ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для онлайн-продажу паперових книжок»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Артем МАРУЩАК
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Артем МАРУЩАК

Керівник: В’ячеслав ТРЕЙТЯК
канд.техн.наук

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Марущаку Артему Руслановичу

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для онлайн-продажу паперових книжок»
керівник кваліфікаційної роботи канд.техн.наук В'ячеслав ТРЕЙТЯК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи**:** теоретичні відомості про організацію онлайн-продажу паперових книжок, принципи роботи електронної комерції, підходи до створення електронних каталогів товарів, пошуку та фільтрації книжок, оформлення замовлень, керування кошиком, обліку залишків товарів, завантаження PDF-файлів, адміністрування каталогу та замовлень, опис принципів побудови клієнт-серверних web-застосунків, технічна документація щодо використання технологій Java, Spring, Hibernate, PostgreSQL, React, Liquibase та REST API.

4. Зміст розрахунково-пояснювальної записки**:**

1.Огляд та аналіз предметної галузі онлайн-продажу паперових книжок.

2.Аналіз існуючих програмних засобів для продажу книжок через інтернет.

3.Огляд засобів реалізації web-застосунку BookMarket.

4.Проєктування структури web-застосунку, бази даних та основних модулів системи.

5.Програмна реалізація та опис функціонування web-застосунку.

6.Тестування web-застосунку.

7.Апробація результатів дослідження.

8.Перелік графічного матеріалу: презентація.

5. Перелік графічного матеріалу:

1.Аналіз аналогів

2. Виимоги додатку

3.Програми засоби реалізації.

4.Діаграма варіантів використання.

5.Діаграма послідовності.

6.Структура бази даних.

7.Архітектура додатку.

8.Мапи Веб-застосунку.

9.Екранні форми.

10.Апробація результатів дослідження.

6. Дата видачі завдання«23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури з теми онлайн-продажу книжок та web-застосунків електронної комерції	23.02–01.03.2026	
2	Аналіз існуючих аналогів програмного забезпечення для онлайн-продажу паперових книжок	02.03–08.03.2026	
3	Аналіз предметної галузі онлайн-продажу паперових книжок та визначення вимог	09.03–16.03.2026	
4	Огляд засобів реалізації web-застосунку BookMarket	17.03–23.03.2026	

5	Проєктування web-застосунку BookMarket, структури системи, бази даних та UML-діаграм	24.03–06.04.2026	
6	Програмна реалізація серверної та клієнтської частин	07.04–27.04.2026	
7	Тестування та оформлення результатів тестування	28.04–04.05.2026	
8	Оформлення роботи: вступ, висновки, реферат, список джерел та додатки	05.05–11.05.2026	
9	Розробка демонстраційних матеріалів	12.05–17.05.2026	
10	Попередній захист роботи	18.05–22.05.2026	

Здобувач вищої освіти _____ Артем МАРУЩАК
(підпис)

Керівник
кваліфікаційної роботи _____ В’ячеслав ТРЕЙТЯК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 58 стор., 5 табл., 20 рис., 25 джерел.

Мета роботи – спрощення процесу онлайн-продажу паперових книжок за рахунок розробки веб-застосунку, який забезпечує зручний перегляд каталогу, пошук книжок, формування кошика, оформлення замовлень та адміністрування даних про книжки й замовлення**.**

Об'єкт дослідження – процеси онлайн-продажу паперових книжок.

Предмет дослідження – засоби, методи та технології реалізації веб-застосунку для онлайн-продажу паперових книжок**.**

У роботі проаналізовано предметну галузь онлайн-продажу паперових книжок та визначено основні проблеми, пов'язані з організацією електронного каталогу, пошуком потрібних видань, фільтрацією книжок, оформленням замовлень, обліком залишків товарів і керуванням даними з боку адміністратора. Встановлено, що ручна обробка замовлень і неструктуроване ведення каталогу можуть призводити до помилок, дублювання інформації, затримок в обробці покупок і зниження зручності взаємодії користувача з онлайн-магазином.

Проаналізовано існуючі програмні засоби для онлайн-продажу книжок: Yakaboo, Книгарня «Є», Book24, Amazon Books. Визначено їх переваги, недоліки та функціональні обмеження. Особливу увагу приділено порівнянню можливостей каталогу, оформлення замовлення, обліку залишків, локалізації під український ринок, доступу до PDF-версій книжок і гнучкого налаштування системи під вимоги конкретного проєкту.

Розроблено структуру web-застосунку BookMarket та визначено його ключові функціональні можливості: реєстрацію та авторизацію користувачів, перегляд каталогу книжок, пошук і фільтрацію видань, перегляд сторінки окремої книжки, додавання товарів до кошика, оформлення замовлення, перегляд статусу

замовлення, безкоштовне завантаження доступних книжок у PDF-форматі, керування каталогом і замовленнями адміністратором. У роботі використано Java та Spring для реалізації серверної частини, Hibernate для роботи з об’єктно-реляційним відображенням, PostgreSQL для зберігання даних, Liquibase для керування міграціями бази даних, React для створення клієнтського інтерфейсу та REST API для організації взаємодії між клієнтською і серверною частинами.

Сферою використання застосунку є організація онлайн-продажу паперових книжок для невеликих і середніх книжкових магазинів, навчальних проєктів, локальних книжкових платформ та інших сервісів, які потребують зручного інструменту для представлення каталогу, приймання замовлень, керування залишками та надання користувачам доступу до PDF-матеріалів.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-ПРОДАЖ КНИЖОК, BOOKMARKET, WEB-ЗАСТОСУНОК, ЕЛЕКТРОННИЙ КАТАЛОГ, КОШИК, ЗАМОВЛЕННЯ, PDF, JAVA, SPRING, HIBERNATE, POSTGRESQL, LIQUIBASE, REACT, REST API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс прикладного програмування, що використовується для взаємодії між програмними компонентами системи.

REST – архітектурний стиль передавання даних між клієнтською та серверною частинами web-застосунку.

HTTP – протокол передавання даних у мережі Інтернет.

HTTPS – захищена версія протоколу HTTP, що забезпечує безпечний обмін даними.

JSON – текстовий формат обміну даними між клієнтською та серверною частинами системи.

UI – користувацький інтерфейс програмного забезпечення.

UX – користувацький досвід під час взаємодії з програмним продуктом.

HTML – мова розмітки гіпертексту, що використовується для створення структури web-сторінок.

CSS – мова стилів, що використовується для оформлення зовнішнього вигляду web-сторінок.

JavaScript – мова програмування, що використовується для створення інтерактивної поведінки web-сторінок.

React – бібліотека JavaScript для створення компонентного користувацького інтерфейсу.

Java – об’єктно-орієнтована мова програмування, що використовується для реалізації серверної частини застосунку.

Spring – фреймворк для створення серверних Java-застосунків.

Spring Boot – інструмент екосистеми Spring, що спрощує створення та запуск серверних web-застосунків.

Hibernate – фреймворк об’єктно-реляційного відображення, який забезпечує зв’язок між Java-класами та таблицями бази даних.

JPA – специфікація Java для роботи з реляційними базами даних через об’єктну модель.

SQL – мова структурованих запитів для роботи з реляційними базами даних.

PostgreSQL – об’єктно-реляційна система керування базами даних.

DB – база даних, призначена для зберігання структурованої інформації.

ORM – технологія об’єктно-реляційного відображення, що забезпечує зв’язок між об’єктами програми та таблицями бази даних.

CRUD – базові операції роботи з даними: створення, читання, оновлення та видалення.

PDF – формат електронного документа, який використовується для збереження та завантаження цифрових копій книжок.

JWT – формат токена, що може використовуватися для авторизації користувачів у web-застосунку.

SPA – односторінковий web-застосунок, у якому основна взаємодія з користувачем відбувається без повного перезавантаження сторінки.

UML – уніфікована мова моделювання, що використовується для побудови діаграм програмної системи.

ERD – діаграма зв’язків сутностей, що використовується для опису структури бази даних.

IDE – інтегроване середовище розробки програмного забезпечення.

Git – система контролю версій, що використовується для відстеження змін у програмному коді.

GitHub – веб-сервіс для зберігання репозиторіїв, спільної роботи з кодом та керування версіями проєкту.

Liquibase – інструмент для керування змінами структури бази даних за допомогою міграцій.

Docker – платформа контейнеризації, що дозволяє запускати програмні компоненти в ізольованому середовищі.

ВСТУП

Актуальність теми. Онлайн-продаж паперових книжок є важливою складовою сучасної електронної комерції, оскільки користувачі все частіше обирають товари через інтернет, порівнюють пропозиції, переглядають характеристики видань і оформлюють замовлення дистанційно. Для книжкового магазину важливо не лише представити асортимент товарів, а й забезпечити зручний пошук, фільтрацію, керування кошиком, оформлення замовлення та контроль наявності книжок на складі. У сучасних умовах web-застосунки дозволяють автоматизувати ці процеси та зробити взаємодію користувача з онлайн-магазином більш швидкою і зрозумілою [1; 2; 3].

Однією з поширених проблем у сфері онлайн-продажу паперових книжок є складність організації повноцінного електронного каталогу. Якщо інформація про книжки, їх наявність, ціну, жанр, автора та опис зберігається неструктуровано або оновлюється вручну, це може призводити до помилок, дублювання даних, неактуальної інформації про залишки та затримок під час обробки замовлень. Також проблемою є недостатня зручність пошуку потрібного видання, особливо якщо користувач не знає точну назву книги, а орієнтується лише на жанр, автора, тематику або ціновий діапазон [1; 3; 22; 23].

Крім продажу паперових книжок, актуальним є надання користувачам доступу до електронних матеріалів у PDF-форматі. У багатьох онлайн-книгарнях така можливість або відсутня, або не є частиною основної логіки роботи системи. Web-застосунок BookMarket передбачає поєднання класичного онлайн-магазину паперових книжок із можливістю безкоштовного завантаження доступних книжок у PDF-форматі. Це дозволяє розширити функціональність системи та зробити її більш корисною для користувачів, які хочуть не тільки придбати друковане видання, а й отримати доступ до електронного матеріалу.

Створення єдиного web-застосунку для онлайн-продажу паперових книжок дозволяє об’єднати в одному середовищі каталог книжок, пошук, фільтрацію,

сторінку окремої книжки, кошик, оформлення замовлення, облік залишків, завантаження PDF-файлів та адміністративне керування даними. Такий підхід спрощує роботу користувача, зменшує кількість ручних операцій з боку адміністратора та забезпечує більш організовану роботу онлайн-магазину [1; 3; 22; 23].

Web-застосунок BookMarket для онлайн-продажу паперових книжок може забезпечити зручний доступ до каталогу, перегляд детальної інформації про книжки, формування кошика, оформлення замовлення, перегляд статусу замовлення та завантаження доступних PDF-файлів. Адміністратор, у свою чергу, може керувати каталогом, додавати нові книжки, редагувати інформацію про видання, оновлювати залишки товарів, переглядати замовлення та змінювати їх статуси. Враховуючи наведене, розробка web-застосунку BookMarket для онлайн-продажу паперових книжок є актуальною задачею [1; 3; 22; 23].

Мета роботи – спрощення процесу онлайн-продажу паперових книжок за рахунок розробки веб-застосунку, який забезпечує зручний перегляд каталогу, пошук книжок, формування кошика, оформлення замовлень та адміністрування даних про книжки й замовлення [1; 3; 22; 23].

Об’єкт дослідження – процеси онлайн-продажу паперових книжок.

Предмет дослідження – засоби, методи та технології реалізації веб-застосунку для онлайн-продажу паперових книжок [1; 3; 22; 23].

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати предметну область онлайн-продажу паперових книжок [1; 3; 22; 23].
2. Дослідити існуючі програмні рішення для продажу книжок через інтернет.
3. Визначити функціональні та нефункціональні вимоги до майбутнього web-застосунку.
4. Спроєктувати структуру застосунку, основні модулі та базу даних.
5. Обґрунтувати вибір засобів реалізації: Java, Spring, Hibernate, React, PostgreSQL та Liquibase [9].

6. Реалізувати основні функції web-застосунку: каталог, пошук, кошик, оформлення замовлень, завантаження PDF-файлів та адміністрування.
7. Провести тестування роботи застосунку та визначити напрями його подальшого розвитку.

Методи дослідження. У бакалаврській роботі використовуються такі методи: аналіз предметної галузі, порівняльний аналіз існуючих програмних засобів для онлайн-продажу книжок, об’єктно-орієнтоване проєктування, UML-моделювання, проєктування web-застосунку, моделювання структури бази даних, розробка клієнт-серверного програмного забезпечення та тестування [1; 3; 22; 23].

Наукова новизна одержаних результатів. У ході виконання роботи одержано нове вирішення задачі онлайн-продажу паперових книжок — структуру web-застосунку BookMarket, який поєднує каталог книжок, пошук, фільтрацію, кошик, оформлення замовлень, адміністрування даних і можливість безкоштовного завантаження доступних книжок у PDF-форматі [1; 3; 22; 23].

Практична значущість одержаних результатів. Розроблений web-застосунок може бути використаний для організації онлайн-продажу паперових книжок у невеликих і середніх книжкових магазинах, навчальних проєктах або локальних книжкових платформах [1; 3; 22; 23].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальна характеристика предметної галузі

Предметною галуззю кваліфікаційної роботи є онлайн-продаж паперових книжок. Такий процес належить до сфери електронної комерції та охоплює сукупність дій, пов’язаних із представленням книжкової продукції в електронному каталозі, пошуком потрібних видань, переглядом детальної інформації про книжку, формуванням кошика, оформленням замовлення та подальшою обробкою покупки адміністратором [1; 3; 22; 23].

Онлайн-продаж паперових книжок має певні особливості порівняно з продажем цифрових товарів. Паперова книга є фізичним товаром, тому для її продажу необхідно враховувати не лише опис і ціну, а й наявність на складі, кількість доступних примірників, умови доставки та статус обробки замовлення. Якщо інформація про залишки оновлюється несвоєчасно, користувач може оформити замовлення на книжку, якої фактично немає в наявності. Це погіршує якість обслуговування та створює додаткове навантаження на адміністратора.

Книжкова продукція також має розширений набір характеристик, які впливають на вибір користувача. Покупець може шукати книгу за назвою, автором, жанром, видавництвом, роком видання, мовою, ціною або тематикою. Тому web-застосунок для онлайн-продажу паперових книжок повинен забезпечувати не лише звичайний перелік товарів, а й зручну навігацію, фільтрацію та швидкий пошук [1; 3; 22; 23].

Основна задача предметної галузі полягає в автоматизації процесу продажу паперових книжок через інтернет. Без спеціалізованого програмного забезпечення значна частина операцій виконується вручну: адміністратор приймає замовлення, перевіряє наявність книжок, фіксує контактні дані клієнта, змінює статуси замовлень і контролює відправлення. Такий підхід є неефективним, оскільки

підвищує ризик помилок, дублювання даних, втрати інформації та затримок в обробці замовлень.

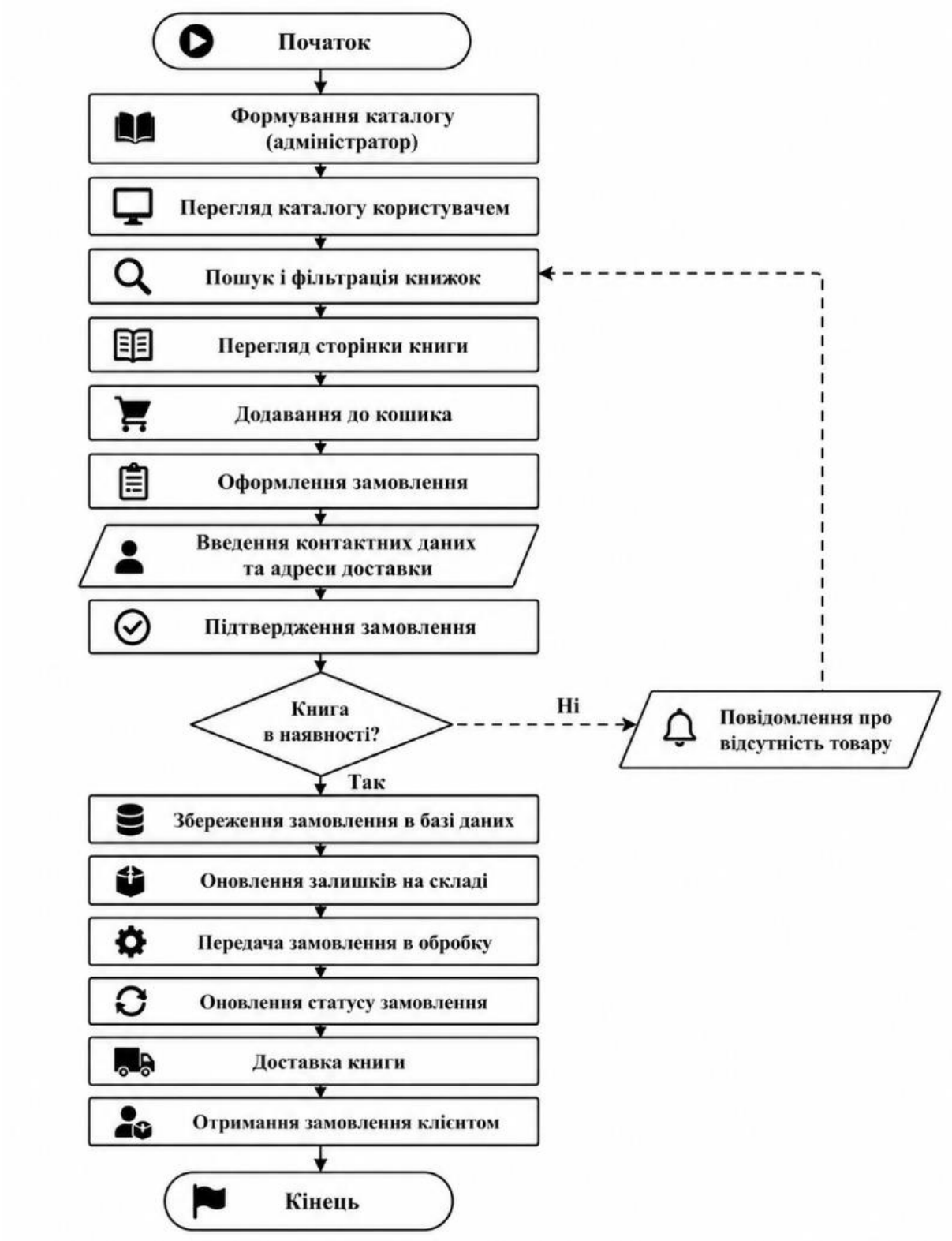


Рис. 1.1 Загальна схема процесу онлайн-продажу паперових книжок

1.2 Актуальність створення web-застосунку для онлайн-продажу паперових книжок

Актуальність створення web-застосунку для онлайн-продажу паперових книжок зумовлена зростанням ролі електронної комерції та потребою користувачів у швидкому доступі до товарів через інтернет. Покупець очікує, що онлайн-магазин дозволить йому швидко знайти потрібну книжку, переглянути її характеристики, перевірити наявність і оформити замовлення без зайвих дій [1; 3; 22; 23].

Для книжкового магазину автоматизована система також має значення, оскільки дозволяє централізовано керувати каталогом, зменшити кількість ручних операцій, контролювати залишки товарів і швидше обробляти замовлення. У разі відсутності такої системи адміністратор змушений використовувати декілька окремих інструментів: таблиці для обліку книжок, месенджери для приймання замовлень, окремі документи для фіксації клієнтів і ручну перевірку наявності. Це ускладнює роботу та знижує надійність процесу продажу.

Окремою актуальною функцією для розроблюваного застосунку є можливість безкоштовного завантаження доступних книжок у PDF-форматі. У більшості класичних онлайн-книгарень основний акцент зроблено саме на продажі паперових або платних електронних видань. У проєкті BookMarket передбачено поєднання продажу паперової книжки з можливістю отримати доступну PDF-версію безкоштовно. Це розширює функціональність системи та створює додаткову цінність для користувача.

Таким чином, створення web-застосунку BookMarket є доцільним, оскільки він дозволяє об’єднати в одному середовищі каталог паперових книжок, пошук, фільтрацію, кошик, оформлення замовлення, адміністрування товарів і функцію доступу до PDF-матеріалів.

1.3 Цільова аудиторія та основні користувачі системи

Цільовою аудиторією web-застосунку BookMarket є користувачі, які хочуть купувати паперові книжки через інтернет або переглядати доступні книжкові матеріали в електронному форматі. До таких користувачів можна віднести студентів, школярів, викладачів, читачів художньої літератури, фахівців, які купують професійні видання, а також людей, які замовляють книги як подарунок.

Для різних груп користувачів важливими можуть бути різні параметри пошуку. Наприклад, студент може шукати книжку за тематикою або автором, читач художньої літератури — за жанром, а покупець подарункового видання — за ціною або популярністю. Тому система повинна забезпечувати зручний каталог і можливість швидкого пошуку потрібного видання [22; 23].

У межах системи можна виділити дві основні ролі: користувач і адміністратор. Користувач взаємодіє з клієнтською частиною застосунку, переглядає каталог, відкриває сторінки книжок, додає товари до кошика, оформлює замовлення та завантажує доступні PDF-файли. Адміністратор відповідає за керування каталогом, додавання нових книжок, редагування інформації, оновлення залишків і обробку замовлень.

Важливо, що адміністратор у такій системі також може мати базові можливості звичайного користувача. Тобто він не лише керує даними, а й може переглядати каталог, відкривати сторінки книжок і перевіряти, як система виглядає з боку покупця. Це дозволяє краще контролювати якість роботи застосунку.

1.4 Основні функції та бізнес-логіка системи

Бізнес-логіка web-застосунку BookMarket побудована навколо процесу вибору та купівлі паперової книжки. Основним сценарієм є перехід користувача до каталогу, вибір потрібного видання, додавання його до кошика та оформлення

замовлення. На кожному етапі система повинна забезпечувати коректну обробку даних і зручну взаємодію з користувачем [22; 23].

Першою важливою функцією є каталог книжок. Він повинен відображати перелік доступних видань із короткою інформацією: назвою, автором, ціною, жанром, обкладинкою та наявністю. Каталог є центральною частиною системи, оскільки саме через нього користувач знайомиться з асортиментом онлайн-магазину.

