [4]

Проектування також допомагає визначити технічні обмеження та можливості. Можна випробувати різні технологічні рішення та інструменти, щоб знайти оптимальні способи реалізації системних функцій. Це важливо для планування подальших етапів розробки та визначення ресурсів, необхідних для завершення проекту. Проектування завершується створенням детального плану та технічної документації, яка служить фундаментом для подальшої

роботи над системою, допомагаючи уникнути серйозних помилок і заощаджуючи час та ресурси.

2.3.3 Реалізація

Етап реалізації є найбільш інтенсивним і трудомістким у процесі розробки інформаційної системи. На цьому етапі вся команда зосереджується на створенні та завершенні всіх програмних компонентів системи, спираючись на детальний план, розроблений на етапі проектування.

Основна частина роботи на етапі реалізації припадає на програмістів. Розробники серверної частини займаються написанням логіки, реалізацією взаємодії з базою даних через DAO-шаблон, створенням сервлетів для обробки HTTP-запитів та управління бізнес-логікою. Також проводиться робота над реалізацією управлінням даними, замовленнями, користувачами та іншими важливими аспектами системи.

Паралельно, розробники інтерфейсної частини активно працюють над створенням візуальних елементів веб-додатку. Вони розробляють структуру HTML-сторінок, застосовують CSS для стилізації та візуального оформлення, а також пишуть JavaScript-сценарії для інтерактивності, валідації форм та динамічного оновлення елементів інтерфейсу. Забезпечується відповідність інтерфейсу розробленим макетам та адаптивність для коректного відображення на різних пристроях.

На етапі реалізації також проводиться постійне внутрішнє тестування коду. Перевіряються окремі модулі та їх інтеграцію на наявність помилок та відповідність заявленому функціоналу. Цей ітеративний процес дозволяє вчасно виявляти та виправляти проблеми, а також вносити необхідні зміни для покращення функціональності та користувацького досвіду. Завершення етапу реалізації знаменується створенням готової до тестування версії системи, де всі основні компоненти завершені та інтегровані в єдиний продукт.

2.3.4 Реалізація

Етап тестування та впровадження включає завершення всіх деталей системи, виправлення залишкових помилок, оптимізацію продуктивності та підготовку до запуску інформаційної системи.

При тестуванні все зосереджується на вдосконаленні всіх аспектів системи, щоб забезпечити високу якість кінцевого продукту. Програмісти продовжують працювати над оптимізацією коду, підвищенням продуктивності та стабільності системи. Проводиться комплексне тестування, включаючи:

- Модульне тестування: перевірка коректності роботи окремих програмних модулів.
- Інтеграційне тестування: перевірка взаємодії між компонентами системи інтерфейсної частини та серверної.
- Системне тестування: перевірка всієї системи на відповідність функціональним та нефункціональним вимогам, включаючи тестування безпеки та продуктивності.
- Функціональне тестування: перевірка виконання всіх заявлених бізнес-функцій.
- Тестування юзабіліті**:** оцінка зручності використання інтерфейсу, простоти навігації.

Будь-які виявлені проблеми негайно виправляються командою розробників.

Фінальним етапом є підготовка до впровадження системи. Це включає створення остаточних збірок веб-додатку, підготовку SQL-скриптів для розгортання бази даних, розробку інструкцій з встановлення та конфігурації. Після успішного тестування та підготовки, система розгортається на сервері. Після впровадження команда продовжує здійснювати підтримку системи, моніторинг роботи системи, випуск оновлень та виправлень для забезпечення стабільної та приємної взаємодії з користувачами.

2.3.5 Виснвоки

На кожному з етапів потрібно здійснити глибокий аналіз потреб, проєктування структури системи, реалізацію програмних компонентів та тестування. Крім того, важливо враховувати вимоги до безпеки, масштабованості та зручності користування, а також забезпечити якісну інтеграцію між усіма модулями системи. Такий підхід забезпечує створення функціонального, надійного та зручного веб-рішення для онлайн-продажу текстилю, який буде відповідати зазначеним на етапі планування функціоналу та задовольняти потреби кінцевих користувачів.

2.4 CASE-засоби для розробки інформаційної системи

Основними інструментами для створення веб-додатку є технології клієнтської та серверної частин, що відповідають за формування користувацького інтерфейсу, обробку бізнес-логіки та взаємодію з базою даних. Середовища розробки та сервери застосунків також відіграють важливу роль у процесі створення та розгортання програмного продукту.

Під час вибору середовища розробки та технологічного стеку основну увагу було зосереджено на надійності та продуктивності. Це передбачало використання таких інструментів і технологій, які забезпечують ефективну роботу системи як у процесі розробки, так і під час подальшої експлуатації.

Для аналізу було обрано такі групи технологій та інструментів: клієнтська частина, серверна частина, база даних, середовище розробки та сервер застосунків.

Для клієнтської частини системи було обрано такі технології для створення інтерфейсу: HTML, CSS, JavaScript та Bootstrap.

HTML (HyperText Markup Language) є фундаментальною мовою розмітки, що забезпечує структуру будь-якої веб-сторінки. Це основа, на якій будується весь візуальний контент. Без HTML створення веб-додатку неможливе, тому він був обраний як базова технологія.

CSS (Cascading Style Sheets) використовується для стилізації HTML-елементів, керуючи їхнім зовнішнім виглядом, кольорами, шрифтами, розташуванням та адаптивністю. CSS дозволяє відділити вміст від представлення, роблячи код чистішим та легшим для підтримки.

JavaScript – це мова програмування, що надає інтерактивність на стороні клієнта. Вона дозволяє реалізувати динамічні елементи інтерфейсу, валідацію даних у формах та асинхронну взаємодію з сервером без перезавантаження сторінки [5].

Bootstrap [6] є провідним CSS-фреймворком, що надає набір готових стилів та JavaScript-компонентів для швидкої розробки адаптивних та сучасних інтерфейсів. Його використання значно прискорює процес верстки та забезпечує коректне відображення на різних пристроях.

Плюси:

- Швидкість розробки: Наявність готових компонентів та класів.
- Адаптивність: Вбудована підтримка адаптивного дизайну.

Мінуси:

- Типовий вигляд: Сайти на Bootstrap можуть виглядати однотипно без додаткової кастомізації.
- Надмірність коду: Може включати функціонал, який не використовується у проекті.

Альтернативні рішення включають сучасні JavaScript-фреймворки, такі як React, Angular або Vue.js. Вони добре підходять для створення складних, динамічних веб-додатків із насиченим інтерфейсом та активною взаємодією користувача [7]. Проте їх використання не було доцільним у даному проєкті, оскільки розроблювана система має обмежений масштаб і відносно просту функціональність. Застосування таких фреймворків лише ускладнило б архітектуру проєкту, збільшило час розробки та потребу в додаткових ресурсах без суттєвої вигоди для кінцевого результату.

Було обрано, що серверна частина системи буде реалізована на платформі Java з використанням технологій JDBC, Servlet та JSP.

Java (JDK) [8] була обрана як основна мова для серверної розробки. Вона відома своєю надійністю, високою продуктивністю, масштабованістю та здатністю працювати на будь-якій платформі. Java є індустріальним стандартом для великих корпоративних систем, що забезпечує її стабільність та довгострокову підтримку.

Плюси:

- Продуктивність та надійність: Створена для високо навантажених систем.

- Платформонезалежність: Код працює на різних операційних системах.
- Велика екосистема: Величезна кількість бібліотек, фреймворків та інструментів.

Мінуси:

- Вищий поріг входу: Може бути складнішою для вивчення початківцям.
- Навантаженість коду: Порівняно з деякими мовами, Java може вимагати більше рядків коду для вирішення тих самих завдань.

JDBC (Java Database Connectivity) є стандартизованим API для підключення Java-додатків до реляційних баз даних. Це дозволяє здійснювати прямі SQL-запити, керувати транзакціями та отримувати результати.

Плюси:

- Прямий контроль: Дозволяє повністю контролювати SQL-запити та оптимізувати їх.
- Універсальність: Може підключатися до будь-якої реляційної СУБД.

Мінуси:

- Багато шаблонного коду: Вимагає написання більшої кількості коду для типових операцій з базою даних порівняно з ORM-фреймворками.

Сервлети та JSP (JavaServer Pages) є ключовими технологіями Java EE для розробки веб-додатків. Сервлети обробляють HTTP-запити [9], реалізуючи бізнес-логіку та взаємодію з моделлю даних, тоді як JSP відповідають за динамічну генерацію HTML-сторінок, слугуючи рівнем представлення. Ця комбінація дозволяє чітко розділяти логіку та представлення, дотримуючись архітектурного шаблону MVC.

Плюси:

- Фундаментальність: Надають глибоке розуміння основ роботи Java EE веб-додатків.

- Гнучкість: Дозволяють контролювати кожен аспект обробки запитів та відображення.

Мінуси:

- Менша автоматизація: Порівняно з повноцінними фреймворками, вимагають більше ручного конфігурування.
- Складність для великих проектів: Можуть стати незручними для дуже великих систем без додаткових абстракцій.

Альтернативами для серверної частини могли б бути такі фреймворки, як Spring MVC або Struts. Вони пропонують більш високорівневі абстракції та автоматизацію, але вимагають вищого порогу входу та більшої початкової конфігурації.

Як система управління базами даних (СУБД) обрано MySQL. Це одна з найпопулярніших реляційних СУБД [10] з відкритим вихідним кодом, що ідеально підходить для веб-додатків. Вона забезпечує надійне зберігання структурованих даних, високу продуктивність для типових веб-операцій та легкість у розгортанні.

Плюси:

- Відкритий вихідний код: Безкоштовна для використання.
- Надійність та продуктивність: Добре зарекомендувала себе для веб-розробки.
- Велика спільнота: Легко знайти підтримку та ресурси.

Мінуси:

- Можливі обмеження для великих систем: Хоча для більшості веб-додатків її можливостей достатньо але для великих систем з великою схемою даних вона може себе проявити занадто обмеженою.

Альтернативні реляційні СУБД включають PostgreSQL або комерційні рішення, як-от Oracle чи SQL Server. NoSQL СУБД, наприклад, MongoDB, також є варіантом, особливо для гнучких або масштабованих систем. Проте вони краще підходять для зберігання неструктурованих або слабо пов’язаних

даних. У випадку системи електронної комерції, де існують чіткі зв’язки між сутностями, такими як товари, категорії, замовлення та користувачі, класична реляційна модель є більш природною, ефективною та зручною для реалізації бізнес-логіки.

Eclipse EE було обрано як інтегроване середовище розробки. Це потужний та багатофункціональний інструмент для розробки Java-додатків, зокрема для Java EE. Він надає зручний редактор коду, інструменти для налагодження, інтеграцію з системами контролю версій та легке розгортання на серверах застосунків.

Плюси:

- Потужний функціонал: Підтримує всі необхідні аспекти розробки Java EE.
- Відкритий вихідний код: Доступний безкоштовно.
- Гнучкість та розширюваність: Велика кількість плагінів.

Мінуси:

- Ресурсоємність: Може вимагати значних системних ресурсів.
- Крива навчання: Для новачків може бути складним освоєння всіх можливостей.

Для розгортання та виконання веб-додатку використовувався Apache Tomcat v9.0. Tomcat – це легкий та ефективний веб-сервер та сервлет-контейнер, який імплементує специфікації Servlet та JSP. Він є відмінним вибором для веб-додатків, що не потребують повного набору функцій великих Java EE серверів застосунків.

Плюси:

- Легкість та простота розгортання: Швидкий запуск та конфігурація.
- Продуктивність: Ефективний для більшості веб-додатків.
- Відкритий вихідний код: Безкоштовний.

Мінуси:

- Не є повноцінним Java EE сервером: Не підтримує деякі розширені специфікації.

3. РОЗРОБКА ПРОЕКТУ *«TEXTILESHOP»*

3.1 Розробка інформаційного забезпечення

Інформаційне забезпечення реалізовано в MySQL [11] шляхом розробки бази даних за реляційними принципами, яка зберігає дані про користувачів, товари, тканини, замовлення, кошик і транзакції інтернет-магазину.

З самого початку ми створюємо саму базу данніх за допомогою sql-запиту. (Рис. 3.1)

```
CREATE DATABASE `textileshop`;
```

Рис. 3.1. sql-запит створення бази даних

Далі створюємо таблицю "Розміри", яка зберігатиме інформацію про доступні розміри товарів. Вона містить унікальний ідентифікатор розміру, що є первинним ключем, назву розміру, тип продукту з індексом для швидкого пошуку, а також необов’язкові параметри довжини і ширини. Ця таблиця використовується для зручного присвоєння розмірів при додаванні товарів та їх фільтрації. (Рис. 3.2)

```
-- Створення таблиці sizes
CREATE TABLE sizes (
    size_id VARCHAR(45) PRIMARY KEY,
    size_name VARCHAR(50) NOT NULL,
    product_type VARCHAR(50) NOT NULL,
    length DECIMAL(10,2),
    width DECIMAL(10,2)
);
```

Рис. 3.2. sql-запит створення таблиці розмірів

Таблицю "Типи тканин" містить інформацію про різні види тканин, доступних у магазині, і їхню ціну за метр. Назва типу тканини є унікальним ідентифікатором і первинним ключем.(Рис. 3.3.)

```
CREATE TABLE fabric_types (
    fabric_type_name VARCHAR(50) PRIMARY KEY,
    price_per_meter DECIMAL(10,2) NOT NULL
);
```

Рис. 3.3. sql-запит створення таблиці типів тканин

Таблицю "Користувач", вона зберігає дані про всіх користувачів системи — клієнтів та адміністраторів. Кожен користувач має унікальну електронну адресу, яка одночасно є логіном і первинним ключем. Також тут зберігаються ім’я, мобільний телефон, повна адреса з поштовим індексом, пароль і роль користувача, що визначає його статус у системі.(Рис. 3.4.)

```
CREATE TABLE user (
    email VARCHAR(60) PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    mobile BIGINT,
    address VARCHAR(250),
    pincode INT,
    password VARCHAR(20) NOT NULL,
    role VARCHAR(50) NOT NULL
);
```

Рис. 3.4. sql-запит створення таблиці користувачів

Таблицю "Тканини" в ній зберігаються конкретні партії тканин, що використовуються для виробництва або продажу. Вона містить унікальний ідентифікатор тканини, посилання на тип тканини з відповідної таблиці, колір та зображення у вигляді бінарних даних.(Рис. 3.5.)

```
CREATE TABLE fabrics (
    fabric_id VARCHAR(45) PRIMARY KEY,
    fabric_type_name VARCHAR(50) NOT NULL,
    color VARCHAR(30) NOT NULL,
    image MEDIUMBLOB,
    FOREIGN KEY (fabric_type_name) REFERENCES fabric_types(fabric_type_name)
);
```

