МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційне та програмне забезпечення онлайн магазину книг»
здобувача групи ІН - 02 Круть Назара Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.
Назар КРУТЬ
(підпис)

Керівник,
доцент кафедри комп'ютерних наук,
доцент, кандидат фізико-математичних наук
Галина ОЛЕКСІЄНКО
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-02 Круть Назара Сергійовича

1. Тема роботи: «Інформаційне та програмне забезпечення онлайн магазину книг»
затверджую наказом по СумДУ від _____
2. Термін здачі здобувачем кваліфікаційної роботи *до* _____
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для створення інформаційної системи онлайн магазину книг. 3) Розробка інформаційної системи онлайн магазину книг. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «01» квітня 2024 р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	10.03.2024	
2	*Огляд технологій, що використовуються для розробки веб-додатків*	15.04.2024	
3	*Розробка інформаційної системи Інтернет-магазину*	25.04.2024	
4	*Аналіз отриманих результатів*	05.05.2024	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	20.25.2024	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 83 стор., 26 рис., 7 табл., 1 додаток, 12 використаних джерел.

Обґрунтування актуальності теми роботи – актуальність теми полягає у необхідності створення інтернет-магазину книг, який би забезпечував зручний інтерфейс для користувачів та ефективну обробку замовлень. Важливим аспектом є також безпека даних користувачів та швидкість обробки інформації.

Об'єкт дослідження — процес створення та функціонування інтернет-магазину для продажу книг.

Предмет дослідження — методи та технології розробки інтернет-магазину, включаючи проектування бази даних, розробку клієнтської та серверної частин, а також забезпечення інтеграції з зовнішніми системами.

Мета роботи — розробка веб-застосунку «BookStore». Додаток повинен містити ознайомлювальну головну сторінку та каталог товарів, мати можливість оформлення замовлення через кошик.

Методи дослідження — аналіз ринку, системний аналіз, проектування, програмування.

Результати — було проаналізовано аналогічні проекти, обрано метод вирішення задачі, розроблено інформаційну систему та програмне забезпечення, які надають можливість користувачам переглядати книги в каталозі та оформлювати замовлення.

ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРНЕТ-МАГАЗИН, ВЕБ-ДОДАТОК, SPRIN BOOT, THYMLEAF

Зміст

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки у сучасному світі цифрових технологій та інтернету, інтернет-магазини стають все більш популярними серед споживачів у різних сферах, включаючи роздрібну торгівлю книгами. Створення інформаційної системи онлайн магазину книг є особливо актуальним.

Згідно з дослідженням [1], онлайн-сегмент книжкового ринку продовжує рости, що підкреслює необхідність розвитку інтернет-магазинів книг, дозволяючи читачам знайти та придбати книги більш зручним способом. Інтеграція сучасних технологій, таких як штучний інтелект, машинне навчання для персоналізації рекомендацій, і блокчейн для забезпечення безпеки транзакцій, робить створення інтернет-магазинів більш привабливим і ефективним. Дослідження [2] показує, що інновації у сфері електронної комерції значно впливають на поведінку покупців. Онлайн-магазини надають можливість купувати книги 24/7 з будь-якої точки світу, не виходячи з дому, що є особливо актуальним для людей з обмеженими можливостями пересування, мешканців віддалених районів та всіх, хто цінує свій час.

Об’єкт дослідження. Процес створення та функціонування інтернет-магазину для продажу книг.

Предмет дослідження. Методи та технології розробки інтернет-магазину, включаючи проектування бази даних, розробку клієнтської та серверної частин, а також забезпечення інтеграції з зовнішніми системами.

Гіпотеза. Впровадження сучасного інформаційного та програмного забезпечення для онлайн магазину книг дозволить суттєво підвищити ефективність управління асортиментом та обробки замовлень, а також забезпечить покращений користувацький досвід, що, у свою чергу, призведе до збільшення кількості задоволених клієнтів та зростання продажів.

Новизна. Новизна даної роботи полягає у розробці інтегрованої інформаційної системи для онлайн магазину книг, яка поєднує в собі сучасні технології управління даними, зручні інтерфейси для користувачів та адміністраторів, а також ефективні інструменти для обробки замовлень і взаємодії з платіжними системами та службами доставки.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз аналогічних проєктів

Перед початком розробки, потрібно проаналізувати існуючі аналоги інтернет-магазинів книг.

Перший аналог – Клуб Сімейного Дозвілля[3]. Видавництво Клуб Сімейного Дозвілля, скорочено «КСД» — одне з найбільших видавництв України, засноване 2000 року. В інтернет-магазині можна переглянути асортимент книг та зробити замовлення через веб-застосунок.

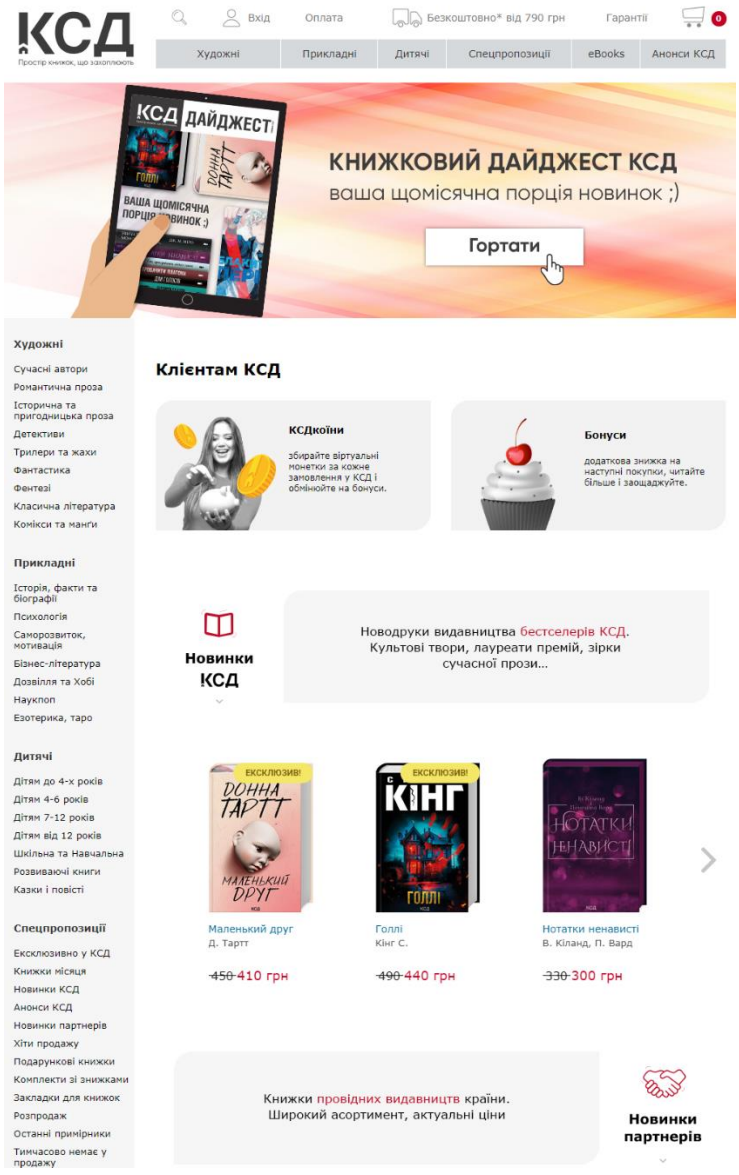

Рисунок 1.1 - Головна сторінка сайту КСД

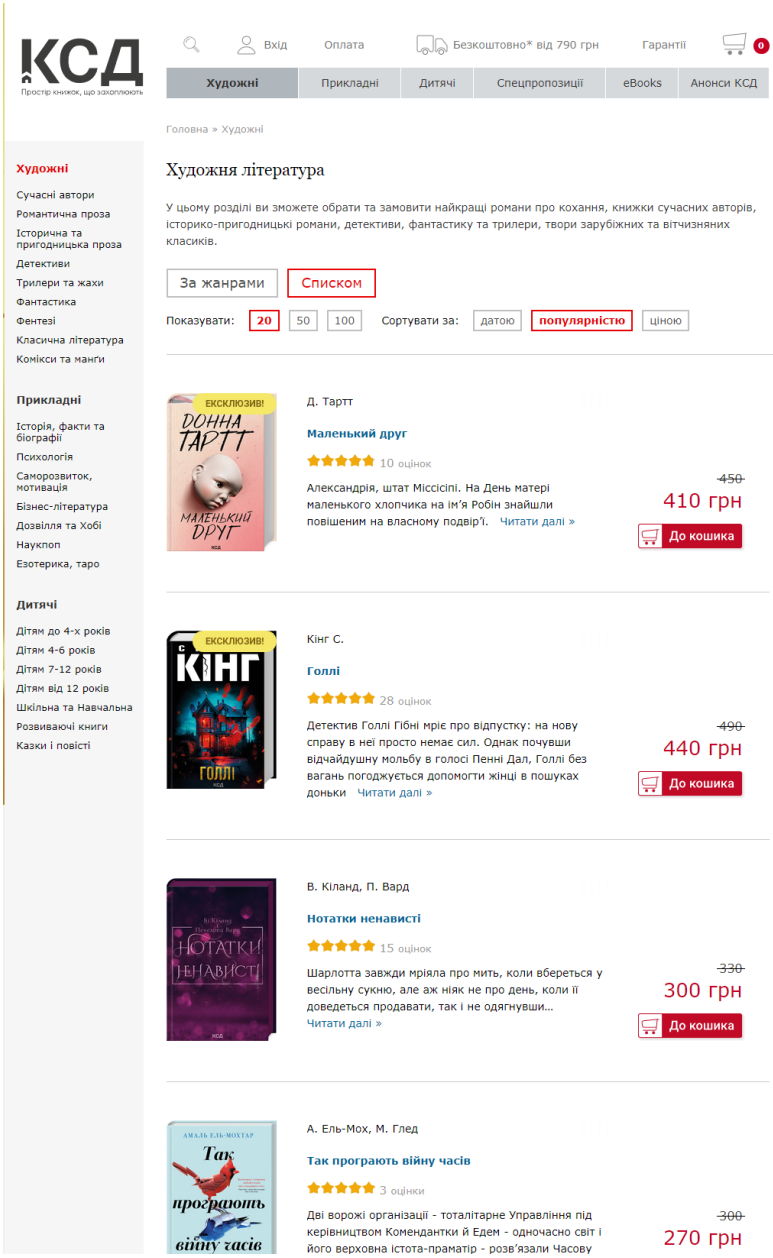

Рисунок 1.2 - Сторінка каталогу товарів

Переваги:

- Простий та зрозумілий інтерфейс для пошуку та завантаження електронних книг.
- Можливість швидко знайти потрібну книгу за назвою, автором або жанром.
- Наявність рейтингів та відгуків користувачів, що допомагає вибрати книгу.

Недоліки:

- Не дуже привабливий дизайн, що може відлякати деяких користувачів.
- Відсутність можливості перегляду книг перед завантаженням.

Наступний аналог – Yakaboo[4]. Yakaboo — одна з найбільших в Україні інтернет-книгарень, що пропонує книги 71 мовою у паперовому, електронному та аудіо форматах. На головній сторінці представлені банери та спеціальні пропозиції. На сайті зручний та зрозумілий інтерфейс, нічого не заважає перегляду.

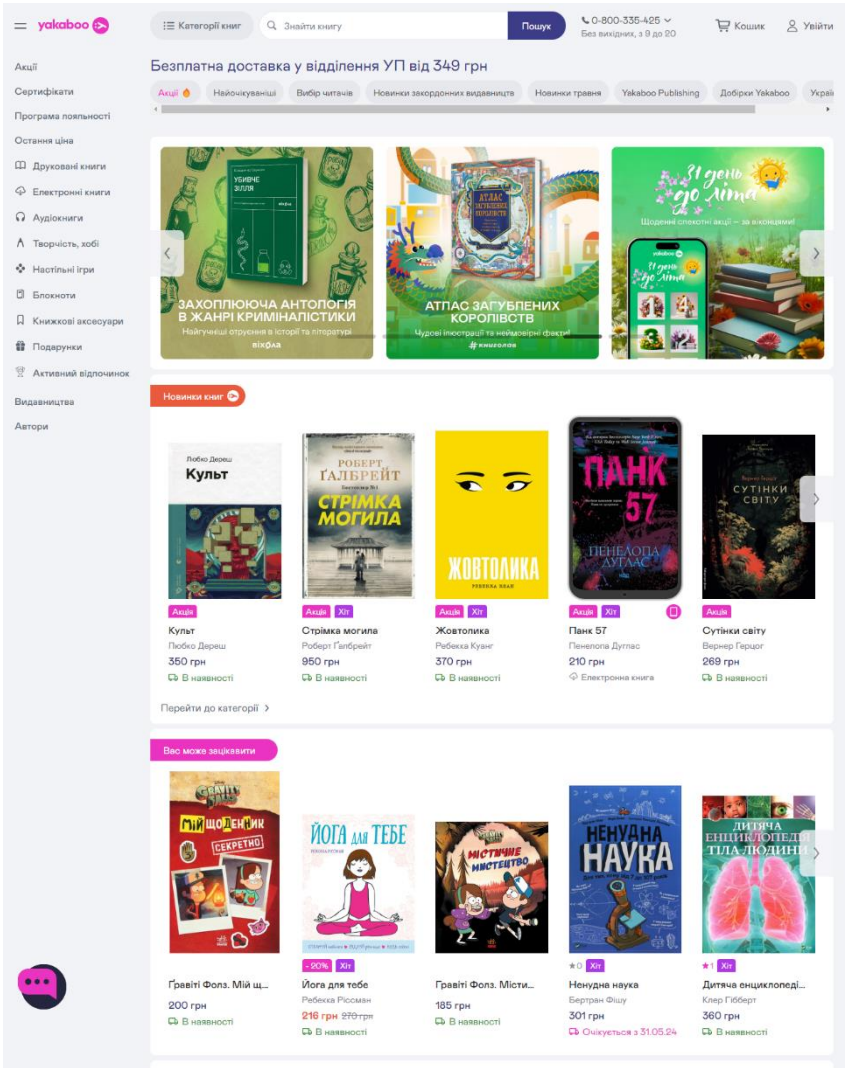

Рисунок 1.3 - Головна сторінка сайту Yakaboo.