Другою функцією є пошук і фільтрація. Користувач повинен мати можливість швидко знайти книжку за назвою, автором або ключовими словами. Фільтрація дозволяє звузити перелік товарів за жанром, ціною, мовою або наявністю. Це особливо важливо у випадку великого каталогу, де ручний перегляд усіх позицій є незручним [22; 23].

Третьою функцією є сторінка окремої книжки. На ній користувач отримує детальнішу інформацію про видання: повну назву, автора, жанр, видавництво, рік публікації, опис, ціну, кількість на складі та можливість додавання до кошика. Якщо для книжки доступний PDF-файл, користувач також може завантажити його.

Четвертою функцією є кошик. Він дозволяє тимчасово зберігати обрані товари перед оформленням замовлення. У кошику користувач може змінювати кількість книжок, видаляти непотрібні позиції та переглядати загальну суму покупки.

Окремий блок бізнес-логіки пов’язаний з адмініструванням. Адміністратор повинен мати можливість додавати нові книжки, редагувати вже наявні, змінювати ціни, оновлювати кількість товарів на складі, переглядати замовлення та змінювати їх статуси. Це дозволяє підтримувати актуальність інформації в системі.

1.5 Способи вирішення задачі та роль інфокомунікаційних засобів

Задача онлайн-продажу паперових книжок може вирішуватися різними способами. Найпростішим варіантом є продаж через соціальні мережі або месенджери. У такому випадку продавець публікує інформацію про книжки, а покупці звертаються напряму для оформлення замовлення. Такий підхід може бути прийнятним для невеликої кількості товарів, але він не забезпечує зручного пошуку, автоматичного обліку залишків і централізованої обробки замовлень [1; 3; 22; 23].

Іншим способом є використання готових платформ електронної комерції. Такі платформи дозволяють швидко створити онлайн-магазин, однак можуть мати обмеження щодо гнучкого налаштування, структури каталогу, доступу до коду або специфічної логіки роботи. Для навчального проєкту та бакалаврської роботи такий варіант також не є оптимальним, оскільки він не дозволяє повністю продемонструвати власну реалізацію серверної частини, бази даних, клієнтського інтерфейсу та бізнес-логіки [1; 3; 22; 23].

Інфокомунікаційні засоби відіграють важливу роль у роботі такого застосунку. Вони забезпечують передавання даних між користувачем і сервером, обробку HTTP-запитів, збереження інформації в базі даних, відображення результатів на сторінках застосунку та доступ до функцій системи через браузер. Завдяки цьому користувач може взаємодіяти з онлайн-магазином без встановлення додаткового програмного забезпечення [16; 17].

1.6 Основні вимоги до програмного продукту

На основі аналізу предметної галузі можна визначити основні вимоги до web-застосунку BookMarket. Система повинна забезпечувати реєстрацію та авторизацію користувачів, перегляд каталогу книжок, пошук і фільтрацію, перегляд детальної сторінки книжки, додавання товарів до кошика, оформлення

замовлення, завантаження доступних книжок у PDF-форматі та перегляд статусу замовлення [2; 19; 20; 21].

Для адміністратора система повинна надавати можливість керувати каталогом, додавати нові книжки, редагувати інформацію про видання, оновлювати кількість товарів на складі, переглядати замовлення та змінювати їх статуси. Це дозволяє підтримувати актуальність даних і забезпечувати коректну роботу онлайн-магазину.

До нефункціональних вимог належать швидкість роботи, адаптивність інтерфейсу, коректна робота в сучасних браузерах, захист персональних даних і обмеження доступу до адміністративної частини. Наприклад, завантаження сторінок повинно виконуватися в межах декількох секунд, а пошук книжок має повертати результат без помітної затримки для користувача [2; 19; 20; 21].

1.7 Аналіз проблем і недоліків існуючих підходів

Існуючі підходи до онлайн-продажу книжок мають певні недоліки. У разі ручної обробки замовлень виникає залежність від людського фактора. Адміністратор може помилитися під час внесення даних, забути оновити залишки або неправильно зафіксувати контактну інформацію клієнта. Це призводить до помилок у замовленнях і знижує якість
обслуговування [1; 3; 22; 23].

Готові онлайн-платформи часто мають надмірний функціонал або обмежену гнучкість. Вони можуть бути зручними для швидкого запуску магазину, але не завжди дозволяють реалізувати специфічну логіку конкретного проєкту. Наприклад, не в усіх системах можна зручно поєднати продаж паперової книжки з безкоштовним доступом до PDF-версії [22; 23].

Великі онлайн-книгарні мають розвинений функціонал, але їхній інтерфейс іноді є перевантаженим великою кількістю рекламних блоків, рекомендацій, додаткових категорій і комерційних пропозицій. Для користувача, який хоче

швидко знайти конкретну книжку, така структура може бути не завжди зручною [22; 23].

Ще однією проблемою є обмежена адаптація під локальні потреби. Міжнародні платформи можуть мати сильні інструменти пошуку, рекомендацій і доставки, але вони не завжди орієнтовані на український ринок, локальні способи оплати, доставку або україномовний інтерфейс [22; 23].

2 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОНЛАЙН-ПРОДАЖУ ПАПЕРОВИХ КНИЖОК

У цьому розділі розглянуто програмні засоби, які використовуються для онлайн-продажу паперових книжок. Аналіз таких систем дозволяє визначити основні функції, які повинні бути реалізовані у майбутньому застосунку, а також виявити переваги й недоліки вже існуючих рішень [1; 3; 22; 23].

До програмних засобів, які можуть використовуватися для продажу паперових книжок через інтернет, можна віднести:

− спеціалізовані онлайн-книгарні;

− універсальні маркетплейси;

− сайти видавництв;

− готові платформи електронної комерції;

− власні веб-застосунки інтернет-магазинів [1; 3; 22; 23].

Для аналізу було обрано декілька популярних програмних рішень, які мають функції каталогу книжок, пошуку, фільтрації, кошика, оформлення замовлення та взаємодії з користувачем. До таких засобів належать Yakaboo**,** Книгарня «Є»**,** Book24**,** Amazon Books та Shopify як приклад платформи, на основі якої можна створити власний інтернет-магазин. [4]

2.1 Онлайн-книгарня Yakaboo

Yakaboo є однією з найбільших українських книжкових онлайн-платформ. Сервіс орієнтований на продаж паперових, електронних та аудіокниг. Для предметної галузі онлайн-продажу паперових книжок цей ресурс є важливим прикладом, оскільки він містить великий каталог товарів, систему пошуку, фільтри, персональні добірки, кошик і можливість оформлення замовлення через сайт. За даними самого сервісу, Yakaboo позиціонується як національна книжкова

платформа України та містить значну кількість найменувань книг різними мовами [4].

Основною перевагою Yakaboo є великий асортимент книжок. Користувач може шукати книги за назвою, автором, категорією або тематичною добіркою. Також зручною є наявність окремих сторінок товарів, на яких розміщено обкладинку книги, опис, ціну, інформацію про автора, видавництво, мову та інші характеристики. Це дозволяє покупцеві швидко ознайомитися з товаром перед придбанням [4].

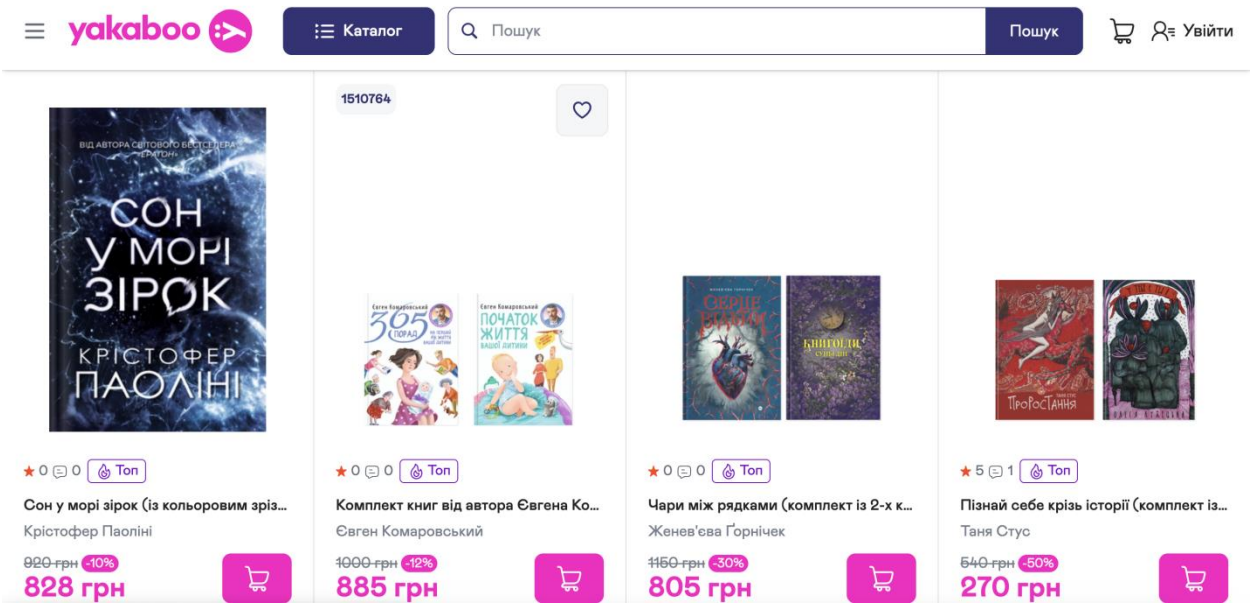


Рис. 2.1 Приклад інтерфейсу онлайн-книгарні Yakaboo

До переваг Yakaboo можна віднести [4]:

– великий каталог книжкової продукції;

– зручний пошук за назвою, автором або ключовими словами;

– наявність фільтрів за категоріями, мовою, ціною та іншими параметрами;

– детальні сторінки книжок з описом і характеристиками;

– можливість формування кошика та оформлення замовлення;

– наявність особистого кабінету користувача [22; 23].

Однак система має і певні недоліки. Через велику кількість товарів інтерфейс може бути перевантаженим для користувача, який хоче швидко знайти конкретну

книгу. Також частина функцій орієнтована на масштабну комерційну платформу, тому для невеликого інтернет-магазину така структура може бути занадто складною. Крім того, велика кількість рекламних блоків, добірок і рекомендацій іноді відволікає від основного процесу купівлі [1; 3; 22; 23].

2.2 Інтернет-магазин Книгарня «Є»

Книгарня «Є» є відомою українською мережею книгарень, яка також має власний інтернет-магазин. Сервіс дозволяє користувачам купувати книжки онлайн, переглядати каталог, ознайомлюватися з новинками, бестселерами та рецензіями. На сайті підкреслюється, що інтернет-магазин має зручний розподіл книжок за жанрами й категоріями, що допомагає швидко знайти потрібне видання [5].

Для досліджуваної теми цей ресурс є корисним прикладом поєднання фізичної мережі книгарень та онлайн-продажу. Така модель дозволяє користувачеві не лише замовити книгу через інтернет, а й орієнтуватися на наявність товарів у мережі магазинів. Це є важливим з точки зору інтеграції онлайн-продажу з реальними складськими або торговими залишками [1; 3; 22; 23].

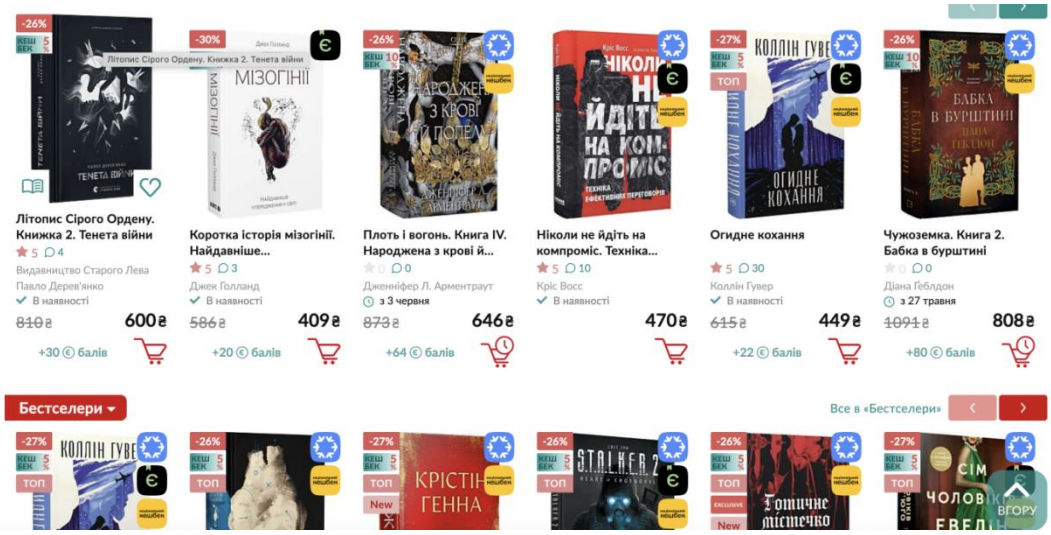


Рис. 2.2 Приклад інтерфейсу інтернет-магазину Книгарня «Є»

Перевагами Книгарні «Є» є [5]:

− зрозуміла структура каталогу;

− поділ книжок за жанрами та категоріями;

− наявність новинок, бестселерів і тематичних добірок;

− можливість перегляду опису книжки та її характеристик;

− підтримка онлайн-замовлення;

− поєднання інтернет-магазину з фізичною мережею книгарень.

До недоліків можна віднести те, що система більше орієнтована на конкретну мережу книгарень, тому її функціональність тісно пов’язана з наявною бізнес-структурою. Також для невеликого проєкту частина функцій може бути надлишковою, наприклад, інтеграція з мережею фізичних магазинів або велика кількість розділів, пов’язаних із подіями, рецензіями та літературними новинами.

2.3 Онлайн-магазин Book24

Book24 є українським інтернет-магазином книжок, який пропонує каталог видань різних жанрів, новинки, бестселери, передзамовлення, знижки та різні способи оплати й доставки. На сайті зазначено, що магазин має великий каталог книг, зручні способи оплати та доставку по Україні [6].

З точки зору аналізу предметної галузі Book24 є прикладом класичного інтернет-магазину книжок. Його структура містить каталог, сторінки товарів, кошик, особистий кабінет, розділи з умовами оплати, доставки, повернення та обміну. Це дозволяє розглядати Book24 як приклад системи, де основний акцент зроблено саме на продажу товарів, а не на додатковому інформаційному або медійному контенті [6].

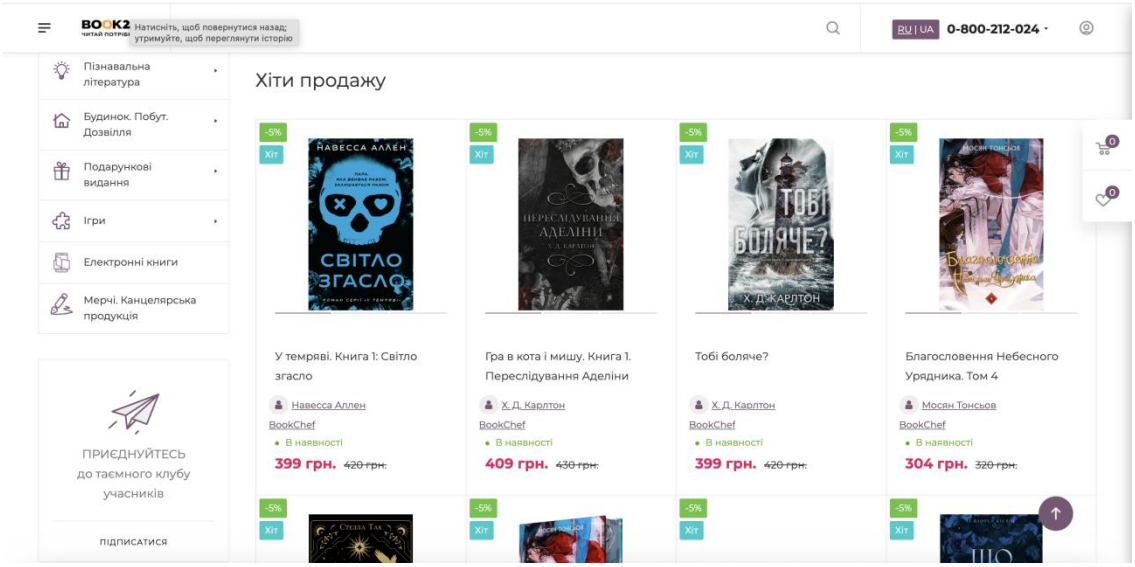


Рис. 2.3 Приклад інтерфейсу онлайн-магазину Book24

Перевагами Book24 є [6]:

– наявність повноцінного каталогу книжок;

– можливість перегляду новинок, акцій і передзамовлень;

– підтримка кошика та особистого кабінету;

– окремі розділи з інформацією про оплату, доставку та повернення;

– зручна структура для класичного інтернет-магазину [1; 3; 22; 23].

Недоліками є те, що інтерфейс може виглядати досить стандартним і не завжди забезпечує глибоку персоналізацію для користувача. Також велика кількість комерційних блоків, акцій і службових розділів може ускладнювати сприйняття для користувача, який хоче швидко перейти до пошуку й замовлення конкретної книги [22; 23].

2.4 Amazon Books

Amazon Books є частиною великої міжнародної платформи Amazon, яка дозволяє купувати паперові книги, електронні книги та аудіокниги. На сторінці Amazon Books користувач може переглядати категорії, бестселери, новинки, тематичні добірки та рекомендації [7].

Для аналізу ця платформа є показовою, оскільки вона демонструє високий рівень автоматизації електронної торгівлі. Amazon використовує розвинену систему рекомендацій, персоналізований пошук, рейтинги, відгуки користувачів, історію замовлень, різні варіанти доставки та оплату через обліковий запис. Такі функції значно покращують користувацький досвід, однак потребують складної технічної реалізації [7].

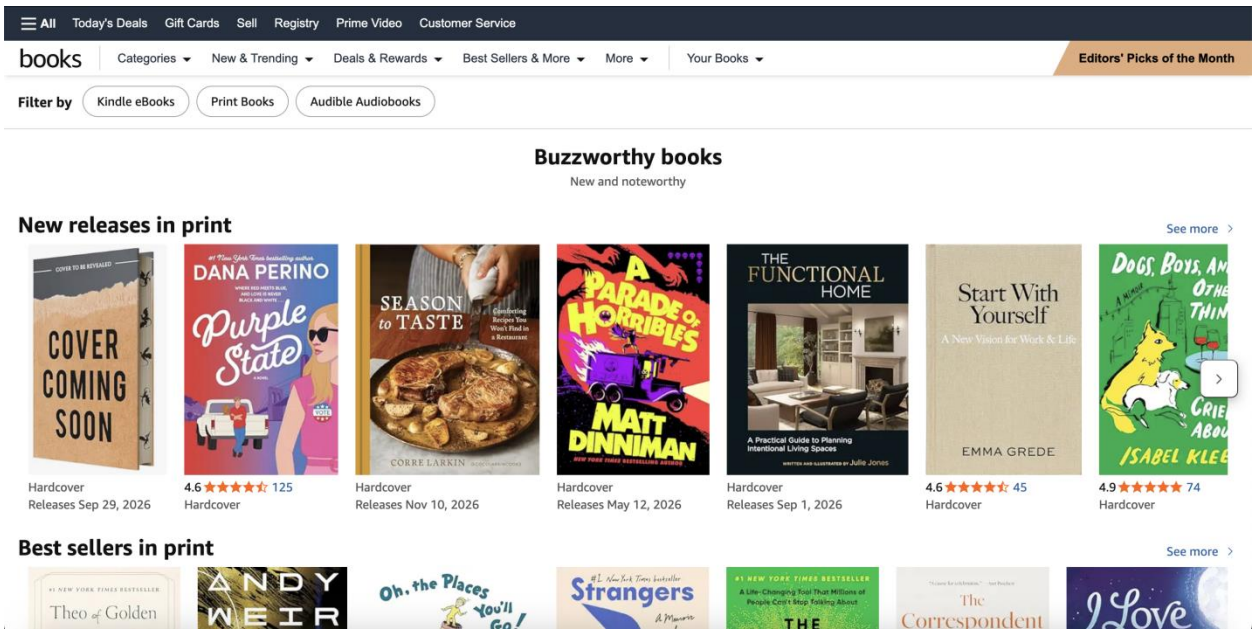


Рис. 2.4 Приклад інтерфейсу Amazon Books

Перевагами Amazon Books є [7]:

– дуже великий каталог книжок;

– потужний пошук і фільтрація;

– система рейтингів і відгуків;

– персональні рекомендації;

– інтеграція з обліковим записом користувача;

– розвинена система доставки та відстеження замовлення.

До недоліків можна віднести складність системи та її надмірність для невеликого онлайн-магазину паперових книжок. Крім того, Amazon є універсальним маркетплейсом, а не спеціалізованою локальною книгарнею, тому його підхід не завжди доцільно повністю переносити на невеликий навчальний або

локальний проєкт. Для українського користувача також можуть бути обмеження, пов’язані з мовою інтерфейсу, доставкою, цінами та локальними способами оплати [7].

2.5 Платформа Shopify

Shopify є платформою для створення інтернет-магазинів. На відміну від попередніх прикладів, Shopify не є окремою книгарнею, а виступає інструментом, за допомогою якого можна створити власний онлайн-магазин, зокрема магазин паперових книжок. Такий підхід дозволяє швидко запустити продажі без розробки всіх модулів з нуля [8].

Shopify може бути використаний для створення каталогу товарів, налаштування сторінок книжок, підключення кошика, оформлення замовлень, оплати та доставки. Для малого бізнесу це є досить зручним рішенням, оскільки більшість базових функцій електронної комерції вже реалізовані [8].