Рис. 3.5. sql-запит створення таблиці тканин

Таблицю "Продукт", що містить інформацію про всі товари, доступні для продажу. Кожен продукт має унікальний ідентифікатор, назву, тип товару з індексом для швидкого пошуку, опис, ціну, кількість на складі, зображення,

прапорець для приховування, а також посилання на стандартний розмір і тип тканини, з яких він виготовлений.(Рис. 3.6.)

```
CREATE TABLE product (
    pid VARCHAR(45) PRIMARY KEY,
    pname VARCHAR(100) NOT NULL,
    ptype VARCHAR(20) NOT NULL,
    pinfo VARCHAR(350),
    pprice DECIMAL(12,2) NOT NULL,
    pquantity INT NOT NULL,
    image MEDIUMBLOB,
    hide TINYINT(1) DEFAULT 0,
    size VARCHAR(45),
    fabric_type VARCHAR(50),
    INDEX (ptype),
    FOREIGN KEY (fabric_type) REFERENCES fabric_types(fabric_type_name),
    FOREIGN KEY (size) REFERENCES sizes(size_id)
);
```

Рис. 3.6. sql-запит створення таблиці продуктів

Таблицю "Кошик користувача"де тимчасово зберігаться товари, додані користувачем до кошика перед оформленням замовлення. Вона містить ім’я користувача і ідентифікатор продукту як складений ключ, кількість товару та опціональне посилання на конкретну партію тканини.(Рис 3.7.)

```
CREATE TABLE usercart (
    username VARCHAR(60) NOT NULL,
    prodid VARCHAR(45) NOT NULL,
    quantity INT NOT NULL,
    fid VARCHAR(45),
    PRIMARY KEY (username, prodid),
    FOREIGN KEY (username) REFERENCES user(email),
    FOREIGN KEY (prodid) REFERENCES product(pid),
    FOREIGN KEY (fid) REFERENCES fabrics(fabric_id)
);
```

Рис. 3.7. sql-запит створення таблиці кошику

Таблиця "Замовлення" деталізує склад кожного замовлення, зберігаючи унікальний ідентифікатор замовлення, ідентифікатор товару, кількість, суму, статус відправки та необов’язкове посилання на тканину, використану для цього товару.(Рис. 3.8.)

```
CREATE TABLE orders (
    orderid VARCHAR(45) NOT NULL,
    prodid VARCHAR(45) NOT NULL,
    quantity INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    shipped INT DEFAULT 0,
    fabricId VARCHAR(50),
    PRIMARY KEY (orderid, prodid),
    FOREIGN KEY (prodid) REFERENCES product(pid),
    FOREIGN KEY (fabricId) REFERENCES fabrics(fabric_id)
);
```

Рис. 3.8. sql-запит створення таблиці замовлень

Таблиця "Транзакції" зберігає інформацію про фінансові платежі, пов’язані з замовленнями. Вона містить унікальний ідентифікатор транзакції, посилання на замовлення, ім’я користувача, дату і час операції, а також суму платежу.(Рис. 3.9.)

```
CREATE TABLE transactions (
    transid VARCHAR(45),
    username VARCHAR(60) NOT NULL,
    time DATETIME NOT NULL,
    amount DECIMAL(10,2),
    FOREIGN KEY (transid) REFERENCES orders(orderid),
    FOREIGN KEY (`username`) REFERENCES `shopping-cart`.`user` (`email`)
);
```

Рис. 3.9. sql-запит створення таблиці кошику

Після цього наша база даних готова для зберігання інформації яку буде надавати сервер. Візуальне представлення нашого сховища даних я представив на Er діаграмі.(Рис. 3.10.)

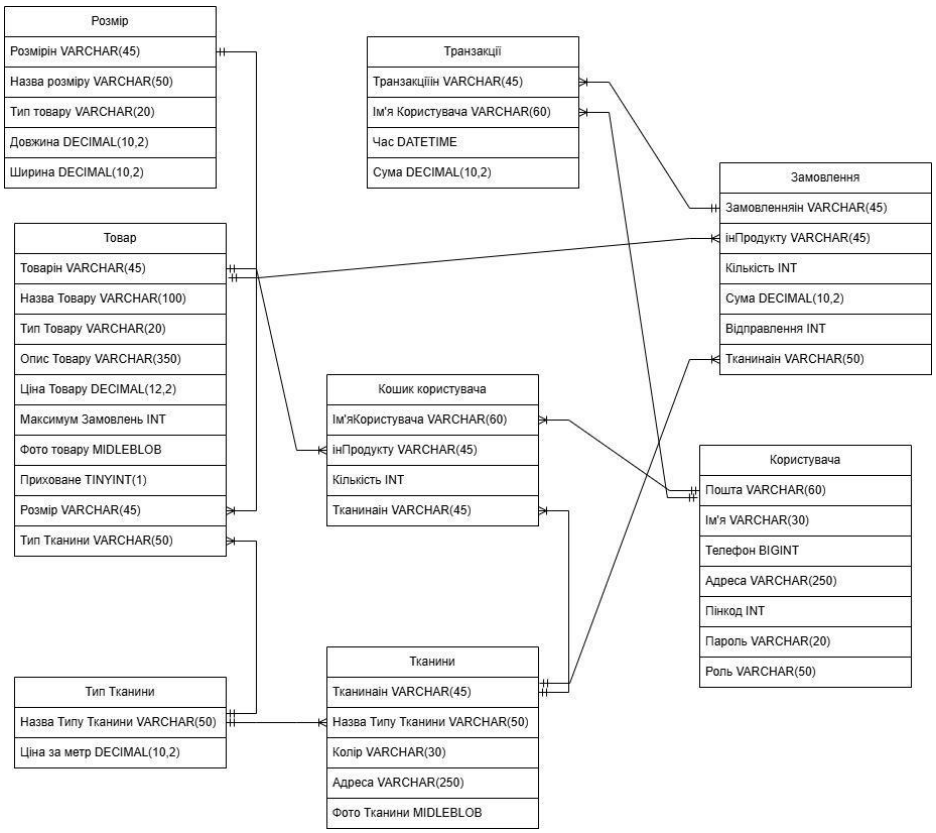

Рис 3.10. Er діаграма

3.2 Розробка серверної частини

На початковому етапі розробки було приділено увагу визначенню структури даних, з якою працюватиме система. Для цього створено пакет .beans, у якому зосереджено класи, такі як ProductBean, UserBean, FabricBean та інші. Вони представляють ключові сутності бізнес-логіки й відповідають структурі таблиць бази даних.

Класи мають приватні поля, публічні гетери й сетери, а також конструктори — як без параметрів, так і параметризовані. Така структура забезпечує інкапсуляцію, зручне створення об'єктів і чітке представлення даних, які зберігаються в базі.(Рис. 3.11.)

```
public class CartBean implements Serializable {

    public CartBean() {
    }

    public String userId;

    public String prodId;

    public int quantity;

    public String fabricid;

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getProdId() {
        return prodId;
    }

    public void setProdId(String prodId) {
        this.prodId = prodId;
    }

    public int getQuantity() {
        return quantity;
    }

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

    public String getFabricId() {
        return fabricid;
    }

    public void setFabricId(String fabricid) {
        this.fabricid = fabricid;
    }

    public CartBean(String userId, String prodId, int quantity, String fabricid) {
        super();
        this.userId = userId;
        this.prodId = prodId;
        this.quantity = quantity;
        this.fabricid=fabricid;
    }

}
```

Рис. 3.11. Код файлу даних кошика

Після визначення структури даних наступним логічним кроком стало формування контрактів для бізнес-логіки. Це реалізовано у пакеті .service, де оголошено інтерфейси, такі як ProductService, UserService та інші. Кожен з них визначає набір операцій, які система може виконувати над відповідними моделями — наприклад, додати товар або отримати користувача за email.

Такий підхід відокремлює опис дій від їх реалізації, що сприяє гнучкості архітектури, підвищує модульність і спрощує заміну реалізацій без впливу на інші компоненти системи.(Рис. 3.12.)

```
public interface CartService {
    String addProductToCart(String userId, String prodId, int prodQty, String fid);

    List<CartBean> getAllCartItems(String userId);

    int getCartCount(String userId);

    String removeProductFromCart(String userId, String prodId, String fid);

    boolean removeAProduct(String userId, String prodId, String fid);

    String updateProductToCart(String userId, String prodId, int prodQty, String fid);

    int getProductCount(String userId, String prodId, String fid);

    int getCartItemCount(String userId, String itemId, String fid);
}
```

Рис. 3.12. Код файлу інтерфейсного класу кошика

Разом з написанням цих інтерфейсів була розробка допоміжних функцій, які знаходяться в пакеті .util. В ньому ми реалізовуємо DBUtil – клас, що буде відповідати за управління з’єднаннями з базою даних через JDBC. Він буде виконувати логіку відкриття, повторного використання та безпечного закриття з’єднань. Щоб він знав з якою базою даних йому з’єднуватись було створено конфігураційний файл application.properties звідки параметри драйвера, URL, логін і пароль. Також в ньому добавлено клас генерації унікальних ідентифікаторів з префіксами, які ми будемо присвоювати продуктам, транзакціям та тканинам.(Рис. 3.13.)

Також було додано файл JavaMailUtil – це клас для відправки електронних листів через SMTP, спрощуючи складні налаштування JavaMail API [12]. Він читає параметри сервера та облікові дані також з application.properties, надаючи методи для формування та відправлення листів.(Рис. 3.14)

```java
public class DBUtil {
    private static Connection conn;

    public DBUtil() {
    }

    public static Connection provideConnection() {
        try {
            if (conn == null || conn.isClosed()) {
                ResourceBundle rb = ResourceBundle.getBundle("application");
                String connectionString = rb.getString("db.connectionString");
                // Додаємо параметри UTF-8 до connection string, якщо їх ще немає
                if (!connectionString.contains("useUnicode") && !connectionString.contains("characterEncoding")) {
                    if (connectionString.contains("?")) {
                        connectionString += "&useUnicode=true&characterEncoding=UTF-8";
                    } else {
                        connectionString += "?useUnicode=true&characterEncoding=UTF-8";
                    }
                }
                String driverName = rb.getString("db.driverName");
                String username = rb.getString("db.username");
                String password = rb.getString("db.password");
                try {
                    Class.forName(driverName);
                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                }
                conn = DriverManager.getConnection(connectionString, username, password);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return conn;
    }

    public static void closeConnection(Connection con) {
    }

    public static void closeConnection(ResultSet rs) {
        try {
            if (rs != null && !rs.isClosed()) {
                try {
                    rs.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void closeConnection(PreparedStatement ps) {
        try {
            if (ps != null && !ps.isClosed()) {
                try {
                    ps.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
```

Рис. 3.13. Код файлу з’єднання з бд

Лише після того, як були визначені інтерфейси бізнес-логіки та створені базові утиліти для роботи з даними та повідомленнями, ми переходимо до реалізації бізнес-логіки в пакеті .service.impl. Тут класи, наприклад ProductServiceImpl, реалізують визначені інтерфейси, об'єднуючи складні операції в одному місці. Саме в цих класах відбувається основна робота: перевірка правильності та повноти вхідних даних, виконання необхідних обчислень і, що найважливіше, взаємодія з базою даних. За допомогою JDBC ці класи формують і виконують SQL-запити, отримуючи дані з бази або записуючи їх у базу. При успішному виконанні операції наприклад відправки замовлення, викликається функція відправки повідомлення користувачу на його пошту. Таким чином, вони є центральною частиною серверної логіки, де бізнес-правила реалізуються та керують роботою додатка.(Рис. 3.15.)

```java
public class JavaMailUtil {
    public static void sendMail(String recipientMailId) throws MessagingException {
        System.out.println("Готуємося до відправки листа");
        Properties properties = new Properties();
        String host = "smtp.gmail.com";

        properties.put("mail.smtp.host", host);
        properties.put("mail.transport.protocol", "smtp");
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.starttls.enable", "true");
        properties.put("mail.smtp.port", "587");
        properties.put("mail.mime.charset", "UTF-8");

        ResourceBundle rb = ResourceBundle.getBundle("application");
        String emailId = rb.getString("mailer.email");
        String passWord = rb.getString("mailer.password");

        Session session = Session.getInstance(properties, new Authenticator() {
        });

        Message message = prepareMessage(session, emailId, recipientMailId);
        Transport.send(message);
        System.out.println("Лист успішно відправлено!");
    }

    private static Message prepareMessage(Session session, String myAccountEmail, String recipientEmail) {
        try {
            MimeMessage message = new MimeMessage(session);
            message.setFrom(new InternetAddress(myAccountEmail));
            message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipientEmail));

            // Встановлення теми з кодуванням UTF-8
            message.setSubject("Ласкаво просимо до TextileShop", "UTF-8");

            // Встановлення тексту з кодуванням UTF-8
            message.setText("Привіт, " + recipientEmail + "! Дякуємо за реєстрацію у нашому сервісі!", "UTF-8");

            return message;
        } catch (Exception exception) {
            Logger.getLogger(JavaMailUtil.class.getName()).log(Level.SEVERE, null, exception);
        }
        return null;
    }

    protected static void sendMail(String recipient, String subject, String htmlTextMessage) throws MessagingException {
        System.out.println("Готуємося до відправки листа");
        Properties properties = new Properties();
        String host = "smtp.gmail.com";

        properties.put("mail.smtp.host", host);
        properties.put("mail.transport.protocol", "smtp");
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.starttls.enable", "true");
        properties.put("mail.smtp.port", "587");
        properties.put("mail.mime.charset", "UTF-8");

        ResourceBundle rb = ResourceBundle.getBundle("application");
        String emailId = rb.getString("mailer.email");
        String passWord = rb.getString("mailer.password");

        Session session = Session.getInstance(properties, new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(emailId, passWord);
            }
        });

        Message message = prepareMessage(session, emailId, recipient, subject, htmlTextMessage);
```

Рис. 3.14. Код файлу відправлення повідомлень

Завершаючий файлом структури серверної структури, що всю внутрішню логіку з зовнішнім з інфтерфейсом являється пакет .srv. Тут знаходяться сервлети, які виступають у ролі контролерів у архітектурі MVC. Вони є точками входу для HTTP-запитів, що надходять від браузеру. Вони обробляють запити, витягують необхідні параметри, встановлюють кодування для коректної роботи з даними, взаємодіють із сесією користувача та виконують валідацію і авторизацію.

Після цього сервлети викликають відповідні методи з раніше розроблених сервісних шарів .service.impl для виконання бізнес-операцій, таких як CRUD-операці товарів, зміну статусу замовлення, реєстрація користувачів, обробка замовлень, логін користувача та інше. В разі

необхідності, сервлети формують атрибути для передачі даних, які потім передаються до JSP-сторінок, що відповідають за представлення. JSP-сторінки використовують ці дані для динамічного формування HTML-відповідей [13], які відправляються назад клієнту.(Рис. 3.16.)