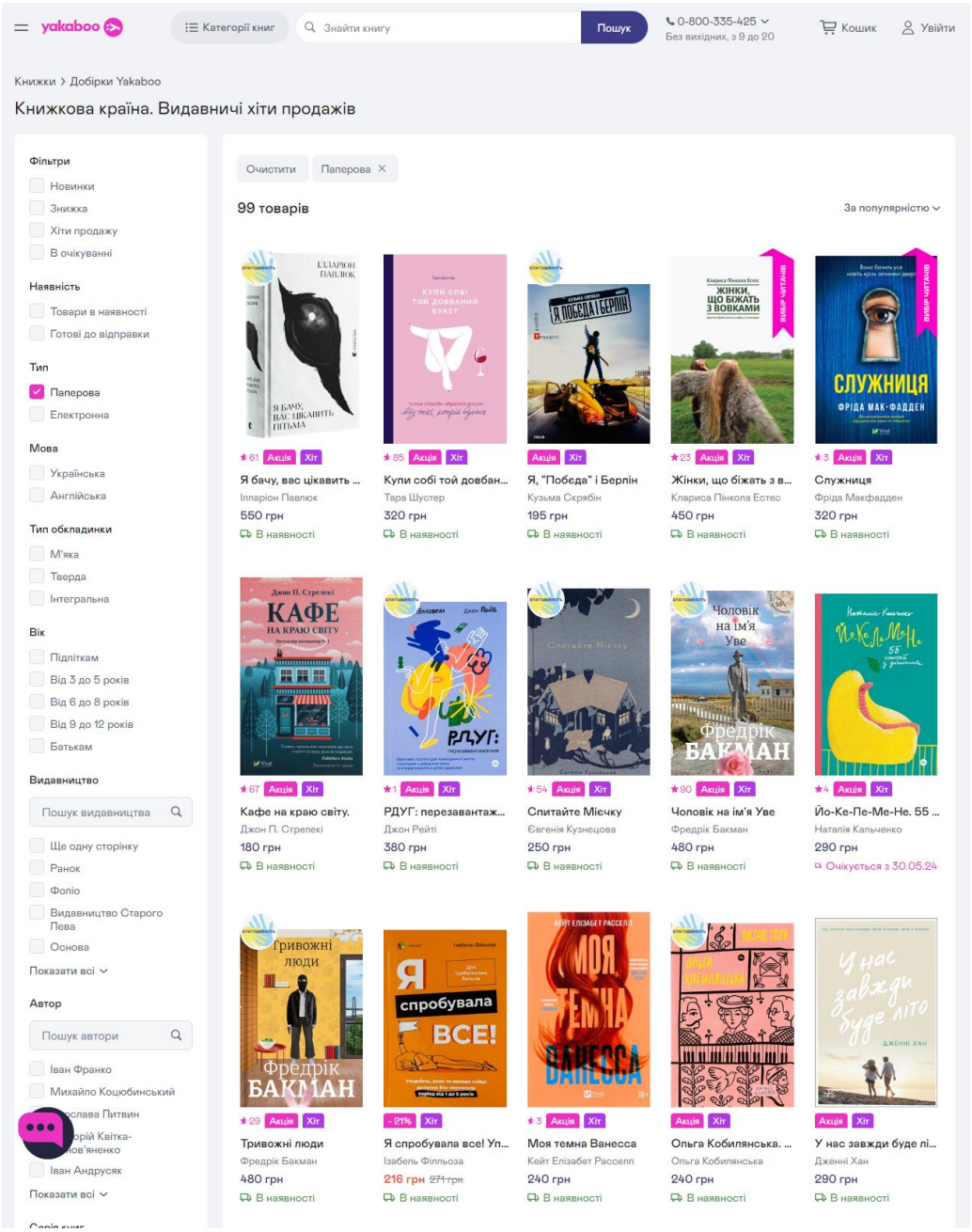

Рисунок 1.4 - Сторінка каталогу товарів

Переваги:

- Сучасний та естетично приємний дизайн сайту.
- Швидка та зручна навігація, дозволяє швидко знайти потрібні товари.
- Великий вибір книг та інших товарів, можливість замовлення книг та отримання їх доставки.

Недоліки:

- Можливість повільної реакції сайту при великому навантаженні.
- Не завжди коректна робота функції пошуку.

Третій аналог – Книгарня «Є»[5]. Це всеукраїнська мережа книгарень, заснована наприкінці 2007 року. Сайт зустрічає агресивною спливаючою рекламою на всю сторінку. На головній сторінці можна переглянути спеціальні пропозиції, акції та ін.

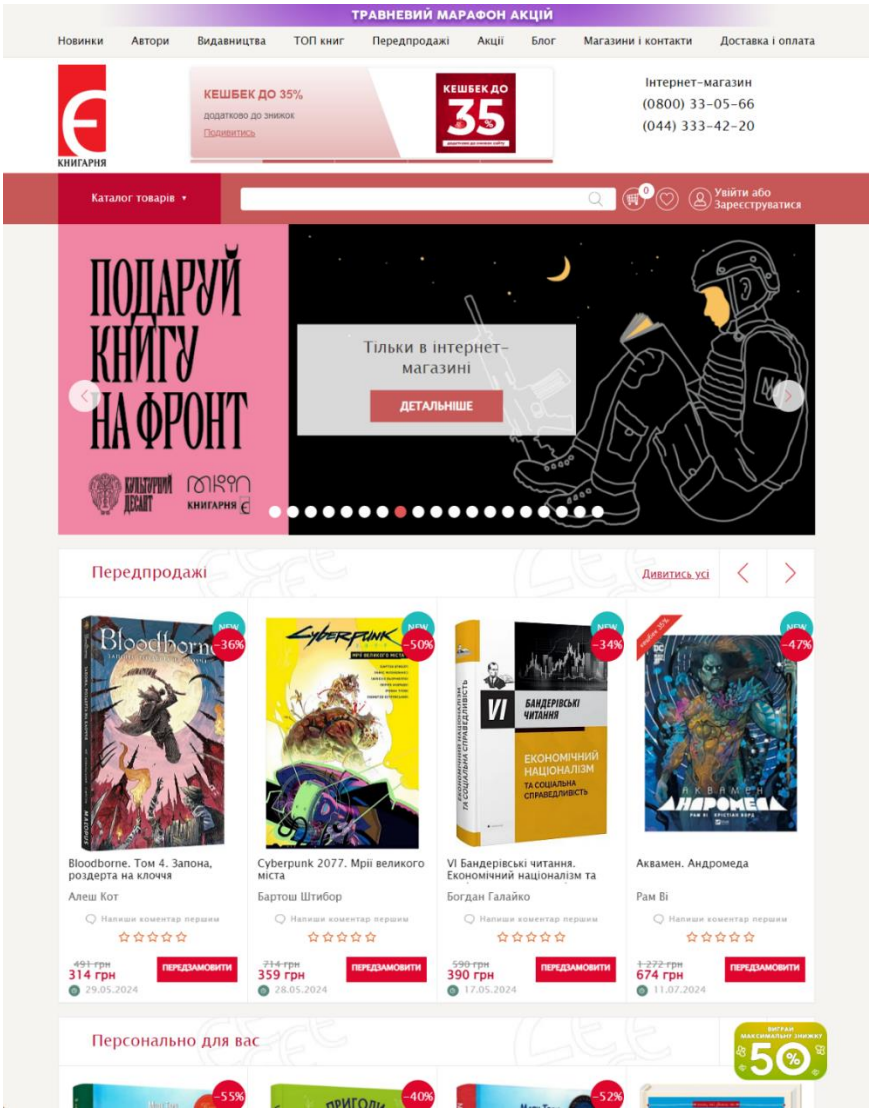

Рисунок 1.5 - Головна сторінка сайту .

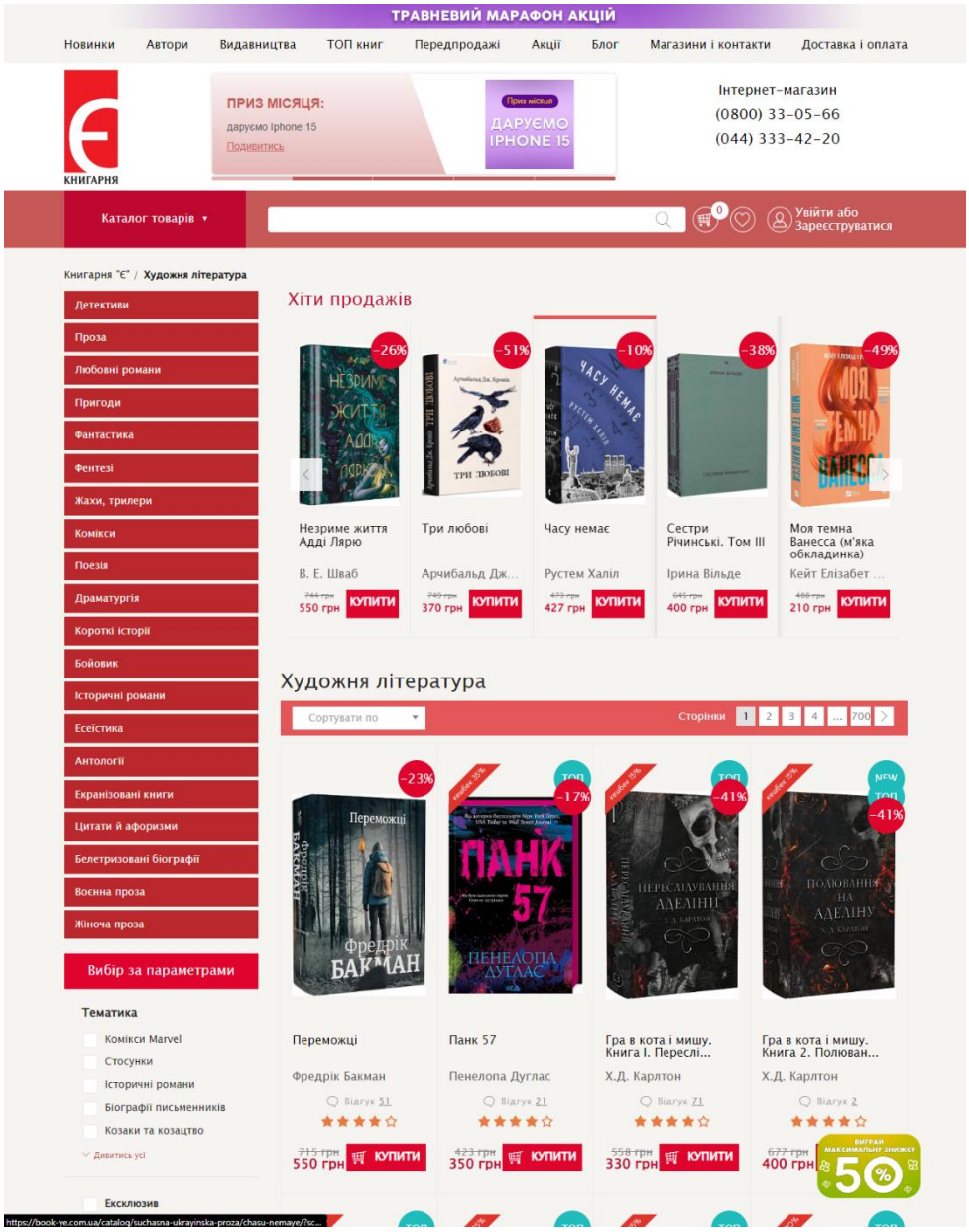

Рисунок 1.6 - Сторінка каталогу товарів

Переваги:

- Інтуїтивний та зручний інтерфейс, легко знаходити потрібні книги.
- Швидкий пошук і фільтрація за різними параметрами (автор, жанр, видавництво тощо).
- Інформативні сторінки з деталями про книги, включаючи опис, рецензії та рейтинги.
- Простий процес оформлення замовлення та оплати.

Недоліки:

- Можливість повільної роботи сайту при великій кількості відвідувачів.
- Відсутність персоналізованих рекомендацій для користувачів.
- Різка вспливаюча реклама

1.2 Постановка задачі

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) аналіз методів та засобів побудови сайтів;
2) вибір методів рішення;
3) проектування сайту;
4) проектування бази даних;
5) розробка дизайну;
6) написання веб-додатку;
7) тестування готового продукту.

2 ВИБІР МЕТОДУ РІШЕННЯ

Виділяють 3 найбільш популярних методів розробки веб-застосунків:

1) за допомогою HTML/CSS/JavaScript (ручний метод);

2) на основі SaaS-платформ (англ. software as a service – програмне забезпечення, як послуга;

3) розробка на CMS (англ. content management system – система управління сайтом);

2.1 Розробка веб-додатку власноруч

• Створення веб-додатку з використанням мов програмування, таких як HTML, CSS та JavaScript[8].

• Використання бази даних для зберігання записів та їх категоризації.

• Розробка інтерфейсу користувача для перегляду товарів, їх замовлення та пошуку.

• Забезпечення безпеки даних і доступу.

Перший метод вимагає знання мов програмування, багато часу та уваги. Метод підходить для створення чогось унікального.

2.2 Використання SaaS платформ

Метод полягає в оренді програмного забезпечення через інтернет. Ці системи також відомі як конструктори. Достатньо просто зареєструватися на відповідному сервісі, який надає необхідне програмне забезпечення. Потім ви можете вибрати готовий шаблон та заповнити його власною інформацією або створити сайт з використанням готових блоків та власним дизайном. Плюси використання SaaS-платформ для створення сайтів включають:

- швидкість створення - можна створити готове рішення за кілька днів;

- низька вартість розробки порівняно з CMS - достатньо вибрати необхідний пакет та оплатити передплату;
- у рішення вже включені всі необхідні інструменти для повноцінної роботи сайту - панель управління, хостинг, сервер.

Мінуси використання SaaS-платформ:

- обмежена можливість налаштування блоків сайту;
- більшість таких систем не оптимізована для високого навантаження на сайт.

Цей метод розробки підходить для запуску нових нескладних проектів та компаній, що мають обмежений час та бюджет. Приклада SaaS є Shopify[6]:

Shopify це одна з найпоширеніших платформ для електронної комерції, яка пропонує рішення "з коробки" для створення та керування онлайн магазинами. Shopify має широкий вибір готових тем і додатків, що дозволяє швидко налаштувати магазин під власні потреби.