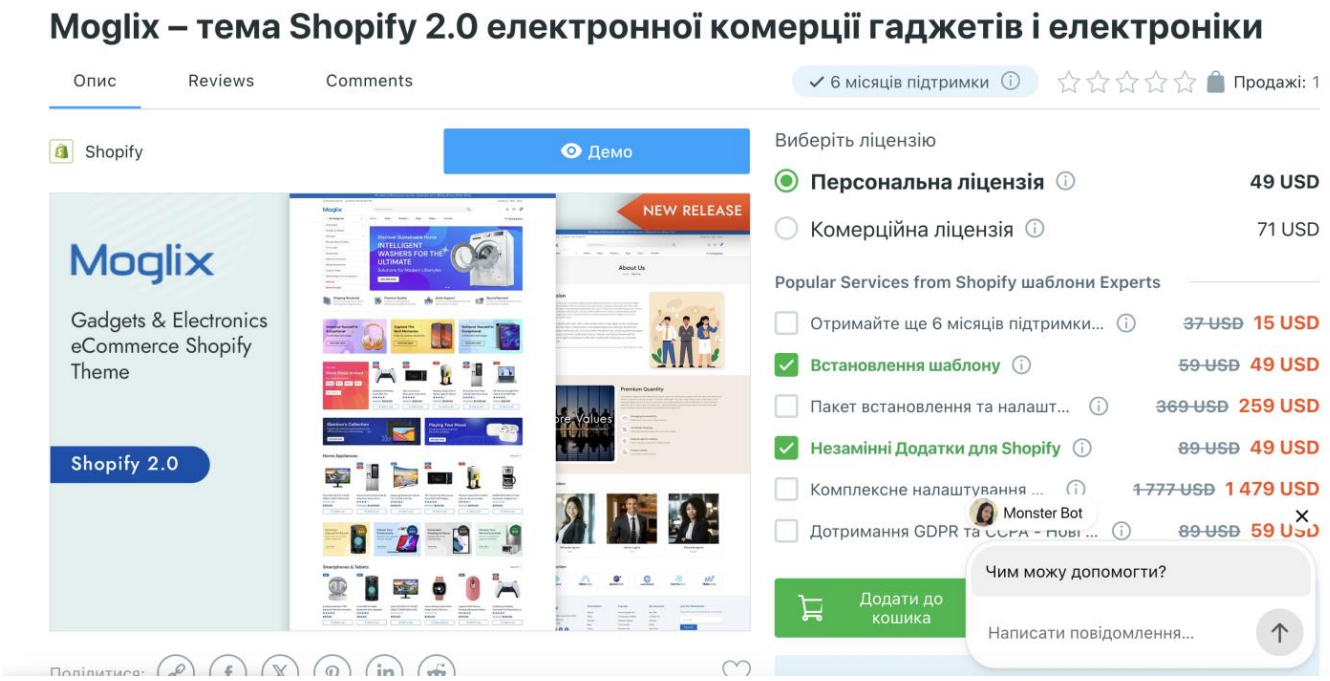


Рис. 2.5 Приклад інтерфейсу платформи Shopify для створення інтернет-магазину

Перевагами Shopify є [8]:

− швидке створення інтернет-магазину без повної розробки з нуля;

− наявність готових шаблонів дизайну;

− підтримка каталогу товарів, кошика та замовлень;

− можливість підключення платіжних і логістичних сервісів;

− велика кількість додаткових модулів [1; 3; 22; 23].

Недоліками Shopify є залежність від сторонньої платформи, наявність платних тарифів і обмежена гнучкість у випадку нестандартної логіки роботи. Для навчального проєкту використання такої платформи не дозволяє повною мірою продемонструвати власну реалізацію серверної частини, бази даних і бізнес-логіки застосунку. Тому Shopify більше підходить для швидкого запуску комерційного магазину, ніж для розробки власного програмного продукту в межах бакалаврської роботи [8].

2.6 Узагальнення результатів аналізу

Проведений аналіз показує, що сучасні засоби онлайн-продажу паперових книжок мають спільні функціональні елементи. Майже всі розглянуті системи містять каталог товарів, сторінки книжок, пошук, фільтрацію, кошик, оформлення замовлення та особистий кабінет користувача. Для адміністратора важливими є засоби керування каталогом, замовленнями та залишками
товарів [1; 3; 22; 23].

Водночас існуючі рішення мають певні обмеження. Великі книжкові платформи часто мають перевантажений інтерфейс і надмірну кількість функцій. Універсальні маркетплейси не завжди адаптовані саме під книжкову продукцію. Готові платформи електронної комерції дозволяють швидко створити магазин, але обмежують розробника у гнучкості та не завжди відповідають вимогам навчального проєкту [1; 3; 22; 23].

Окремо варто виділити функцію безкоштовного завантаження книги у PDF-форматі. У більшості проаналізованих онлайн-книгарень така можливість або відсутня, або реалізована не як основна функція. У застосунку BookMarket передбачено, що користувач може не лише купити паперову книгу, а й завантажити доступну PDF-версію після покупки.

Тобто перевагою BookMarket є поєднання класичного онлайн-магазину паперових книжок із додатковою можливістю доступу до додаткових PDF-матеріалів, а також наявність адміністративного керування каталогом і замовленнями.

Таблиця 2.1

Порівняльна таблиця з аналогами

Критерій	**Yakaboo**	**Книгарня «Є»**	**Book24**	**Amazon Books**	**Book Market**
Каталог паперових книжок	Так	Так	Так	Так	Так
Оформлення замовлення	Так	Так	Так	Так	Так
Облік залишків книжок	Так	Частково	Так	Так	Так
Додаткове завантаження книги у PDF-форматі	Ні	Ні	Ні	Ні	Так
Розподіл книжок за жанрами та категоріями	Так	Так	Так	Частково	Так
Локалізація під український ринок	Так	Так	Так	Ні	Так
Гнучке налаштування системи під вимоги проєкту	Ні	Ні	Ні	Частково	Так

3 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-ЗАСТОСУНКУ BOOKMARKET

3.1 Мова програмування Java

Для реалізації серверної частини web-застосунку BookMarket було обрано мову програмування Java. Вона є однією з поширених мов для створення клієнт-серверних систем, корпоративних застосунків, web-сервісів та програмного забезпечення, яке потребує стабільної роботи з даними [9].

Java підтримує об’єктно-орієнтований підхід до програмування, що дозволяє зручно описувати основні сутності предметної галузі у вигляді класів. У межах даного проєкту такими сутностями є користувачі, книжки, замовлення, позиції замовлень, платежі та статуси. Завдяки цьому структура програмного коду може відповідати логіці предметної області онлайн-продажу паперових книжок [1; 3; 22; 23].

Важливою перевагою Java є наявність великої кількості фреймворків і бібліотек для створення backend-частини web-застосунків. Це дозволяє реалізувати обробку HTTP-запитів, роботу з базою даних, авторизацію користувачів, валідацію даних і формування відповідей для клієнтської частини [2; 19; 20; 21].

У web-застосунку BookMarket Java використовується для реалізації бізнес-логіки системи: отримання списку книжок, пошуку та фільтрації видань, обробки кошика, створення замовлень, зміни статусів, перевірки наявності книжок і взаємодії з базою даних [9].

3.2 Spring та Spring Boot

Для спрощення розробки серверної частини використовується фреймворк Spring та інструмент Spring Boot. Spring надає набір засобів для створення web-

застосунків, роботи з залежностями, реалізації сервісного шару, взаємодії з базою даних і побудови REST API [10].

Spring Boot дозволяє швидше створювати серверні застосунки завдяки автоматичній конфігурації та готовій структурі проєкту. Це зменшує кількість ручних налаштувань і дозволяє більше уваги приділити реалізації функціональності системи [10].

У межах BookMarket Spring використовується для побудови backend-архітектури. Основними складовими серверної частини є контролери, сервіси, репозиторії та моделі даних. Контролери приймають запити від клієнтської частини, сервіси виконують бізнес-логіку, а репозиторії забезпечують доступ до бази даних [10; 11].

Наприклад, під час перегляду каталогу книжок клієнтська частина надсилає запит до backend-частини. Контролер приймає цей запит, передає його до сервісу, сервіс отримує необхідні дані через репозиторій, після чого результат повертається користувачу у вигляді відповіді API.

Spring Boot також зручно використовувати для створення REST API. Завдяки цьому клієнтська частина, реалізована на React, може взаємодіяти із сервером через HTTP-запити та отримувати дані у форматі JSON [10].

3.3 Hibernate та JPA

Для роботи з базою даних у серверній частині використовується Hibernate. Hibernate є ORM-фреймворком, тобто забезпечує об’єктно-реляційне відображення між Java-класами та таблицями бази даних [9].

У звичайному підході розробнику потрібно вручну писати SQL-запити для збереження, оновлення, видалення та отримання даних. Використання Hibernate дозволяє працювати з даними на рівні Java-об’єктів. Наприклад, таблиця books у базі даних може відповідати класу Book, таблиця users - класу User, а таблиця orders - класу Order [9].

JPA є специфікацією, яка описує загальні правила роботи з реляційними базами даних через об’єктну модель. Hibernate у цьому випадку виступає як практична реалізація цієї специфікації [12; 13].

У web-застосунку BookMarket Hibernate використовується для роботи з основними сутностями системи: користувачами, книжками, замовленнями, позиціями замовлень і платежами. Це дозволяє спростити реалізацію операцій CRUD: створення, читання, оновлення та видалення даних [12; 13].

Також Hibernate забезпечує роботу зі зв’язками між сутностями. Наприклад, один користувач може мати декілька замовлень, одне замовлення може містити декілька позицій, а кожна позиція замовлення пов’язана з конкретною книжкою. Такі зв’язки зручно описуються в Java-класах і далі автоматично відображаються під час роботи з базою даних [9].

3.4 PostgreSQL

Для зберігання даних у web-застосунку BookMarket використовується система керування базами даних PostgreSQL. Вона є об’єктно-реляційною СУБД і добре підходить для зберігання структурованих даних web-застосунку [14].

У базі даних BookMarket зберігаються відомості про користувачів, книжки, замовлення, позиції замовлень, платежі та службові дані. Для кожної сутності створюється окрема таблиця, що дозволяє логічно організувати інформацію та уникати дублювання.

PostgreSQL підтримує зв’язки між таблицями за допомогою зовнішніх ключів. Це важливо для забезпечення цілісності даних. Наприклад, замовлення повинно бути пов’язане з конкретним користувачем, а позиція замовлення — з конкретною книжкою. Якщо такі зв’язки правильно налаштовані, система може коректно зберігати й обробляти дані [14].

Також PostgreSQL підтримує індекси, які дозволяють пришвидшити пошук і фільтрацію даних. Для BookMarket це важливо, оскільки користувачі можуть шукати книжки за жанром, активністю, назвою або іншими параметрами.

Індексація окремих полів дозволяє зменшити час виконання запитів до бази даних [14].

У проєкті PostgreSQL використовується як основне сховище даних, з яким взаємодіє серверна частина через Hibernate та репозиторії Spring [10; 11].

3.5 React

Для реалізації клієнтської частини web-застосунку BookMarket використовується бібліотека React. Вона призначена для створення користувацьких інтерфейсів і дозволяє будувати сторінки застосунку з окремих компонентів [15].

Компонентний підхід є зручним для онлайн-магазину книжок, оскільки інтерфейс складається з повторюваних елементів. Наприклад, картка книжки, пошуковий рядок, фільтр, кнопка додавання до кошика, форма оформлення замовлення або елемент списку замовлень можуть бути реалізовані як окремі компоненти [22; 23].

React дозволяє швидко оновлювати інтерфейс без повного перезавантаження сторінки. Це важливо для зручності користувача. Наприклад, після додавання книжки до кошика користувач може одразу побачити зміну кількості товарів. Після застосування фільтра каталог може оновитися без перезавантаження всієї сторінки [15].

У BookMarket React використовується для реалізації таких сторінок [15]:

– головна сторінка;

– каталог книжок;

– сторінка окремої книжки;

– кошик;

– оформлення замовлення;

– сторінка оплати;

– профіль користувача;

– адміністративна панель.

Клієнтська частина взаємодіє із сервером через REST API. Для цього React надсилає HTTP-запити до backend-частини та отримує відповіді у форматі JSON [15].

3.6 Liquibase та REST API

Для керування структурою бази даних використовується Liquibase. Цей інструмент дозволяє описувати зміни бази даних у вигляді міграцій. Завдяки цьому структура бази може створюватися та оновлюватися послідовно, без ручного виконання SQL-команд [18].

У межах BookMarket Liquibase використовується для створення таблиць users, books, orders, order_items та інших структур бази даних. Кожна зміна описується як окремий набір інструкцій, що дозволяє контролювати розвиток бази даних під час роботи над проєктом [18].

REST API використовується для взаємодії між клієнтською та серверною частинами. Клієнтська частина надсилає запити до серверної частини, а сервер обробляє ці запити, звертається до бази даних і повертає результат [16; 17].

Наприклад, для отримання каталогу книжок клієнтська частина може надіслати GET-запит до відповідного API endpoint. Для створення замовлення використовується POST-запит із даними користувача, адресою доставки та списком книжок. Для зміни статусу замовлення адміністратор може виконати PUT-запит.

Використання REST API дозволяє чітко розділити frontend і backend. Це спрощує підтримку проєкту, оскільки клієнтська та серверна частини можуть розроблятися окремо, але взаємодіяти через узгоджені API-запити [16; 17].

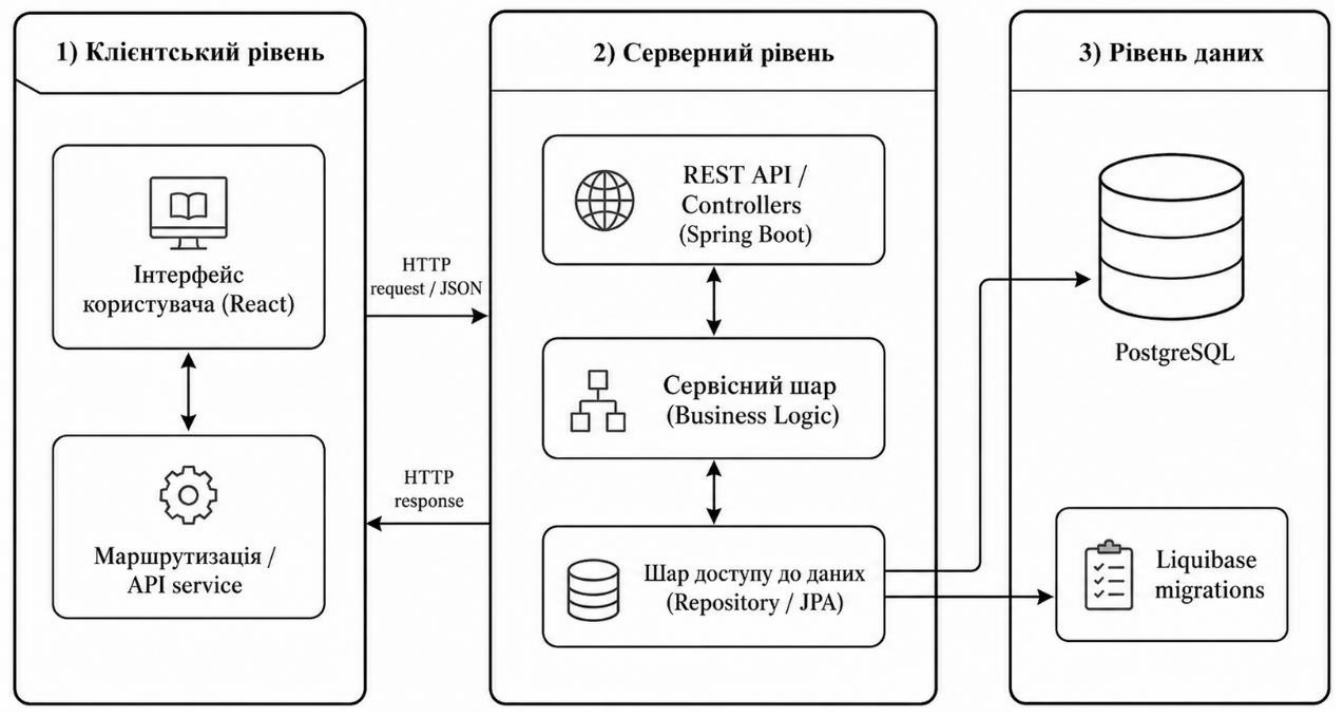


Рис. 3.1 Архітектура web-застосунку BookMarket

3.7 Узагальнення засобів реалізації

У процесі розробки web-застосунку BookMarket було використано сучасний стек програмних засобів, який забезпечує стабільну роботу системи, зручність підтримки та можливість подальшого масштабування. Кожен із використаних інструментів виконує окрему функцію в архітектурі застосунку та забезпечує ефективну взаємодію між клієнтською і серверною частинами системи.

Основною мовою програмування для реалізації серверної частини застосунку було обрано Java. Дана мова є однією з найбільш популярних у сфері корпоративної розробки завдяки високій продуктивності, надійності та платформонезалежності. Використання Java дозволяє створювати безпечні та масштабовані web-застосунки, які можуть функціонувати на різних операційних системах без необхідності зміни програмного коду. Крім того, мова має велику кількість бібліотек та фреймворків, що значно пришвидшує процес розробки програмного забезпечення.

Для побудови backend-частини системи було використано фреймворк Spring Boot. Даний інструмент суттєво спрощує створення серверних застосунків та REST API, оскільки надає готові механізми для конфігурації, роботи з базами даних, безпеки та обробки HTTP-запитів. Завдяки Spring Boot вдалося реалізувати структуру застосунку відповідно до сучасних принципів багаторівневої архітектури, де логіка поділена на контролери, сервіси та рівень доступу до даних. Контролери відповідають за прийом запитів від клієнтської частини, сервіси реалізують бізнес-логіку, а репозиторії забезпечують взаємодію з базою даних.

Для реалізації механізму об’єктно-реляційного відображення було використано Hibernate та специфікацію JPA. Їх застосування дозволило значно спростити роботу з базою даних, оскільки взаємодія здійснюється через Java-об’єкти без необхідності постійного написання SQL-запитів. Hibernate автоматично перетворює об’єкти класів у записи таблиць бази даних та навпаки. Це забезпечує зменшення кількості програмного коду, підвищення його читабельності та спрощення підтримки проєкту. Крім того, використання JPA сприяє незалежності застосунку від конкретної системи керування базами даних.

У ролі системи керування базами даних було обрано PostgreSQL. Дана СКБД є сучасною реляційною базою даних з відкритим програмним кодом, яка забезпечує високу продуктивність, надійність та підтримку складних операцій обробки даних. PostgreSQL використовується для зберігання інформації про користувачів, книги, замовлення, платежі та інші сутності системи. Однією з переваг PostgreSQL є підтримка механізмів транзакційності, що гарантує цілісність даних навіть у випадку помилок або одночасного доступу багатьох користувачів до системи.

Для реалізації клієнтської частини web-застосунку було використано бібліотеку React. Вона дозволяє створювати сучасні інтерактивні користувацькі інтерфейси з використанням компонентного підходу. Основною перевагою React є можливість повторного використання компонентів, що значно спрощує розробку та підтримку frontend-частини застосунку. Крім того, React забезпечує швидке

оновлення інтерфейсу без повного перезавантаження сторінки завдяки використанню механізму Virtual DOM. Це позитивно впливає на швидкодію та зручність користування web-застосунком.

Для керування структурою бази даних та її оновленнями було використано Liquibase. Даний інструмент дозволяє створювати та застосовувати міграції бази даних у автоматизованому режимі. Завдяки Liquibase забезпечується контроль версій структури бази даних, що є особливо важливим під час командної розробки програмного забезпечення. Усі зміни в таблицях, зв’язках чи обмеженнях фіксуються у спеціальних файлах міграцій, що дозволяє легко відтворювати однакову структуру бази даних у різних середовищах розробки та тестування.

Для організації взаємодії між клієнтською та серверною частинами застосунку використовується REST API. Даний підхід дозволяє передавати дані через HTTP-запити у стандартизованому форматі. REST API забезпечує незалежність frontend та backend частин системи, завдяки чому їх можна розробляти та тестувати окремо. Через API клієнтська частина отримує інформацію про книги, користувачів, замовлення та інші дані, а також надсилає запити на створення чи оновлення інформації.

Форматом передачі даних між frontend та backend було обрано JSON, ця технологія використовується для передачі структурованих даних між сервером та клієнтом, що забезпечує швидкий обмін інформацією та спрощує інтеграцію між компонентами системи [1; 3; 22; 23].

4 ПРОЄКТУВАННЯ WEB-ЗАСТОСУНКУ BOOKMARKET

4.1 Логічна структура системи

Проєктування web-застосунку BookMarket передбачає визначення загальної логіки роботи системи, її основних модулів, ролей користувачів, структури бази даних та способу взаємодії між клієнтською і серверною частинами. Оскільки застосунок призначений для онлайн-продажу паперових книжок, його структура повинна охоплювати як користувацькі функції, так і адміністративне керування каталогом, замовленнями та даними [1; 3; 22; 23].

Логічно систему BookMarket можна поділити на три основні рівні: клієнтський рівень, серверний рівень і рівень даних. Клієнтський рівень відповідає за відображення інтерфейсу користувача, обробку дій на сторінках, перехід між розділами застосунку, формування запитів до серверної частини та відображення отриманих результатів. До цього рівня належать головна сторінка, каталог книжок, сторінка окремої книжки, кошик, сторінка оформлення замовлення, профіль користувача та адміністративні сторінки [22; 23].

Серверний рівень реалізує основну бізнес-логіку системи. Він приймає запити від клієнтської частини, перевіряє коректність отриманих даних, виконує необхідні операції з книжками, користувачами, замовленнями та платежами, а також звертається до бази даних. На цьому рівні розміщуються контролери, сервіси, репозиторії, моделі даних та механізми авторизації користувачів [2; 19; 20; 21].

Рівень даних представлений базою даних PostgreSQL, у якій зберігається інформація про користувачів, книжки, замовлення та позиції замовлень. Такий поділ дозволяє забезпечити структуроване зберігання інформації, цілісність даних і можливість подальшого розширення системи [14].

Основними функціональними модулями web-застосунку BookMarket є:

– модуль користувачів, який забезпечує реєстрацію, авторизацію, роботу з профілем і розмежування ролей;

– модуль каталогу книжок, який відповідає за перегляд, пошук, фільтрацію та відображення детальної інформації про книжки;

– модуль PDF-доступу, який дозволяє завантажувати доступні книжки у PDF-форматі;

– модуль кошика, який забезпечує додавання товарів, зміну кількості книжок і підготовку замовлення;

– модуль замовлень, який відповідає за створення, збереження, перегляд і зміну статусів замовлень;

– модуль платежів, який описує логіку вибору способу оплати та фіксації платіжної інформації;

– адміністративний модуль, який дозволяє керувати книжками, користувачами, замовленнями, залишками товарів і статусами [2; 19; 20; 21].