Таким чином, сервлети забезпечують зв’язок між користувацьким інтерфейсом і бізнес-логікою системи, керуючи повним циклом обробки запитів — від прийому параметрів до відображення результатів, що робить серверну частину системи "TextileShop" ефективною та структурованою.

```
public class ProductServiceImpl implements ProductService {

    @Override
    public String addProduct(String prodName, String prodType, String prodInfo, double prodPrice, int prodQuantity,
            InputStream prodImage, String size, String fabricType) {
        String status = null;
        String prodId = IDUtil.generateId();

        ProductBean product = new ProductBean(prodId, prodName, prodType, prodInfo, prodPrice, prodQuantity, prodImage, false, size, fabricType);

        status = addProduct(product);

        return status;
    }

    @Override
    public String addProduct(ProductBean product) {
        String status = "Product Registration Failed!";

        if (product.getProdId() == null)
            product.setProdId(IDUtil.generateId());

        Connection con = DBUtil.provideConnection();
        PreparedStatement ps = null;

        try {
            ps = con.prepareStatement("insert into product values(?,?,?,?,?,?,?,?,?,?)");
            ps.setString(1, product.getProdId());
            ps.setString(2, product.getProdName());
            ps.setString(3, product.getProdType());
            ps.setString(4, product.getProdInfo());
            ps.setDouble(5, product.getProdPrice());
            ps.setInt(6, product.getProdQuantity());
            ps.setBlob(7, product.getProdImage());
            ps.setBoolean(8, product.getHide());
            ps.setString(9, product.getSize());
            ps.setString(10, product.getFabricType());

            int k = ps.executeUpdate();

            if (k > 0) {
                status = "Product Added Successfully with Product Id: " + product.getProdId();
            }

        } catch (SQLException e) {
            status = "Error: " + e.getMessage();
            e.printStackTrace();
        } finally {
            DBUtil.closeConnection(con);
            DBUtil.closeConnection(ps);
        }

        return status;
    }

    public String getSizeNameById(String sizeId) {
        if (sizeId == null || sizeId.isEmpty()) {
            return "-";
        }

        SizeBean size = new SizeServiceImpl().getSizeDetails(sizeId);
        return size != null ? size.getSizeName() : sizeId;
    }
```

Рис. 3.15. Код файлу реалізації класу товару

```
public class AddProductSrv extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Встановлюємо кодування UTF-8 для запиту та відповіді
        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");

        HttpSession session = request.getSession();
        String userType = (String) session.getAttribute("usertype");
        String userName = (String) session.getAttribute("username");
        String password = (String) session.getAttribute("password");

        if (userType == null || !userType.equals("admin")) {
            response.sendRedirect("login.jsp?message=Access Denied!");
            return;
        }
        else if (userName == null || password == null) {
            response.sendRedirect("login.jsp?message=Session Expired, Login Again to Continue!");
            return;
        }

        String status = "Product Registration Failed!";
        String prodName = request.getParameter("name");
        String prodType = request.getParameter("type");
        String prodInfo = request.getParameter("info");
        double prodPrice = Double.parseDouble(request.getParameter("price"));
        int prodQuantity = Integer.parseInt(request.getParameter("quantity"));
        String size = request.getParameter("size"); // New parameter
        String fabricType = request.getParameter("fabricType"); // New parameter

        Part part = request.getPart("image");
        InputStream prodImage = part.getInputStream();

        ProductServiceImpl product = new ProductServiceImpl();
        status = product.addProduct(prodName, prodType, prodInfo, prodPrice, prodQuantity, prodImage, size, fabricType);

        RequestDispatcher rd = request.getRequestDispatcher("addProduct.jsp?message=" + status);
        rd.forward(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

Рис. 3.16. Код файлу HTTP-запиту на додавання товару

3.3 Розробка інтерфейсної яастини

Перед початком розробки інтерфейсної частини проекту я зайнявся налаштуванням середовища, щоб забезпечити зручну та ефективну роботу з веб-сторінками. Для цього обрав фреймворк Bootstrap [14], який допомагає швидко створювати стильний і адаптивний дизайн.(Рис. 3.17.)

За допомогою CDN Google спеціальні сервери, які швидко доставляють необхідні ресурси з сервера при відтворенні сторінки завантажуватиметься фреймворк js та css файли, так само і завантажується jQuery він є необхідним для роботи інтерактивних компонентів Bootstrap. Потім в кожній JSP-сторінці в розділі <head> через тег <link> я підключав бібліотеку допоміжну а потім після неї локальний файл стилів Bootstrap, вказавши відносний шлях до файлу CSS у папці проекту. Тепер в нас підключено все необхідне для розробки інтерфейсної частини і ми можемо переходити до реалізації сторінок нашої інформаційної системи.

```
<script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
<style>
```

Рис. 3.17.Скріпт сторінці який активує під завантаження фрейсворка

Першу сторінку яку потрібно було реалізувати це була головна сторінка де відображався б товар, тому я почав робити, щоб вона відображала динамічно додані та показані товарів в групі. Для отримання даних про товари використовується об'єкт products, який формується на сервері у ProductServiceImpl, а інформація про доступні тканини попередньо обробляється у fabricsByType, що теж формується серверною логікою до завантаження сторінки.

У верхній частині реалізовано пошукову форму, яка надсилає GET-запит на ту ж саму сторінку index.jsp із параметром search. Якщо параметри type або search не задані, користувачу відображається список доступних категорій товарів, який будується на основі списку categories.

Коли ж визначено type або search, формується детальний список товарів. Дані групуються в groupedProducts, де використовується вкладена структура Map<String, Map<String, List<ProductBean>>>, що дозволяє поєднувати товари з однаковою назвою і типом тканини.(Рис. 3.18.) Відповідно до цього на сторінці відображаються блоки, що містять перший товар як представника, а також доступні варіанти за розміром і тканиною. Для кожного товару виводиться зображення через ShowImage сервлет, опис, ціна, та доступні тканини через allFabricsData.

Вибір тканини реалізовано динамічно: натискання на варіант тканини змінює приховані input-поля selectedFabric та form-fabric-id-input, які використовуються під час розміщення форми для додавання товару в кошик. Додатково використовується об’єкт sharedSelectedFabrics для збереження обраної тканини окремо для кожної групи товарів.

Для дій із товарами реалізовано обробку подій форми. При спробі надіслати форму виконується перевірка, чи користувач авторизований перевіряється userName != null. Якщо ні — виконується переадресування на

сторінку входу з параметром returnUrl, щоб після авторизації користувач міг повернутися до цієї сторінки. Якщо ж користувач авторизований, дані форми передаються методом POST на сервлет AddtoCart, який обробляє дію в залежності від натиснутої кнопки.

```
<div class="container" style="margin-top: 25px;">
    <% if (type != null) { %>
        <div class="back-to-categories">
            <a href="index.jsp" class="btn btn-primary">
                <span class="glyphicon glyphicon-arrow-left"></span> Повернутись до категорій
            </a>
        </div>
    <% } %>

    <div class="message-title text-center">
        <%=message%>
    </div>

    <% if (type == null && search == null) { %>
        <div class="row">
            <% for (String category : categories) { %>
                <div class="col-md-4 col-sm-6">
                    <div class="category-tile" onclick="window.location.href='index.jsp?type=<%=category%>'"
                         data-category="<%=category%>">
                        <h3><%=category%></h3>
                    </div>
                </div>
            <% } %>
        </div>
    <% } else { %>
        <div class="product-list">
            <%
            // Групуємо товари спочатку за назвою, потім за типом тканини
            // Використовуємо LinkedHashMap для збереження порядку додавання
            Map<String, Map<String, List<ProductBean>>> groupedProducts = new LinkedHashMap<>();

            for (ProductBean product : products) {
                groupedProducts
                    .computeIfAbsent(product.getProdName(), k -> new LinkedHashMap<>())
                    .computeIfAbsent(product.getFabricType(), k -> new ArrayList<>())
                    .add(product);
            }

            for (Map.Entry<String, Map<String, List<ProductBean>>> nameEntry : groupedProducts.entrySet()) {
                String productName = nameEntry.getKey();
                Map<String, List<ProductBean>> productsByFabricType = nameEntry.getValue();

                for (Map.Entry<String, List<ProductBean>> fabricTypeEntry : productsByFabricType.entrySet()) {
                    String fabricTypeName = fabricTypeEntry.getKey();
                    List<ProductBean> sameNameAndFabricProducts = fabricTypeEntry.getValue();
                    ProductBean firstProduct = sameNameAndFabricProducts.get(0); // Беремо перший товар для відображення загальної інформації
                    List<SizeBean> sizes = sizeService.getSizesByProductType(firstProduct.getProdType());
            %>
            <div class="product-row" data-product-name="<%=firstProduct.getProdName()%>" data-fabric-type-group="<%=fabricTypeName%>">
                <div class="product-image-container">
                    <img src="./ShowImage?pid=<%=firstProduct.getProdId()%>" alt="<%=firstProduct.getProdName()%>" class="product-image">
                </div>

                <div class="product-info-container">
                    <h3 class="product-title"><%=firstProduct.getProdName()%></h3>

                    <div class="description-selector">
                        <span class="option-label">Оберіть варіант:</span>
                        <div class="description-options">
                            <% for (int i = 0; i < sameNameAndFabricProducts.size(); i++) {
                                ProductBean product = sameNameAndFabricProducts.get(i);
                                boolean isOutOfStock = product.getProdQuantity() <= 0;
                                String sizeName = prodDao.getSizeNameById(product.getSize());
                            %>
                            <div class="description-option <%= i == 0 ? "selected" : "" %> <%= isOutOfStock ? "out-of-stock" : "" %>"
                                 data-product-id="<%=product.getProdId()%>"
```

Рис. 3.18. Частка коду головної сторінки відображення товару

Наступною була створена сторінка з полями для логіна, що має виконувати функціонал авторизації користувача. Вона містить два поля, під електронну пошту та пароль, які користувач заповнює для входу. Після натискання кнопки «Увійти» дані відправляються методом POST на сервлет LoginSrv, який відповідає за обробку авторизації. Цей сервлет перевіряє надані облікові дані з базою даних, через UserService. Якщо авторизація успішна, в сесію зберігається інформація про користувача, його тип usertype та ім’я користувача username, і виконується перенаправлення на відповідну домашню сторінку, для покупця це index.jsp, адміністратора adminViewProduct.jsp.(Рис. 3.18.)

Далі перевіряється наявність параметра message у запиті request.getParameter("message") він використовується для виводу повідомлень, наприклад, про невдалий вхід або інші повідомлення сервера, які повертає сервлет. У разі помилкових даних сервлет перенаправляє користувача назад на цю сторінку з параметром message, який і відображається у верхній частині форми. Також реалізовано посилання для переходу на сторінку реєстрації register.jsp, що дозволяє користувачам, які ще не мають облікового запису, створити новий.

```
</head>
<body>

<%@ include file="header.jsp"%>

<%
String message = request.getParameter("message");
%>

<div class="container">
    <div class="login-container">
        <div class="login-header">
            <h2>Вхід до системи</h2>
            <p class="text-muted">Будь ласка, введіть ваші облікові дані</p>

            <%
            if (message != null) {
            %>
            <div class="message-box">
                <%=message%>
            </div>
            <%
            }
            %>
        </div>

        <form action="./LoginSrv" method="post">
            <div class="form-group">
                <label for="username">Електронна пошта</label>
                <input type="email" class="form-control" id="username"
                       name="username" placeholder="Ваша електронна пошта" required>
            </div>

            <div class="form-group">
                <label for="password">Пароль</label>
                <input type="password" class="form-control" id="password"
                       name="password" placeholder="Ваш пароль" required>
            </div>

            <div class="form-group text-center">
                <button type="submit" class="btn btn-primary btn-login">Увійти</button>
                <p class="text-muted mt-3">Ще не зареєстровані?
                    <a href="register.jsp" style="color: #3498db;">Створити акаунт</a>
                </p>
            </div>
        </form>
    </div>
</div>

<%@ include file="footer.html"%>

</body>
</html>
```

Рис. 3.18. Код сторінки авторизації

Для можливості зареєструватись користувачу, була створена ця сторінка. Після введення даних форма відправляється методом POST на сервлет RegisterSrv, який відповідає за обробку введених даних, перевірку коректності введених паролей та на унікальність пошти, якщо все підходе то зберігає нового користувача в базу даних. Для цього сервлет використовує

сервісний клас, UserService який виконує вставку даних у таблицю користувачів.(Рис. 3.19.)

Форма включає кілька полів: ім’я користувача, електронна пошта, адреса доставки, номер телефону, поштовий індекс, пароль і його підтвердження. На стороні сервера в сервлеті додатково перевіряється, що пароль містить щонайменше 6 символів і збігається з полем підтвердження. У випадку успішної реєстрації сервлет перенаправляє користувача на сторінку входу login.jsp. У разі невдачі на цю ж сторінку з відповідним повідомленням через параметр message.

```
            <%
            if (message != null) {
            %>
            <div class="message-box">
                <%=message%>
            </div>
            <%
            }
            %>
        </div>

        <form action="./RegisterSrv" method="post">
            <div class="row">
                <div class="col-md-6 form-group">
                    <label for="username">Ім'я користувача</label>
                    <input type="text" class="form-control" id="username"
                           name="username" placeholder="Ваше ім'я" required>
                </div>
                <div class="col-md-6 form-group">
                    <label for="email">Електронна пошта</label>
                    <input type="email" class="form-control" id="email"
                           name="email" placeholder="example@example.com" required>
                </div>
            </div>

            <div class="form-group">
                <label for="address">Адреса доставки</label>
                <textarea class="form-control" id="address"
                          name="address" placeholder="Вулиця, будинок, квартира" required></textarea>
            </div>

            <div class="row">
                <div class="col-md-6 form-group">
                    <label for="mobile">Номер телефону</label>
                    <input type="tel" class="form-control" id="mobile"
                           name="mobile" placeholder="+380XXXXXXXXX" required>
                </div>
                <div class="col-md-6 form-group">
                    <label for="pincode">Поштовий індекс</label>
                    <input type="text" class="form-control" id="pincode"
                           name="pincode" placeholder="XXXXX" required>
                </div>
            </div>

            <div class="row">
                <div class="col-md-6 form-group">
                    <label for="password">Пароль</label>
                    <input type="password" class="form-control" id="password"
                           name="password" placeholder="Не менше 6 символів" required>
                    <p class="password-hint">Пароль має містити щонайменше 6 символів</p>
                </div>
                <div class="col-md-6 form-group">
                    <label for="confirmPassword">Підтвердження паролю</label>
                    <input type="password" class="form-control" id="confirmPassword"
                           name="confirmPassword" placeholder="Повторіть пароль" required>
                </div>
            </div>

            <div class="row" style="margin-top: 20px;">
                <div class="col-md-6" style="margin-bottom: 15px;">
                    <button type="reset" class="btn btn-reset">Очистити форму</button>
                </div>
                <div class="col-md-6">
                    <button type="submit" class="btn btn-register">Зареєструватися</button>
                </div>
            </div>

            <div class="text-center" style="margin-top: 20px;">
                <p class="text-muted">Вже є акаунт?
                    <a href="login.jsp" style="color: #3498db;">Увійти</a>
                </p>
            </div>
        </form>
    </div>
</div>
```

Рис. 3.19. Код форми реєстрації

Було реалізовано динамічну навігаційну панель, яка залежить від типу користувача, що зберігається в атрибуті сесії usertype. Відповідно до значення

цього атрибута, здійснюється розгалуження логіки відображення: якщо користувач неавторизований, йому пропонується лише перегляд товарів за index.jsp та можливість увійти до системи через login.jsp. У разі авторизації як "customer", система отримує ім’я користувача з сесії session.getAttribute("username") і викликає метод getCartCount() з класу CartServiceImpl, який повертає кількість товарів у кошику. Це число використовується для виведення відповідної іконки кошика з badge-лічильником. Клік по кошику веде на сторінку cartDetails.jsp, на замовлення orderDetails.jsp, на профіль користувача userProfile.jsp та вихід через сервлет LogoutSrv, який очищує сесію користувача.(Рис. 3.20.)

Якщо користувач має роль адміністратора, то відображається панель, яка орієнтована на керування товарами та замовленнями. Тут реалізовано доступ до таких сторінок, як adminStock.jsp перегляд та керування товарами, viewFabrics.jsp перелік доступних тканин, viewFabricTypes.jsp типи тканин, viewSizes.jsp доступні розміри. Також передбачено доступ до перегляду вже відправлених замовлень shippedItems.jsp і необроблених замовлень unshippedItems.jsp функціонал виходу аналогічний через LogoutSrv.