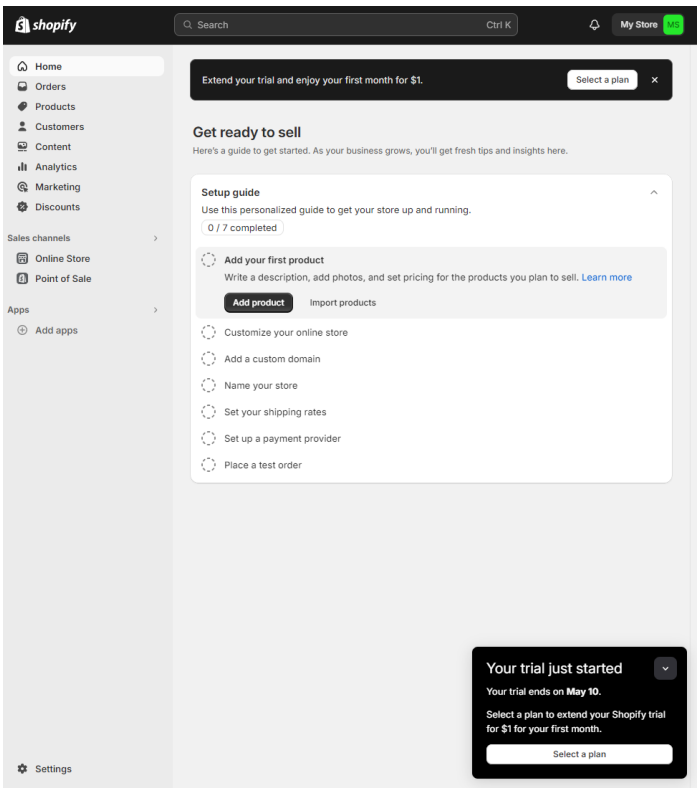

Рисунок 2.1 - Інтерфейс Shopify

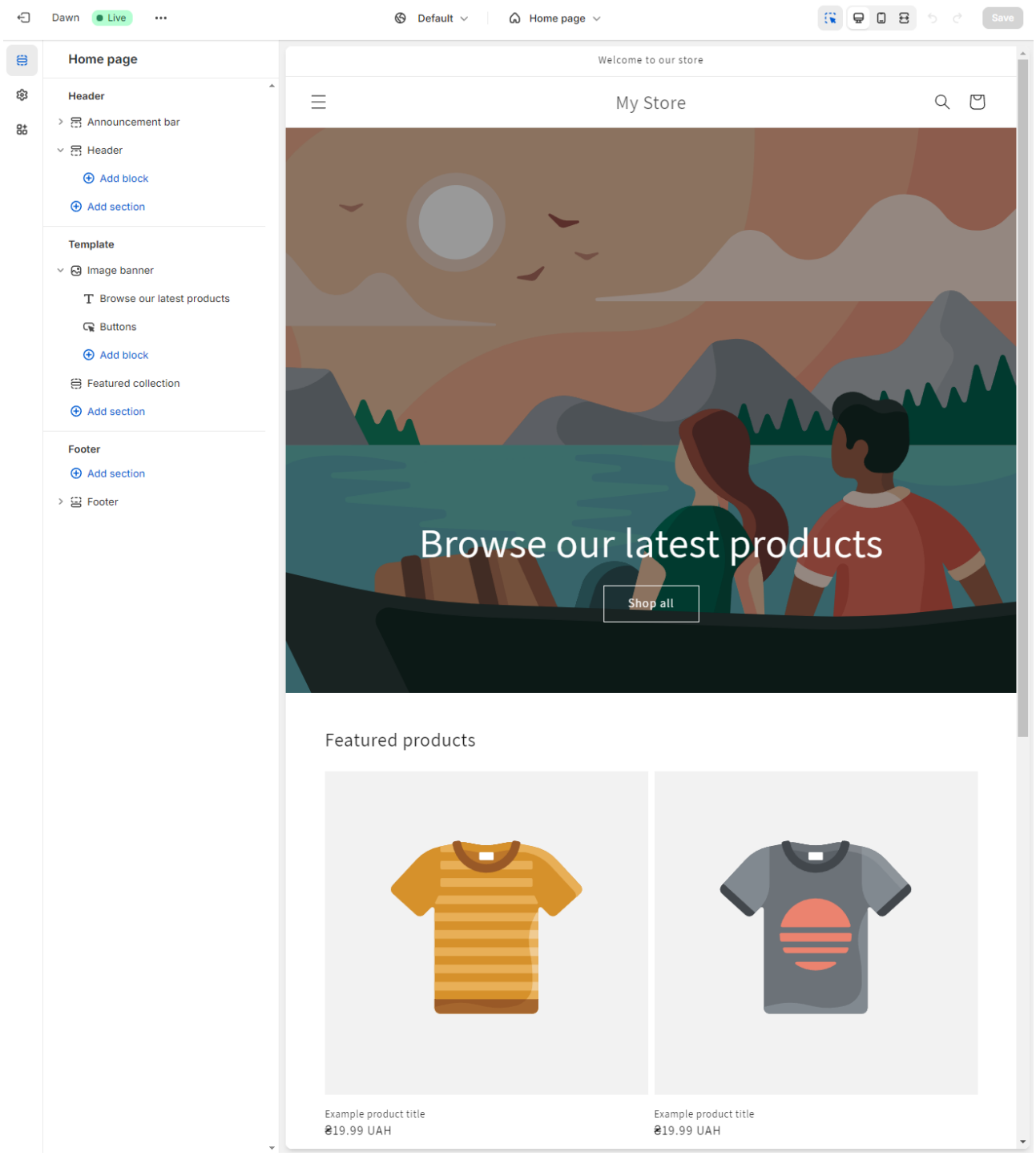

Рисунок 2.2 - Головна сторінка шаблону

Shopify надає можливість змінювати дизайн, переміщувати блоки з контентом та додавати нові.

Переваги:

- Простий у використанні та налаштуванні.

- Широкий вибір готових тем і додатків.
- Надійна підтримка та безпека.

Недоліки:

- Високі комісійні від продажів у деяких тарифах.
- Обмежена можливість налаштування, порівняно з деякими іншими платформами.
- Безкоштовний період лише 3 дні, далі 1 місяць за 1 долар, а потім 24 долари на місяць.

2.3 CMS

Третій підхід CMS – це готове програмне забезпечення, що виглядає як готова основа для веб-сайту і призначене для управління, створення та зміни цифрового контенту. Основною перевагою цього методу є зменшення часу і витрат на розробку порівняно з власноруч написаними системами. Проте, гнучкість власноруч написаних систем значно краща, оскільки вони лишені зайвого та повністю залежать від потреб замовника. Головна проблема безкоштовних CMS - це проблеми безпеки. Будь-хто знайомий з програмуванням може освоїти ці CMS та, майже будь-хто може їх взломати, що може призвести до серйозних проблем для компанії. Тому рекомендується використовувати платні модулі, які є більш надійними та безпечними щодо злому. Прикладом CMS є OpenCart[7].

OpenCart це безкоштовна платформа з відкритим вихідним кодом, яка пропонує просте інтерфейс для створення і керування магазином. OpenCart має ряд безкоштовних і платних розширень для розширення функціоналу вашого магазину.

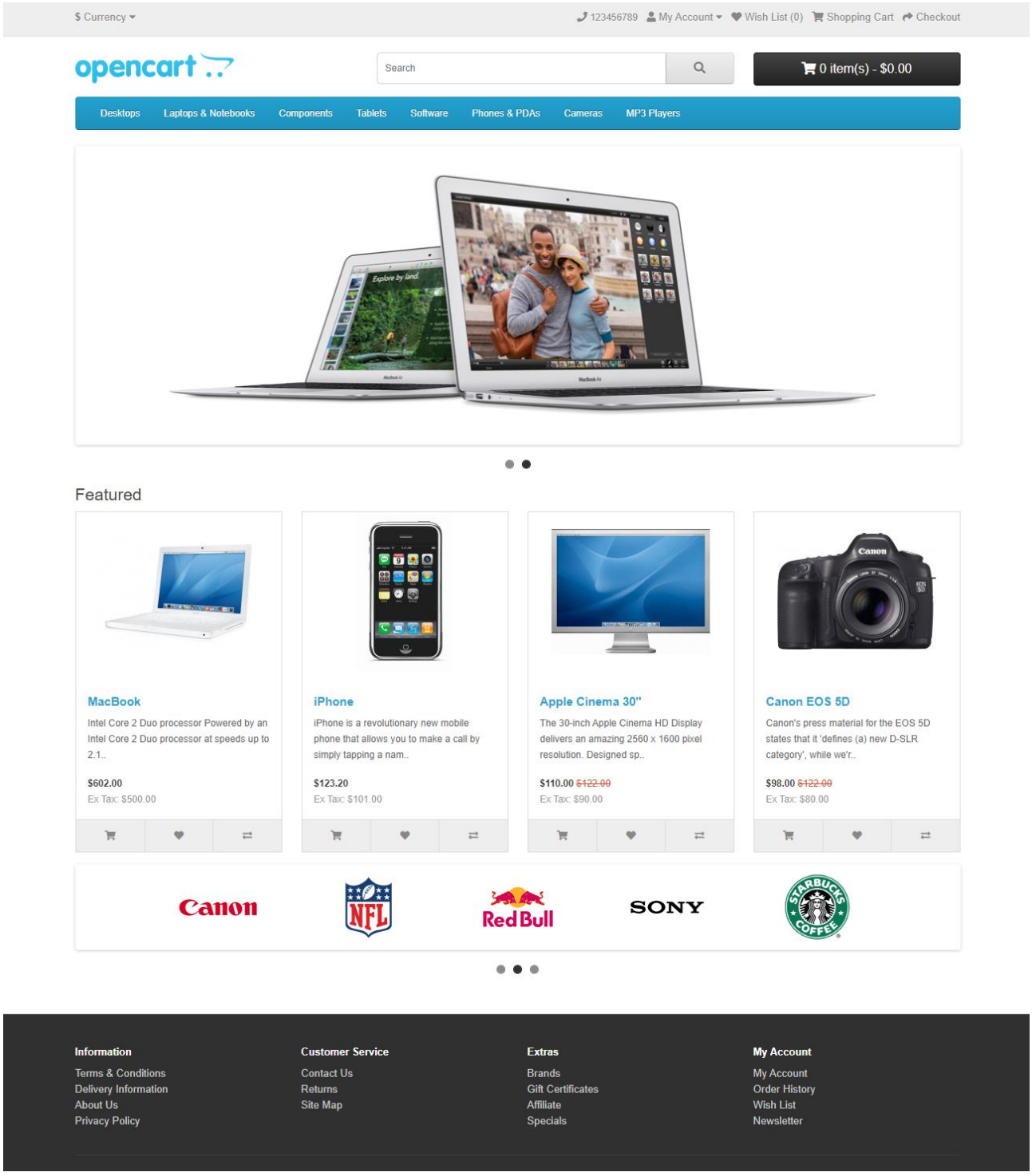

Рисунок 2.3 - Приклад магазину створеного за допомогою OpenCart

Переваги:

- Легкість встановлення та використання.
- Широкий вибір безкоштовних розширень та тем.
- Невибагливий до хостингу.

Недоліки:

- Менш розвинута спільнота користувачів та підтримки порівняно з іншими платформами.

Кожен із цих інструментів має свої переваги та недоліки, і вибір залежить від потреб користувача, бюджету та рівня технічних навичок. При створенні онлайн магазину, важливо враховувати спрощений інтерфейс та зручність використання для користувачів.

2.4 Вимоги до проектування інформаційної системи

На основі аналізу наявних інструментів для інтернет-магазину книг, ми можемо сформулювати технічне завдання для розробки спеціального інформаційного забезпечення. Це програмне забезпечення призначене для оптимізації управління та обслуговування онлайн магазину. Задача має наступний характер:

Розробка інформаційної системи для ефективного управління онлайн магазином книг з розширеним функціоналом для забезпечення зручності покупців та оптимізації внутрішніх процесів. Основні вимоги до системи такі:

1. Каталог товарів: Система повинна забезпечувати можливість додавання, редагування та видалення книг з каталогу, включаючи опис, ціну, обкладинку та інші характеристики.

2. Оформлення замовлень: Користувачі мають змогу додавати книги до кошика, оформляти замовлення та обирати зручний метод оплати та доставки.

3. Управління замовленнями: Адміністратори мають можливість переглядати, виконувати та відстежувати статуси замовлень, а також керувати історією замовлень користувачів.

4. Авторизація та реєстрація користувачів: Система повинна забезпечити можливість реєстрації нових користувачів та авторизації вже існуючих для здійснення покупок та перегляду історії замовлень.

5. Спеціальні пропозиції та рекомендації: Система може пропонувати рекомендації та спеціальні пропозиції для користувачів.

2.5 Вибір методу рішення

В процесі розробки інформаційної системи онлайна магазину книг було прийнято ряд ключових рішень з вибору технологій та методів, які гармонізувалися з вимогами проекту та допомогли досягти поставлених цілей.

Так як ми маємо обмежений бюджет та достатньо часу було вирішено створювати сайт власноруч. Основними аспектами вибору методів рішення були:

Вибір веб-технологій: Для розробки фронтенду веб-додатка було використано HTML, CSS і JavaScript. Використання HTML забезпечує структуру та розміщення елементів на сторінці, CSS - стилізацію та вигляд, а JavaScript - динаміку та можливість взаємодії з користувачем. Це дозволило створити дружній та легкий у використанні інтерфейс.

Вибір мови програмування: Для реалізації бекенду веб-додатка та взаємодії з базою даних була обрана мова програмування Java з фреймворком Spring[9] та Thymeleaf[11]. Spring це фреймворк та інструментарій для розробки Java-додатків. Він надає широкий спектр функцій та можливостей, спрощуючи процес розробки, підтримки та розгортання програмного забезпечення.

Thymeleaf - це шаблонний двигун для веб-додатків у Java, який використовується для створення HTML-сторінок з динамічним вмістом.

При виборі СУБД (системи управління базами даних) для вашого проекту важливо враховувати декілька факторів, таких як тип даних, обсяг і швидкодія доступу до даних, потреби в масштабованості, вартість, командна експертиза та інші особливості проекту. Ось деякі з найпопулярніших SQL СУБД та їхні особливості:

- MySQL: Це одна з найпоширеніших відкритих SQL-баз даних. Вона має широку підтримку, велику спільноту користувачів і відносно низьку вартість. MySQL підходить для багатьох типів проектів, особливо для веб-застосунків та додатків з великим обсягом читань.
- PostgreSQL: Це міцна, потужна та відкрита СУБД з високим рівнем стандартів SQL-сумісності. PostgreSQL має широкий набір функцій, таких як високорівнева підтримка транзакцій, відправлення журналу та геопросторові можливості. Вона підходить для проектів, де важлива надійність даних та потужність.
- Microsoft SQL Server: Це комерційна СУБД, розроблена Microsoft, яка пропонує широкі можливості для управління даними, включаючи бізнес-аналітику, звіти та інтеграцію з іншими продуктами Microsoft. Це може бути хорошим вибором для проектів, що використовують інші продукти Microsoft або потребують високого рівня безпеки.
- SQLite: Це легка, вбудована СУБД, яка не потребує окремого серверу. Вона ідеально підходить для малих проектів, використання в мобільних додатках або в тих випадках, коли не потрібні великі обсяги даних або високі навантаження.
- Oracle Database: Це масштабована комерційна СУБД з великим набором функцій і високою продуктивністю. Oracle часто використовується в корпоративних середовищах, де вимоги до масштабованості і надійності є критичними.

Вибір між цими СУБД залежить від конкретних потреб проекту. Наприклад, для простого веб-сайту, то MySQL або PostgreSQL можуть бути добрими виборами. Якщо це корпоративний додаток, то Microsoft SQL Server або Oracle можуть виявитися більш підходящими. Для нашого веб-додатку для зберігання інформації було обрано PostgreSQL як СУБД.

Також під час розробки буде використовуватись Git. Git - це розподілена система керування версіями, що використовується для відстеження змін у програмному коді під час розробки програмного забезпечення. Повний програмний код розміщено в git-репозиторії[12].