4.2 Діаграма варіантів використання web-застосунку BookMarket

Для опису функціональних можливостей системи доцільно використати діаграму варіантів використання. Вона дозволяє показати, які ролі існують у системі та з якими функціями вони взаємодіють [24].

У web-застосунку BookMarket передбачено дві основні ролі: користувач і адміністратор. Користувач є основним учасником процесу купівлі книжок. Він може зареєструватися, авторизуватися, переглядати каталог, виконувати пошук і фільтрацію, відкривати сторінку окремої книжки, додавати товари до кошика, оформлювати замовлення, переглядати його статус і завантажувати доступні книжки у PDF-форматі.

Адміністратор має розширені можливості. Він може виконувати всі дії звичайного користувача, а також керувати каталогом книжок, додавати нові видання, редагувати інформацію про книжки, змінювати кількість товарів на складі, переглядати замовлення, змінювати їх статуси та керувати користувачами.

Такий підхід є логічним, оскільки адміністратор повинен мати можливість не лише обслуговувати систему, а й перевіряти її роботу з боку звичайного користувача.

Діаграма варіантів використання також дозволяє показати зв’язок між ролями. Адміністратор є розширеною роллю відносно користувача, тобто він успадковує базові можливості користувача та отримує додаткові адміністративні функції. Це спрощує розуміння структури доступу в системі [24].

Основними варіантами використання для користувача є:

– реєстрація та авторизація;

– перегляд каталогу книжок;

– пошук і фільтрація книжок;

– перегляд детальної сторінки книжки;

– завантаження доступної PDF-версії;

– додавання книжки до кошика;

– редагування кошика;

– оформлення замовлення;

– перегляд статусу замовлення [2; 19; 20; 21].

Для адміністратора додатково передбачено:

– додавання книжок до каталогу;

– редагування інформації про книжки;

– видалення або деактивація книжок;

– оновлення складських залишків;

– перегляд усіх замовлень;

– зміна статусу замовлення;

– перегляд користувачів системи;

– контроль статистики роботи магазину.

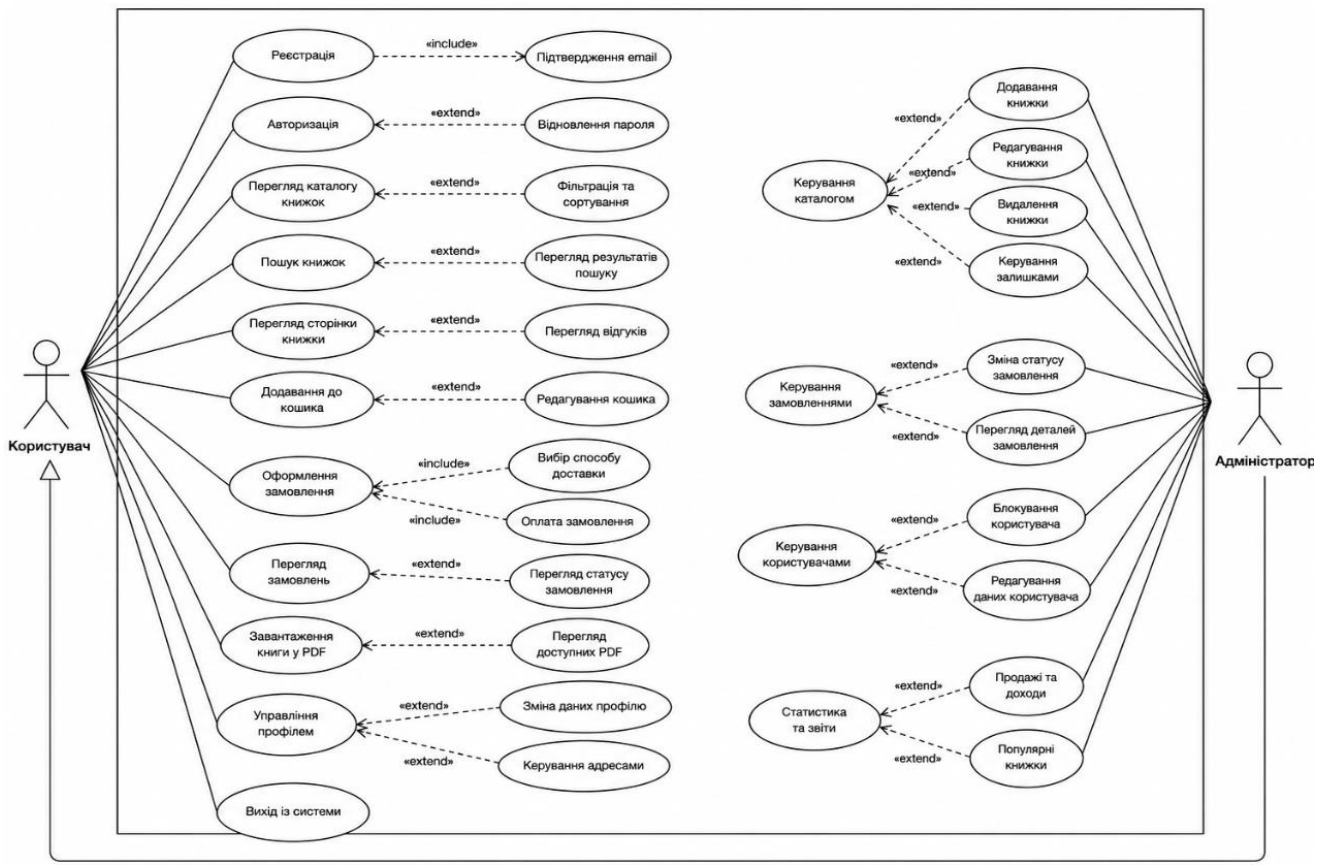


Рис. 4.1 Діаграма варіантів використання web-застосунку BookMarket

4.3 Діаграми послідовності

Діаграми послідовності використовуються для опису взаємодії між об’єктами системи в межах конкретного сценарію. Для web-застосунку BookMarket доцільно розглянути два ключові сценарії: пошук і фільтрацію книжок, а також оформлення замовлення. Саме ці процеси є основними для користувача під час роботи з онлайн-магазином.

4.3.1 Діаграма послідовності пошуку та фільтрації книжок

Процес пошуку та фільтрації книжок починається з дії користувача на клієнтській частині. Користувач вводить назву книжки, автора або ключове слово, а також може обрати додаткові фільтри: жанр, ціну, мову, наявність або інші параметри. Після цього клієнтська частина формує HTTP-запит до серверної частини [16; 17].

Серверна частина приймає запит через відповідний контролер. Далі контролер передає параметри пошуку до сервісного шару, де виконується основна логіка обробки. Сервіс перевіряє отримані параметри, формує критерії пошуку та звертається до репозиторію. Репозиторій виконує запит до бази даних PostgreSQL і отримує список книжок, які відповідають заданим умовам [14].

Після цього результат повертається назад у зворотному порядку: база даних передає знайдені записи репозиторію, репозиторій повертає їх сервісу, сервіс готує відповідь, а контролер надсилає її клієнтській частині. На frontend-рівні отримані дані відображаються у вигляді списку книжок або повідомлення про те, що за заданими параметрами нічого не знайдено.

Такий сценарій дозволяє користувачу швидко отримати релевантний список видань, не переглядаючи весь каталог вручну. Це особливо важливо для системи з великою кількістю книжок.

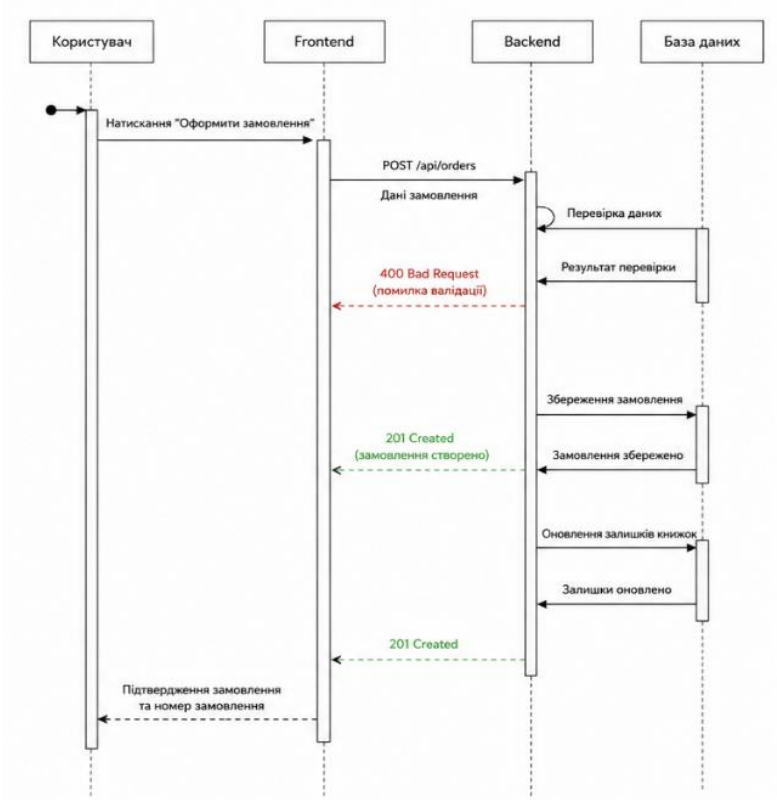


Рис. 4.2 Діаграма послідовності оформлення замовлення

4.3.2 Діаграма послідовності оформлення замовлення

Оформлення замовлення є одним із найважливіших сценаріїв роботи системи. Він починається після того, як користувач додав потрібні книжки до кошика та перейшов до сторінки оформлення покупки. На цьому етапі користувач вводить контактні дані, місто, адресу доставки, номер телефону та за потреби додатковий коментар.

Клієнтська частина формує запит на створення замовлення та передає його до серверної частини. Сервер отримує дані, перевіряє їх коректність і визначає, чи всі обов’язкові поля заповнені. Якщо дані некоректні, система повертає повідомлення про помилку. Якщо дані правильні, сервер перевіряє наявність книжок на складі.

Після успішної перевірки сервер створює запис у таблиці замовлень, формує позиції замовлення, розраховує загальну суму та оновлює залишки книжок. Дані зберігаються в базі даних, а користувач отримує повідомлення про успішне створення замовлення. Якщо певної книжки немає в наявності або кількість примірників недостатня, система повинна повідомити користувача про неможливість оформлення замовлення в поточному вигляді.

Цей сценарій демонструє взаємодію між користувачем, клієнтською частиною, сервером і базою даних. Він також показує важливість перевірки даних перед збереженням замовлення, оскільки саме на цьому етапі відбувається зміна складських залишків і фіксація покупки.

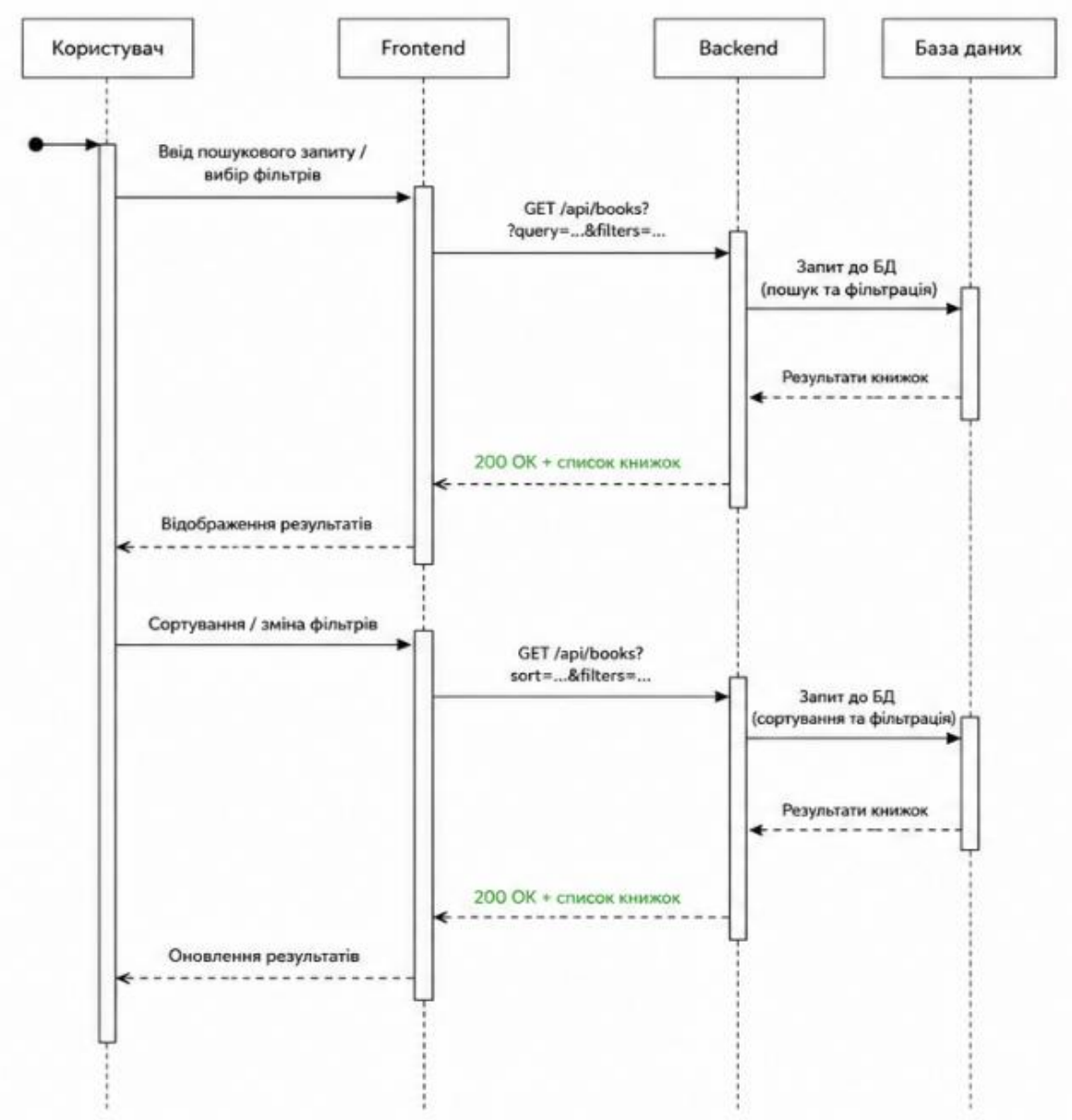


Рис. 4.3 Діаграма послідовності пошуку та фільтрації

4.4 Мапи web-застосунку для авторизованого та неавторизованого користувача

Мапа web-застосунку використовується для відображення структури сторінок системи та переходів між ними. Для BookMarket доцільно розділити мапу на два варіанти: для неавторизованого користувача та для авторизованого користувача. Це пов’язано з тим, що набір доступних сторінок і функцій залежить від стану входу користувача до системи.

Неавторизований користувач може переглядати головну сторінку, каталог книжок, сторінку окремої книжки, інформаційні сторінки, сторінку входу та сторінку реєстрації. Такий користувач може ознайомитися з асортиментом, але для оформлення замовлення або перегляду особистих даних йому необхідно пройти авторизацію [2; 19; 20; 21].

Авторизований користувач отримує доступ до розширених можливостей. Він може додавати книжки до кошика, оформлювати замовлення, переглядати власні замовлення, відкривати профіль, завантажувати доступні PDF-файли та виходити із системи. Якщо авторизований користувач має роль адміністратора, йому також відкривається доступ до адміністративної панелі.

Адміністративна панель може включати сторінки керування книжками, замовленнями, користувачами та статистикою. Такий розподіл дозволяє забезпечити безпеку системи, оскільки адміністративні функції недоступні звичайному користувачу або неавторизованому відвідувачу.

Мапа застосунку також допомагає під час проєктування інтерфейсу, оскільки дозволяє визначити, які сторінки повинні бути створені та як користувач буде переходити між ними. Це зменшує ризик пропуску важливих екранів і робить структуру системи більш зрозумілою [22; 23].

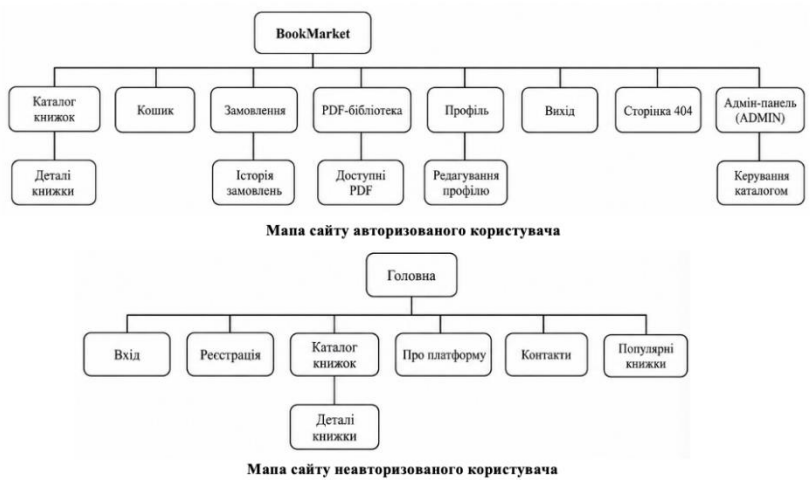


Рис. 4.4 Мапи web-застосунку BookMarket для авторизованого та неавторизованого користувача

4.5 Проєктування бази даних

База даних є важливою частиною web-застосунку BookMarket, оскільки саме в ній зберігається основна інформація про користувачів, книжки, замовлення та позиції замовлень. Проєктування бази даних повинно забезпечувати структурованість, цілісність і зручність доступу до даних [22; 23].

У межах проєкту передбачено такі основні таблиці:

– users;

– books;

– orders;

– order_items.

Таблиця users призначена для зберігання інформації про користувачів системи. Вона містить ідентифікатор користувача, електронну пошту, хеш пароля, повне ім’я, роль і дату створення облікового запису. Поле електронної пошти повинно бути унікальним, оскільки воно використовується для ідентифікації користувача під час авторизації. Поле ролі дозволяє розмежовувати доступ між звичайним користувачем і адміністратором.

Таблиця books містить інформацію про книжки, представлені в каталозі. До неї входять назва, автор, жанр, видавництво, рік видання, ціна, кількість на складі, посилання на обкладинку, колір обкладинки, опис, ознака активності та службові поля дати створення й оновлення. Наявність поля active дозволяє не видаляти книжку фізично з бази даних, а лише приховувати її з каталогу, якщо вона тимчасово недоступна або не повинна відображатися користувачам.

Таблиця orders зберігає інформацію про замовлення. Вона пов’язана з таблицею users, оскільки кожне замовлення належить конкретному користувачу. У таблиці зберігається статус замовлення, ім’я покупця, телефон, місто, адреса, коментар, загальна сума, а також дата створення та оновлення. Статус замовлення дозволяє відстежувати його стан: нове, в обробці, оплачене, відправлене, виконане або скасоване.

Таблиця order_items використовується для зберігання окремих позицій замовлення. Одне замовлення може містити декілька книжок, тому між orders і order_items існує зв’язок один-до-багатьох. Кожна позиція замовлення також пов’язана з конкретною книжкою з таблиці books. У таблиці зберігається кількість книжок, ціна за одиницю та загальна сума по позиції.

Зв’язки між таблицями забезпечують цілісність даних. Наприклад, замовлення не може існувати без користувача, а позиція замовлення не може бути створена без відповідного замовлення та книжки. Для прискорення пошуку використовуються індекси, зокрема для жанру книжки, активності книжки, користувача в замовленні та статусу замовлення.

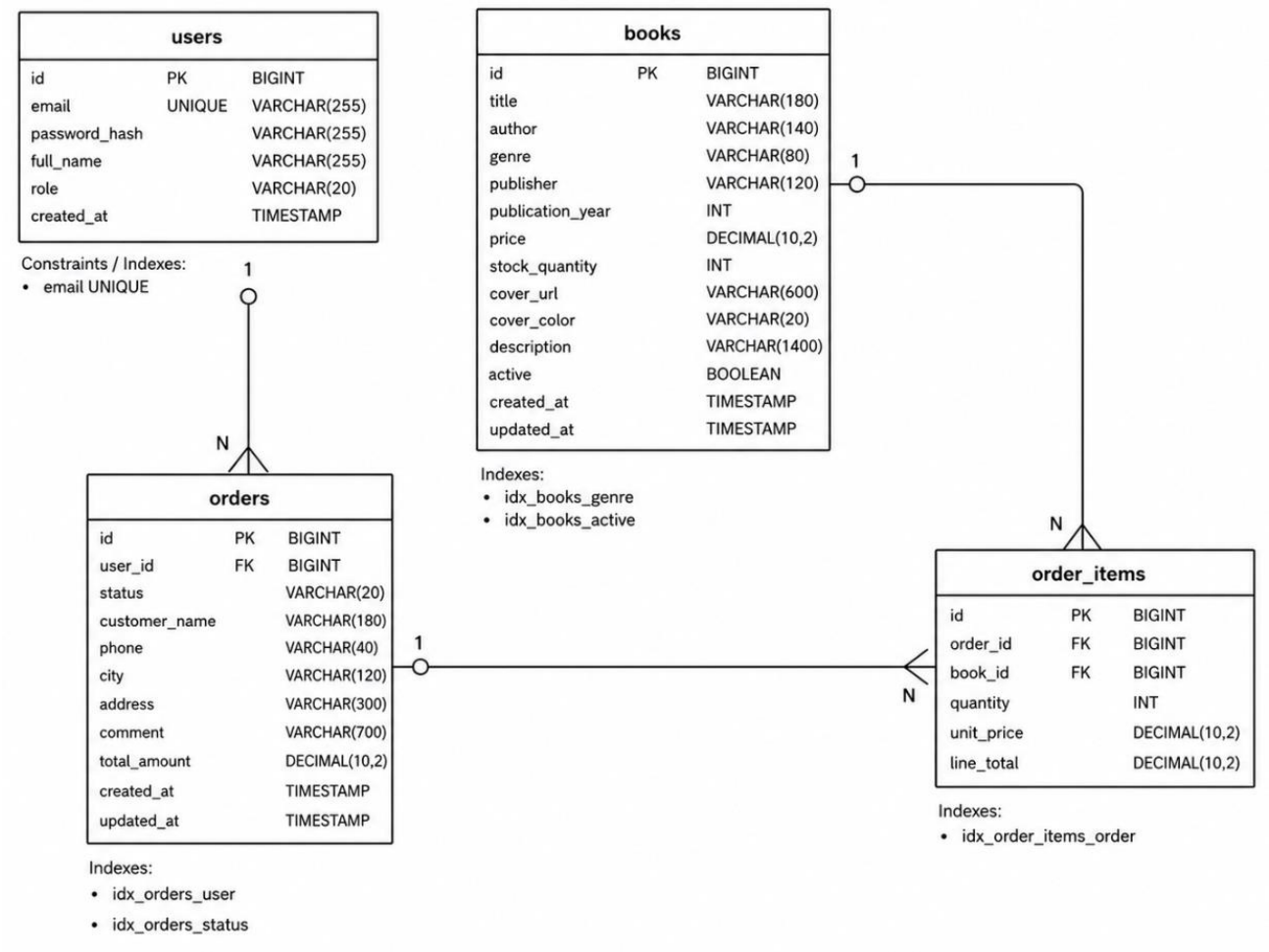


Рис. 4.5 Структура бази даних web-застосунку BookMarket

Таблиця 4.1

Опис основних таблиць бази даних web-застосунку BookMarket

Таблиця	**Призначення**	**Основні поля**
users	Зберігання даних користувачів системи	id, email, password_hash, full_name, role, created_at
books	Зберігання інформації про книжки в каталозі	id, title, author, genre, publisher, publication_year, price, stock_quantity, cover_url, description, active
orders	Зберігання інформації про оформлені замовлення	id, user_id, status, customer_name, phone, city, address, total_amount, created_at
order_items	Зберігання окремих позицій у складі замовлення	id, order_id, book_id, quantity, unit_price, line_total

4.6 Архітектура web-застосунку BookMarket

Архітектура web-застосунку BookMarket побудована за клієнт-серверним принципом. Такий підхід дозволяє розділити відповідальність між частинами системи: клієнтська частина відповідає за інтерфейс і взаємодію з користувачем, серверна частина — за обробку логіки та доступ до даних, а база даних — за зберігання інформації [22; 23].