```
} else if ("customer".equalsIgnoreCase(userType)) { //Заголовок для покупця

int notf = new CartServiceImpl().getCartCount((String) session.getAttribute("username"));
%>
<nav class="navbar navbar-default navbar-fixed-top">

    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse"
                data-target="#myNavbar">
                <span class="icon-bar"></span> <span class="icon-bar"></span> <span
                    class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="userHome.jsp"><span
                class="glyphicon glyphicon-home"> </span>Textileshop</a>
        </div>

        <div class="collapse navbar-collapse" id="myNavbar">
            <ul class="nav navbar-nav navbar-right">
                <li><a href="userHome.jsp"><span>Товари</span></a></li>
                <%
                if (notf == 0) {
                %>
                <li><a href="cartDetails.jsp"> <span
                        class="glyphicon glyphicon-shopping-cart"></span>Кошик
                </a></li>

                <%
                } else {
                %>
                <li><a href="cartDetails.jsp"
                    style="margin: 0px; padding: 0px;" id="mycart"><i
                        data-count="<%=notf%>"
                        class="fa fa-shopping-cart fa-3x icon-white badge"
                        style="background-color: #0066cc; margin: 0px; padding: 0px; padding-bottom: 0px; padding-top: 5px;">
                    </i></a></li>
                <%
                }
                %>
                <li><a href="orderDetails.jsp">Замовлення</a></li>
                <li><a href="userProfile.jsp">Профіль</a></li>
                <li><a href="./LogoutSrv">Вийти</a></li>
            </ul>
        </div>
    </div>
</nav>
<%
```

Рис. 3.20. Частина кода навігаціної панелі «виведення даних для покупця»

Після того як авторизацію було реалізовано, я вирішив зробити сторінку кошика, на ній я спочатку створюю об'єкти класів CartServiceImpl, ProductServiceImpl і FabricServiceImpl, які відповідають за отримання даних про товари, тканини та сам кошик. Метод getAllCartItems(userName) класу CartServiceImpl викликається з метою отримання списку всіх позицій у кошику поточного користувача. Отримані об’єкти типу CartBean містять інформацію про товар, тканину та кількість.

Далі через ProductServiceImpl викликається метод getProductDetails(), який повертає ProductBean з інформацією про товар, а також викликається метод getFabricDetails() з FabricServiceImpl, що повертає FabricBean з даними про обрану тканину. При заповненні таблиці відображаються всі позиції в кошику: зображення товару та тканини яке завантажується через сервлет ShowImage із параметрами pid або fabricId, назва товару, обраний колір тканини, ціна, кількість і сума.

Форма також реалізує зміну кількості товару без перезавантаження всієї сторінки. Для цього використовуються посилання на цю ж JSP-сторінку cartDetails.jsp, із параметром add=0 для зменшення кількості або add=1 для збільшення. Ці параметри передаються разом з ідентифікатором користувача, товару, тканини, наявною кількістю товару та поточною кількістю у кошику. Зміна кількості обробляється у тій же сторінці, де логіка перевіряє значення параметра add і відповідно оновлює запис у базі даних через CartServiceImpl.

У нижній частині таблиці автоматично обчислюється і виводиться загальна сума до сплати. Якщо в кошику є товари, з’являються кнопки для продовження покупок перехід на index.jsp або переходу до оплати перенаправлення на payment.jsp, куди передається сума через параметр amount.(Рис. 3.21.)

Рис. 3.21. Код виведення товару в кошику

Так як можливість замовлення товару було реалізовано, настав час реалізувати відображення замовлень користувача. Це було реалізовано таким чином, що при завантаженні сторінки здійснюється перевірка автентифікації: із сесії отримуються параметри username та password, і якщо хоча б один із них

відсутній, виконується переадресація на login.jsp із повідомленням про завершення сесії. Це забезпечує захист доступу до персональних даних користувача.

Після перевірки автентифікації створюються екземпляри сервісів OrderServiceImpl і FabricServiceImpl, які забезпечують доступ до відповідних даних у базі. Через метод getAllOrderDetails(userName) об'єкта OrderServiceImpl отримується список усіх замовлень користувача у вигляді списку об’єктів OrderDetails.

Кожен об’єкт OrderDetails містить інформацію про товар, кількість, суму, час оформлення, стан доставки, а також ідентифікатор тканини. Для кожного замовлення у циклі здійснюється визначення статусу, вони можуть бути тільки в тьох: замовлено, відправлено, скасовано відповідно до числового поля shipped. Визначений текстовий статус і відповідний клас оформлення text-success, text-primary, text-danger використовуються для наочного відображення стану кожного замовлення.(Рис. 3.22.)

```
<jsp:include page="header.jsp" />

<div class="text-center"
    style="color: green; font-size: 24px; font-weight: bold;">
    Деталі замовлення
</div>

<div class="container">
    <div class="table-responsive">
        <table class="table table-hover table-sm">
            <thead style="background-color: black; color: white; font-size: 14px; font-weight: bold;">
                <tr>
                    <th>Зображення товару</th>
                    <th>Назва товару</th>
                    <th>ID замовлення</th>
                    <th>Кількість</th>
                    <th>Ціна</th>
                    <th>Тканина</th> <%-- Змінено заголовок --%>
                    <th>Час</th>
                    <th>Статус</th>
                </tr>
            </thead>
            <tbody style="background-color: white; font-size: 15px; font-weight: bold;">
                <%
                for (OrderDetails order : orders) {
                    String statusText;
                    String statusClass;
                    if (order.getShipped() == 0) {
                        statusText = "ЗАМОВЛЕНО";
                        statusClass = "text-success"; // Зелений колір
                    } else if (order.getShipped() == 1) {
                        statusText = "ВІДПРАВЛЕНО";
                        statusClass = "text-primary"; // Синій колір
                    } else if (order.getShipped() == 2) {
                        statusText = "СКАСОВАНО";
                        statusClass = "text-danger"; // Червоний колір
                    } else {
                        statusText = "НЕВІДОМО";
                        statusClass = "text-muted"; // Сірий колір
                    }

                    // Отримуємо інформацію про тканину
                    FabricBean fabric = null;
                    if (order.getFabricId() != null && !order.getFabricId().isEmpty()) {
                        fabric = fabricDao.getFabricDetails(order.getFabricId());
                    }
                %>
                <tr>
                    <td><img src="./ShowImage?pid=<%=order.getProductId()%>" style="width: 50px; height: 50px;"></td>
                    <td><%=order.getProdName()%></td>
                    <td><%=order.getOrderId()%></td>
                    <td><%=order.getQty()%></td>
                    <td><%=order.getAmount()%></td>
                    <td class="fabric-info-cell"> <%-- Новий клас для стилізації --%>
                        <% if (fabric != null) { %>
                            <img src="./ShowImage?fabricId=<%=fabric.getFabricId()%>" alt="<%=fabric.getColor()%>" class="fabric-order-thumbnail">
                            <%=fabric.getColor()%>
                        <% } else { %>
                            - <%-- або можна вивести "Тканина не вказана" --%>
                        <% } %>
                    </td>
                    <td><%=order.getTime()%></td>
                    <td class="<%=statusClass%>"><%=statusText%></td>
                </tr>
                <%
                }
                %>
```

Рис. 3.22. Код таблиці замовлень користувача

Додатково через сервіс FabricServiceImpl викликається метод getFabricDetails(), який повертає об’єкт FabricBean, з якого витягується колір тканини та зображення. Це дозволяє користувачу бачити обраний колір тканини разом із товаром.

Таблиця відображає всю інформацію у зручному для користувача форматі: зображення товару завантажується через сервлет ShowImage з параметром pid, назву товару, ідентифікатор замовлення, кількість, суму, тканину, дату й час замовлення та його поточний статус.

Для покупців реалізовані всі інфтерфейси, тепер для адміністратора реалізація сторінки для управління товарами. На початку сторінки виконується перевірка прав доступу користувача шляхом зчитування

атрибутів сесії: usertype, username та password. Якщо користувач не є адміністратором або відсутні дані сесії, відбувається перенаправлення на сторінку авторизації через метод response.sendRedirect(). Таким чином реалізується базовий захист адміністративної частини системи. Якщо користувач проходить перевірку, тоді здійснюється ініціалізація сервісу ProductServiceImpl, з якого за допомогою методу getAllProductsadmin() отримується повний перелік товарів, що доступні для адміністратора. Після цього список сортується в мапу productsByCategory, де ключем є категорія, а значенням список об'єктів ProductBean.(Рис. 3.23.)

Для кожної категорії виводиться окрема таблиця з товарами. Кожен рядок таблиці представляє окремий товар, в якому відображається його зображення через сервлет ShowImage з параметром pid, ID, назва, ціна, тип тканини, розмір який ми отримуємо з функції productService.getSizeNameById(), визначення кількість проданих одиниць через OrderServiceImpl().countSoldItem(), кількість на складі, статус видимості та кнопки для редагування і видалення.

Кнопка редагування веде на сторінку updateProduct.jsp, передаючи ID товару як параметр, а кнопка видалення викликає сервлет RemoveProductSrv, що обробляє запит на видалення товару з бази даних. Перед видаленням реалізовано підтвердження дії через стандартний JavaScript confirm.

```
<%
/* Перевірка авторизації адміністратора */
String userType = (String) session.getAttribute("usertype");
String userName = (String) session.getAttribute("username");
String password = (String) session.getAttribute("password");

if (userType == null || !userType.equals("admin")) {
    response.sendRedirect("login.jsp?message=Доступ заборонено! Увійдіть як адміністратор");
}
else if (userName == null || password == null) {
    response.sendRedirect("login.jsp?message=Сесія закінчилась, увійдіть знову");
}
%>

<jsp:include page="header.jsp" />

<div class="container">
    <div class="stock-container">
        <div class="stock-header">
            <h2>Склад товарів</h2>
            <p class="text-muted">Перегляд та управління товарами</p>
        </div>

        <div class="text-right" style="margin-bottom: 20px;">
            <a href="addProduct.jsp" class="btn btn-success btn-add">
                <span class="glyphicon glyphicon-plus"></span> Додати новий товар
            </a>
        </div>

        <%
        ProductServiceImpl productDao = new ProductServiceImpl();
        List<ProductBean> allProducts = productDao.getAllProductsadmin();

        // Групуємо товари за категоріями
        Map<String, List<ProductBean>> productsByCategory = new HashMap<>();

        for (ProductBean product : allProducts) {
            String category = product.getProdType().toUpperCase();
            if (!productsByCategory.containsKey(category)) {
                productsByCategory.put(category, new ArrayList<>());
            }
            productsByCategory.get(category).add(product);
        }

        if (productsByCategory.isEmpty()) {
        %>
        <div class="no-items text-center">
            Немає товарів
        </div>
        <%
        } else {
            for (Map.Entry<String, List<ProductBean>> entry : productsByCategory.entrySet()) {
                String category = entry.getKey();
                List<ProductBean> products = entry.getValue();
        %>
        <div class="category-section">
            <div class="category-title"><%=category%></div>

            <div class="table-responsive stock-table">
                <table class="table table-hover">
<thead>
    <tr>
        <th>Зображення</th>
        <th>ID</th>
        <th>Назва</th>
        <th>Ціна (грн)</th>
        <th>Тип тканини</th>
        <th>Розмір</th>
        <th>Продано</th>
        <th>Можливо замовити</th>
        <th>Статус</th>
        <th>Дії</th>
    </tr>
```

Рис. 3.23. Частка коду сторінки управління товарами

Оснвоний функціонал сторінки оновлення реалізований через виклик методів сервісних класів. Дані про товар отримуються за допомогою ProductServiceImpl.getProductDetails(prodid), що повертає об’єкт ProductBean із усією необхідною інформацією для заповнення форми. Для формування випадаючих списків із розмірами та типами тканин викликаються методи getAllSizes() і getAllFabricTypes() відповідно з класів SizeServiceImpl та FabricTypeServiceImpl.

У формі редагування товару відображається поточне зображення товару, яке підвантажується через сервлет ShowImage із параметром pid, що ідентифікує товар. Користувач має змогу змінити назву, тип, опис, ціну, кількість на складі, розмір, тип тканини та статус видимості товару. Також реалізована можливість завантаження нового зображення, при цьому старе збережеться, якщо файл не вибрано.

Після внесення змін форма надсилається методом POST на сервлет UpdateProductSrv, який обробляє оновлення інформації в базі даних. Перед цим у клієнта проходить базова перевірка введених даних. Повідомлення про успішне оновлення або помилку передаються назад через параметри message або error і відображаються відповідними блоками на сторінці.(Рис. 3.24.)