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Опис бізнес логіки

Для нашого інтернет магазину була розроблена така бізнелогіка:

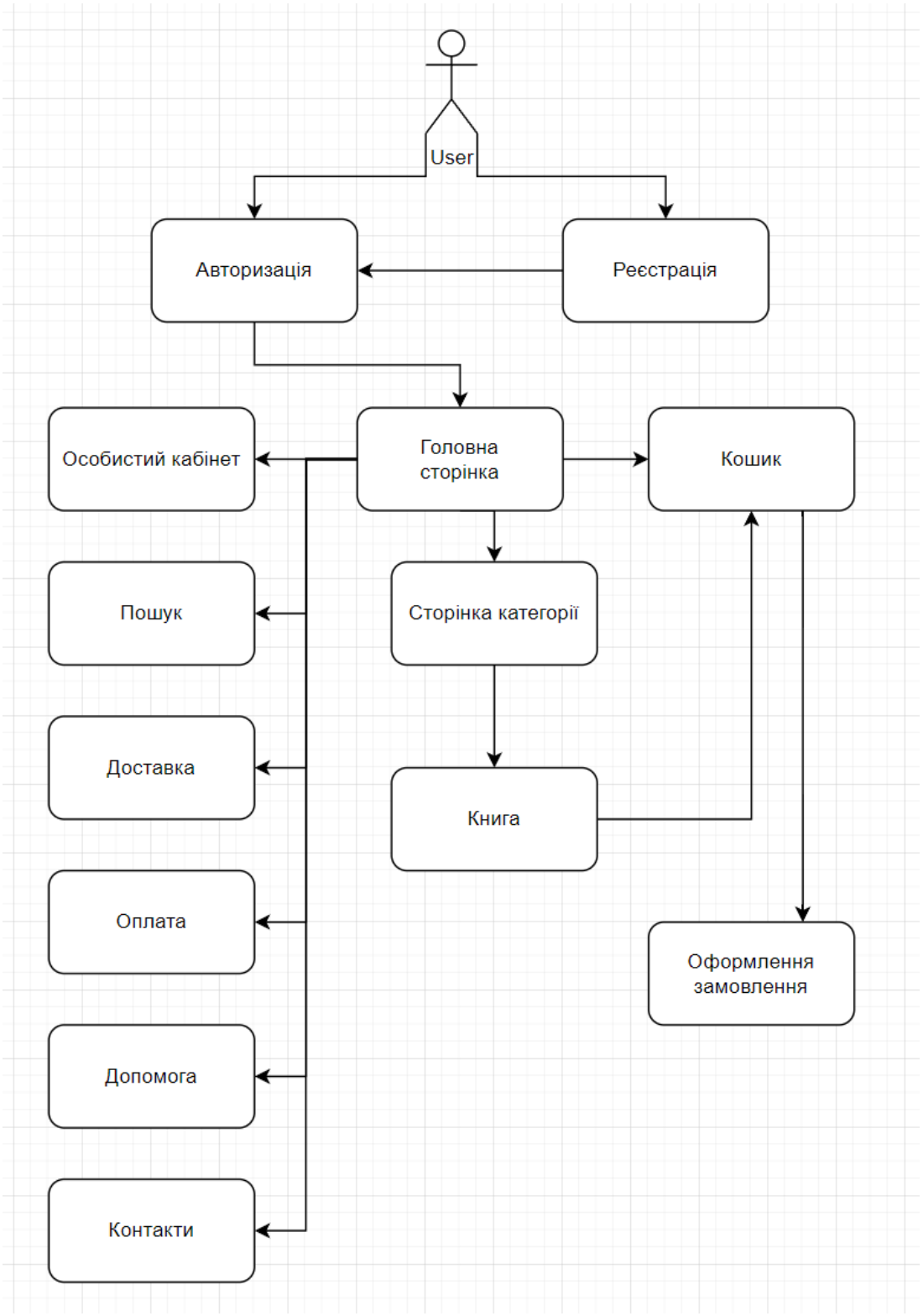

Рисунок 3.1 - Бізнес-логіка веб-додатку

На основі наданої діаграми бізнес-логіки для інтернет-магазину книг, можна зробити такі висновки:

1. Користувач має можливість:
    - Авторизуватися або зареєструватися в системі.
    - Після авторизації доступні: особистий кабінет, головна сторінка з каталогом книг, сторінка категорій, перегляд книг, кошик для покупок та оформлення замовлення.
2. Головна сторінка:
    - Дозволяє доступ до кошика та сторінки категорій.
    - З головної сторінки можна перейти до особистого кабінету (якщо авторизований).
3. Сторінка категорій:
    - Відображає список книг за категоріями.
    - З цієї сторінки можна перейти до перегляду детальної інформації про книгу.
4. Книга:
    - Користувач може додавати книги до кошика.
    - Після додавання до кошика можна перейти до оформлення замовлення.
5. Оформлення замовлення:
    - Включає етапи вибору доставки та оплати.
    - Після завершення замовлення інформація зберігається у системі.
6. Особистий кабінет:
    - Містить опції для перегляду історії замовлень.

Діаграма бізнес-логіки представляє інтуїтивно зрозумілу структуру для користувача, де чітко визначені основні кроки для роботи з інтернет-магазином, від реєстрації до оформлення замовлення та управління особистими даними.

3.2 Проектування та нормалізація бази даних.

Для зберігання інформації було обрано PostgreSQL як СКБД. PostgreSQL - це безкоштовна та дуже поширена СКБД, яка надійно зберігає дані та надає швидкий доступ до них.

Була спроектована база даних та приведена до 3 номальної форми[10]:

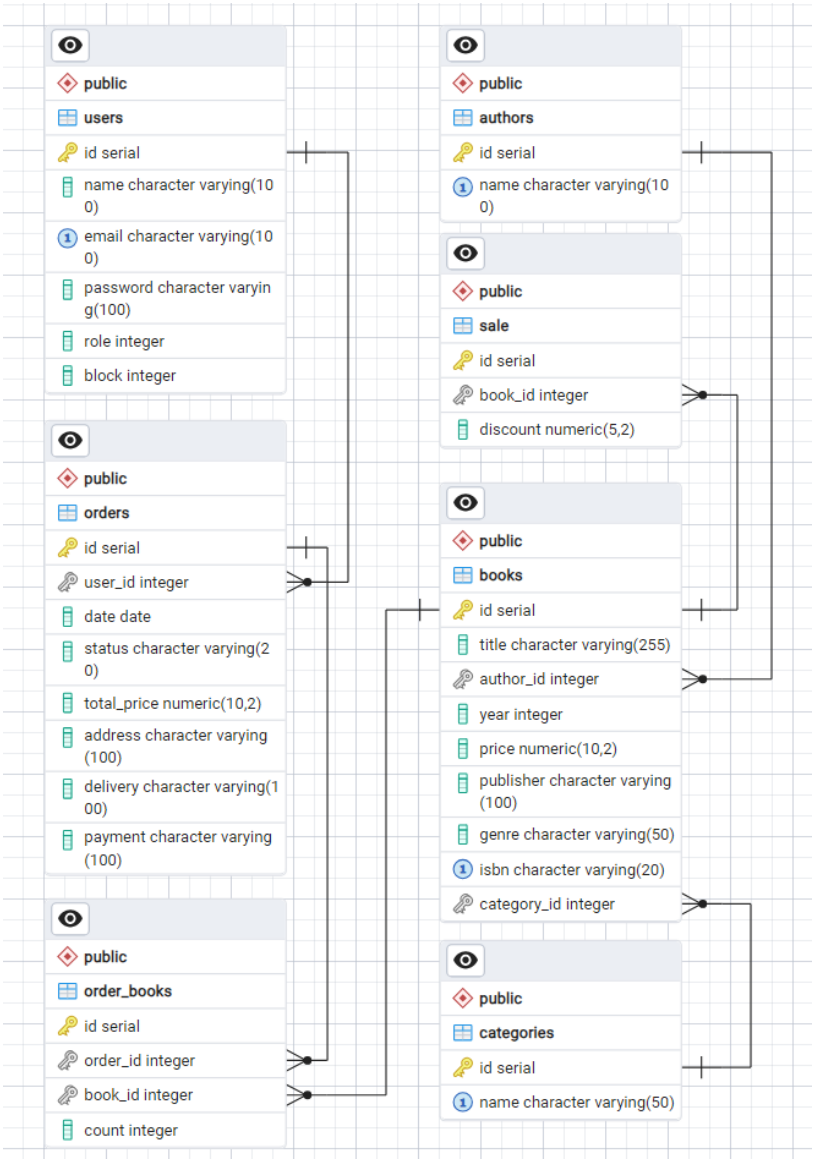

Рисунок 3.2 - Схема бази даних

Давайте розглянемо діаграму ERD:

Сутності:

- `authors`: Представляє авторів книг.
- `categories`: Визначає категорії книг.
- `users`: Представляє користувачів системи.
- `books`: Представляє книги, що доступні для покупки.
- `orders`: Представляє замовлення, зроблені користувачами.
- `order_books`: Визначає зв'язок між замовленнями та книгами, що були замовлені.
- `sale`: Визначає знижки на книги.

Відносини:

- Відношення між `authors` та `books` є один до багатьох, оскільки один автор може мати багато книг, але кожна книга має лише одного автора.
- Відношення між `categories` та `books` також є один до багатьох, оскільки кожна категорія може мати багато книг, але кожна книга може належати лише до однієї категорії.
- У таблиці `orders` є відношення багато до одного з `users`, оскільки кожне замовлення належить лише одному користувачеві, але один користувач може мати багато замовлень.
- В таблиці `order_books` відношення багато до одного як до `orders`, так і до `books`, оскільки кожне замовлення може містити багато книг, а кожна книга може бути частиною багатьох замовлень.

Атрибути:

У кожній таблиці перераховані атрибути, такі як `id`, який є унікальним ідентифікатором, а також інші атрибути, такі як `name`, `title`, `year` та інші, які характеризують кожну сутність.

Отже, ми можемо побачити, що діаграма ERD буде містити сутності `authors`, `categories`, `users`, `books`, `orders`, `order_books` та `sale`, з відповідними відносинами між ними, як описано вище.

На основі наданої діаграми бази даних (ERD), наведеної на зображенні, логічна модель бази даних може бути представлена у вигляді таблиць з відповідними полями та їхніми типами даних.

Таблиця 3.1 Табиця users

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор користувача
name	VARCHAR(100)	NOT NULL	Ім'я користувача
email	VARCHAR(100)	UNIQUE, NOT NULL	Електронна пошта користувача
password	VARCHAR(100)	NOT NULL	Пароль користувача
role	INTEGER	CHECK (role >= 0)	Роль користувача
block	INTEGER	DEFAULT 0	Статус блокування користувача

Таблиця 3.2 Таблиця authors

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор автора
name	VARCHAR(100)	UNIQUE, NOT NULL	Ім'я автора

Таблиця 3.3 Таблиця categories

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор категорії
name	VARCHAR(50)	UNIQUE, NOT NULL	Назва категорії

Таблиця 3.4 Таблиця books

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор книги
title	VARCHAR(255)	NOT NULL	Назва книги
author_id	INTEGER	REFERENCES authors(id) ON DELETE SET NULL	Ідентифікатор автора

Продовження таблиці 3.4

year	INTEGER	CHECK (year >= 0)	Рік видання
price	NUMERIC(10,2)	CHECK (price >= 0)	Ціна книги
publisher	VARCHAR(100)	NOT NULL	Видавець
genre	VARCHAR(50)	NOT NULL	Жанр
isbn	VARCHAR(20)	UNIQUE, NOT NULL	ISBN
category_id	INTEGER	REFERENCES categories(id) ON DELETE SET NULL	Ідентифікатор категорії

Таблиця 3.5 Таблиця orders

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор замовлення
user_id	INTEGER	REFERENCES users(id) ON DELETE CASCADE	Ідентифікатор користувача
date	DATE	NOT NULL	Дата замовлення
status	VARCHAR(20)	NOT NULL	Статус замовлення

Продовження таблиці 3.5

total_price	NUMERIC(10,2)	CHECK (total_price >= 0)	Загальна сума замовлення
address	VARCHAR(100)	NOT NULL	Адреса доставки
delivery	VARCHAR(100)	NOT NULL	Спосіб доставки
payment	VARCHAR(100)	NOT NULL	Спосіб оплати

Таблиця 3.6 Таблиця order_books

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор запису
order_id	INTEGER	REFERENCES orders(id) ON DELETE CASCADE	Ідентифікатор замовлення
book_id	INTEGER	REFERENCES books(id) ON DELETE CASCADE	Ідентифікатор книги
count	INTEGER	CHECK (count > 0)	Кількість

Таблиця 3.7 Таблиця sale

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор знижки
book_id	INTEGER	REFERENCES books(id) ON DELETE CASCADE	Ідентифікатор книги
discount	NUMERIC(5,2)	CHECK (discount >= 0 AND discount <= 100)	Величина знижки

Ця логічна модель відображає структуру бази даних з усіма необхідними полями, типовими зв'язками між таблицями (Foreign Key), та основними типами даних для кожного поля.

Під час розробки веб-додатку будемро використовувати паттерн MVC(Model-View-Controller). MVC це архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

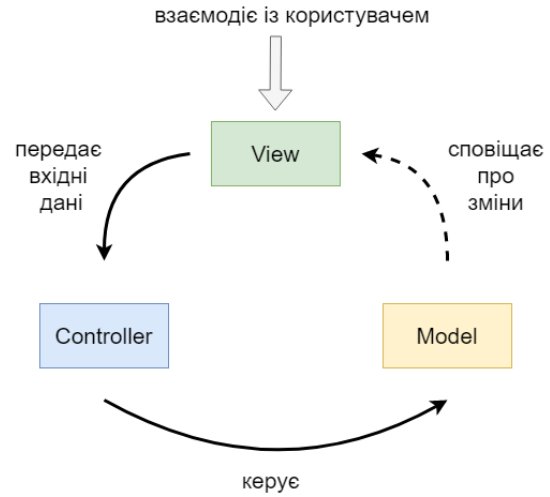

Рисунок 3.3 - MVC

3.3 Опис програмної реалізації

Архітектура інтернет-магазину складається з трьох основних компонентів: клієнтська частина (frontend), серверна частина (backend) і база даних.

1. Клієнтська частина (Frontend): Клієнтська частина розроблена з використанням сучасних веб-технологій, таких як HTML, CSS, JavaScript, та шаблонізатора Thymeleaf. Основні функції клієнтської частини включають:

   - Інтерфейс користувача для перегляду та пошуку книг.
   - Механізм авторизації та реєстрації користувачів.
   - Кошик покупок та оформлення замовлень.
   - Сторінки для перегляду інформації про книги, авторів та категорії.

2. Серверна частина (Backend): Серверна частина реалізована на основі фреймворку Spring Boot. Основні функції серверної частини включають:

   - Обробка запитів від клієнта та взаємодія з базою даних.
   - Реалізація логіки авторизації та управління сесіями користувачів.
   - Обробка замовлень, включаючи збереження даних про замовлення та оновлення статусу.
   - Інтеграція з платіжними системами для обробки транзакцій.

Реалізація функціоналу:

1. Реєстрація та авторизація користувачів: Реалізовано систему реєстрації нових користувачів з перевіркою унікальності електронної

пошти та безпечним збереженням паролів з використанням хешування. Система авторизації забезпечує безпечний вхід користувачів до їх особистих кабінетів.

2. Кошик та оформлення замовлень: Користувачі можуть додавати книги до кошика, переглядати його вміст, змінювати кількість товарів та оформлювати замовлення. Під час оформлення замовлення користувач вибирає спосіб доставки та оплати.

3. Управління контентом: Адміністратори мають можливість додавати, змінювати та видаляти книги, авторів та категорії, що дозволяє динамічно оновлювати контент сайту.

4. Пошук та фільтрація: Реалізовано механізм пошуку книг за ключовими словами та фільтрації за категоріями, що дозволяє користувачам швидко знаходити потрібні товари.