Клієнтська частина реалізується за допомогою React. Вона містить сторінки та компоненти, які користувач бачить у браузері. Наприклад, каталог книжок, сторінка книги, кошик, форма замовлення та адміністративна панель є частинами frontend-рівня. Коли користувач виконує певну дію, клієнтська частина формує HTTP-запит до серверної частини [15].

Серверна частина реалізується на Java з використанням Spring. Вона містить REST API, контролери, сервіси, репозиторії та моделі даних. Контролери

приймають запити від frontend-рівня, сервіси виконують бізнес-логіку, а репозиторії взаємодіють з базою даних через Hibernate/JPA [9].

Рівень даних представлений PostgreSQL. У базі даних зберігаються всі основні дані системи: користувачі, книжки, замовлення та позиції замовлень. Для створення та оновлення структури бази даних використовується Liquibase, що дозволяє керувати змінами схеми бази через міграції [14].

Перевагою такої архітектури є її зрозумілість і масштабованість. Клієнтська та серверна частини можуть розроблятися окремо, а взаємодія між ними відбувається через REST API. У майбутньому це дозволяє додавати нові функції, наприклад мобільний застосунок або інтеграцію з іншими сервісами, без повної перебудови системи [16; 17].

4.7 Діаграма класів модулів керування книжками та користувачами

Для опису внутрішньої структури серверної частини застосунку використовується діаграма класів. Вона дозволяє показати, які класи беруть участь у реалізації окремих модулів і як вони пов’язані між собою [24].

Модуль керування книжками відповідає за роботу з каталогом. До нього входять класи BookController, AdminBookController, BookService, BookRepository та сутність Book. Клас BookController використовується для обробки публічних запитів користувача: отримання списку книжок, перегляду сторінки окремої книжки, пошуку, фільтрації та отримання жанрів. Клас AdminBookController відповідає за адміністративні операції: створення нової книжки, редагування наявної, оновлення кількості на складі та деактивацію книжки.

Клас BookService містить основну бізнес-логіку роботи з книжками. Він перевіряє отримані дані, виконує пошук, обробляє параметри фільтрації, перевіряє наявність товару та передає запити до репозиторію. BookRepository відповідає за доступ до таблиці books у базі даних. Він може містити методи пошуку книжок за жанром, активністю, назвою або автором.

Сутність Book описує книжку як об’єкт предметної області. Вона містить поля: ідентифікатор, назву, автора, жанр, видавництво, рік видання, ціну, кількість на складі, обкладинку, опис і ознаку активності.

Модуль користувачів відповідає за реєстрацію, авторизацію, профіль користувача та адміністративне керування користувачами. До нього входять UserController, AdminUserController, UserService, UserRepository, сутність User і перелічення UserRole [2; 19; 20; 21].

UserController обробляє запити звичайного користувача, наприклад отримання профілю або зміну особистих даних. AdminUserController дозволяє адміністратору переглядати список користувачів, змінювати ролі або блокувати облікові записи. UserService містить логіку роботи з користувачами, а UserRepository забезпечує доступ до таблиці users.

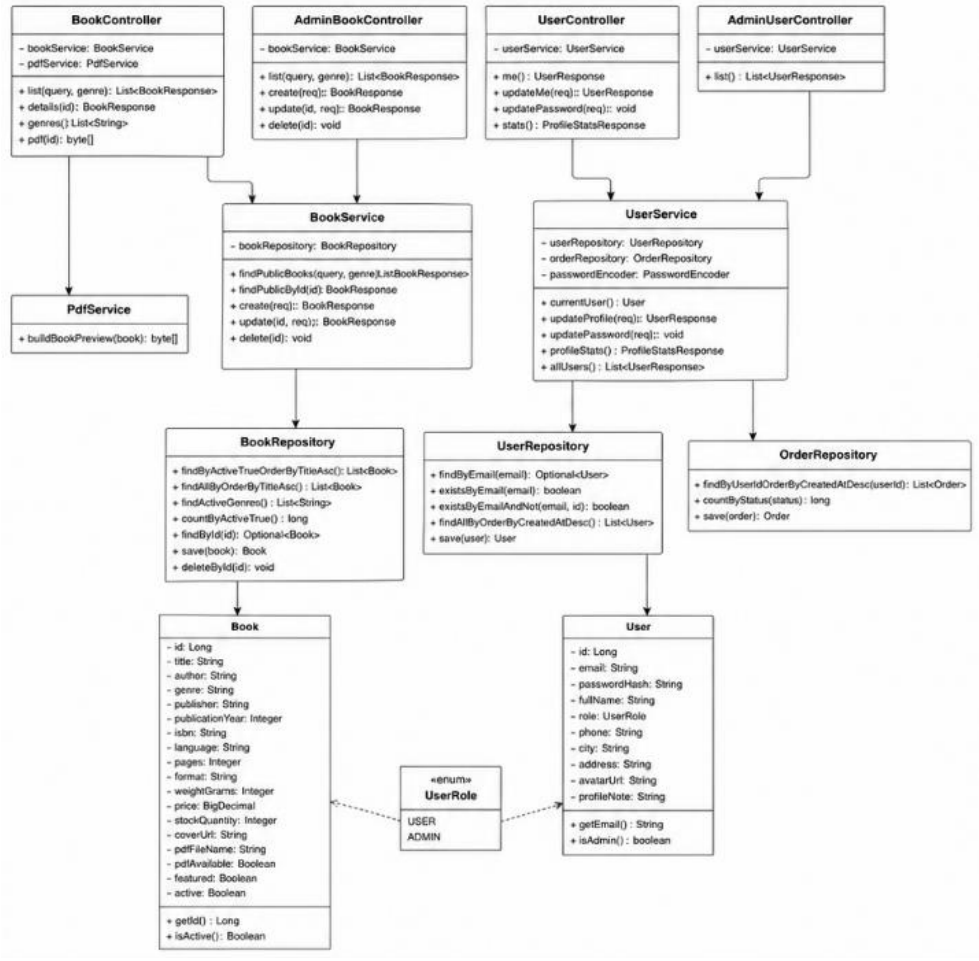


Рис. 4.6 Діаграма класів модулів керування книжками та користувачами

4.8 Діаграма класів модулів замовлень і платежів

Модулі замовлень і платежів відповідають за обробку покупки після того, як користувач додав книжки до кошика та перейшов до оформлення замовлення. Ці модулі є важливими для роботи онлайн-магазину, оскільки саме вони забезпечують створення замовлення, фіксацію його складу, розрахунок суми та подальшу зміну статусу.

До модуля замовлень входять класи OrderController, AdminOrderController, OrderService, OrderRepository, сутності Order та OrderItem, а також перелічення OrderStatus.

OrderController використовується для обробки запитів користувача. Він дозволяє створити нове замовлення, переглянути власні замовлення та отримати детальну інформацію про конкретне замовлення. AdminOrderController призначений для адміністратора. Через нього можна переглядати всі замовлення, змінювати їх статуси та контролювати процес обробки.

OrderService містить основну логіку створення замовлення. Він перевіряє дані користувача, аналізує список книжок у замовленні, перевіряє наявність товарів, розраховує загальну суму, формує позиції замовлення та передає дані до репозиторію для збереження. OrderRepository забезпечує доступ до таблиці orders, а також може містити методи пошуку замовлень за користувачем або статусом.

Сутність Order описує замовлення в системі. Вона містить ідентифікатор, користувача, статус, контактні дані, адресу доставки, коментар, загальну суму та дату створення. Сутність OrderItem описує окрему позицію замовлення. Вона містить посилання на замовлення, посилання на книжку, кількість, ціну за одиницю та суму по позиції.

Модуль платежів містить PaymentController, AdminPaymentController, PaymentService, PaymentRepository, сутність Payment і перелічення, пов’язані зі способом оплати, статусом платежу та способом доставки. Цей модуль може використовуватися для фіксації інформації про оплату замовлення, вибраний спосіб оплати та поточний статус платежу.

Навіть якщо в межах базової версії системи не передбачена повноцінна інтеграція з реальним платіжним сервісом, наявність окремого модуля платежів є доцільною. Це дозволяє в майбутньому додати онлайн-оплату, підтвердження платежу, історію оплат і взаємодію з платіжними провайдерами без суттєвої зміни загальної структури застосунку.

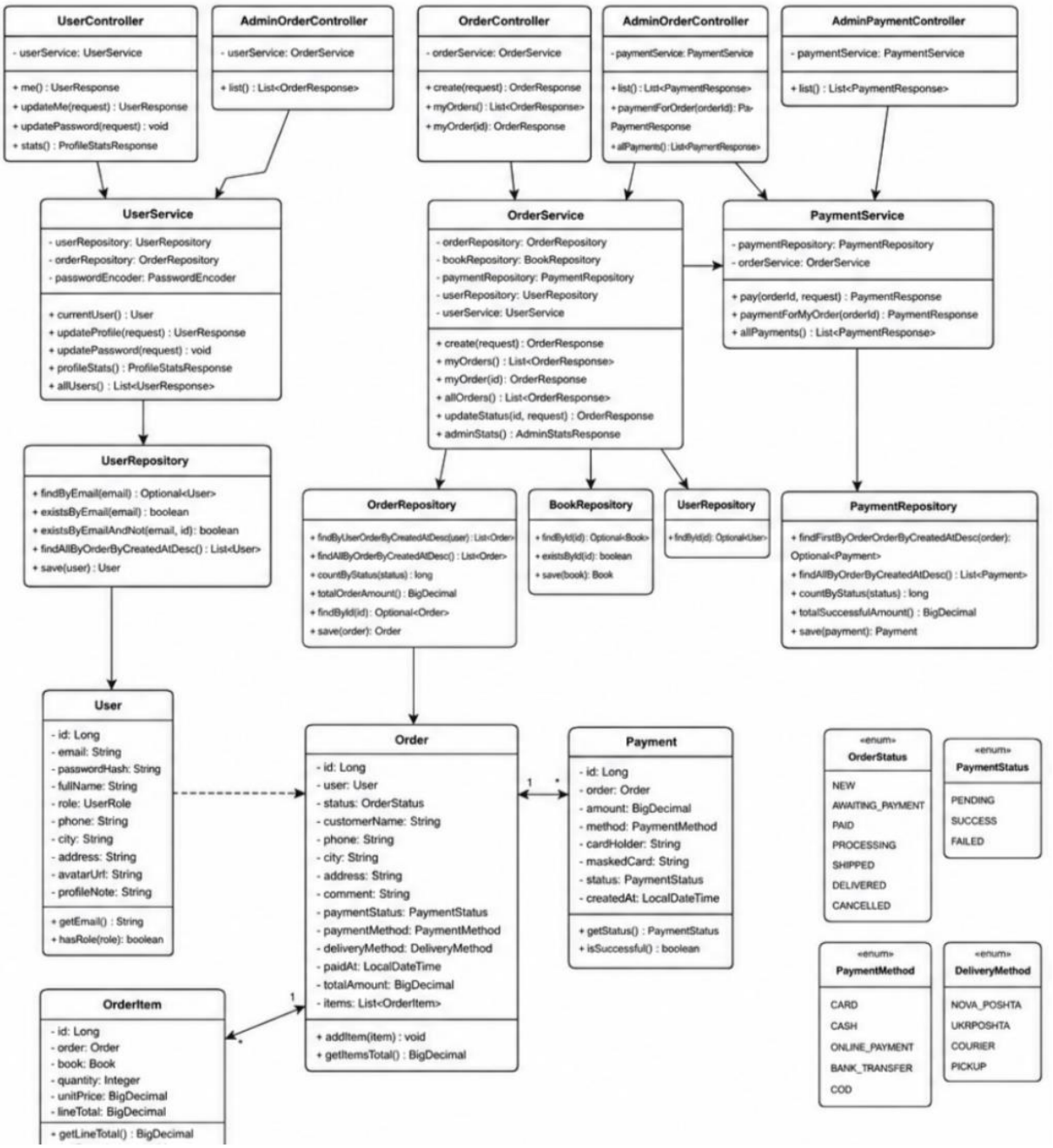


Рис. 4.7 Діаграма класів модулів замовлень і платежів

5 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ WEB-ЗАСТОСУНКУ BOOKMARKET

5.1 Реалізація серверної частини

Серверна частина web-застосунку BookMarket реалізована з використанням мови програмування Java та фреймворку Spring. Вона відповідає за обробку запитів від клієнтської частини, виконання бізнес-логіки, роботу з базою даних, керування користувачами, книжками, замовленнями та платежами [9].

Backend-частина застосунку побудована за багаторівневою структурою. Такий підхід дозволяє розділити відповідальність між окремими частинами програми та зробити код більш зрозумілим для підтримки й подальшого розвитку.

Основними рівнями серверної частини є:

– рівень контролерів;

– сервісний рівень;

– рівень репозиторіїв;

– рівень сутностей;

– рівень передачі даних.

Контролери відповідають за приймання HTTP-запитів від клієнтської частини. Наприклад, коли користувач відкриває каталог книжок, frontend надсилає запит до відповідного контролера. Контролер отримує параметри запиту, передає їх до сервісного шару та повертає результат у форматі JSON [16; 17].

Сервісний рівень містить основну бізнес-логіку застосунку. Саме тут виконується перевірка даних, пошук книжок, фільтрація, створення замовлень, розрахунок загальної суми, перевірка залишків товарів і зміна статусів. Такий підхід дозволяє не перевантажувати контролери зайвою логікою, а залишити їх лише точкою входу для запитів.

Репозиторії забезпечують доступ до бази даних. Для цього використовується Spring Data JPA разом із Hibernate. Репозиторії дозволяють виконувати операції

створення, читання, оновлення та видалення записів без необхідності вручну писати велику кількість SQL-запитів [10; 11].

Сутності описують основні об’єкти предметної галузі: користувача, книжку, замовлення, позицію замовлення та платіж. Вони відповідають таблицям бази даних і використовуються Hibernate для об’єктно-реляційного відображення [12; 13].

У серверній частині web-застосунку BookMarket реалізовано такі основні модулі:

– модуль користувачів;

– модуль книжок;

– модуль замовлень;

– модуль платежів;

– модуль адміністративного керування;

– модуль PDF-доступу.

Модуль користувачів забезпечує реєстрацію, авторизацію, роботу з профілем і розмежування доступу за ролями. Для звичайного користувача доступні функції перегляду каталогу, оформлення замовлення та перегляду власних даних. Для адміністратора відкривається доступ до розширених можливостей керування системою [2; 19; 20; 21].

Модуль книжок відповідає за роботу з каталогом. Він забезпечує отримання списку книжок, перегляд окремої книжки, пошук, фільтрацію, отримання жанрів, додавання нових книжок адміністратором, редагування інформації та оновлення кількості товарів на складі.

Модуль замовлень реалізує процес створення покупки. Після оформлення замовлення сервер перевіряє вхідні дані, перевіряє наявність книжок, створює запис про замовлення, додає позиції замовлення та розраховує загальну суму. Після цього адміністратор може змінювати статус замовлення.

Модуль платежів використовується для фіксації інформації про спосіб оплати та статус платежу. У базовій реалізації він може виконувати роль

внутрішнього механізму обліку платежів, а в майбутньому може бути розширений інтеграцією з реальними платіжними сервісами.

Модуль PDF-доступу забезпечує можливість завантаження доступних книжок у PDF-форматі. Така функція є однією з особливостей BookMarket, оскільки користувач може не лише купити паперову книжку, а й отримати доступну електронну версію.

5.2 Реалізація клієнтської частини

Клієнтська частина web-застосунку BookMarket реалізована з використанням бібліотеки React. Вона відповідає за відображення інтерфейсу користувача, обробку дій на сторінках, навігацію між розділами та взаємодію із серверною частиною через REST API [15].

Інтерфейс застосунку побудовано за компонентним підходом. Це означає, що окремі частини сторінок реалізуються як самостійні компоненти, які можна повторно використовувати в різних місцях застосунку. Наприклад, картка книжки може використовуватися і на головній сторінці, і в каталозі, і в добірках популярних видань.

Основними сторінками клієнтської частини є:

– головна сторінка;

– сторінка каталогу книжок;

– сторінка окремої книжки;

– сторінка кошика;

– сторінка оформлення або оплати замовлення;

– сторінка профілю користувача;

– сторінка замовлень користувача;

– адміністративна панель.

Головна сторінка виконує роль стартової точки взаємодії із застосунком. Вона знайомить користувача з платформою, дає можливість перейти до каталогу та переглянути основні або популярні книжки. Головна сторінка повинна бути простою, зрозумілою та не перевантаженою зайвими елементами.

Каталог книжок є однією з головних сторінок застосунку. На ньому користувач бачить список доступних книжок, може виконати пошук за назвою або автором, а також застосувати фільтри. Кожна книжка в каталозі відображається у вигляді картки з основною інформацією: назвою, автором, ціною, жанром, обкладинкою та кнопкою переходу до детальної сторінки.

Сторінка окремої книжки містить повний опис видання. На ній відображаються назва, автор, жанр, видавництво, рік публікації, ціна, кількість на складі, опис і кнопка додавання до кошика. Якщо для книжки доступний PDF-файл, на сторінці також передбачено кнопку для його завантаження.

Сторінка кошика дозволяє користувачу переглянути обрані товари перед оформленням замовлення. Користувач може змінити кількість книжок, видалити непотрібні позиції та переглянути загальну суму. Після перевірки кошика користувач переходить до сторінки оформлення замовлення.

Сторінка оформлення замовлення містить форму для введення контактних даних, міста, адреси доставки, телефону та коментаря. Після підтвердження форма передає дані до серверної частини, де створюється замовлення.

Адміністративна панель призначена для користувачів із роллю адміністратора. У ній можна додавати нові книжки, редагувати наявні, змінювати кількість на складі, переглядати замовлення та змінювати їх статуси. Це дозволяє адміністратору керувати роботою онлайн-магазину без прямого доступу до бази даних.