```
</div>

<input type="hidden" name="pid" value="<%=product.getProdId()%>">

<div class="form-group">
    <label for="productName">Назва товару *</label>
    <input type="text" placeholder="Введіть назву товару"
        name="name" class="form-control"
        value="<%=product.getProdName() != null ? product.getProdName() : ""%>"
        id="productName" required>
</div>

<div class="form-group">
    <%
    String ptype = product.getProdType() != null ? product.getProdType() : "Інші товари";
    %>
    <label for="producttype">Тип товару *</label>
    <select name="type" id="producttype" class="form-control" required>
        <option value="Подушка" <%="Подушка".equalsIgnoreCase(ptype) ? "selected" : ""%>>Подушка</option>
        <option value="Постільна білизна" <%="Постільна білизна".equalsIgnoreCase(ptype) ? "selected" : ""%>>Постільна білизна</option>
        <option value="Ковдра" <%="Ковдра".equalsIgnoreCase(ptype) ? "selected" : ""%>>Ковдра</option>
        <option value="Плед" <%="Плед".equalsIgnoreCase(ptype) ? "selected" : ""%>>Плед</option>
        <option value="Рушники" <%="Рушники".equalsIgnoreCase(ptype) ? "selected" : ""%>>Рушники</option>
        <option value="Крісла" <%="Крісла".equalsIgnoreCase(ptype) ? "selected" : ""%>>Крісла</option>
    </select>
</div>

<div class="form-group">
    <label for="productDescription">Опис товару *</label>
    <textarea name="info" class="form-control"
        id="productDescription" required><%=product.getProdInfo() != null ? product.getProdInfo() : ""%></textarea>
</div>

<div class="row">
    <div class="col-md-4 form-group">
        <label for="productPrice">Ціна за одиницю (₴) *</label>
        <input type="number" step="0.01" min="0.01"
            value="<%=product.getProdPrice()%>"
            placeholder="Введіть ціну" name="price"
            class="form-control" id="productPrice" required>
        <div class="error-message" id="priceError"></div>
    </div>
    <div class="col-md-4 form-group">
        <label for="productQuantity">Кількість яку можуть замовити *</label>
        <input type="number" min="0"
            value="<%=product.getProdQuantity()%>"
            placeholder="Введіть кількість" class="form-control"
            id="productQuantity" name="quantity" required>
        <div class="error-message" id="quantityError"></div>
    </div>
    <div class="col-md-4 form-group">
        <label for="productSize">Розмір *</label>
        <select class="form-control" id="productSize" name="size" required>
            <% for (SizeBean size : sizes) { %>
                <option value="<%= size.getSizeId() %>"
                    <%= size.getSizeId().equals(product.getSize()) ? "selected" : "" %>>
                    <%= size.getSizeName() %> (<%= size.getLength() %>x<%= size.getWidth() %>)
                </option>
            <% } %>
        </select>
    </div>
</div>

<div class="row">
    <div class="col-md-6 form-group">
        <label for="fabricType">Тип тканини *</label>
        <select class="form-control" id="fabricType" name="fabricType" required>
            <% for (FabricTypeBean fabricType : fabricTypes) { %>
                <option value="<%= fabricType.getFabricTypeName() %>"
                    <%= fabricType.getFabricTypeName().equals(product.getFabricType()) ? "selected" : "" %>>
                    <%= fabricType.getFabricTypeName() %>
                </option>
            <% } %>
        </select>
```

Рис. 3.24. Частина коду форми оновлення товару

Видалення спрацьовує автоматично, адже через кнопку передається id товару надсилає ці дані методом POST на сервлет RemoveProductSrv. Даний сервлет обробляє запит, зчитує передане значення prodid, звертається до

відповідного сервісного шару ProductServiceImpl та видаляє товар, та повернення користувач назад до товарів.

Сторінка додавання товару так само спочатку перевіряє авторизованість користувача через атрибути сесії. Після проходження авторизації ініціалізуються сервіси ProductServiceImpl, SizeServiceImpl та FabricTypeServiceImpl, які відповідають за взаємодію з базою даних. За допомогою їхніх методів getAllProductsadmin(), getAllSizes() та getAllFabricTypes() відбувається отримання списків усіх товарів, доступних розмірів та типів тканин відповідно. Оскільки адміністратор може ввести нову позицію, яка вже існує в системі, я реалізував додатковий механізм фільтрації унікальних назв товарів через HashSet, щоб забезпечити підказки у формі.

Основна частина сторінки представлена HTML-формою з полями введення назви, категорії, опису, ціни, кількості, розміру, типу тканини та зображення товару. Значення з полів після заповнення відправляються методом POST на сервлет AddProductSrv, який відповідає за обробку введених даних, збереження товару у базу даних та збереження зображення на сервері. Передбачено також повідомлення, яке відображається у разі успішного додавання або помилки, та передається через параметр message.

Окрему увагу було приділено динамічній фільтрації списку розмірів залежно від вибраного типу товару. Для цього використовується JavaScript-функція updateSizesByType, яка на основі значення data-type елементів <option> у селекті product_size приховує або відображає доступні варіанти. Цей скрипт активується як при зміні типу товару, так і при завантаженні сторінки.(3.4.10)

Також я реалізовував і для інших даних сторінки з функціями створення, оновлення та видалення, та реалізація в них така сама, Тільки замість ProductServiceImpl використовується спеціалізований під той тип даних.

```
<script>
function updateSizesByType(productType) {
    var sizeSelect = document.getElementById('product_size');
    var options = sizeSelect.options;

    // Show all options if no specific type is selected
    if (!productType) {
        for (var i = 0; i < options.length; i++) {
            options[i].style.display = '';
        }
        return;
    }

    // Filter sizes by product type
    for (var i = 0; i < options.length; i++) {
        var option = options[i];
        if (option.getAttribute('data-type') === productType) {
            option.style.display = '';
        } else {
            option.style.display = 'none';
        }
    }

    // Select the first visible option
    for (var i = 0; i < options.length; i++) {
        if (options[i].style.display !== 'none') {
            sizeSelect.selectedIndex = i;
            break;
        }
    }
}

// Initialize sizes based on default product type
document.addEventListener('DOMContentLoaded', function() {
    var defaultType = document.getElementById('product_type').value;
    updateSizesByType(defaultType);
});
</script>

<%@ include file="footer.html"%>
```

Рис. 3.25. Код сторінки додавання товару «фільтр розмірів»

Для адміністатора також створена була сторінка для відображення замовлень які не мали статусу «відправлено». Отже реалізація починається з створення екземпляра сервісів: OrderServiceImpl, FabricServiceImpl, TransServiceImpl, UserServiceImpl та ProductServiceImpl, які використовуються для отримання необхідних даних. Метод getAllOrders() з OrderServiceImpl витягує повний список замовлень, після чого на сторінці окремо відображаються необроблені замовлення з ознакою shipped 0 та скасовані з ознакою shipped 2.

Для кожного замовлення витягується ID транзакції, назва товару, тканина, email клієнта, адреса доставки та кількість товару. Дані про товар отримуються через метод getProductDetails() з ProductServiceImpl, інформація про тканину — через getFabricDetails() з FabricServiceImpl, адреса клієнта — через getUserAddr() з UserServiceImpl, а ID користувача визначається за допомогою getUserId() з TransServiceImpl.(Рис. 3.26.)

Адміністратор має можливість натиснути кнопку “Відправити”, яка формує запит до ShipmentServlet, передаючи параметри orderid, amount, userid та prodid. Це запускає процес позначення замовлення як відправленого. Або він може натиснути кнопку “Скасувати”, яка надсилає POST-запит до

CancelOrderServlet для скасування відповідного замовлення. У таблицях скасованих замовлень та відправлених показується аналогічна інформація, однак без можливості виконання дій.

```
    int unprocessedCount = 0;

    for (OrderBean order : orders) {
        if (order.getShipped() == 0) { // Перевіряємо, що shipped = 0 (очікує відправлення)
            unprocessedCount++;
            String transId = order.getTransactionId();
            String prodId = order.getProductId();
            String fabricId = order.getFabricId();
            int quantity = order.getQuantity();

            String userId = transService.getUserId(transId);
            String userAddr = userService.getUserAddr(userId);

            // Отримуємо назву товару
            ProductBean product = prodDao.getProductDetails(prodId);
            String productName = (product != null) ? product.getProdName() : "Невідомий товар";