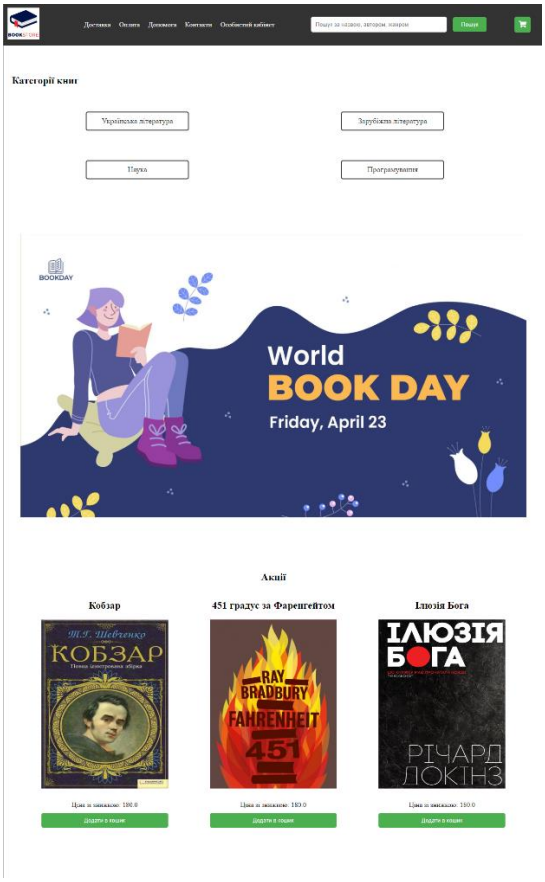

Рисунок 3.4 - Головна сторінка

Головна сторінка є першою точкою взаємодії користувача з сайтом. Вона містить:

- Логотип компанії, який слугує посиланням на цю сторінку з будь-якого місця сайту.
- Поле пошуку для швидкого знаходження книг за ключовими словами.
- Навігаційне меню для доступу до основних розділів сайту, таких як категорії, кошик, і особистий кабінет.
- Вітальне повідомлення або актуальні пропозиції, акції.
- Перелік популярних або нових книг, що може зацікавити користувача.

Головна сторінка розроблена так, щоб забезпечити користувачу зручний доступ до всіх основних функцій сайту та заохотити подальшу взаємодію.

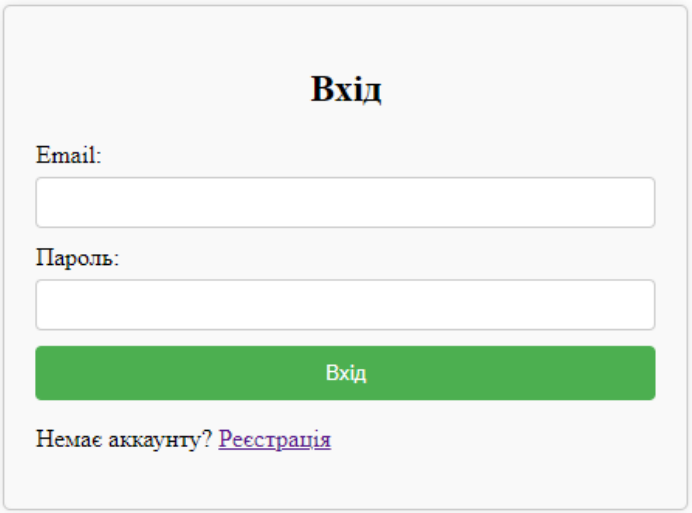

Рисунок 3.5 - Сторінка авторизації

Ця сторінка потрібна для входу користувачів до їхніх облікових записів. Вона містить:

- Поля для введення електронної пошти та пароля.
- Кнопку для входу в систему.
- Посилання на сторінку відновлення пароля у випадку його втрати.
- Посилання на сторінку реєстрації для нових користувачів.

Сторінка авторизації забезпечує безпечний вхід користувачів і дозволяє їм отримати доступ до персоналізованих функцій сайту, таких як перегляд замовлень або оформлення нових.

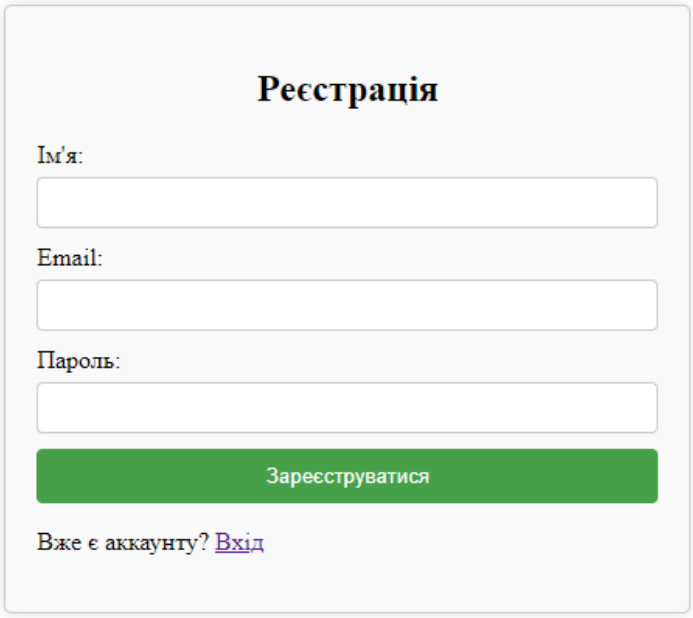

Рисунок 3.6 - Сторінка реєстрації

На цій сторінці нові користувачі можуть створити свій обліковий запис. Вона включає:

- Поля для введення необхідної інформації: ім'я, електронна пошта, пароль.
- Кнопку для завершення реєстрації.
- Інформацію про захист даних користувача, включаючи політику конфіденційності.

Реєстрація є простою і швидкою, забезпечуючи користувачам можливість швидко почати користуватися всіма функціями сайту.

Рисунок 3.7 - Сторінка особистого кабінету

Ця сторінка надає користувачам доступ до їхньої особистої інформації та історії замовлень. Вона містить:

- Список минулих замовлень з детальною інформацією про кожне з них.
- Можливість змінювати особисті дані, такі як адреса чи спосіб оплати.
- Налаштування облікового запису, включаючи зміну пароля.

Особистий кабінет забезпечує зручний інтерфейс для управління всім, що стосується користувацького досвіду на сайті.

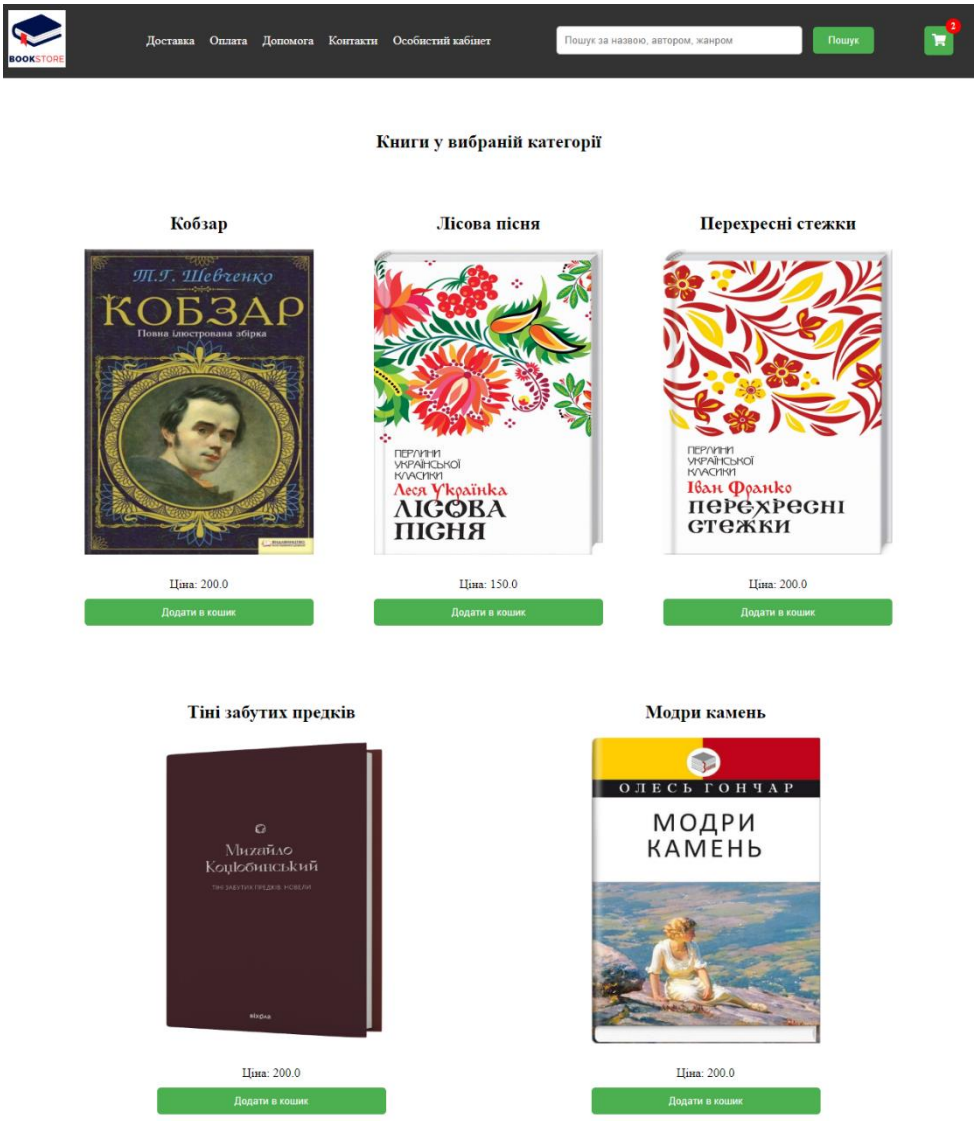

Рисунок 3.8 - Сторінка категорії

На цій сторінці користувач може переглядати книги певної категорії. Вона включає:

- Список книг з коротким описом та ціною.
- Фільтри для уточнення пошуку за різними параметрами (наприклад, ціна, автор).
- Можливість додавання книг до кошика або переходу на сторінку детальної інформації про книгу.

Сторінка категорій допомагає користувачам швидко знаходити книги, що відповідають їхнім інтересам.

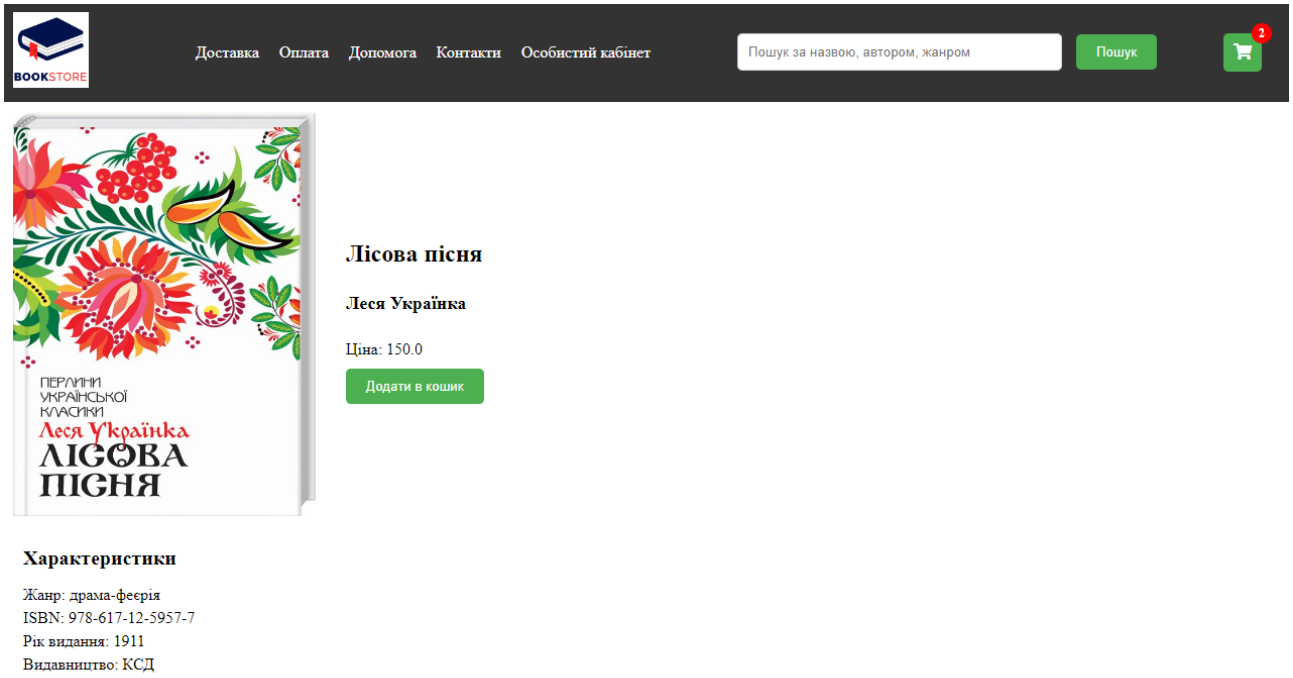

Рисунок 3.9 - Сторінка книги

Ця сторінка надає детальну інформацію про конкретну книгу, включаючи:

- Назву, автора, рік видання, жанр, ціну.
- Опис книги та відгуки інших користувачів.
- Кнопки для додавання книги до кошика або купівлі зараз.
- Рекомендації схожих книг.

Детальна сторінка книги допомагає користувачам прийняти рішення про покупку, надаючи всю необхідну інформацію.

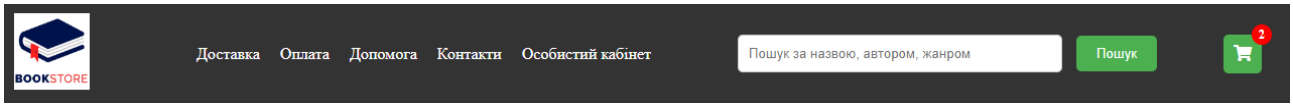

Рисунок 3.10 - Хедер

Хедер розташований на верхній частині кожної сторінки (крім сторінок реєстрації та входу) і включає:

- Логотип компанії.
- Поле пошуку.
- Посилання на основні розділи сайту: головна сторінка, категорії, кошик, особистий кабінет.
- Іконку кошика з кількістю доданих товарів.

Хедер забезпечує швидкий доступ до ключових функцій сайту, полегшуючи навігацію.