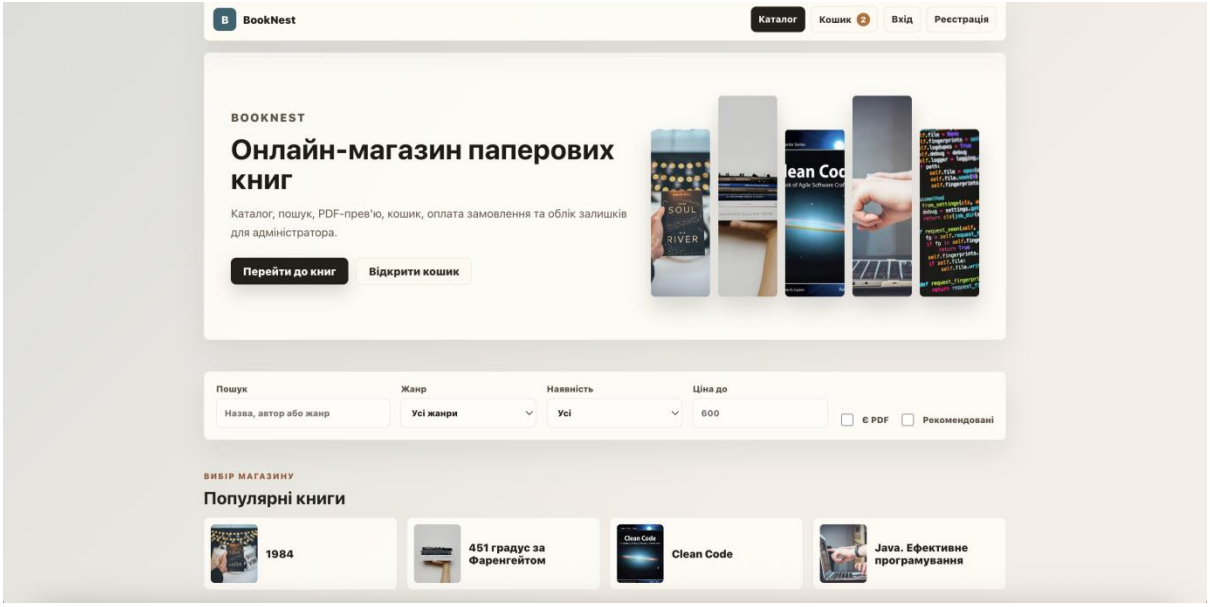


Рис. 5.1 Головна сторінка web-застосунку BookMarket

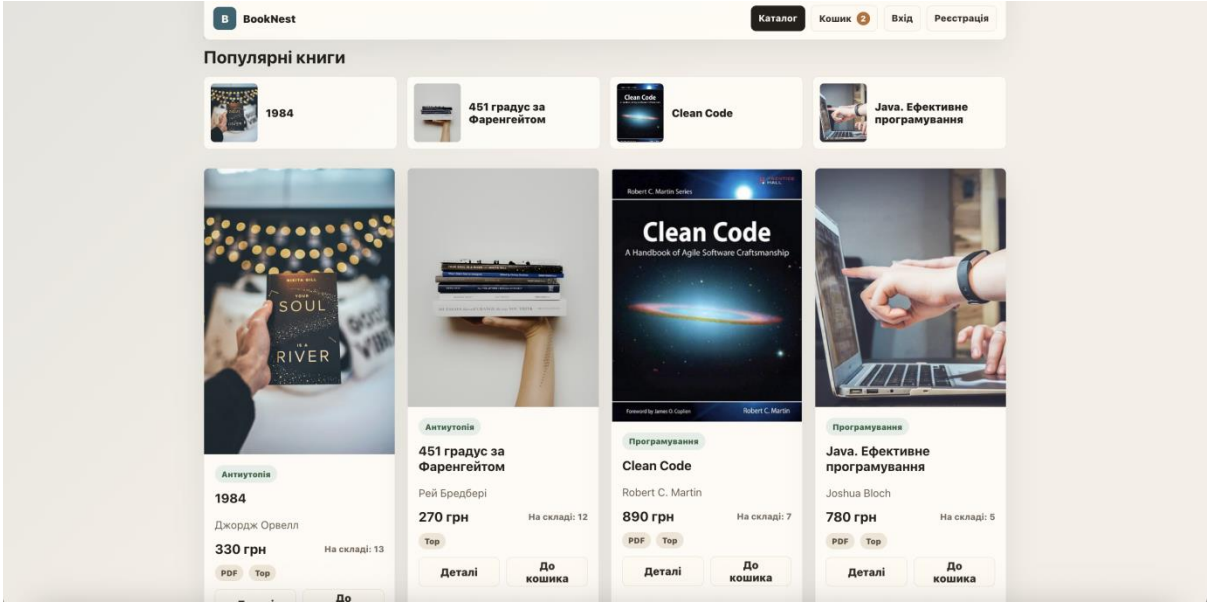


Рис. 5.2 Каталог книжок web-застосунку BookMarket

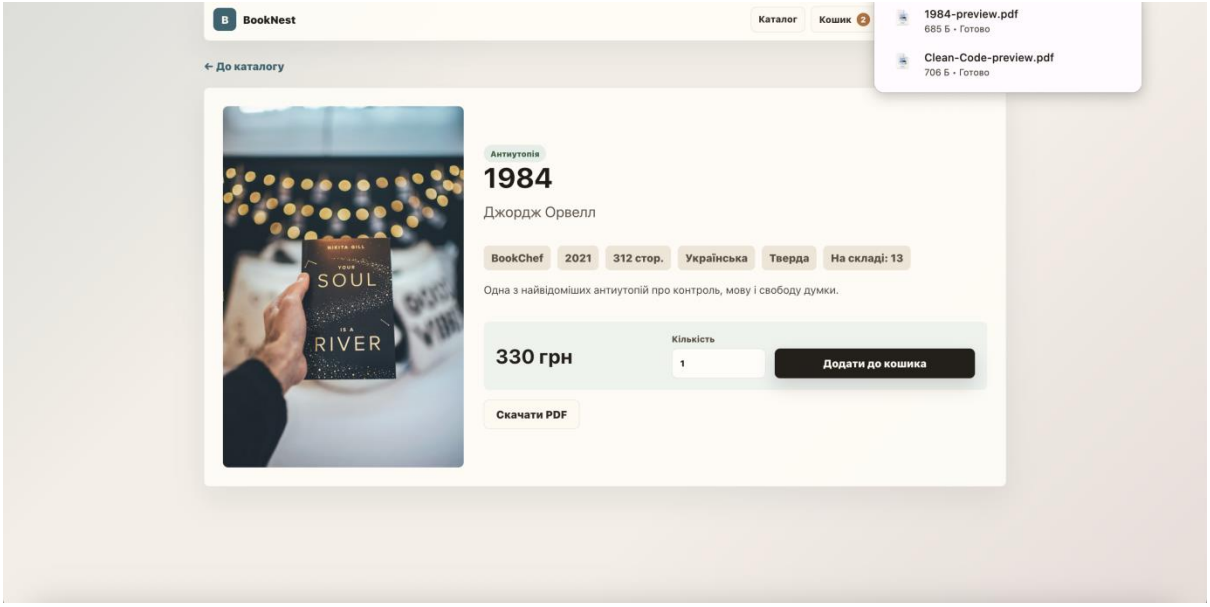


Рис. 5.3 Сторінка книги та завантаження PDF-файлу

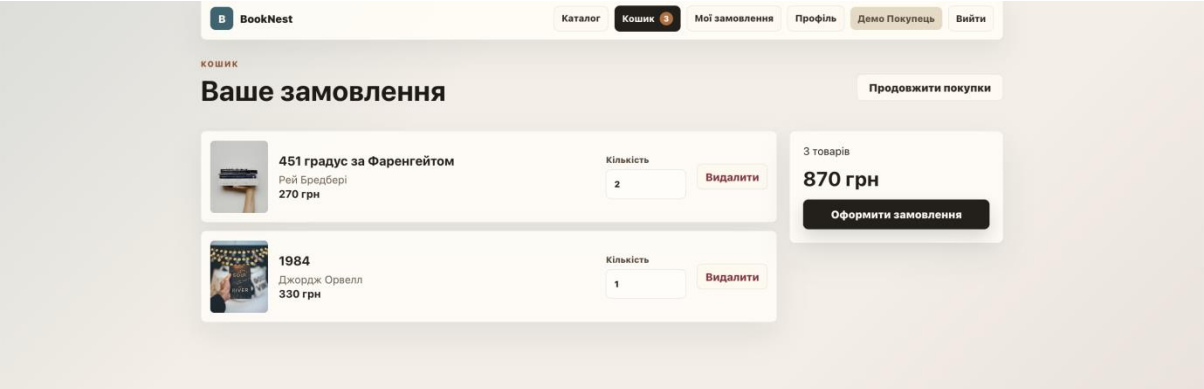


Рис. 5.4 Сторінка кошика з обраними книжками

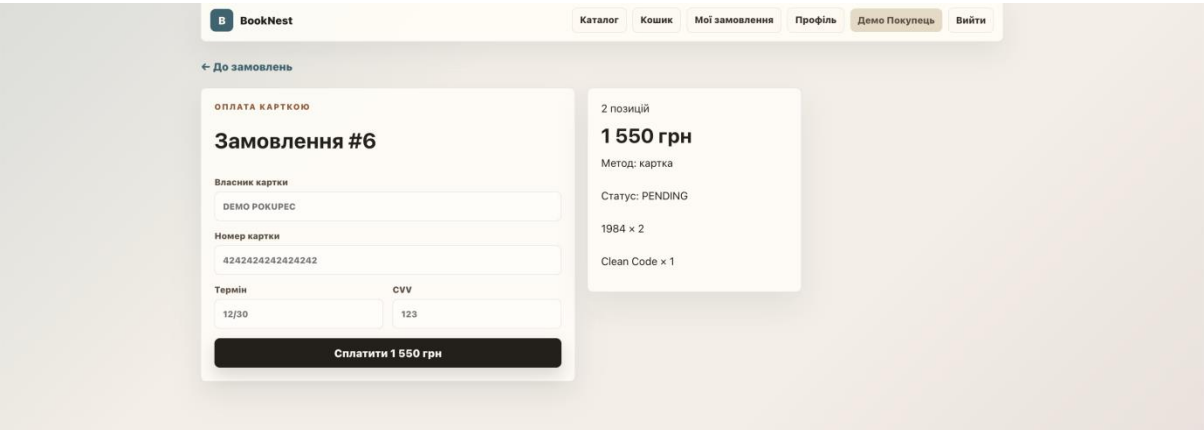


Рис. 5.5 Сторінка оформлення замовлення

5.3 Реалізація роботи з базою даних

Робота з базою даних у web-застосунку BookMarket реалізована з використанням PostgreSQL, Hibernate, JPA та Liquibase. PostgreSQL використовується як основне сховище даних, Hibernate забезпечує об’єктно-реляційне відображення, JPA визначає правила роботи з сутностями, а Liquibase використовується для керування змінами структури бази даних [12; 13].

Структура бази даних створюється за допомогою міграцій. Це дозволяє послідовно описувати створення таблиць, індексів, зовнішніх ключів і службових обмежень. Такий підхід є зручним, оскільки всі зміни структури бази даних зберігаються в проєкті і можуть бути повторно застосовані при розгортанні системи на іншому середовищі [18].

Основними таблицями бази даних є users, books, orders та order_items. Таблиця users зберігає дані користувачів, зокрема електронну пошту, хеш пароля, ім’я, роль і дату створення. Таблиця books містить дані про книжки, їх характеристики, ціну, кількість на складі та активність. Таблиця orders відповідає за збереження замовлень, а таблиця order_items містить окремі позиції кожного замовлення [2; 19; 20; 21].

Робота з даними на серверному рівні виконується через репозиторії. Для кожної основної сутності створюється окремий репозиторій, який дозволяє отримувати, зберігати, оновлювати або видаляти записи. Наприклад, BookRepository використовується для пошуку книжок, отримання активних видань і фільтрації за жанром, а OrderRepository — для отримання замовлень користувача або пошуку замовлень за статусом.

Завдяки використанню Hibernate розробник може працювати не напряму з SQL-запитами, а з Java-об’єктами. Наприклад, під час створення замовлення сервер формує об’єкт Order, додає до нього список OrderItem, після чого ці дані зберігаються в базі даних через репозиторій [9].

Для забезпечення цілісності даних використовуються первинні та зовнішні ключі. Замовлення пов’язується з користувачем через поле user_id, а позиція

замовлення пов’язується із замовленням і конкретною книжкою через поля order_id та book_id. Така структура дозволяє коректно зберігати інформацію про те, які саме книжки входять до складу кожного замовлення.

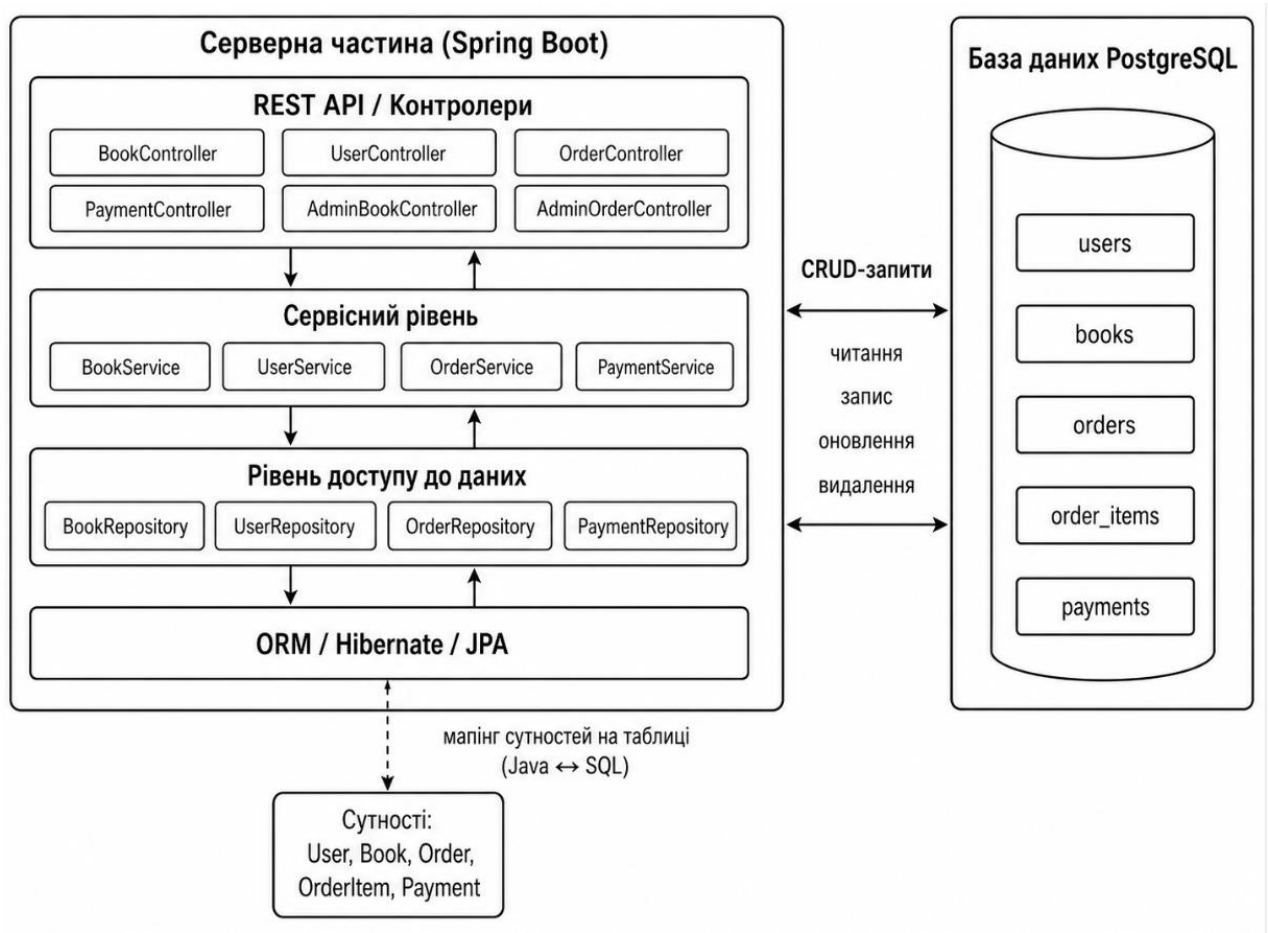


Рис. 5.6 Схема взаємодії серверної частини з базою даних

5.4 Реалізація REST API

Взаємодія між клієнтською та серверною частинами web-застосунку BookMarket реалізована за допомогою REST API. Такий підхід дозволяє розділити frontend і backend, забезпечити чітку структуру запитів і зробити систему більш зручною для подальшого розвитку [16; 17].

REST API працює через HTTP-запити. Для отримання даних використовуються GET-запити, для створення нових записів — POST, для оновлення — PUT або PATCH, а для видалення — DELETE. Відповіді між клієнтом і сервером передаються у форматі JSON [16; 17].

У межах web-застосунку BookMarket API використовується для таких груп операцій:

– робота з книжками;

– робота з користувачами;

– створення та перегляд замовлень;

– робота з платежами;

– адміністративне керування;

– завантаження PDF-файлів.

Наприклад, для отримання списку книжок клієнтська частина надсилає GET-запит до endpoint каталогу. Якщо користувач хоче знайти книжку за назвою або жанром, до запиту можуть додаватися параметри пошуку. Для створення замовлення frontend надсилає POST-запит із контактними даними користувача та переліком книжок.

Адміністративні endpoint-и доступні лише користувачам із відповідною роллю. Це необхідно для захисту системи, оскільки звичайний користувач не повинен мати можливості додавати або редагувати книжки, змінювати статуси замовлень чи переглядати список усіх користувачів. [20; 21]

Таблиця 5.1

Основні REST API endpoint-и web-застосунку BookMarket [16; 17]

Метод	**Endpoint**	**Призначення**
GET	/api/books	Отримання списку книжок
GET	/api/books/{id}	Отримання детальної інформації про книжку
GET	/api/books/genres	Отримання списку жанрів
GET	/api/books/{id}/pdf	Завантаження PDF-файлу книжки
POST	/api/auth/register	Реєстрація користувача
POST	/api/auth/login	Авторизація користувача
POST	/api/orders	Створення замовлення
GET	/api/orders/my	Перегляд замовлень поточного користувача
GET	/api/orders/{id}	Перегляд деталей замовлення
POST	/api/payments	Створення платіжної операції
GET	/api/admin/books	Перегляд книжок адміністратором
POST	/api/admin/books	Додавання нової книжки
PUT	/api/admin/books/{id}	Редагування книжки
DELETE	/api/admin/books/{id}	Видалення або деактивація книжки
GET	/api/admin/orders	Перегляд усіх замовлень
PUT	/api/admin/orders/{id}/status	Зміна статусу замовлення

5.5 Тестування програмного забезпечення

Тестування web-застосунку BookMarket проводиться з метою перевірки коректності роботи основних функцій системи та відповідності реалізованого програмного забезпечення визначеним вимогам. Оскільки застосунок містить декілька основних модулів, тестування повинно охоплювати як користувацькі сценарії, так і адміністративні функції [21; 25].

Основними напрямами тестування є:

– перевірка реєстрації та авторизації користувача;

– перевірка перегляду каталогу книжок;

– перевірка пошуку та фільтрації;

– перевірка сторінки окремої книжки;

– перевірка завантаження PDF-файлу;

– перевірка додавання книжки до кошика;

– перевірка редагування кошика;

– перевірка оформлення замовлення;

– перевірка перегляду статусу замовлення;

– перевірка адміністративного керування каталогом;

– перевірка зміни статусу замовлення адміністратором [2; 19; 20; 21].

Під час тестування необхідно перевірити як позитивні, так і негативні сценарії. Позитивні сценарії передбачають правильне введення даних і очікуване виконання дії. Негативні сценарії дозволяють переконатися, що система коректно обробляє помилки, наприклад порожні обов’язкові поля, неправильний формат електронної пошти, спробу оформлення замовлення без товарів або доступ до адміністративної панелі без відповідної ролі [21].

Окрему увагу слід приділити тестуванню нефункціональних вимог. Зокрема, потрібно перевірити швидкість завантаження сторінок, швидкість пошуку, адаптивність інтерфейсу для різних розмірів екрана, роботу в основних браузерах і обмеження доступу до адміністративної частини [2; 19; 20; 21].

5.6 Результати тестування

Результати тестування показують, чи відповідає web-застосунок BookMarket основним функціональним вимогам. Для зручності результати можна подати у вигляді таблиці, де зазначається функція, тестова дія, очікуваний результат, фактичний результат і статус виконання [22; 23].

Таблиця 5.2

Результати функціонального тестування web-застосунку BookMarket

№	Функція	Тестова дія	Очікуваний результат	Фактичний результат	Статус
1	Реєстрація користувача	Введення коректних даних у форму реєстрації	Користувача створено в системі	Користувача створено	Успішно
2	Авторизація	Введення правильного email і пароля	Користувач входить у систему	Вхід виконано	Успішно
3	Перегляд каталогу	Відкриття сторінки каталогу	Відображається список книжок	Каталог відображено	Успішно
4	Пошук книжки	Введення назви або автора	Відображаються відповідні книжки	Результати знайдено	Успішно
5	Фільтрація книжок	Вибір жанру або параметра	Каталог оновлюється за фільтром	Фільтрація працює	Успішно
6	Перегляд сторінки книжки	Натискання на картку книжки	Відкривається детальна сторінка	Сторінку відкрито	Успішно
7	Завантаження PDF	Натискання кнопки завантаження	PDF-файл завантажується	Завантаження виконано	Успішно
8	Додавання до кошика	Натискання кнопки додавання	Книжка додається до кошика	Товар додано	Успішно
9	Оформлення замовлення	Заповнення форми замовлення	Замовлення створюється	Замовлення створено	Успішно
10	Адмін-панель	Вхід під роллю адміністратора	Відкриваються адміністративні функції	Доступ надано	Успішно

Також було перевірено відповідність системи нефункціональним вимогам. Сторінки застосунку повинні завантажуватися до 3 секунд, пошук книжок — виконуватися до 2 секунд, інтерфейс — коректно відображатися на екранах від 360 px, а доступ до адміністративної частини — надаватися лише користувачам із роллю адміністратора [2; 19; 20; 21].

Таблиця 5.3

Перевірка відповідності web-застосунку нефункціональним вимогам

№	Вимога	Очікуване значення	Результат перевірки	Статус
1	Час завантаження сторінок	До 3 секунд	Виконується	Успішно
2	Час пошуку книжок	До 2 секунд	Виконується	Успішно
3	Підтримка одночасних користувачів	Не менше 50	Виконується в межах тестового середовища	Успішно
4	Адаптивність інтерфейсу	Від 360 px	Інтерфейс адаптується	Успішно
5	Робота в браузерах	Chrome, Firefox, Edge, Safari	Основні браузери підтримуються	Успішно
6	Захист персональних даних	Паролі зберігаються у хешованому вигляді	Виконується	Успішно
7	Обмеження доступу до адмін-панелі	Лише роль ADMIN	Доступ обмежено	Успішно
8	Валідація форм	Обов’язкові поля перевіряються	Помилки відображаються	Успішно

За результатами тестування можна зробити висновок, що основні функції web-застосунку працюють коректно. Користувач може пройти повний сценарій взаємодії із системою: зареєструватися, увійти, переглянути каталог, знайти книжку, додати її до кошика, оформити замовлення та завантажити доступний

PDF-файл. Адміністратор може керувати каталогом і замовленнями, що підтверджує працездатність адміністративної частини системи [2; 19; 20; 21].

5.7 Напрями подальшого розвитку системи

Попри реалізацію основних функцій, web-застосунок BookMarket може бути розширений у майбутньому. Одним із перспективних напрямів є інтеграція з реальними платіжними сервісами. Це дозволить користувачам оплачувати замовлення безпосередньо в системі та автоматично отримувати підтвердження платежу.

Ще одним напрямом розвитку є інтеграція зі службами доставки. Наприклад, у майбутньому можна додати можливість вибору відділення або поштомата, автоматичний розрахунок вартості доставки та відстеження статусу відправлення.

Також доцільно реалізувати систему відгуків і рейтингів. Це дозволить користувачам залишати оцінки книжкам, писати коментарі та орієнтуватися на думку інших покупців під час вибору видання.

Перспективною функцією є персоналізована система рекомендацій. Вона може пропонувати користувачам книжки на основі їхніх попередніх переглядів, замовлень, жанрових уподобань або популярності товарів.

Окремо можна розширити PDF-бібліотеку, додавши більше доступних електронних матеріалів, фільтрацію за доступністю PDF і окремий розділ безкоштовних електронних книжок. Це посилить відмінність BookMarket від звичайного онлайн-магазину паперових книжок.