            // Отримуємо інформацію про тканину
            FabricBean fabric = null;
            if (fabricId != null && !fabricId.isEmpty()) {
                fabric = fabricDao.getFabricDetails(fabricId);
            }
%>
    <tr>
        <td><%= transId %></td>
        <td>
            <a href="./updateProduct.jsp?prodid=<%= prodId %>" class="product-link">
                <%= productName %> <%-- Відображаємо назву товару --%>
            </a>
        </td>
        <td class="fabric-info-cell">
            <% if (fabric != null) { %>
                <img src="./ShowImage?fabricId=<%=fabric.getFabricId()%>" alt="<%=fabric.getColor()%>" class="fabric-admin-thumbnail">
                <%=fabric.getColor()%>
            <% } else { %>
                -
            <% } %>
        </td>
        <td><%= userId %></td>
        <td><%= userAddr %></td>
        <td><%= quantity %></td>
        <td>
            <a href="ShipmentServlet?orderid=<%= transId %>&amount=<%= order.getAmount() %>&userid=<%= userId %>&prodid=<%= prodId %>"
               class="btn btn-ship">
               Відправити
            </a>
            <form action="CancelOrderServlet" method="post" style="display:inline-block;">
                <input type="hidden" name="orderid" value="<%= transId %>">
                <input type="hidden" name="prodid" value="<%= prodId %>">
                <button type="submit" class="btn btn-danger btn-cancel">
                   Скасувати
                </button>
            </form>
        </td>
    </tr>
<%
        }
    }
    if (unprocessedCount == 0) {
%>
    <tr class="no-items">
        <td colspan="7" class="text-center">Немає замовлень для обробки</td>
    </tr>
<%
    }
```

Рис. 3.26. Код сторінки замовлень адміністратора «Фільтр не відправлених замовлень»

3.4 Тестування системи «Textileshop»

Після того, як всі заплановані напрацювання для системи були реалізовані, було прийнято рішення про її тестування. Метою тестування програмного продукту є значне зменшення кількості дефектів та забезпечення повної відповідності вимогам до продукту.

Для тестування використовувалась методологія «Чорного ящика».

Методологія чорного ящика - це підхід до тестування програмного забезпечення, який базується на аналізі функціональності системи без знання про її внутрішню структуру або реалізацію. У цьому підході тестувальник

розглядає програму, де він аналізує вхідні дані та очікувані вихідні результати без звернення до деталей внутрішньої реалізації.

Проведення тестів було створено план тестування [15]. Кожен з тестів виконувався декілька разів для досягнення більш точного результату тестування.

Тест-план:

1. Перевірка авторизації адміністратора

Дії: Відкрити сторінку без авторизації або з користувачем не типу "admin".

Очікуваний результат: Користувача буде перенаправлено на сторінку loginFirst.jsp, і сторінка обробки замовлень не буде завантажена.

Отриманий результат: При відсутності авторизації або якщо usertype ≠ "admin", виконується response.sendRedirect("loginFirst.jsp"), сторінка не відкривається.

Підсумок: Пройдено.

2. Перевірка відсутності замовлень

Дії: Очистити всі замовлення або відфільтрувати так, щоб список був порожній.

Очікуваний результат: У таблиці має з’явитися рядок з повідомленням “Немає замовлень для обробки” або “Немає скасованих замовлень”.

Отриманий результат: В обох таблицях реалізована перевірка через змінні unprocessedCount і canceledCount, при нулі виводиться повідомлення у рядку таблиці.

Підсумок: Пройдено.

3. Перевірка кнопки виходу (logout)

Дії: Натиснути кнопку “Logout” в адмін-панелі.

Очікуваний результат: Має відбутися вихід користувача зі сесії та редірект на сторінку авторизації.

Отриманий результат: Сесія завершується, виконується session.invalidate(), користувач потрапляє на login.jsp.

Підсумок: Пройдено.

4. Перевірка масштабування інтерфейсу з великою кількістю замовлень

Дії: Створити багато записів в базі з необробленими замовленнями.

Очікуваний результат: Сторінка повинна залишатись читабельною, можливо — реалізована пагінація або скрол.

Отриманий результат: Інтерфейс витримує велику кількість записів, таблиця прокручується, однак пагінація не реалізована.

Підсумок: Пройдено.

5. Перевірка некоректного формату orderid

Дії: Викликати ShipmentServlet?orderid=abc.

Очікуваний результат: Має з’явитись повідомлення про некоректний формат.

Отриманий результат: Виникає помилка NumberFormatException, немає обробки помилки.

Підсумок: Не пройдено.

4. РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ СИСТЕМИ

4.1 Системні вимоги до програмного додатку

Розроблений проект є добре оптимізованим, тому він не ставить серйозних навантажень на систему, це дозволяє проекту мати невеликі системні вимоги.

Вимоги хостингу:

Мінімальні:

- Процесор: 2 ядра, 2.0+ GHz (Intel Core i3 / AMD Ryzen 3)
- Оперативна пам’ять: 2 ГБ RAM
- Дисковий простір: 10 ГБ SSD
- Мережа: 50 Mbps

Рекомендовані:

- Процесор: 4 ядра, 2.5+ GHz (Intel Xeon / AMD Ryzen 5)
- Оперативна пам’ять: 4+ ГБ RAM
- Дисковий простір: 30+ ГБ SSD
- Мережа: 100+ Mbps
- Резервне копіювання: щоденне автоматичне бекапування

Програмні вимоги

- Операційна система: Ubuntu 20.04+ / Windows Server 2019+

Розгортання бекенду:

- Java Development Kit (JDK) 8 або новіше
- Apache Tomcat 9+
- MySQL Server 5.7+
- Драйвер JDBC для MySQL
- Процесор: 2.0 GHz Dual Core;
- Відеокарта: NVIDIA Geforce GTS 450 Або AMD Radeon HD 6750;
- ОЗУ: 4 GB;

- SSD: 1 GB доступного місця;
- Мережа: широкосмугове підключення до Інтернету;
- DirectX: версії 11;
- ОС: Windows 7;
- Доступ до інтернет мережі.

Вимоги до браузера клієнта:

Мінімальні:

- Процесор: Intel Pentium / AMD Athlon
- Оперативна пам’ять: 2 ГБ RAM
- Графічний процесор: інтегрований
- Мережа: 10 Mbps
- Рекомендовані:
- Процесор: Intel Core i5 / AMD Ryzen 5
- Оперативна пам’ять: 4+ ГБ RAM

Мережа: 30+ Mbps

Програмні вимоги

- Операційна система: Windows 10 / Ubuntu 22.04+
- Підтримувані браузери:
- Google Chrome
- Mozilla Firefox

4.2 Склад дистрибутива

До складу установлюючого дистрибутива входять наступні файли (Рис. 4.1.):

- Папку «src» - містить в собі реалізацію серверної частини інформаційної системи.
- Папки «WebContent» - містять в собі реалізацію інтерфейсної частини інформаційної системи.

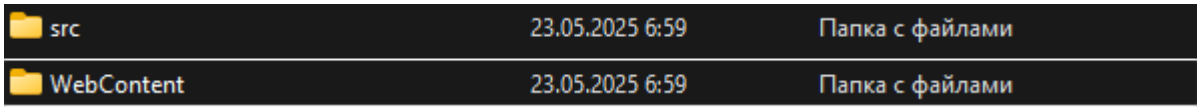

Рис. 4.1. Файли дистрибутива

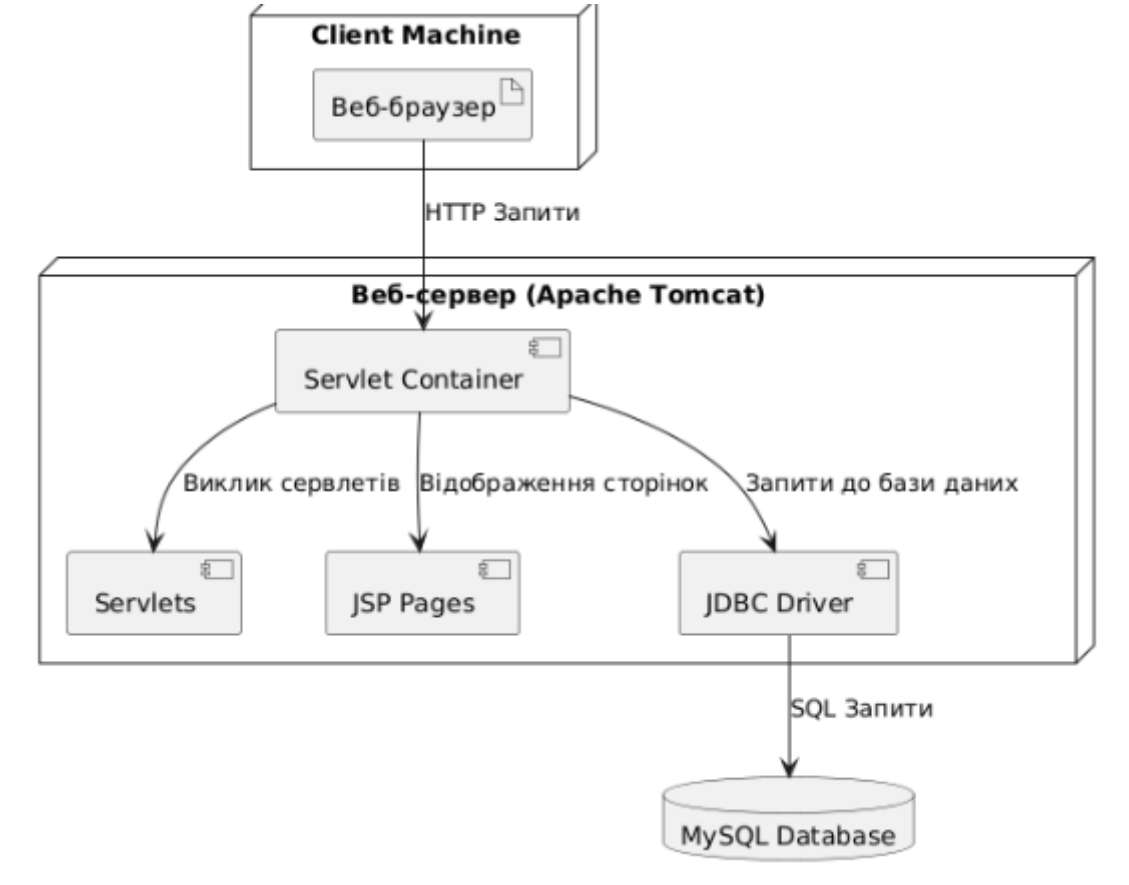

Рис. 4.2. Діаграма розгортання

4.3 Демонстрація веб-додатку

У верхній частині сайту завжди розташований головний навігаційний блок, який поєднує логотип магазину та основні пункти меню. Ліворуч знаходиться назва магазину, яка одночасно є клікабельним логотипом – зазвичай такий елемент веде на головну сторінку з товаром. Праворуч у гостя відображаються на ньому знаходяться дві ключові навігаційні кнопки. Перша кнопка "Увійти" призначена для авторизації зареєстрованих користувачів – вона веде на сторінку входу в особистий кабінет. Наступний пункт "Товари" – це посилання на каталог продукції, де відвідувачі можуть переглядати весь асортимент магазину.

Запуск веб-додатку відбувається шляхом доступу до його URL-адреси у веб-браузері. Після завантаження ви побачите головну сторінку, яка відображає основні категорії товарів (Рис. 4.3.).

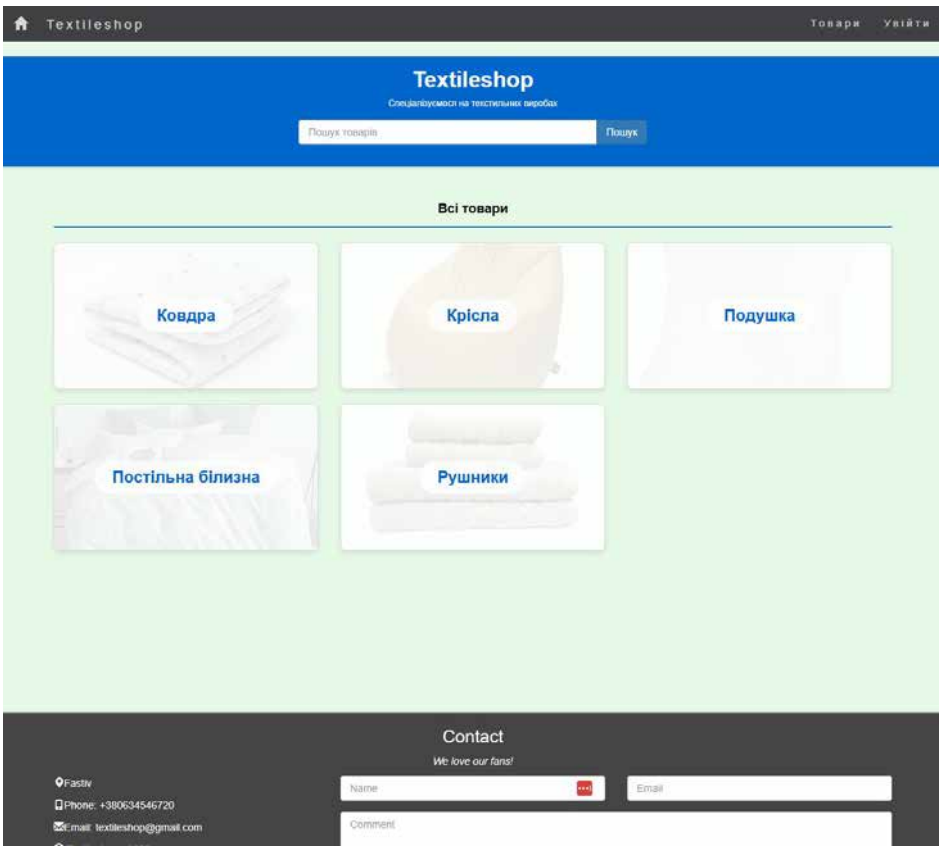

Рис. 4.3. Головна сторінка

На цій же сторінці пошуку конкретних товарів використовуйте поле "Пошук товарів" воно знаходиться у верхній частині вікна, при введенні слова «Подушка» та натисніть кнопку "Пошук" нам покажуться тільки які мають схожу назву. (Рис. 4.4.)

Повернувшись на головну сторінку ми можемо натиснути на одну з плиток категорій, після чого на сторінці лише товари, що належать до обраної категорії. (Рис. 4.5.).

У деяких товарів може бути декілька видів розміру, який ми можемо обрати і в нас зміниться інформація про товар. Також для товарів доступні різні кольора тканин, за стандартом обрана перша тканина але можна обрати іншу просто натиснувши на неї.

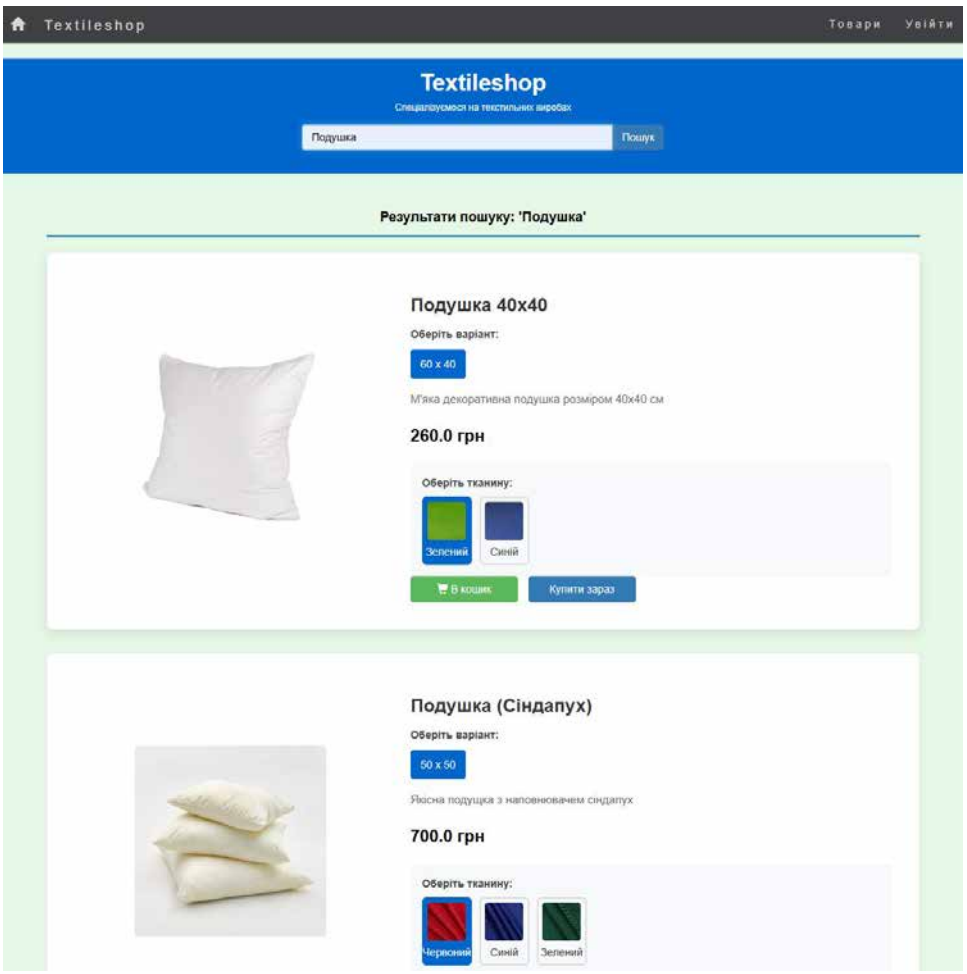

Рис. 4.4. Вікно зі знайденим товаром

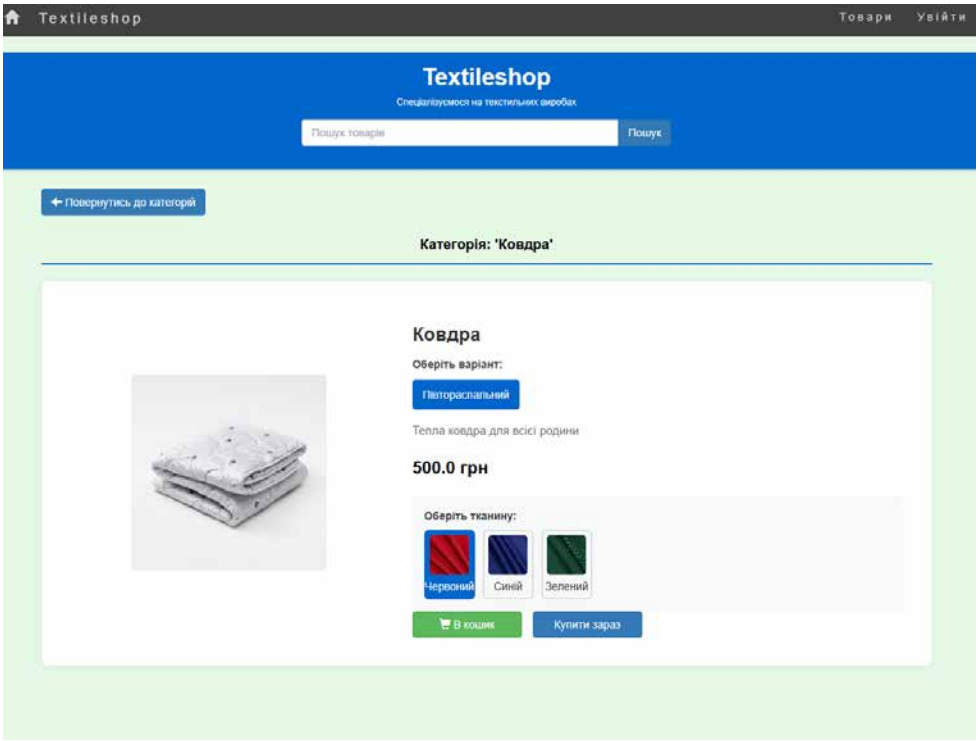

Рис. 4.5. Вікно з товаром обраної категорії

Так як ми не авторизовані то при натисненні кнопки покупки в нас не вийде замовити товар і перекине на сторінку авторизації, якщо ви вже реєструвались то просто вводите данні для авторизації(Рис. 4.6.), в іншому випадку натискаємо на напис створити акаунт та в формі яка з'явилась вводимо дані для реєстрації (Рис. 4.7.), якщо дані були коректно введено то користувач зможе увійти по цим даним.

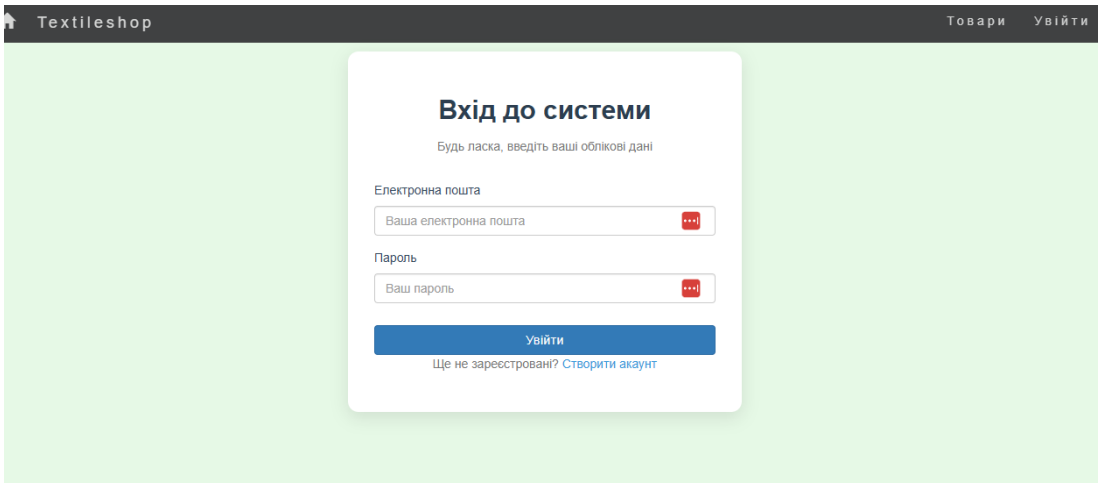

Рис. 4.6. Сторінка авторизації

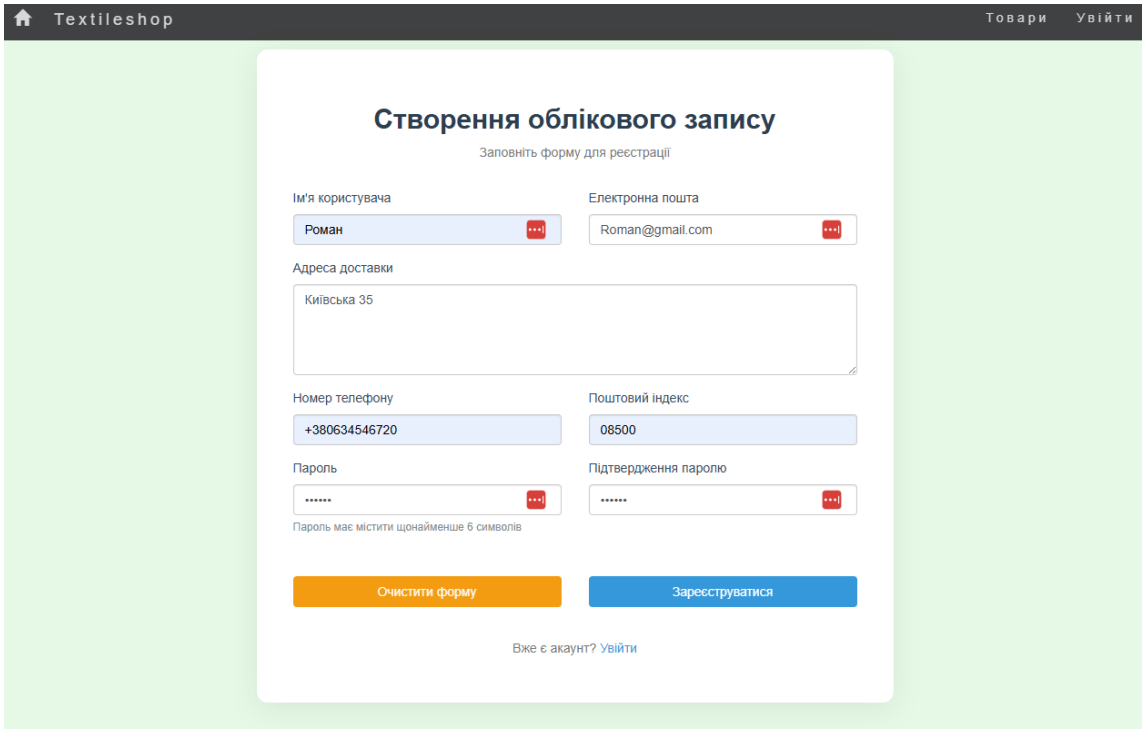

Рис. 4.7. Сторінка реєстрації

В авторизований користувач на панелі з'являються такі кнопки: «Кошик», яка перенаправлятиме користувача на сторінку з доданими товарами для оформлення замовлення. Наступна вкладка — «Замовлення»,

яка відкриватиме форму замовлень, зроблених цим користувачем. Далі йде вкладка «Профіль», при натисканні на яку відображатиметься сторінка з інформацією авторизованого користувача. Остання кнопка «Вийти» розлогінює користувача та відкриває вікно для логіну. Та при вході побачить ту саму сторінку з товаром, але тепер він зможе додавати продукцію в кошик. (Рис. 4.8.) Також користувач може змінювати кількість товару, якщо потрібно буде видалити його, то просто зменшуємо кількість до 0 то він видалиться.

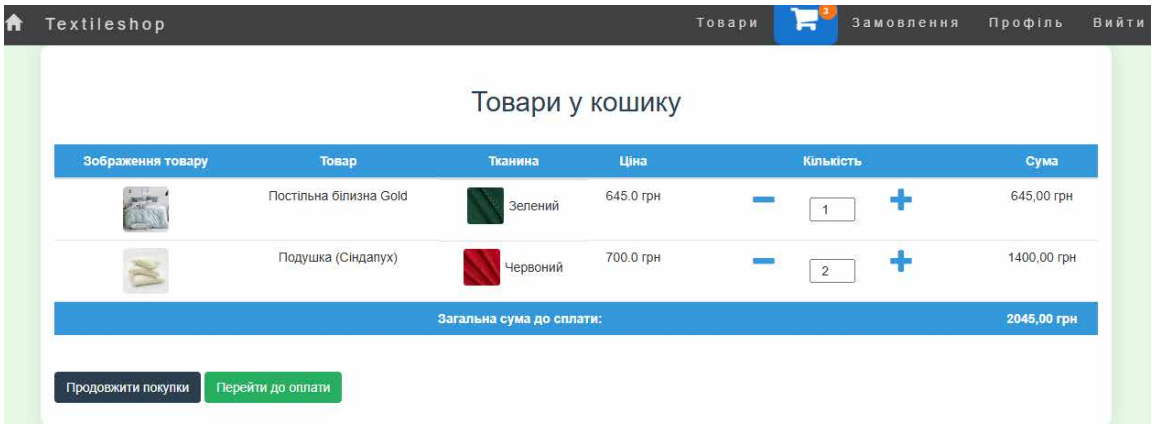

Рис. 4.8. Вікно корзини

Якщо користувач захоче замовити обрані товари, то він повинен натиснути кнопка «оплата» та заповнити форму оплати (Рис. 4.9.), якщо данні були введені коректно, то оплата пройде та його перенесе у вікно з замовленнями, де він може побачити своє замовлення та його статус.(Рис. 4.10.)