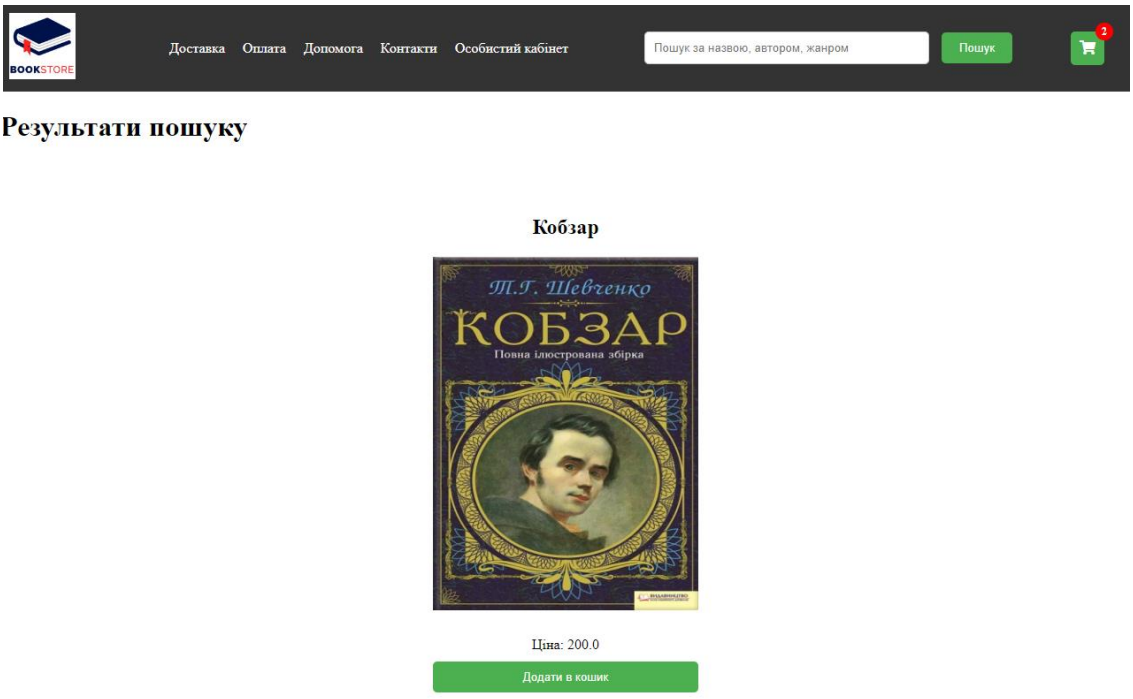

Рисунок 3.11 - Сторінка пошуку

Ця сторінка відображає результати пошуку книг за введеним запитом. Вона включає:

- Список книг, що відповідають пошуковому запиту.
- Можливість фільтрувати результати за різними параметрами.
- Переходи до сторінок книг для більш детальної інформації.

Сторінка пошуку допомагає користувачам швидко знаходити потрібні книги, покращуючи їхній досвід користування сайтом.

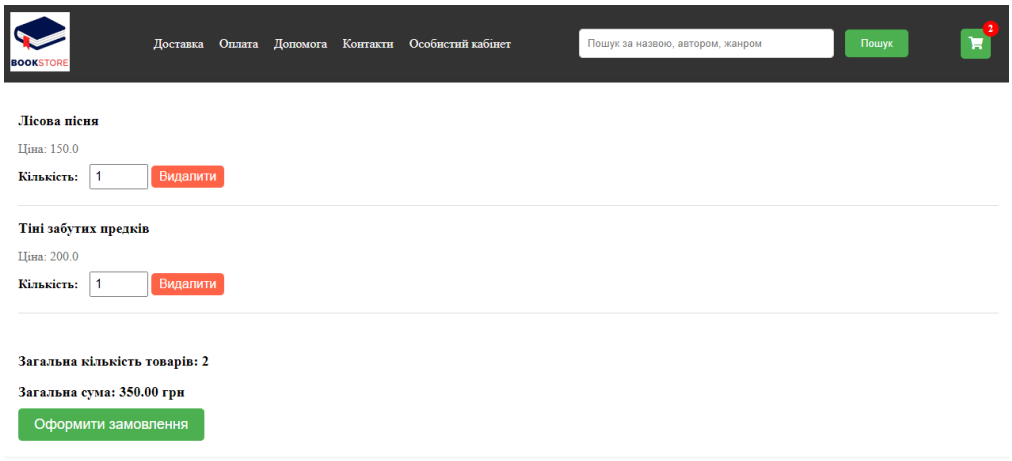

Рисунок 3.12 - Сторінка кошику

На цій сторінці користувачі можуть переглядати вибрані книги, змінювати їх кількість, видаляти товари та оформлювати замовлення. Вона включає:

- Список вибраних книг із загальною сумою.
- Кнопку для переходу до оформлення замовлення.
- Опції для зміни кількості товарів або видалення книг з кошика.

Кошик є зручною точкою для управління покупками перед остаточним оформленням замовлення.

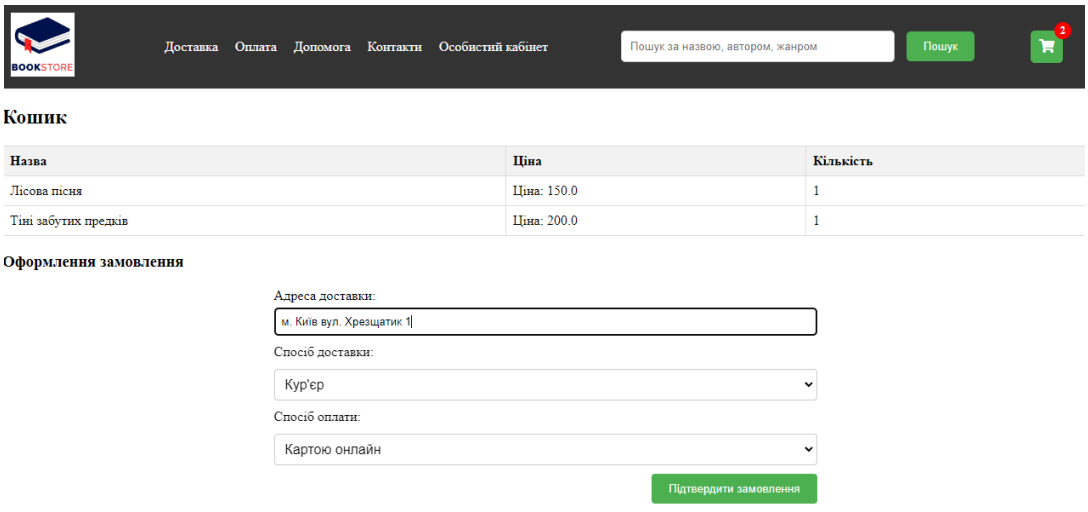

Рисунок 3.13 - Сторінка оформлення замовлення

Ця сторінка є завершальним етапом процесу покупки. Вона включає:

- Остаточний список товарів.
- Поля для введення адреси доставки.
- Вибір способу доставки та оплати.
- Кнопку підтвердження замовлення.

Сторінка оформлення замовлення гарантує, що всі деталі замовлення введені правильно перед його підтвердженням.

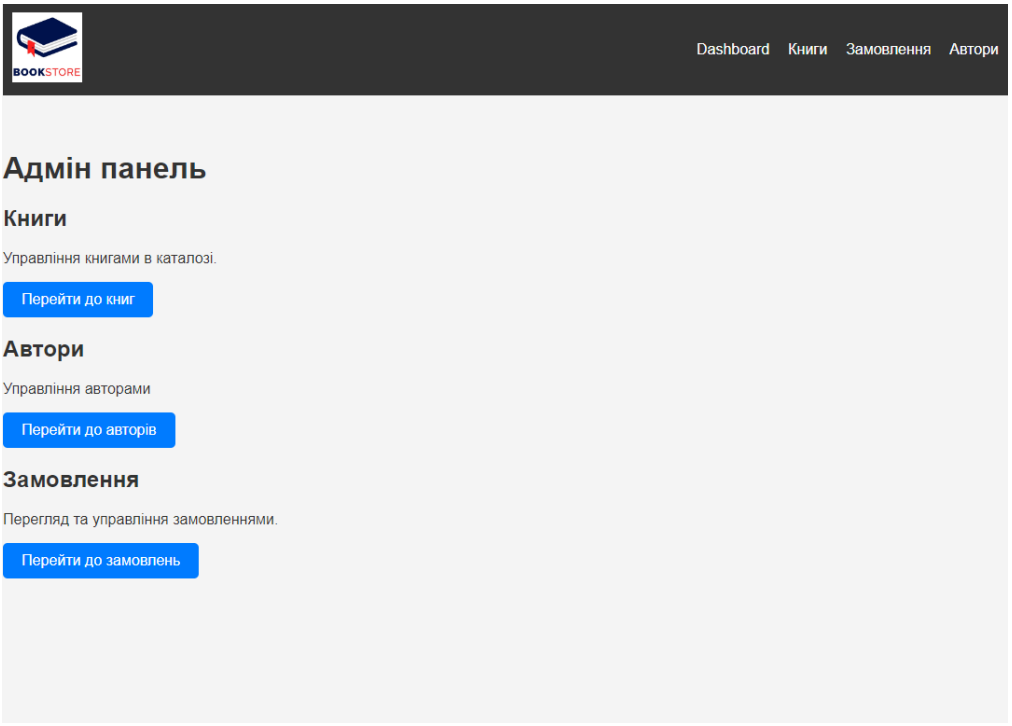

Рисунок 3.14 - Головна сторінка адмін-панелі

Ця сторінка призначена для адміністраторів сайту. Вона включає:

- Інструменти для управління контентом: додавання, редагування, видалення книг, авторів, категорій.
- Управління користувачами та їхніми замовленнями.
- Аналітичні дані про продажі та активність користувачів.

Адмін-панель забезпечує динамічне управління сайтом, дозволяючи підтримувати актуальність та якість контенту.

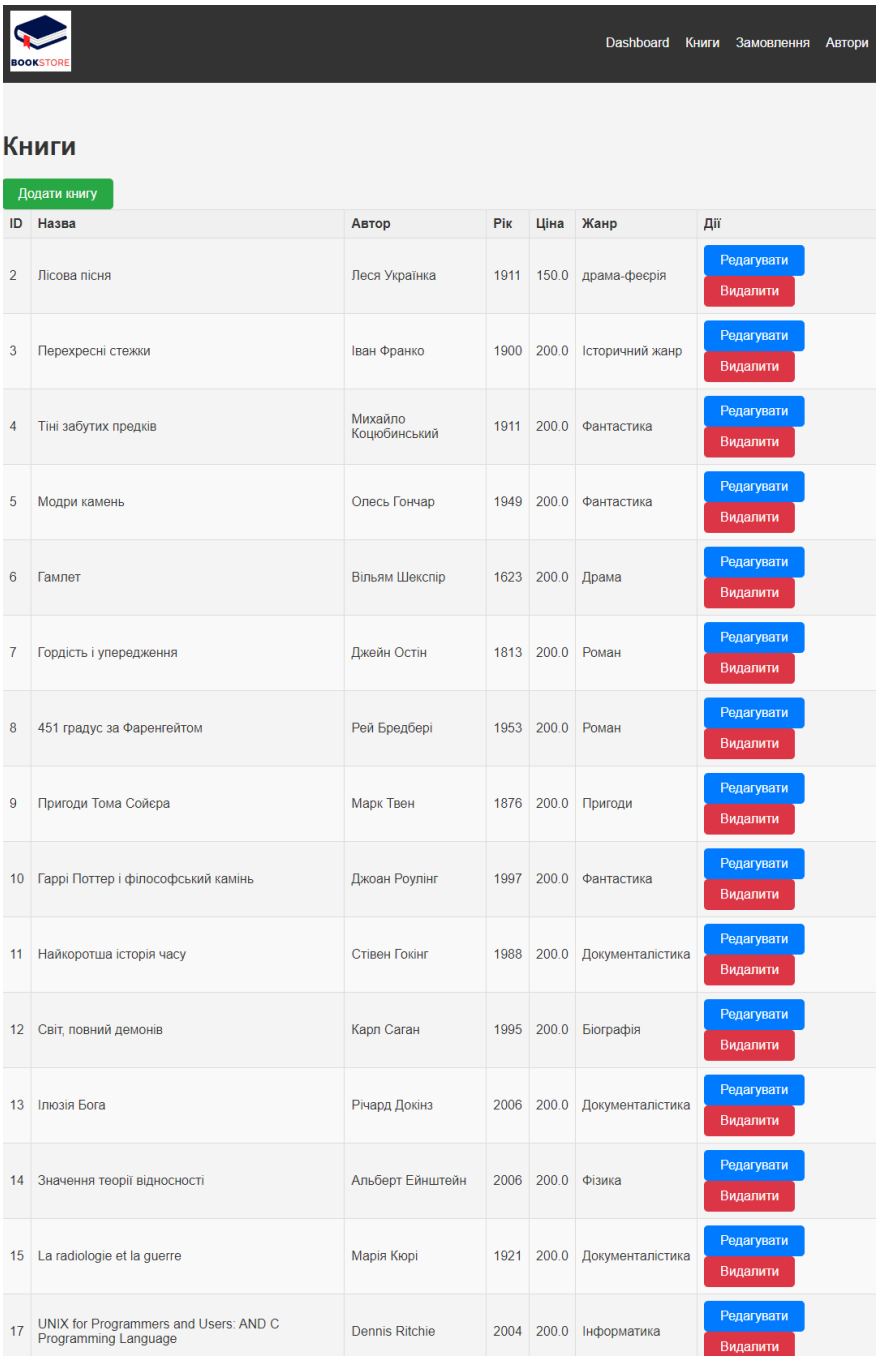

Рисунок 3.15 - Сторінка керуванням книг

Сторінка керуванням книг є важливою частиною адмін-панелі інтернет-магазину. Вона дозволяє адміністраторам здійснювати базові CRUD-операції (створення, читання, оновлення, видалення) з інформацією про книги.

- Додавання нової книги: Адміністратор може заповнити форму, де вказує назву книги, автора, рік видання, ціну, видавця, жанр, ISBN та категорію. Після заповнення необхідних полів книга додається до бази даних.

- Редагування існуючої книги: Адміністратори можуть вибрати книгу зі списку та внести зміни до її інформації. Це дозволяє підтримувати актуальність даних, таких як ціни або доступність.

- Видалення книги: Адміністратори можуть видаляти книги, які більше не доступні для продажу або застарілі.

Рисунок 3.16 - Форма редагування книги

Форма редагування книги є складовою частиною сторінки керуванням книг. Вона дозволяє адміністраторам змінювати деталі конкретної книги, що вже є в базі даних.

- Поля форми: Адміністратори можуть редагувати назву книги, автора, рік видання, ціну, видавця, жанр, ISBN та категорію. Кожне поле може бути змінено відповідно до потреб.

- Збереження змін: Після внесення змін адміністратор може зберегти їх, що оновить інформацію в базі даних.

- Перегляд поточних даних: Форма дозволяє переглядати поточні дані, які знаходяться в базі, що полегшує процес редагування та забезпечує точність.