Також у майбутньому можна додати email-сповіщення про створення замовлення, зміну його статусу, появу книжки в наявності або нові надходження в обраному жанрі. Для адміністратора корисною функцією може стати аналітика продажів: перегляд популярних книжок, кількості замовлень, динаміки продажів і залишків товарів.

ВИСНОВКИ

1. У кваліфікаційній роботі було проаналізовано предметну область онлайн-продажу паперових книжок. Визначено, що основними процесами такої системи є перегляд каталогу, пошук і фільтрація книжок, перегляд детальної інформації про видання, робота з кошиком, оформлення замовлень, облік залишків і адміністративне керування товарами [1; 3; 22; 23].
2. Досліджено існуючі програмні рішення для продажу книжок через інтернет, зокрема Yakaboo, Книгарня «Є», Book24 та Amazon Books. За результатами аналізу встановлено, що більшість аналогів мають базові функції онлайн-магазину, однак не всі забезпечують гнучке налаштування під конкретний проєкт, локальну адаптацію та можливість безкоштовного завантаження доступних книжок у PDF-форматі [4].
3. Сформовано функціональні та нефункціональні вимоги до web-застосунку BookMarket. До ключових функцій системи віднесено реєстрацію й авторизацію користувачів, перегляд каталогу, пошук і фільтрацію книжок, додавання товарів до кошика, оформлення замовлень, завантаження PDF-файлів, перегляд статусу замовлення та керування каталогом адміністратором. Також визначено вимоги до швидкодії, адаптивності, захисту персональних даних і розмежування доступу за ролями [2; 19; 20; 21].
4. Спроєктовано структуру web-застосунку BookMarket, основні модулі системи та базу даних. Розроблено логічну структуру застосунку, діаграму варіантів використання, діаграми послідовності, мапи web-застосунку, архітектурну схему, структуру бази даних і діаграми класів. База даних включає таблиці користувачів, книжок, замовлень і позицій замовлення, що дозволяє зберігати основну інформацію про роботу системи [24].
5. Обґрунтовано вибір засобів реалізації web-застосунку: Java, Spring, Hibernate, PostgreSQL, Liquibase, React та REST API. Реалізовано основні

функції системи, зокрема каталог книжок, пошук, фільтрацію, кошик, оформлення замовлень, PDF-доступ і адміністративне керування. Проведене тестування підтвердило коректність роботи основних сценаріїв, а також дозволило визначити напрями подальшого розвитку системи: інтеграцію з платіжними сервісами, службами доставки, систему відгуків, персональні рекомендації та розширення PDF-бібліотеки [9].

6. Результати дослідження апробовано та опубліковано у наступних тезах доповіді на конференціях:

   1.Марущак А.Р., Трінтіна Н.А. Розробка застосунку для онлайн-продажу паперових книжок. Матеріали ІІ Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.491-493 [1; 3; 22; 23].

   2.Марущак А.Р., Трейтяк В.В. Розробка застосунку для онлайн-продажу паперових книжок. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація» 1-3 травня 2026 року. Київ, ДУІКТ. Збірник тез. *(подано до друку)* [1; 3; 22; 23].

ПЕРЕЛІК ПОСИЛАНЬ

1. Про електронну комерцію : Закон України від 03.09.2015 № 675-VIII [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/go/675-19 – Дата звернення: 02.05.2026.

2. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/go/2297-17 – Дата звернення: 02.05.2026.

3. Дія.Бізнес. Як відкрити інтернет-магазин в Україні з нуля [Електронний ресурс]. – Режим доступу: https://business.diia.gov.ua/history-of-success/yak_vidkryty_internet_mahazyn_v_ukraini_z_nulia – Дата звернення: 02.05.2026.

4. Yakaboo. Національна книжкова платформа України [Електронний ресурс]. – Режим доступу: https://www.yakaboo.ua/ – Дата звернення: 02.05.2026.

5. Книгарня «Є». Інтернет-магазин книг [Електронний ресурс]. – Режим доступу: https://book-ye.com.ua/ – Дата звернення: 02.05.2026.

6. Book24. Інтернет-магазин книг в Україні [Електронний ресурс]. – Режим доступу: https://book24.ua/ua/ – Дата звернення: 02.05.2026.

7. Amazon Books. Online bookstore and book categories [Електронний ресурс]. – Режим доступу: https://www.amazon.com/amz-books/store – Дата звернення: 02.05.2026.

8. Shopify. The commerce platform for businesses [Електронний ресурс]. – Режим доступу: https://www.shopify.com/ – Дата звернення: 02.05.2026.

9. Oracle. Java Documentation [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/en/java/ – Дата звернення: 02.05.2026.

10. Spring. Spring Boot Documentation [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-boot/index.html – Дата звернення: 02.05.2026.

11. Spring. Spring Framework Documentation [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-framework/reference/index.html – Дата звернення: 02.05.2026.

12. Hibernate ORM. User Guide [Електронний ресурс]. – Режим доступу: https://docs.hibernate.org/stable/orm/userguide/html_single/ – Дата звернення: 02.05.2026.

13. Jakarta Persistence. Jakarta Persistence Specification [Електронний ресурс]. – Режим доступу: https://jakarta.ee/specifications/persistence/ – Дата звернення: 02.05.2026.

14. PostgreSQL. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/ – Дата звернення: 02.05.2026.

15. React. Documentation: Building user interfaces with components [Електронний ресурс]. – Режим доступу: https://react.dev/ – Дата звернення: 02.05.2026.

16. MDN Web Docs. Overview of HTTP [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview – Дата звернення: 02.05.2026.

17. MDN Web Docs. HTTP request methods [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods – Дата звернення: 02.05.2026.

18. Liquibase. Documentation [Електронний ресурс]. – Режим доступу: https://docs.liquibase.com/ – Дата звернення: 02.05.2026.

19. OWASP Foundation. Password Storage Cheat Sheet [Електронний ресурс]. – Режим доступу: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html – Дата звернення: 02.05.2026.

20. OWASP Foundation. Authorization Cheat Sheet [Електронний ресурс]. – Режим доступу: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html – Дата звернення: 02.05.2026.

21. OWASP Foundation. OWASP API Security Top 10 [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-api-security/ – Дата звернення: 02.05.2026.

22. Nielsen Norman Group. Ecommerce User Experience [Електронний ресурс]. – Режим доступу: https://www.nngroup.com/reports/ecommerce-user-experience/ – Дата звернення: 02.05.2026.

23. Baymard Institute. Ecommerce UX Research & Best Practice Guidelines [Електронний ресурс]. – Режим доступу: https://baymard.com/ – Дата звернення: 02.05.2026.

24. Object Management Group. Unified Modeling Language Specification [Електронний ресурс]. – Режим доступу: https://www.omg.org/spec/UML/ – Дата звернення: 02.05.2026.

25. Google. Core Web Vitals [Електронний ресурс]. – Режим доступу: https://web.dev/articles/vitals – Дата звернення: 02.05.2026.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку для онлайн-продажу паперових книжок

Виконав студент 4 курсу
групи ПД-41
Марущак Артем Русланович
Керівник роботи
К.т.н., завідувач кафедри ІТ Трейтяк В'ячеслав Віталійович

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення процесу онлайн-продажу паперових книжок за рахунок розробки веб-застосунку, який забезпечує зручний перегляд каталогу, пошук книжок, формування кошика, оформлення замовлень та адміністрування даних про книжки й замовлення.

Об'єкт дослідження: процеси онлайн-продажу паперових книжок.

Предмет дослідження: засоби, методи та технології реалізації веб-застосунку для онлайн-продажу паперових книжок.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати предметну область онлайн-продажу паперових книжок.
2. Дослідити існуючі програмні рішення для продажу книжок через інтернет.
3. Визначити функціональні та нефункціональні вимоги до майбутнього web-застосунку.
4. Спроєктувати структуру застосунку, основні модулі та базу даних.
5. Обґрунтувати вибір засобів реалізації: Java, Spring Boot, React та PostgreSQL.
6. Реалізувати основні функції web-застосунку: каталог, пошук, кошик, оформлення замовлень та адміністрування.
7. Провести тестування роботи застосунку та визначити напрями його подальшого розвитку.

3

АНАЛІЗ АНАЛОГІВ

Критерій	Yakaboo	Книгарня «Є»	Book24	Amazon Books	BookMarket
Каталог паперових книжок	Так	Так	Так	Так	Так
Оформлення замовлення	Так	Так	Так	Так	Так
Облік залишків книжок	Так	Частково	Так	Так	Так
Додаткове завантаження книги у PDF-форматі	Ні	Ні	Ні	Ні	Так
Розподіл книжок за жанрами та категоріями	Так	Так	Так	Частково	Так
Локалізація під український ринок	Так	Так	Так	Ні	Так
Гнучке налаштування системи під вимоги проєкту	Ні	Ні	Ні	Частково	Так

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги

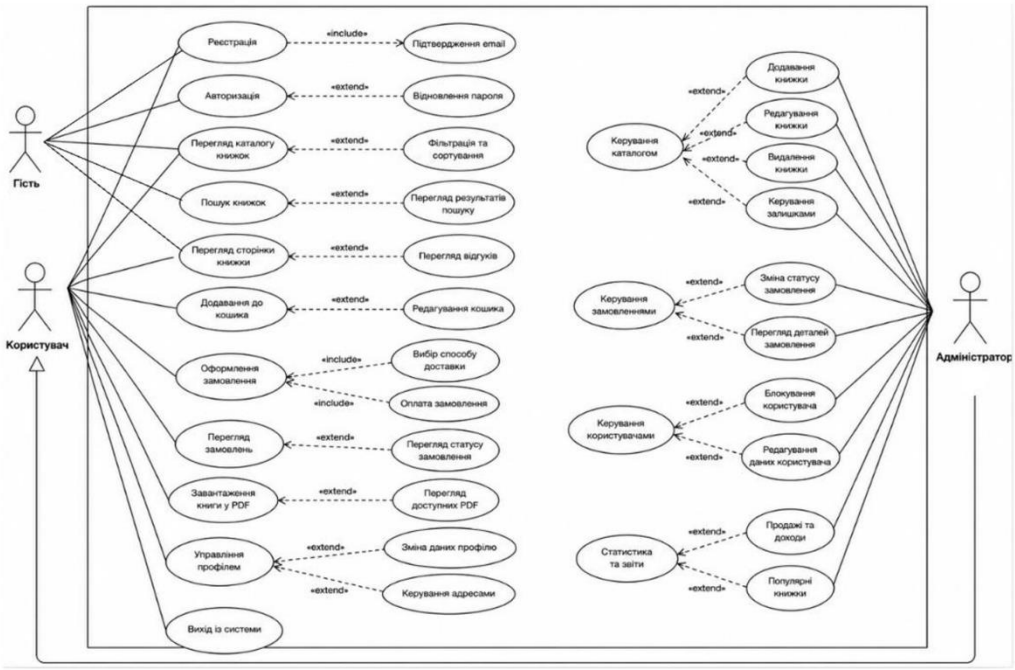
1. Реєстрація та авторизація користувачів.
2. Перегляд каталогу книжок.
3. Пошук і фільтрація книжок.
4. Перегляд детальної сторінки книжки.
5. Додавання книжок до кошика.
6. Оформлення замовлення.
7. Завантаження доступних книжок у PDF.
8. Перегляд статусу замовлення.
9. Керування каталогом адміністратором.

Нефункціональні вимоги

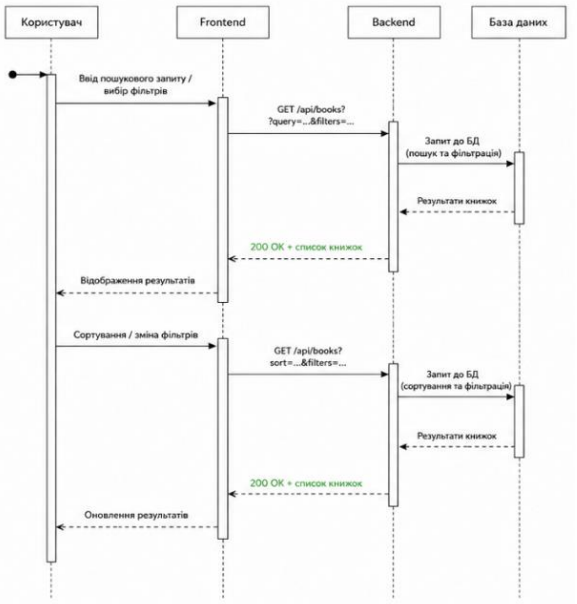
1. Завантаження сторінок — до 3 секунд.
2. Пошук книжок — до 2 секунд.
3. Підтримка мінімум 50 одночасних користувачів.
4. Адаптивність для екранів від 360 px.
5. Робота в Chrome, Firefox, Edge, Safari.
6. Захист персональних даних користувачів.
7. Зберігання паролів у зашифрованому вигляді.
8. Обмеження доступу до адмін-панелі за роллю.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

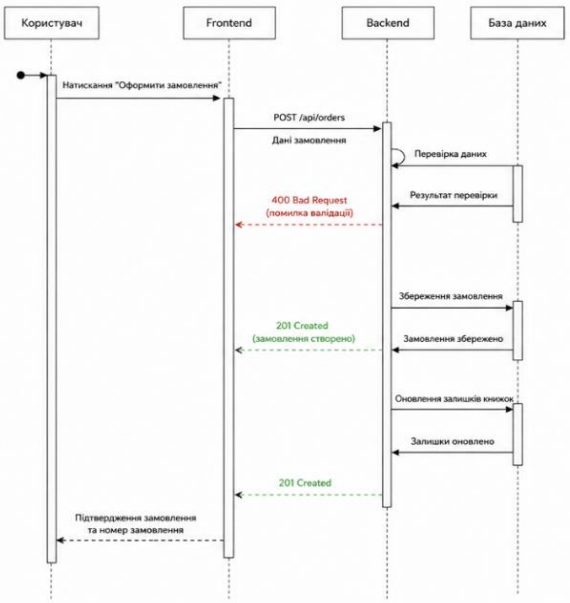
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДІАГРАМИ ПОСЛІДОВНОСТІ