Якщо користувач авторизується, як адміністратор, то йому одразу покаже сторінку з товарами(Рис. 4.11.). Тут він може натиснути кнопку «Додати новий товар» після чого йому виведе форму для заповнення даних про товар, при коретно заповнених даних товар буде додано в базу даних та він зможете повернутись до сторінки з товарами(Рис. 4.12.). Якщо користувач захоче змінити дані товару то він в полі з товару може натиснути кнопку «Змінити» та його перенесе на форму з даними обраного товару, якщо зміни були коретними то вони приміняться та ці зміни можна буде побачити у вікні «Склад товарів».(Рис. 4.13.)

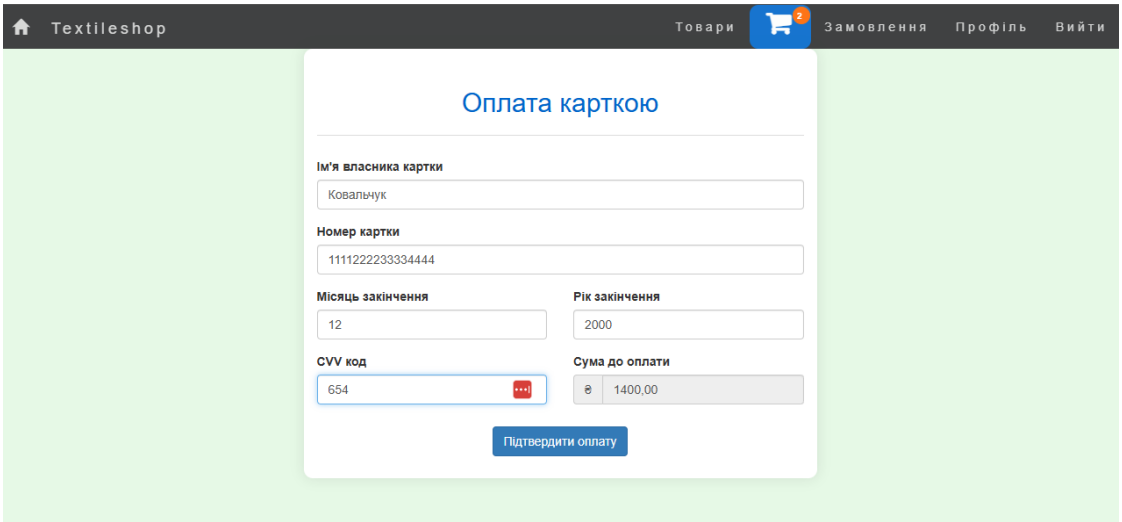

Рис. 4.9. Форма оплати

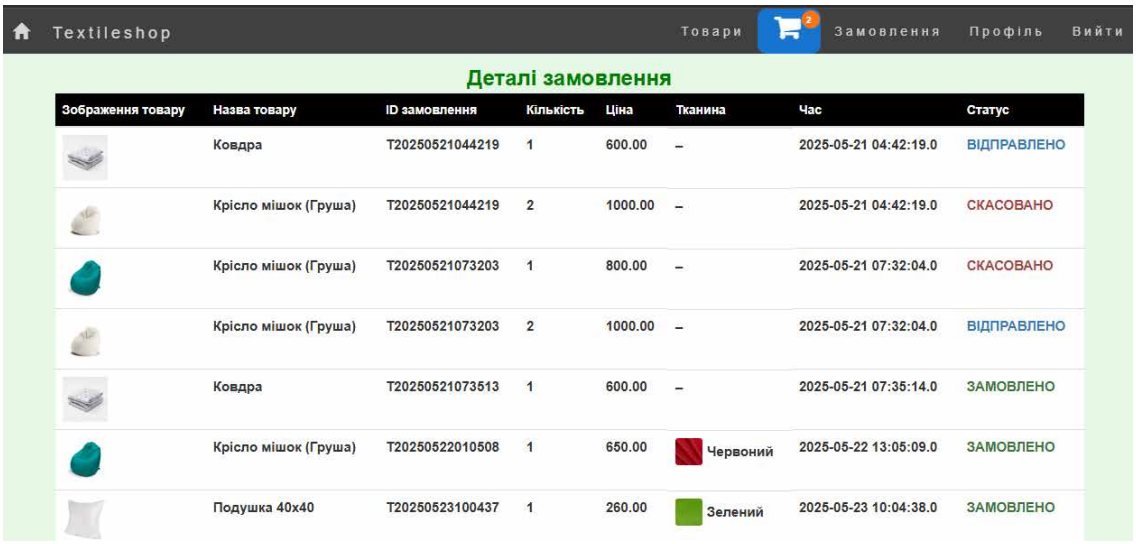

Рис. 4.10. Вікно з замовленнями користувача

Щоб перейти до управління іншими даними, користувач має в меню зверху натиснути на «Склад» після чого з’явиться випадаючий список де він зможе обрати потрібний пункт після цього покажеться сторінка управління продукції яку обрали (Рис. 4.14.).

Також адміністратор може змінювати статус замовлень, для цього йому в верхні панелі потрібно обрати «Замовлення», після чого має вивестись сторінка з таблицею необроблених замовлень (Рис. 4.15.), бравши потрібне замовлення користувач може його відправити і воно переміститься в таблицю відправлень яку можна переглянути обравши в верхньому меню «Відправлення», що відобразить таблицю з відправленими замовленнями

(Рис. 4.16.), або можна скасувати і тоді товар буде відображатись в таблиці зі скасованими замовленнями яка розміщується нижче необроблених.

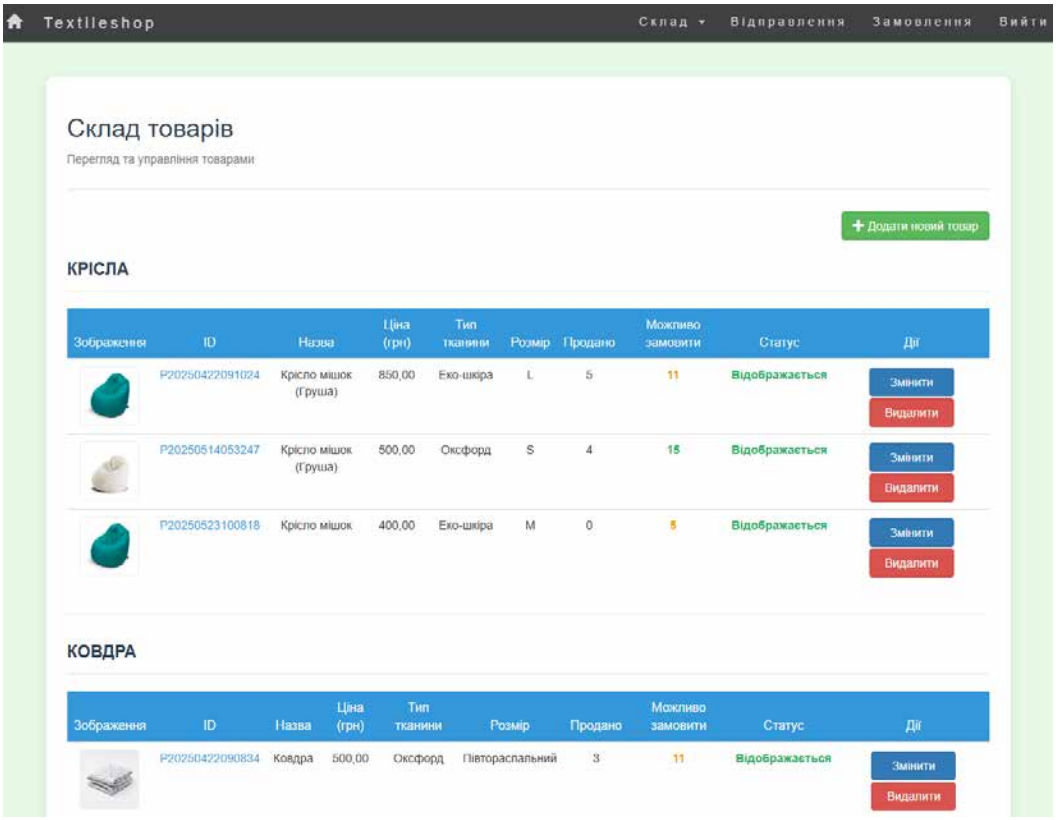

Рис 4.11. Сторінка управління товарами

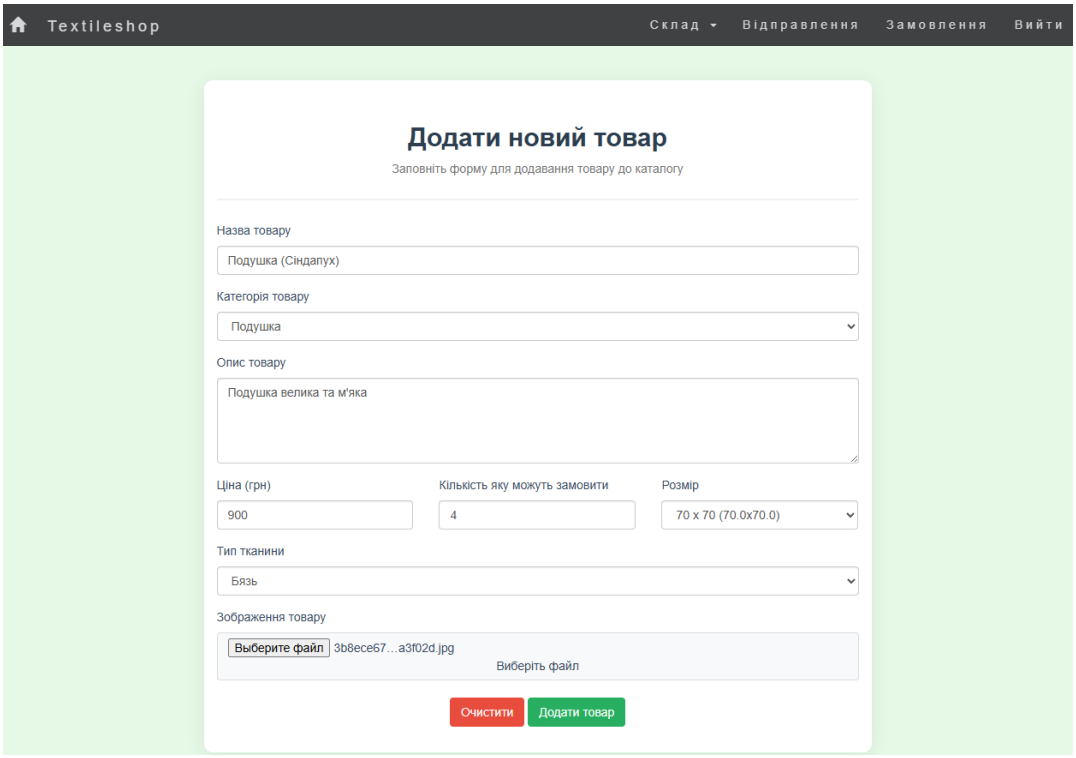

Рис. 4.12. Форма додавання товару

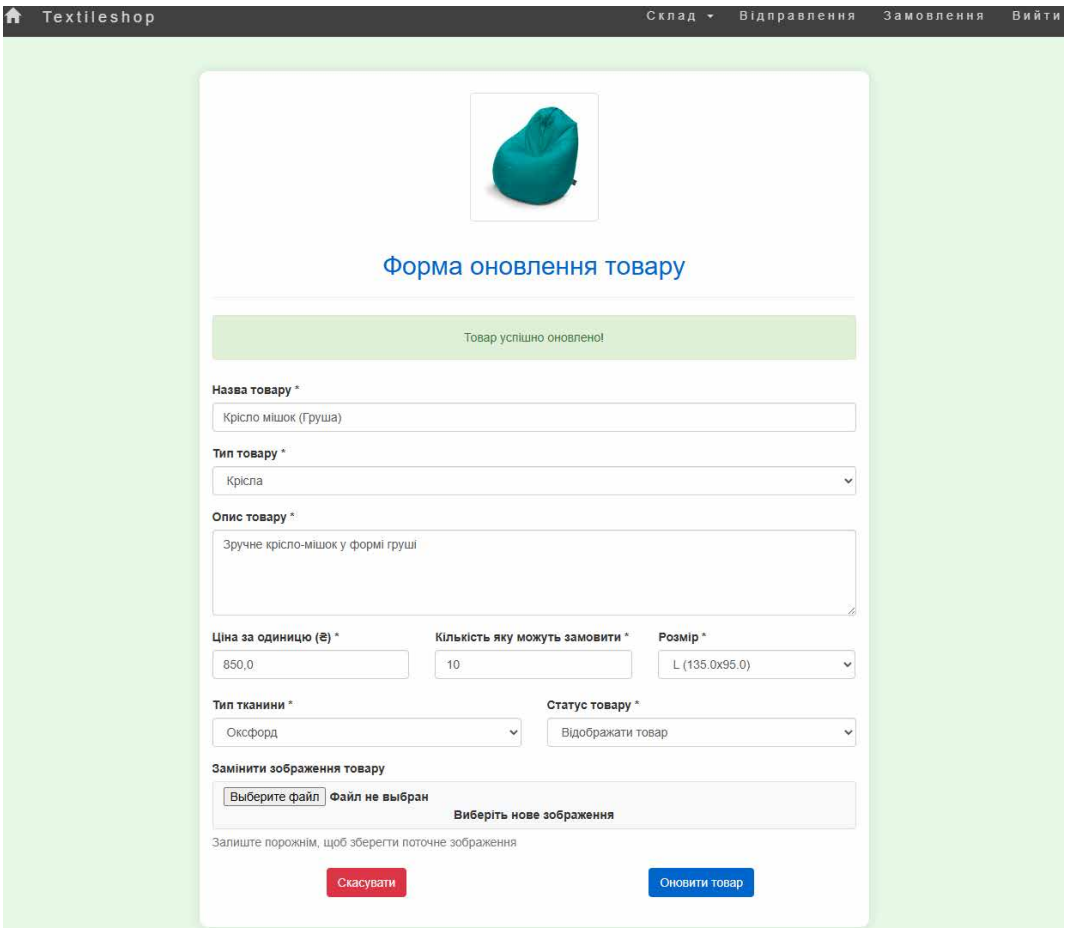

Рис. 4.13. Форма зміни даних товара

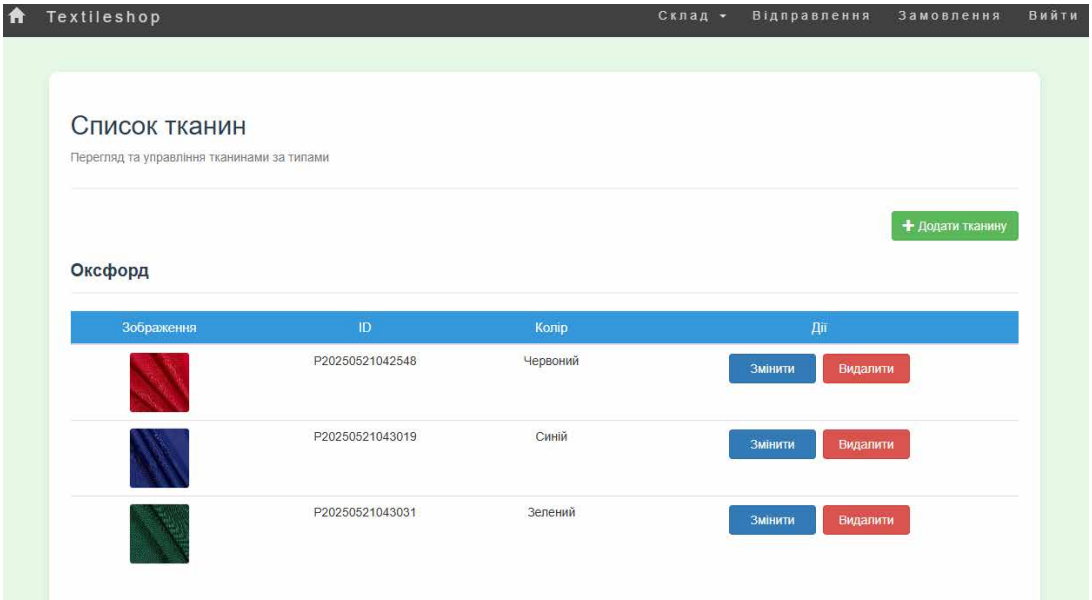

Рис. 4.14. Сторінка управління тканинами

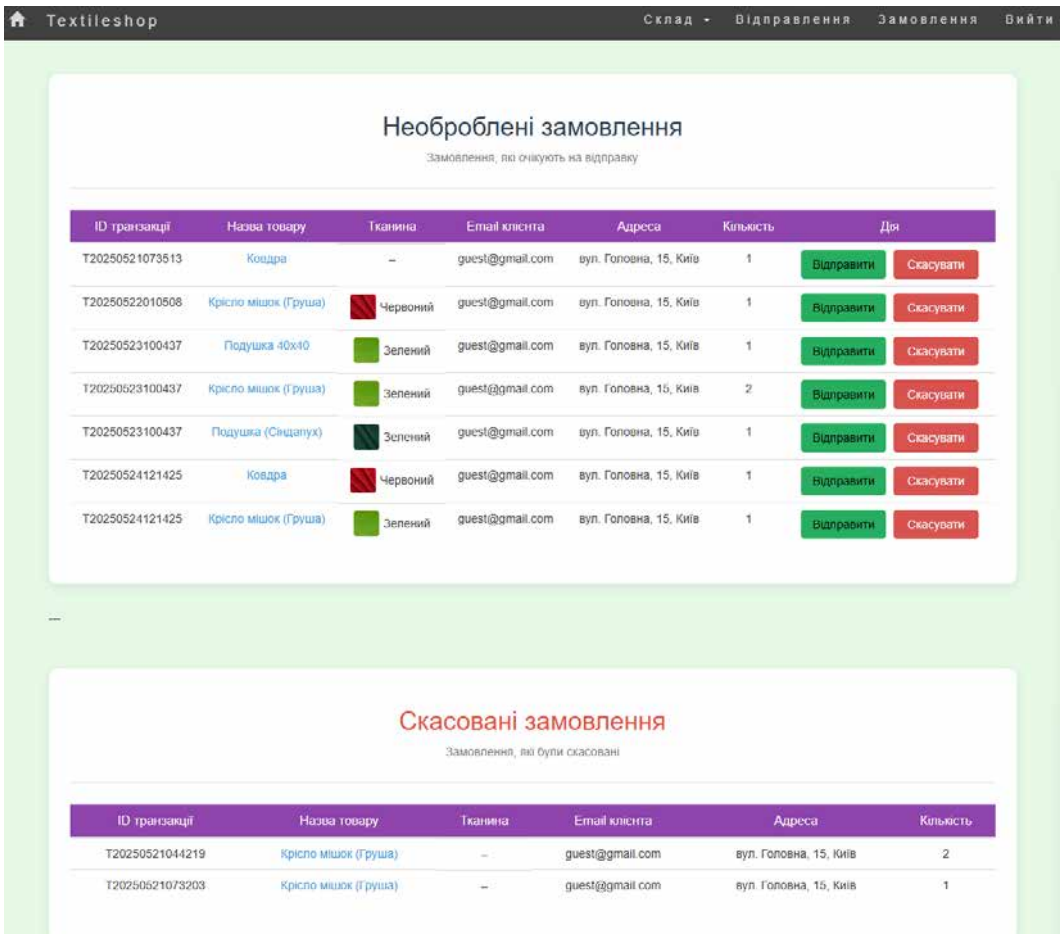

Рис. 4.15. Форма з таблицями не відправлених замовлень

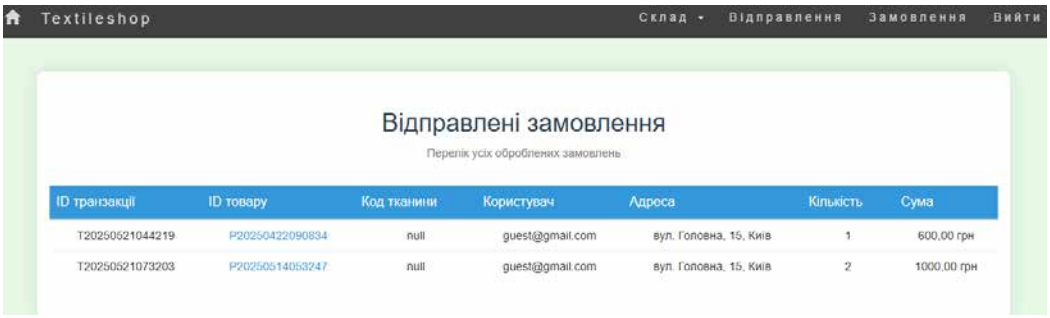

Рис. 4.16. Форма з таблицею відправлених замовлень

ВИСНОВОК

У ході виконання дипломного проєкту було здійснено повноцінне дослідження предметної області, що стосується продажу текстильних виробів, а також проведено аналіз і порівняння існуючих рішень у сфері розробки інформаційних систем для електронної комерції. Це дозволило визначити основні вимоги до майбутньої системи, обрати оптимальні програмні засоби та сформувати чітке бачення архітектури програмного забезпечення.

На основі аналізу було обрано для реалізації мови програмування Java, технологій JavaServer Pages та Servlets, для реалізацієї візуальної частини використовувався фреймворк Bootstrap, а також системи управління базами даних MySQL. Ці інструменти були обрані з огляду на їхню функціональність, відкритість, популярність та активну підтримку з боку спільноти. Їх застосування дозволило забезпечити масштабованість, надійність і зручність у розробці, а також спростити реалізацію основних модулів управління товарами, обробки замовлень, взаємодії з клієнтами та забезпечення автентифікації користувачів.

У процесі розробки було опановано такі важливі аспекти веб-програмування, як створення веб-інтерфейсу з використанням JSP, реалізація серверної логіки на основі Servlets, робота з базами даних за допомогою JDBC, а також побудова системи безпеки через механізми автентифікації та авторизації. Таким чином, здобуті знання охопили всі ключові компоненти створення сучасних веб-систем.

Результатом виконаної роботи стало створення повноцінного програмного забезпечення для інформаційної системи з продажу текстильних виробів, що відповідає сучасним вимогам до подібних систем. Окрім основного функціоналу, система має потенціал для подальшого розвитку — зокрема, інтеграції з платіжними сервісами, розширення модуля аналітики та звітності, а також розробки мобільної версії додатку для зручності кінцевих користувачів.

Таким чином, усі поставлені цілі та завдання дипломного проєкту були досягнуті, що свідчить про практичну значущість отриманих результатів і можливість їх подальшого впровадження в реальні комерційні проєкти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Невідомий, « Веб-розробка: B2B, B2C, C2C, C2B,» 2015.

[2] B. N., «Information Architecture for the World Wide Web,» 2013.

[3] З. С. Васильович, «Основи проектування інформаційних систем,» 2019.

[4] Ю. Н. Власюк Ю. Ю., «Курс лекцій з дисципліни «Веб-проектування»,» 2011.

[5] M. D. Network, «JavaScript technologies overview,» 2025.

[6] ITProger, «17 найпопулярніших CSS-фреймворків,» 2018.

[7] O. Mell, «Огляд фреймворків JavaScript. Що, для чого і коли використовувати,» 2021.

[8] Oracle, «Java SE 8 Documentation,» 2018.

[9] P. H. Noodles, «Частина 3: Протоколи HTTP,» 2023.

[10] Foxminded, «Реляційні бази даних,» 2023.

[11] MySQL, «MySQL Documentation,» 2025.

[12] Oracle, «Class Message,» 2018.

[13] K. J, «Essential Guide to HTML5 Using JavaScript,» 2017, 2017.

[14] L. J, «Bootstrap 5 Quick Start,» 2021.

[15] QATestLab, «HTTP-протокол: що це і де тестувати,» 2023.

ДОДАТКИ

ДОДАТОК А

Посилання на репозиторій мого проекту:
https://github.com/A1lurus/TextileShop