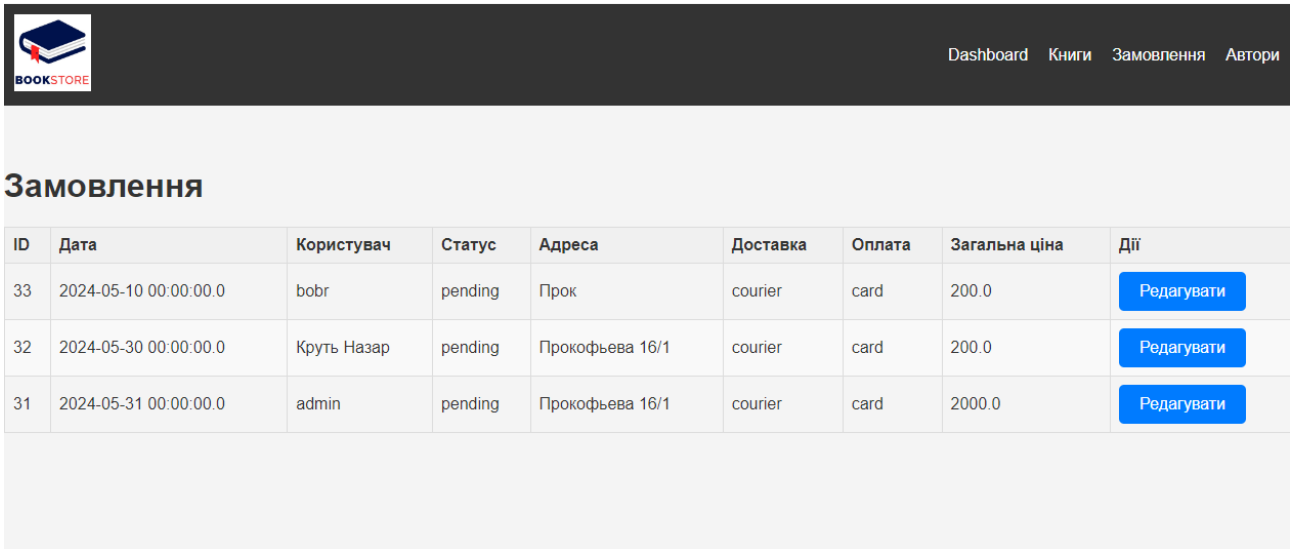

Рисунок 3.17 - Сторінка замовлень

Сторінка замовлень є ключовою для адміністраторів, які відповідають за обробку та управління замовленнями.

- Перегляд замовлень: На цій сторінці адміністратори можуть бачити список всіх замовлень, розміщених користувачами. Кожне замовлення включає інформацію про користувача, дату замовлення, статус, адресу доставки, метод оплати та загальну суму.

- Редагування замовлень: Адміністратори можуть змінювати статус замовлення (наприклад, з "в обробці" на "відправлено" або "доставлено"), що допомагає підтримувати точний облік і забезпечує інформування користувачів про стан їхніх замовлень.

- Деталі замовлення: Адміністратори можуть переглядати деталі кожного замовлення, включаючи список замовлених книг та їх кількість. Це допомагає в управлінні запасами і підготовці замовлень до відправлення.

ВИСНОВКИ

У ході дослідження було розглянуто та вирішено низку теоретичних і практичних завдань, пов'язаних із розробкою інтернет-магазину книг. Зокрема, автором було досягнуто наступних результатів:

- Аналіз вимог та існуючих рішень: Було здійснено глибокий аналіз існуючих інтернет-магазинів книг, визначено ключові функціональні та нефункціональні вимоги до системи, що дозволило розробити проект, який відповідає сучасним стандартам та очікуванням користувачів.
- Огляд технологій, що використовуються для створення інформаційної системи онлайн-магазину книг: Було проведено ретельний аналіз вибору технологій для розробки інформаційної системи. Для фронтенду було використано HTML, CSS та JavaScript, що забезпечує дружній та легкий у використанні інтерфейс. Для бекенду обрано мову програмування Java з фреймворком Spring та шаблонним двигуном Thymeleaf, що сприяє створенню динамічних веб-сторінок. Також було визначено оптимальні СУБД для ефективного зберігання та доступу до даних.
- Проектування системи: Було створено логічну модель бази даних, яка забезпечує ефективне зберігання та доступ до інформації про книги, користувачів, замовлення та інші сутності. Розроблено архітектуру системи, що включає серверну та клієнтську частини, інтегровані між собою для забезпечення безперервної та стабільної роботи.
- Розробка та реалізація: Виконано реалізацію інтерфейсу користувача, який забезпечує зручність та інтуїтивність роботи з системою. Реалізовано функціонал, необхідний для реєстрації та авторизації користувачів, управління кошиком, оформлення замовлень, вибору способу доставки та оплати.

Подальші дослідження за темою роботи можуть бути спрямовані на такі напрямки:

- Розширення функціоналу: Додавання нових функцій, таких як рекомендаційні системи, що надають персоналізовані пропозиції книг користувачам на основі їхніх попередніх покупок та переглядів.
- Інтеграція з іншими сервісами: Інтеграція інтернет-магазину з зовнішніми сервісами, такими як соціальні мережі, платіжні системи, служби доставки, що дозволить підвищити зручність та функціональність системи.
- Підвищення масштабованості: Дослідження та впровадження нових технологій, які забезпечать масштабованість системи для роботи з великим обсягом даних та великою кількістю користувачів.
- Аналіз даних та бізнес-аналітика: Використання методів машинного навчання та аналізу даних для покращення роботи системи, підвищення точності рекомендацій та оптимізації бізнес-процесів.

Результати дослідження та розроблена система мають великий потенціал для подальшого розвитку та вдосконалення, що сприятиме підвищенню ефективності електронної комерції та задоволенню потреб сучасних користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Statista. (2023). Book Market - Statistics & Facts - https://www.statista.com/topics/1177/book-market/ (дата звернення 03.05.2024)
2. McKinsey & Company. (2021). The future of the shopping mall - https://www.mckinsey.com/industries/retail/our-insights/the-future-of-the-shopping-mall (дата звернення 03.05.2024)
3. Інтернет магазин КСД. - https://bookclub.ua/ (дата звернення 03.05.2024)
4. Інтернет магазин Yakaboo - https://www.yakaboo.ua/ (дата звернення 03.05.2024)
5. Інтернет магазин Книгарня Є - https://book-ye.com.ua/ (дата звернення 03.05.2024)
6. Платформа Shopify Ecommerce Platform - https://www.shopify.com/ (дата звернення 03.05.2024)
7. Платформа OpenCart - https://www.opencart.com/ (дата звернення 03.05.2024)
8. Miller Brian D., Ackerman Jason. Principles of Web Design. - 3rd Edition. — Allworth, 2022. — 280 p.
9. Spring Framework Documentation - https://docs.spring.io/spring-framework/reference/ (дата звернення 15.04.2024)
10. Coronel Carlos, Morris Steven. Database Systems: Design, Implementation, and Management. - 14th Edition. — Cengage Learning, 2023. — 818
11. Thymeleaf documentation - https://www.thymeleaf.org/documentation.html (дата звернення 25.04.2024)
12. Повний код на GitHub - https://github.com/Balinvax/BookStore.git (дата звернення 05.05.2024)

ДОДАТОК

BookStoreApplication.java:

```
package com.krutn.bookstore;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BookStoreApplication {

    public static void main(String[] args) {
        SpringApplication.run(BookStoreApplication.class, args);
    }
}
```

SecurityConfig.java:

```
package com.krutn.bookstore.configuration;

import com.krutn.bookstore.repository.UserRepository;
import com.krutn.bookstore.service.CustomUserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.authentication.configuration.Authent
icationConfiguration;
```

```
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecuri
ty;
import
org.springframework.security.config.annotation.web.configurers.LogoutConfigurer;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

   @Autowired
   private UserRepository userRepository;

   @Bean
   public UserDetailsService userDetailsService(UserRepository
userRepository) {
      return new CustomUserDetailsService(userRepository);
   }

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

```
    @Bean
    public AuthenticationManager authenticationManagerBean(AuthenticationConfiguration authenticationConfiguration) throws Exception {
        return authenticationConfiguration.getAuthenticationManager();
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests((requests) -> requests
                .requestMatchers("/admin/**").hasAuthority("ADMIN")
                .requestMatchers("/", "/login", "/home", "/registration", "/cart", "/category/**",
                        "/book/**","/contacts", "/search", "/delivery", "/help",
                        "/payment", "/css/**", "/js/**", "/images/**").permitAll()
                .anyRequest().authenticated()
            )
            .formLogin((form) -> form
                .loginPage("/login")
                .usernameParameter("email") // Тут "email" - ім'я поля у вашій формі логіну
                .passwordParameter("password")
                .defaultSuccessUrl("/", true)
                .permitAll()
            )
            .logout(LogoutConfigurer::permitAll)
            .userDetailsService(userDetailsService(userRepository)); // Використовуйте правильний бін
```

```
            return http.build();
        }
}
```

AdminController.java:

```
package com.krutn.bookstore.controller;

import com.krutn.bookstore.entity.Book;
import com.krutn.bookstore.entity.Author;
import com.krutn.bookstore.entity.Order;
import com.krutn.bookstore.repository.OrderRepository;
import com.krutn.bookstore.service.AuthorService;
import com.krutn.bookstore.service.BookService;
import com.krutn.bookstore.service.CategoryService;
import com.krutn.bookstore.service.OrderService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.List;

@Controller
@RequestMapping("/admin")
@PreAuthorize("hasRole('ADMIN')")
public class AdminController {
```

```
    @Autowired
    BookService bookService;

    @Autowired
    OrderService orderService;

    @Autowired
    AuthorService authorService;

    @Autowired
    CategoryService categoryService;

    @Autowired
    private OrderRepository orderRepository;


    @GetMapping("/dashboard")
    public String adminDashboard(Model model) {
        // Передаємо в модель списки книг та замовлень для відображення на головній сторінці адміністратора
        model.addAttribute("books", bookService.getAllBooks());
        model.addAttribute("orders", orderService.findAll());
        return "admin/dashboard";
    }

    @GetMapping("/books")
    public String listBooks(Model model) {
        model.addAttribute("books", bookService.getAllBooks());
        return "admin/books";
```

```
    }

    @GetMapping("/books/add")
    public String showAddBookForm(Model model) {
        model.addAttribute("book", new Book());
        model.addAttribute("authors", authorService.getAllAuthors());
        model.addAttribute("categories", categoryService.getAllCategories());
        return "admin/add-book";
    }

    @PostMapping("/books/add")
    public String addBook(@ModelAttribute Book book, RedirectAttributes
redirectAttributes) {
        bookService.saveBook(book);
        redirectAttributes.addFlashAttribute("message", "Книгу успішно
додано!");
        return "redirect:/admin/books";
    }

    @GetMapping("/books/edit/{id}")
    public String showEditBookForm(@PathVariable Long id, Model model) {
        Book book = bookService.getBookById(id);
        if (book == null) {
            model.addAttribute("error", "Книга не знайдена!");
            return "admin/books";
        }
        model.addAttribute("book", book);
        model.addAttribute("authors", authorService.getAllAuthors());
        model.addAttribute("categories", categoryService.getAllCategories());
        return "admin/edit-book";
```

```
    }

    @PostMapping("/books/update")
    public String updateBook(@ModelAttribute Book book, RedirectAttributes
redirectAttributes) {
        bookService.saveBook(book);
        redirectAttributes.addFlashAttribute("message", "Книгу успішно
оновлено!");
        return "redirect:/admin/books";
    }

    @GetMapping("/books/delete/{id}")
    public String deleteBook(@PathVariable Long id, RedirectAttributes
redirectAttributes) {
        bookService.deleteBook(id);
        redirectAttributes.addFlashAttribute("message", "Книгу успішно
видалено!");
        return "redirect:/admin/books";
    }

    @GetMapping("/orders")
    public String listOrders(Model model) {
        model.addAttribute("orders", orderService.findAll());
        return "admin/orders";
    }

    @GetMapping("/orders/edit/{id}")
    public String showEditOrderForm(@PathVariable Long id, Model model) {
        Order order = orderService.findById(id);
        if (order == null) {
```

```
            model.addAttribute("error", "Замовлення не знайдено!");
            return "admin/orders";
        }
        model.addAttribute("order", order);
        return "admin/edit-order";
    }

    @PostMapping("/orders/update")
    public String updateOrder(@ModelAttribute Order order, RedirectAttributes
redirectAttributes) {
        orderRepository.save(order);
        redirectAttributes.addFlashAttribute("message", "Статус замовлення
успішно оновлено!");
        return "redirect:/admin/orders";
    }

    @GetMapping("/authors")
    public String listAuthors(Model model) {
        model.addAttribute("authors", authorService.getAllAuthors());
        return "admin/authors";
    }

    @GetMapping("/authors/add")
    public String showAddAuthorForm(Model model) {
        model.addAttribute("author", new Author());
        return "admin/add-author";
    }

    @PostMapping("/authors/add")
```

```
    public String addAuthor(@ModelAttribute Author author, RedirectAttributes
redirectAttributes) {
        authorService.saveAuthor(author);
        redirectAttributes.addFlashAttribute("message", "Автор успішно
доданий!");
        return "redirect:/admin/authors";
    }

    @GetMapping("/authors/edit/{id}")
    public String showEditAuthorForm(@PathVariable Long id, Model model) {
        Author author = authorService.getAuthorById(id);
        if (author == null) {
            model.addAttribute("error", "Автор не знайдений!");
            return "admin/authors";
        }
        model.addAttribute("author", author);
        return "admin/edit-author";
    }

    @PostMapping("/authors/update")
    public String updateAuthor(@ModelAttribute Author author,
RedirectAttributes redirectAttributes) {
        authorService.saveAuthor(author);
        redirectAttributes.addFlashAttribute("message", "Автор успішно
оновлений!");
        return "redirect:/admin/authors";
    }

    @GetMapping("/authors/delete/{id}")
```

```
    public String deleteAuthor(@PathVariable Long id, RedirectAttributes redirectAttributes) {
        authorService.deleteAuthor(id);
        redirectAttributes.addFlashAttribute("message", "Автор успішно видалений!");
        return "redirect:/admin/authors";
    }
}
```

CategoryController.java:

```
    package com.krutn.bookstore.controller;

    import com.krutn.bookstore.entity.Book;
    import com.krutn.bookstore.service.BookService;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Controller;
    import org.springframework.ui.Model;
    import org.springframework.web.bind.annotation.*;

    import java.util.List;

    @Controller
    public class CategoryController {

        @Autowired
        private BookService BookService;

        @GetMapping("/category/{categoryId}")
        public String getCategoryBooks(@PathVariable Long categoryId, Model model) {
```

```
        // Отримати книги для даної категорії за її ідентифікатором
        List<Book> books = BookService.getBooksByCategoryId(categoryId);

        // Передати список книг у модель для відображення на сторінці
        model.addAttribute("books", books);

        return "categories";
    }
}
```

HomeController.java:

```
package com.krutn.bookstore.controller;

import com.krutn.bookstore.entity.Sale;
import com.krutn.bookstore.service.SaleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.List;

@Controller
public class HomeController {

    private final SaleService saleService;

    @Autowired
```

```
    public HomeController(SaleService saleService) {
        this.saleService = saleService;
    }

    @GetMapping("/")
    public String home(Model model) {
        // Отримуємо список акційних товарів
        List<Sale> sales = saleService.getAllSales();
        model.addAttribute("sales", sales); // Передаємо їх на головну сторінку

        return "home";
    }
}
```

LoginController.java:

```
package com.krutn.bookstore.controller;

import com.krutn.bookstore.service.LoginService;
import com.krutn.bookstore.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
@RequestMapping("/login")
public class LoginController {
```

```
    @Autowired
    private UserService userService;

    @Autowired
    private LoginService loginService;

    @GetMapping
    public String showLoginPage(@RequestParam(required = false) String error,
Model model) {
        if (error != null) {
            model.addAttribute("error", "Invalid email or password");
        }
        return "login";
    }

    @PostMapping
    public String login(@RequestParam String email, @RequestParam String
password, RedirectAttributes redirectAttributes) {
        if (loginService.authenticate(email, password)) {
            return "redirect:/";
        } else {
            redirectAttributes.addFlashAttribute("error", "Invalid email or
password");
            return "redirect:/login";
        }
    }
}
```

OrderBookController.java:

```
package com.krutn.bookstore.controller;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/order-books")
public class OrderBookController {

}
```

Book.java:

```
package com.krutn.bookstore.entity;

import jakarta.persistence.*;

@Entity
@Table(name = "books")
public class Book {


    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "title")
    private String title;

    @ManyToOne
```

```
    @JoinColumn(name = "author_id")
    private Author author;

    @Column(name = "year")
    private int year;

    @Column(name = "price")
    private double price;

    @Column(name = "publisher")
    private String publisher;

    @Column(name = "genre")
    private String genre;

    @Column(name = "isbn")
    private String isbn;

    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;

    public Book() {}

    public Book(String title, String author, int year, double price, String publisher,
String genre) {}

    public Long getId() {
       return id;
    }
```

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

public String getIsbn() {
    return isbn;
}

public void setIsbn(String isbn) {
    this.isbn = isbn;
}

public String getGenre() {
    return genre;
}

public void setGenre(String genre) {
    this.genre = genre;
}

public String getPublisher() {
    return publisher;
}

public void setPublisher(String publisher) {
    this.publisher = publisher;
}

public int getYear() {
```

```
    return year;
}

public void setYear(int year) {
    this.year = year;
}

public double getPrice() {
    return price;
}

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

public Author getAuthor() {
    return author;
}

public void setAuthor(Author author) {
    this.author = author;
}

public Category getCategory() {
    return category;
}

public void setCategory(Category category) {
    this.category = category;
}
```

```
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @Override
    public String toString() {
        return "Books{" +
                "id=" + id +
                ", title='" + title + '\'' +
                ", author='" + author + '\'' +
                ", year=" + year +
                ", price=" + price +
                ", publisher='" + publisher + '\'' +
                ", genre='" + genre + '\'' +
                ", isbn='" + isbn + '\'' +
                '}';
    }
}
```

BookRepository.java:

```
package com.krutn.bookstore.repository;

import com.krutn.bookstore.entity.Book;
```

```
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface BookRepository extends JpaRepository<Book, Integer> {

    List<Book> findByCategoryId(Long categoryId);

    public List<Book> findBooksByAuthorId(Long authorId);

    Book findByTitle(String title);

    List<Book> findBooksByTitleContainingIgnoreCase(String searchTerm);

    List<Book> findBooksById(Long id);

    Book findBookById(Long id);
}
```

BookService.java:

```
package com.krutn.bookstore.service;

import com.krutn.bookstore.entity.Book;
import com.krutn.bookstore.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
```

```
import org.springframework.stereotype.Service;

import java.math.BigDecimal;
import java.util.List;
import java.util.Map;
import java.util.Optional;

@Service
public class BookService {

    @Autowired
    private BookRepository bookRepository;

    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    public List<Book> getBooksById(Long id) {
        return bookRepository.findBooksById(id);
    }

    public Book getBookById(Long id) {
        return bookRepository.findBookById(id);
    }

    public Book saveBook(Book book) {
        return bookRepository.save(book);
    }

    public void deleteBook(Long id) {
```

```
        bookRepository.deleteById(Math.toIntExact(id));
    }

    public Page<Book> getBooksPage(Pageable pageable) {
        return bookRepository.findAll(pageable);
    }

    public List<Book> getBooksByCategoryId(Long categoryId) {
        return bookRepository.findByCategoryId(categoryId);
    }
}
```

CustomUserDetailsService.java:

```
package com.krutn.bookstore.service;

import com.krutn.bookstore.entity.User;
import com.krutn.bookstore.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import java.util.*;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.stereotype.Service;
```

```
@Service
public class CustomUserDetailsService implements UserDetailsService {

    private final UserRepository userRepository;

    @Autowired
    public CustomUserDetailsService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepository.findByEmail(username)
                .orElseThrow(() -> new UsernameNotFoundException("User not
found with email: " + username));
        return new CustomUserDetails(user.getId(), user.getEmail(),
user.getPassword(), getAuthorities(user.getRole()));
    }

    private Collection<? extends GrantedAuthority> getAuthorities(int role) {
        List<SimpleGrantedAuthority> authorities = new ArrayList<>();
        if (role == 1) {
            authorities.add(new SimpleGrantedAuthority("ADMIN"));
        } else {
            authorities.add(new SimpleGrantedAuthority("USER"));
        }
        return authorities;
    }
```

```
}
```

Login Service.java:

```
package com.krutn.bookstore.service;

import com.krutn.bookstore.entity.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Optional;
import java.util.concurrent.atomic.AtomicBoolean;

@Service
public class LoginService {

    @Autowired
    private UserService userService;

    @Autowired
    private PasswordEncoder passwordEncoder;

    public boolean authenticate(String email, String password) {
        AtomicBoolean isAuthenticated = new AtomicBoolean(false);
        userService.getUserByEmail(email).ifPresent(user -> {
            if (passwordEncoder.matches(password, user.getPassword())) {
                isAuthenticated.set(true);
            }
        });
```

```
            return isAuthenticated.get();
        }
}
```

styles.css:

```
    .container {
        margin-top: 50px;
        max-width: 400px;
        padding: 20px;
        background-color: #fff;
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    h2 {
        text-align: center;
        margin-bottom: 20px;
    }

    label {
        display: block;
        margin-bottom: 10px;
    }

    .delivery-container {
        max-width: 300px;
        margin: 0 auto;
```

```
        padding: 20px;
    }

    .delivery-content {
        background-color: #f9f9f9;
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    .delivery-title {
        font-size: 24px;
        margin-bottom: 20px;
    }

    .delivery-info {
        margin-bottom: 20px;
    }

    .delivery-info p {
        margin: 5px 0;
    }

    .delivery-info h3 {
        font-size: 18px;
    }

    .delivery-info .delivery-method {
        font-weight: bold;
}
```

script.js:

```
//АНІМАЦІЯ БАНЕРІВ

document.addEventListener("DOMContentLoaded", function() {
    const banners = document.querySelectorAll(".banner");
    let currentIndex = 0;

    function showBanner(index) {
        banners.forEach(function(banner, i) {
            if (i === index) {
                banner.style.transform = "translateX(0)";
            } else {
                banner.style.transform = "translateX(100%)";
            }
        });
    }

    function nextBanner() {
        currentIndex = (currentIndex + 1) % banners.length;
        showBanner(currentIndex);
    }

    setInterval(nextBanner, 3000);
});

//ФІКСАЦІЯ ХЕДЕРА

window.onscroll = function() { myFunction() };
```

```
var header = document.getElementById("myHeader");

var sticky = header.offsetTop;

function myFunction() {
  if (window.pageYOffset >= sticky) {
    header.classList.add("sticky");
    document.body.style.paddingTop = header.offsetHeight + 'px'; // Додати додатковий відступ до верхнього краю сторінки, щоб уникнути перекриття контенту фіксованим хедером
  } else {
    header.classList.remove("sticky");
    document.body.style.paddingTop = 0; // Зняти відступ, коли хедер не фіксований
  }
}



//ДОДАВАННЯ ТОВАРІВ У КОШИК
//ІНДИКАТОР КІЛЬКОСТІ ТОВАРІВ У КОШИКУ

document.addEventListener("DOMContentLoaded", function() {
  const addToCartButtons = document.querySelectorAll(".book-item .add-to-cart-btn, .sale-item .add-to-cart-btn");

  addToCartButtons.forEach(button => {
    button.addEventListener("click", function() {
      const parentElement = this.closest(".book-item, .sale-item");
```

```
            if (!parentElement) {
                return; // Перевіряємо, чи кнопка знаходиться в book-item або sale-item
            }

            const titleElement = parentElement.querySelector("h2");
            const priceElement = parentElement.querySelector("p");

            if (!titleElement || !priceElement) {
                return; // Перевіряємо, чи знайдено всі необхідні елементи
            }

            const title = titleElement.innerText;
            const price = priceElement.innerText;

            addToCart(title, price);
            updateCartQuantity(); // Оновлюємо кількість товарів у кошику
        });
    });

    function addToCart(title, price) {
        let cart = JSON.parse(localStorage.getItem("cart")) || [];
        let existingItem = cart.find(item => item.title === title);

        if (existingItem) {
            existingItem.quantity = (existingItem.quantity || 1) + 1;
        } else {
            cart.push({ title: title, price: price, quantity: 1 });
        }
```

```
        localStorage.setItem("cart", JSON.stringify(cart));
    }

    // Функція для оновлення кількості товарів у червоному кружку
    function updateCartQuantity() {
        // Отримуємо дані з localStorage та підраховуємо кількість товарів
        let cartItems = JSON.parse(localStorage.getItem("cart")) || [];
        let totalQuantity = cartItems.reduce((total, item) => total + item.quantity,
0);

        // Знаходимо елемент, в який будемо вставляти кількість товарів
        let cartQuantityElement = document.getElementById("cart-quantity");

        // Оновлюємо вміст елементу з кількістю товарів
        cartQuantityElement.textContent = totalQuantity.toString(); // або
String(totalQuantity);

        // Відображаємо або приховуємо червоний кружок в залежності від
кількості товарів
        if (totalQuantity > 0) {
            cartQuantityElement.style.display = "block";
        } else {
            cartQuantityElement.style.display = "none";
        }
    }

    // Додаємо обробник події для оновлення кількості при завантаженні
сторінки
    updateCartQuantity();
```

```
        // Додаємо обробник події для оновлення кількості при зміні кошика
        window.addEventListener("storage", updateCartQuantity);
    });
```

book.html:

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Сторінка книги</title>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
        <link rel="stylesheet" href="/static/css/styles.css" th:href="@{/css/styles_header.css}" />
        <link rel="stylesheet" href="/static/css/styles.css" th:href="@{/css/styles_book.css}" />
    </head>
    <body>
    <div th:insert="~{header.html}"></div>

    <div class="container">
        <div class="book-main">
            <img th:src="@{'/images/bookCovers/' + ${book.id} + '.png'}" alt="Обкладинка книги">
            <div class="book-item">
                <h1 class="book-title" th:text="${book.title}">Назва книги</h1>
                <h2 class="book-author" th:text="${book.author.name}">Автор</h2>
```

```
            <p class="book-price" th:text="'Ціна: ' + ${book.price}">Ціна: 200.0</p>
            <button class="add-to-cart-btn">Додати в кошик</button>
          </div>
        </div>
        <div class="book-details">
          <h2>Характеристики</h2>
          <ul>
            <li th:text="'Жанр: ' + ${book.genre}">Жанр: Художня література</li>
            <li th:text="'ISBN: ' + ${book.isbn}">ISBN: 123-456-7890</li>
            <li th:text="'Рік видання: ' + ${book.year}">Рік видання: 2021</li>
            <li th:text="'Видавництво: ' + ${book.publisher}">Видавництво: Example Publisher</li>
          </ul>
        </div>
      </div>


    </body>
</html>
```

header.html:

```
    <!-- Навігаційне меню -->
    <div class="header-container" id="myHeader" xmlns:sec="http://www.w3.org/1999/xhtml">
        <nav class="header-navigation">
            <div class="header-logo">
```

```
            <a href="/"><img th:src="@{/images/logo/logo.png}" alt="Логотип Магазину книг"></a>
        </div>

        <ul class="header-nav-links">
            <li><a href="/delivery">Доставка</a></li>
            <li><a href="/payment">Оплата</a></li>
            <li><a href="/help">Допомога</a></li>
            <li><a href="/contacts">Контакти</a></li>
            <li><a href="/user/profile">Особистий кабінет</a></li>
            <li sec:authorize="hasAuthority('ADMIN')"><a href="/admin/dashboard">Адмін панель</a></li>
        </ul>

        <!-- Поле пошуку, вихідне становище - приховане -->
        <div class="header-search-bar" id="header-searchBar">
            <form action="/search" method="get">
                <input type="text" name="searchTerm" placeholder="Пошук за назвою, автором, жанром">
                <button type="submit">Пошук</button>
            </form>
        </div>

        <div class="header-cart-button" onclick="location.href='/cart'">
            <i class="fas fa-shopping-cart"></i> <!-- Іконка кошика з Font Awesome -->
            <!-- Червоний кружок з кількістю товарів у кошику -->
            <div class="cart-quantity" id="cart-quantity">0</div>
        </div>
```

```
        </nav>
    </div>
<script th:src="@{/js/script.js}"></script>
```

home.html:

```
    <!DOCTYPE html>
    <html lang="ua" xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Магазин книг</title>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
        <link rel="stylesheet" href="/static/css/styles.css" th:href="@{/css/styles_header.css}" />
        <link rel="stylesheet" href="/static/css/styles.css" th:href="@{/css/styles_home.css}" />
    </head>
    <body>
    <div th:insert="~{header.html}" ></div>

    <section class="categories">
        <div class="container">
            <h2>Категорії книг</h2>
            <div class="row">
                <div class="category">
                    <a href="/category/1">Українська література</a>
            </div>
            <div class="category">
```

```
            <a href="/category/2">Зарубіжна література</a>
        </div>
        </div>
        <div class="row">
            <div class="category">
                <a href="/category/3">Наука</a>
        </div>
        <div class="category">
            <a href="/category/4">Програмування</a>
        </div>
        </div>
    </div>
</section>

<section class="animated-banners">
    <div class="banner-container">

        <div class="animated-banners">
            <div class="banner-carousel">
                <div class="banner">
                    <img src="/images/banners/banner1.jpg" alt="Banner 1">
                </div>
                <div class="banner">
                    <img src="/images/banners/banner2.jpg" alt="Banner 2">
                </div>
            </div>
        </div>

    </div>
</section>
```

```
        <section class="promotions">
            <div class="container">
                <h2>Акції</h2>
                <div class="promotion">
                    <div th:each="sale : ${sales}">
                        <div class="sale-item">
                            <a th:href="@{'/book/' + ${sale.book.id}}">
                                <h2 th:text="${sale.book.title}"></h2>
                                <img th:src="@{'/images/bookCovers/' + ${sale.book.id} + '.png'}" alt="Обкладинка книги">
                            </a>
                            <p th:text="'Ціна зі знижкою: ' + ${sale.book.price * (1 - sale.discount / 100)}"></p>
                            <button class="add-to-cart-btn">Додати в кошик</button>
                        </div>
                    </div>
                </div>
            </div>
        </section>

    </body>
</html>
```

login.html:

```
    <!DOCTYPE html>
    <html lang="en" xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Login</title>
        <link rel="stylesheet" href="/static/css/styles.css"
th:href="@{/css/styles.css}" />
        <link rel="stylesheet" href="/static/css/styles.css"
th:href="@{/css/styles_login.css}" />
    </head>
    <body>
    <div class="container">
      <h2>Вхід</h2>
      <form action="#" th:action="@{/login}" method="post">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br>
        <label for="password">Пароль:</label>
        <input type="password" id="password" name="password" required><br>
        <button type="submit">Вхід</button>
        <p th:if="${param.error}" class="error">Невірна пошта або пароль</p>
      </form>
      <p>Немає аккаунту? <a href="/registration">Реєстрація</a></p>
    </div>
    </body>
</html>
```