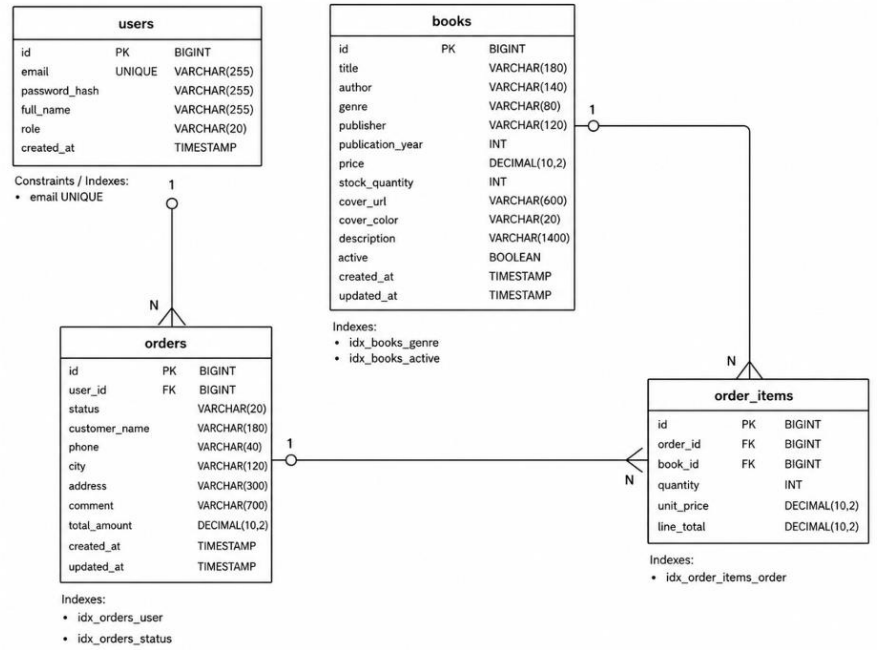


Пошук та фільтрація книжок



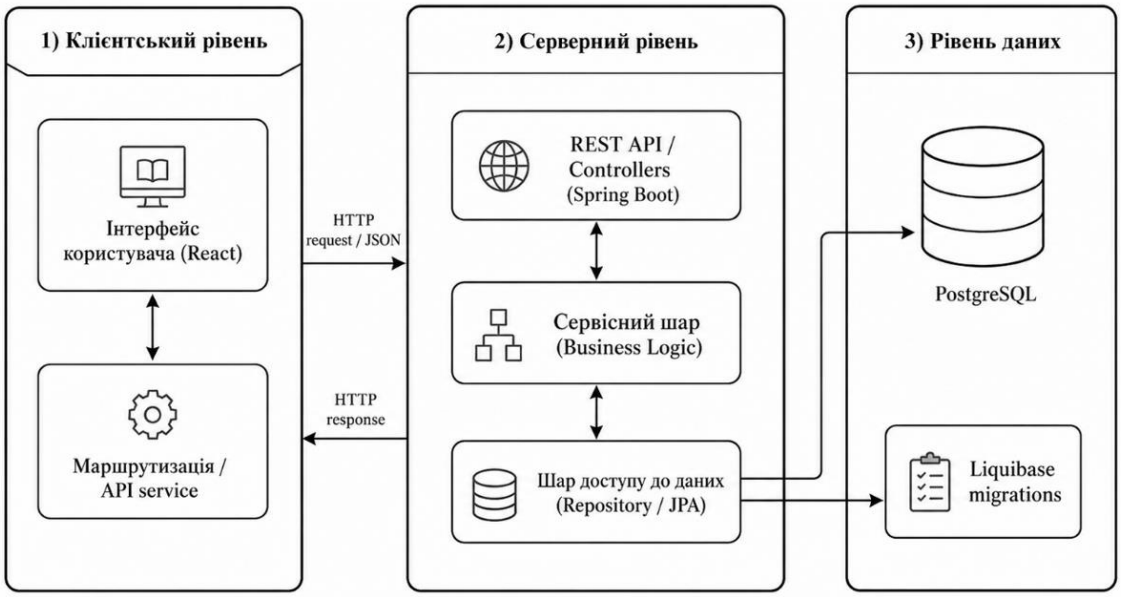
Оформлення заповлення

СТРУКТУРА БАЗИ ДАННИХ



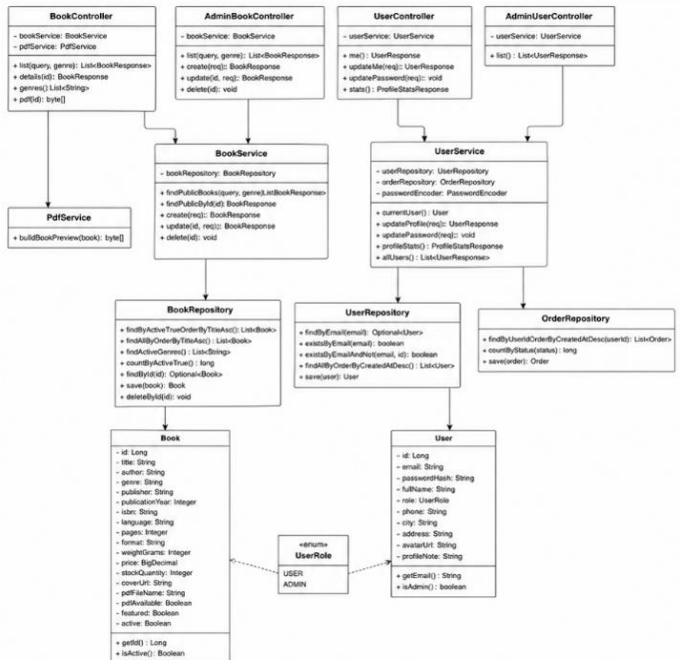
9

АРХІТЕКТУРА ЗАСТОСУНКУ



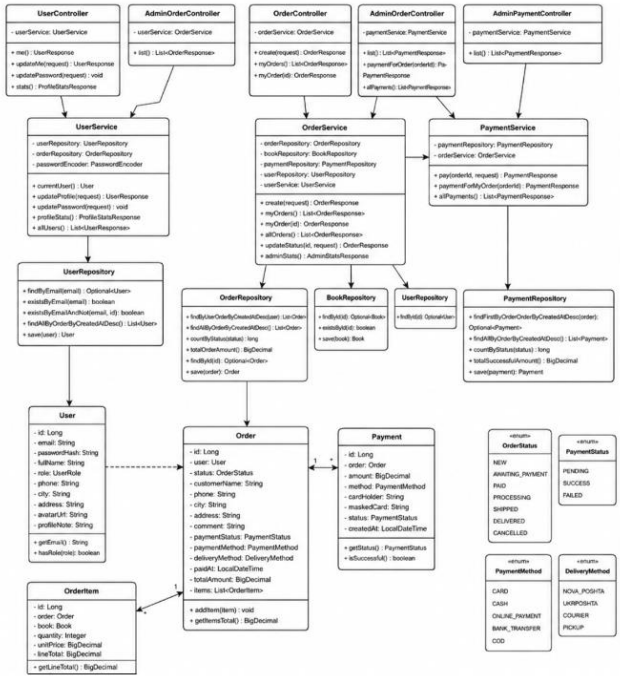
10

ДІАГРАМА КЛАСІВ МОДУЛІВ КЕРУВАННЯ КНИЖКАМИ ТА КОРИСТУВАЧАМИ



11

ДІАГРАМА КЛАСІВ МОДУЛІВ ЗАМОВЛЕНЬ І ПЛАТЕЖІВ



12

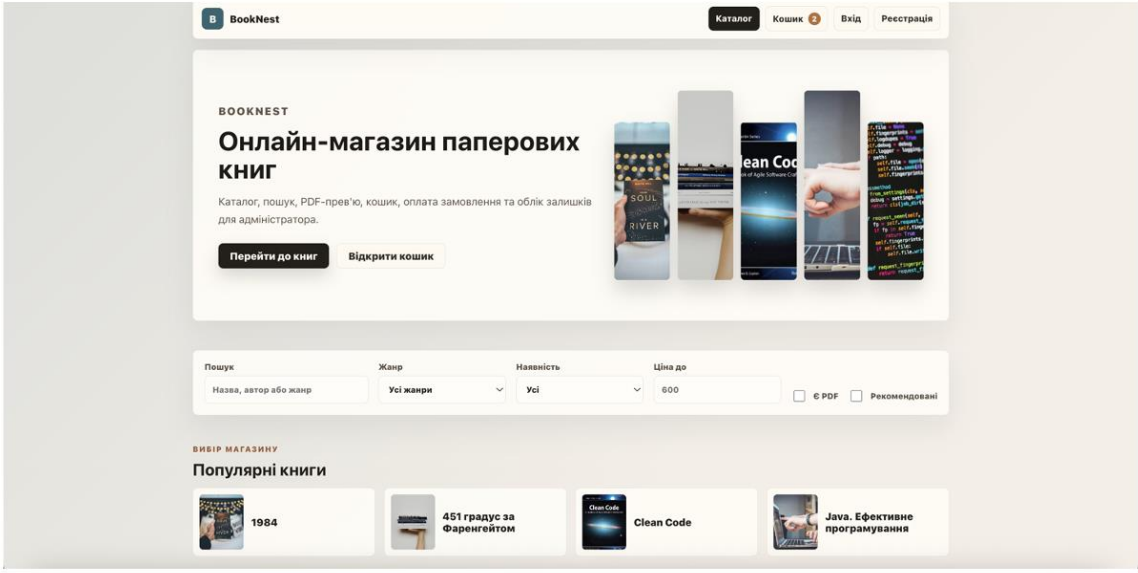
МАПИ ВЕБ-ЗАСТОСУНКУ

Мапа сайту авторизованого користувача

Мапа сайту неавторизованого користувача

13

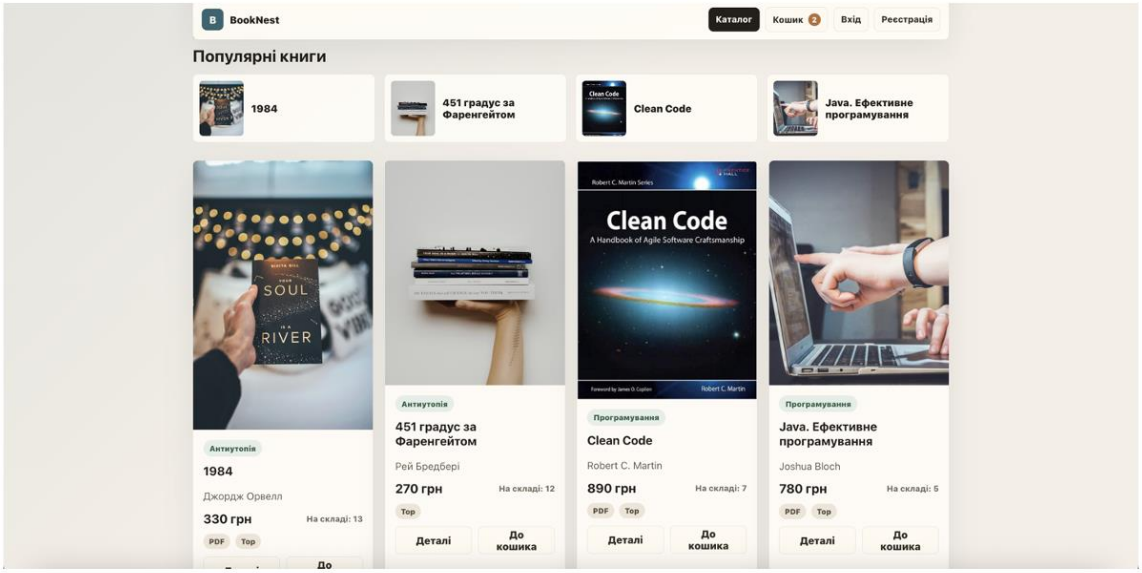
ЕКРАННІ ФОРМИ



Головна сторінка

14

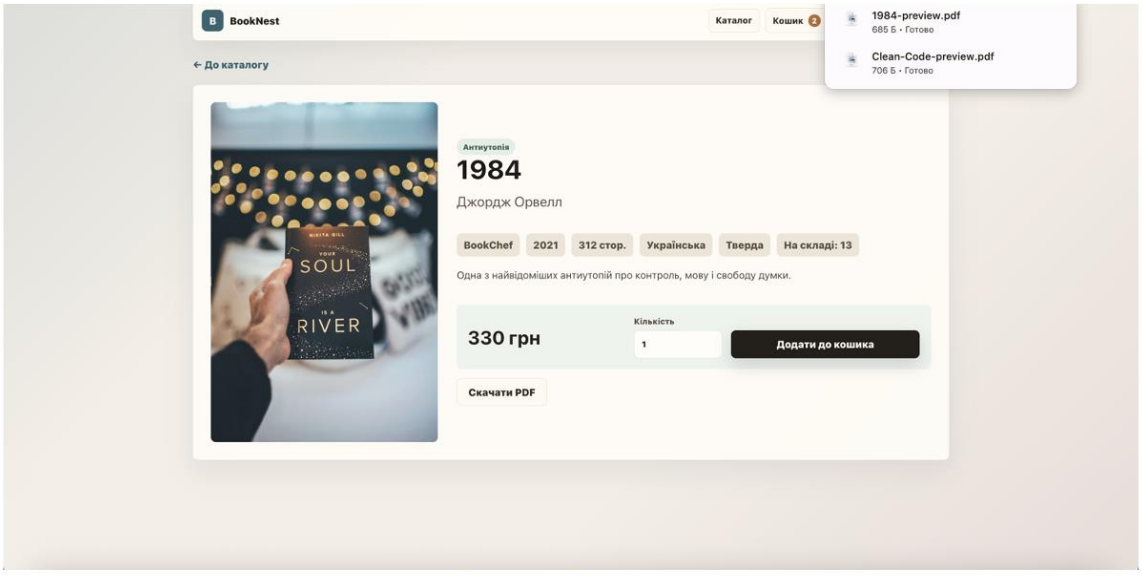
ЕКРАННІ ФОРМИ



Каталог книг

15

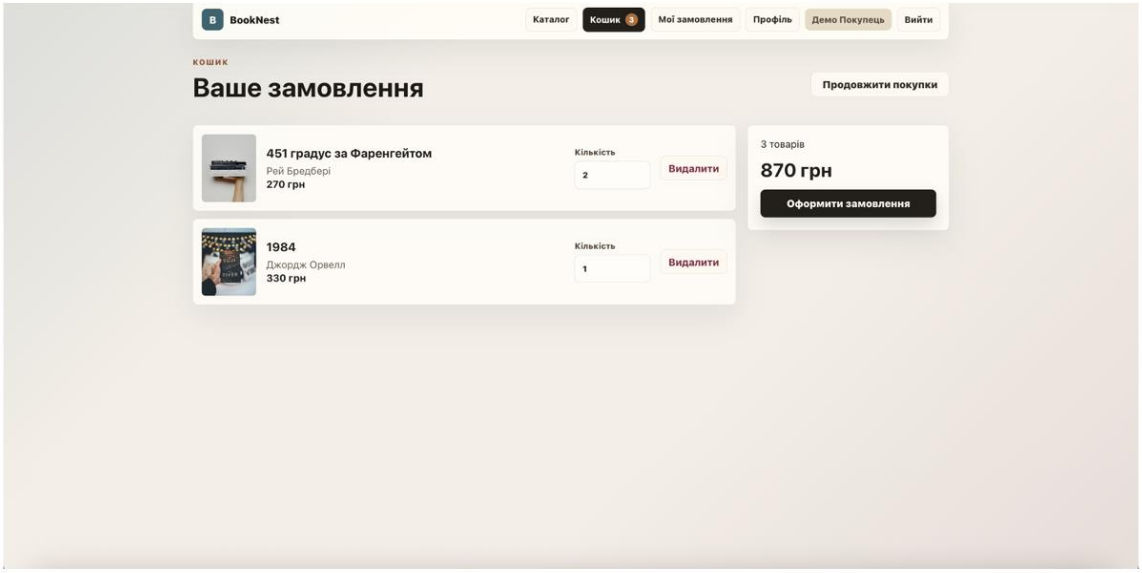
ЕКРАННІ ФОРМИ



Сторінка книги та завантаження у PDF форматі

16

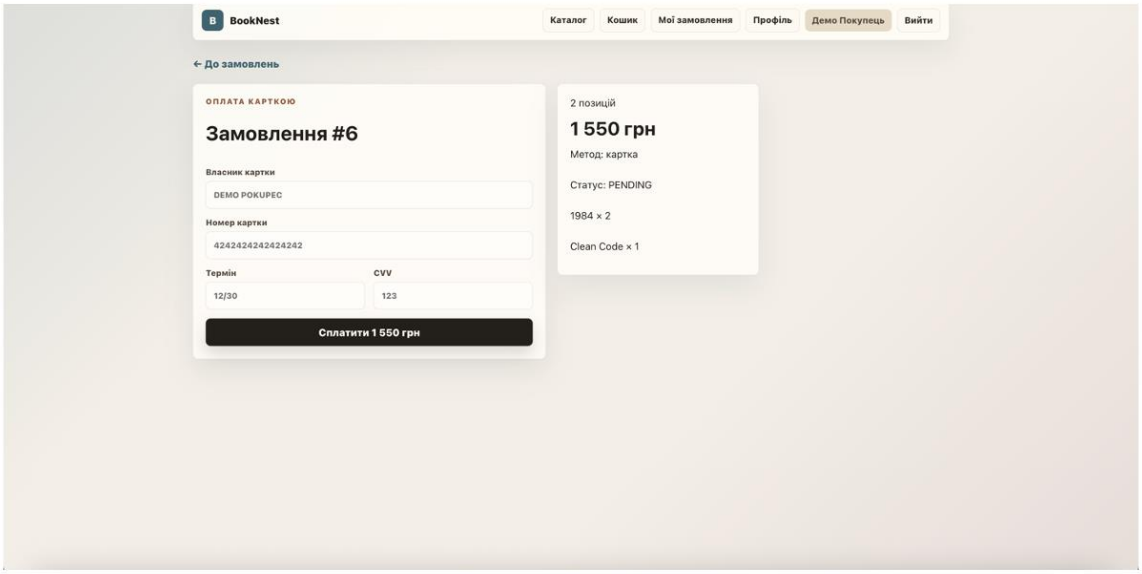
ЕКРАННІ ФОРМИ



Сторінка кошику з обраними книгами

17

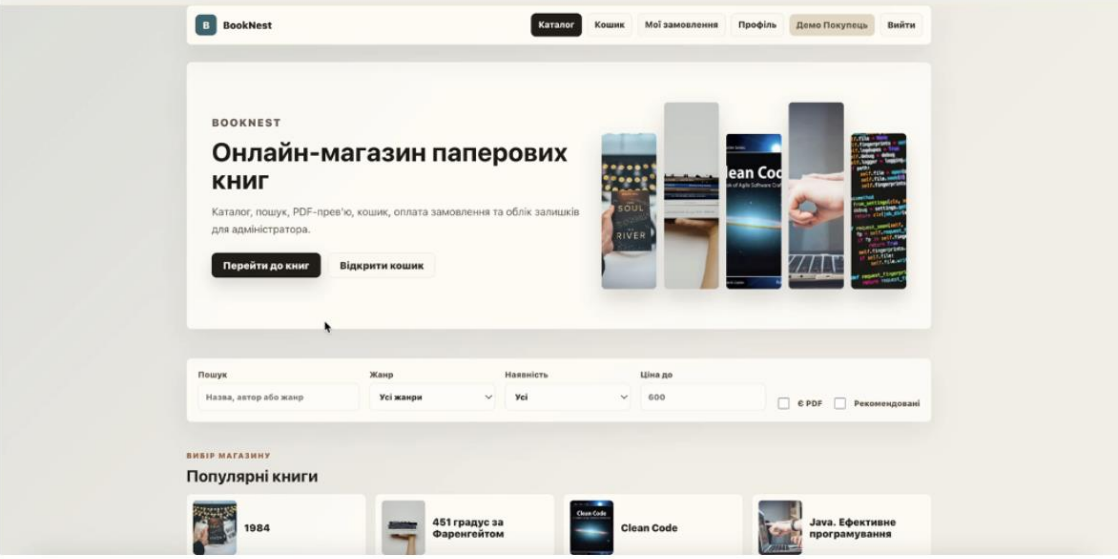
ЕКРАННІ ФОРМИ



Сторінка оплати замовлення

18

ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Марущак А.Р., Трінтіна Н.А. Розробка застосунку для онлайн-продажу паперових книжок. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.491-493.

2. Марущак А.Р., Трейтяк В.В. Розробка застосунку для онлайн-продажу паперових книжок. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація» 1-3 травня 2026 року. Київ, ДУІКТ. Збірник тез. *(подано до друку)*.

ВИСНОВКИ

1. Проаналізовано предметну область онлайн-продажу паперових книжок та визначено основні процеси системи: перегляд каталогу, пошук видань, роботу з кошиком, оформлення замовлень і керування наявністю товарів.
2. Досліджено існуючі програмні рішення для продажу книжок через інтернет, зокрема Yakaboo, Книгарня «Є», Book24, Amazon Books, та визначено їх переваги, недоліки й функціональні обмеження.
3. Сформовано функціональні та нефункціональні вимоги до web-застосунку, включаючи реєстрацію користувачів, пошук і фільтрацію книжок, завантаження PDF, адміністрування каталогу, швидкодію, адаптивність і захист даних.
4. Спроєктовано структуру застосунку, основні модулі системи та базу даних, що включає таблиці користувачів, книжок, замовлень і позицій замовлення.
5. Обґрунтовано вибір Java, Spring Boot, React та PostgreSQL, реалізовано основні функції web-застосунку й проведено тестування, за результатами якого визначено напрями подальшого розвитку системи.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

User.java

```
@Entity
@Table(name = "users")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class User {
    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true, length =
255)
    private String email;

    @Column(name = "password_hash", nullable = false)
    private String passwordHash;

    @Column(name = "full_name", nullable = false,
length = 255)
    private String fullName;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false, length = 20)
    @Builder.Default
    private UserRole role = UserRole.USER;

    private String phone;
    private String city;
    private String address;
    private String avatarUrl;
    private String profileNote;

    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;
}
```

Book.java

```
@Entity
@Table(name = "books")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Book {
    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    private String title;
    private String author;
    private String genre;
    private String publisher;
    private Integer publicationYear;
    private String isbn;
    private String language;
    private Integer pages;
    private String format;
    private Integer weightGrams;

    private BigDecimal price;
    private Integer stockQuantity;

    private String coverUrl;
    private String coverColor;
    private String description;

    private String pdfFileName;

    @Builder.Default
    private Boolean pdfAvailable = false;

    @Builder.Default
    private Boolean featured = false;

    @Builder.Default
    private Boolean active = true;

    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;
}
```

Order.java

```
@Entity
@Table(name = "orders")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Order {
    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY, optional =
false)
    private User user;

    @Enumerated(EnumType.STRING)
    @Builder.Default
    private OrderStatus status = OrderStatus.NEW;

    private String customerName;
    private String phone;
    private String city;
    private String address;
    private String comment;
```

```
    @Enumerated(EnumType.STRING)
    @Builder.Default
    private PaymentStatus paymentStatus =
PaymentStatus.PENDING;

    @Enumerated(EnumType.STRING)
    @Builder.Default
    private PaymentMethod paymentMethod =
PaymentMethod.CASH_ON_DELIVERY;

    private String deliveryMethod;
    private LocalDateTime paidAt;
    private BigDecimal totalAmount;

    @OneToMany(mappedBy = "order", cascade =
CascadeType.ALL, orphanRemoval = true)
    @Builder.Default
    private List<OrderItem> items = new ArrayList<>();

    public void addItem(OrderItem item) {
        items.add(item);
        item.setOrder(this);
    }
}
```

OrderItem.java

```
@Entity
@Table(name = "order_items")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class OrderItem {
    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY, optional =
false)
    private Order order;

    @ManyToOne(fetch = FetchType.LAZY, optional =
false)
    private Book book;

    private Integer quantity;
    private BigDecimal unitPrice;
    private BigDecimal lineTotal;
}
```

Payment.java

```
@Entity
@Table(name = "payments")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Payment {
    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY, optional =
false)
    private Order order;

    private BigDecimal amount;

    @Enumerated(EnumType.STRING)
    private PaymentMethod method;

    private String cardHolder;
    private String maskedCard;

    @Enumerated(EnumType.STRING)
    private PaymentStatus status;

    private LocalDateTime createdAt;
}
```

Enums

```
public enum UserRole {
    USER,
    ADMIN
}

public enum OrderStatus {
    NEW,
    AWAITING_PAYMENT,
    PAID,
    PROCESSING,
    SHIPPED,
    DELIVERED,
    CANCELLED
}

public enum PaymentMethod {
    CARD,
    CASH_ON_DELIVERY
}

public enum PaymentStatus {
    PENDING,
    SUCCESS,
    FAILED
}
```

BookService.java

```
@Service
@RequiredArgsConstructor
public class BookService {
    private final BookRepository bookRepository;

    public List<BookResponse> findPublicBooks(String
query, String genre) {
        return
filterBooks(bookRepository.findByActiveTrueOrderBy
TitleAsc(), query, genre)
                .stream()
                .map(BookResponse::from)
```

```
            .toList();
    }

    public BookResponse findPublicById(Long id) {
        Book book = bookRepository.findById(id)
                .filter(Book::getActive)
                .orElseThrow(() -> new
ResourceNotFoundException("Книгу не знайдено"));

        return BookResponse.from(book);
    }

    public BookResponse create(BookRequest request) {
        Book book = Book.builder().build();
        applyRequest(book, request);
        return
BookResponse.from(bookRepository.save(book));
    }

    public BookResponse update(Long id, BookRequest
request) {
        Book book = getBookEntity(id);
        applyRequest(book, request);
        return BookResponse.from(book);
    }

    public void delete(Long id) {
        bookRepository.delete(getBookEntity(id));
    }
}
```

OrderService.java

```
@Service
@RequiredArgsConstructor
public class OrderService {
    private final OrderRepository orderRepository;
    private final BookRepository bookRepository;
    private final PaymentRepository paymentRepository;
    private final UserService userService;

    public OrderResponse create(OrderRequest request) {
        User user = userService.currentUser();
        PaymentMethod paymentMethod =
parsePaymentMethod(request.paymentMethod());

        Order order = Order.builder()
                .user(user)
                .status(paymentMethod ==
PaymentMethod.CARD
                        ? OrderStatus.AWAITING_PAYMENT
                        : OrderStatus.PROCESSING)
                .paymentMethod(paymentMethod)
                .paymentStatus(PaymentStatus.PENDING)
                .customerName(request.customerName())
                .phone(request.phone())
                .city(request.city())
                .address(request.address())
                .deliveryMethod(request.deliveryMethod())
                .totalAmount(BigDecimal.ZERO)
                .build();

        BigDecimal total = BigDecimal.ZERO;

        for (OrderItemRequest item : request.items()) {
            Book book =
bookRepository.findById(item.bookId())
                    .filter(Book::getActive)
                    .orElseThrow(() -> new
ResourceNotFoundException("Книгу не знайдено"));

            if (book.getStockQuantity() < item.quantity()) {
                throw new
IllegalArgumentException("Недостатньо книг на
складі");
            }

            book.setStockQuantity(book.getStockQuantity()
- item.quantity());

            BigDecimal lineTotal =
book.getPrice().multiply(BigDecimal.valueOf(item.quan
tity()));
            total = total.add(lineTotal);

            order.addItem(OrderItem.builder()
                    .book(book)
                    .quantity(item.quantity())
                    .unitPrice(book.getPrice())
                    .lineTotal(lineTotal)
                    .build());
        }

        order.setTotalAmount(total);
        return
OrderResponse.from(orderRepository.save(order));
    }

    public List<OrderResponse> myOrders() {
        return
orderRepository.findByUserOrderByCreatedAtDesc(use
rService.currentUser())
                .stream()
                .map(OrderResponse::from)
                .toList();
    }

    public OrderResponse updateStatus(Long id,
OrderStatusRequest request) {
        Order order = getOrderEntity(id);
        order.setStatus(request.status());
        return OrderResponse.from(order);
    }
}
```

PaymentService.java

```
@Service
@RequiredArgsConstructor
public class PaymentService {
    private final PaymentRepository paymentRepository;
    private final OrderService orderService;

    public PaymentResponse payOrder(Long orderId,
PaymentRequest request) {
        Order order =
```

```
orderService.getOrderForCurrentUser(orderId);

        if (order.getPaymentMethod() !=
PaymentMethod.CARD) {
            throw new ConflictException("Це замовлення
обрано з оплатою при отриманні");
        }

        if (order.getPaymentStatus() ==
PaymentStatus.SUCCESS) {
            throw new ConflictException("Замовлення вже
оплачено");
        }

        validateExpiry(request.expiry());

        Payment payment =
paymentRepository.save(Payment.builder()
                .order(order)
                .amount(order.getTotalAmount())
                .method(PaymentMethod.CARD)
                .cardHolder(request.cardHolder())
                .maskedCard(mask(request.cardNumber()))
                .status(PaymentStatus.SUCCESS)
                .build());


order.setPaymentStatus(PaymentStatus.SUCCESS);
        order.setStatus(OrderStatus.PAID);
        order.setPaidAt(LocalDateTime.now());

        return PaymentResponse.from(payment);
    }

    private String mask(String cardNumber) {
        return "**** **** **** " +
cardNumber.substring(cardNumber.length() - 4);
    }
}
UserService.java

@Service
@RequiredArgsConstructor
public class UserService {
    private final UserRepository userRepository;
    private final OrderRepository orderRepository;
    private final PasswordEncoder passwordEncoder;

    public User currentUser() {
        String email =
SecurityContextHolder.getContext().getAuthentication()
.getName();

        return userRepository.findByEmail(email)
                .orElseThrow(() -> new
ResourceNotFoundException("Користувача не
знайдено"));
    }

    public UserResponse
updateProfile(ProfileUpdateRequest request) {
        User user = currentUser();

        if
(userRepository.existsByEmailAndIdNot(request.email(
), user.getId())) {
            throw new ConflictException("Email вже
використовується");
        }

        user.setEmail(request.email());
        user.setFullName(request.fullName());
        user.setPhone(request.phone());
        user.setCity(request.city());
        user.setAddress(request.address());
        user.setAvatarUrl(request.avatarUrl());
        user.setProfileNote(request.profileNote());

        return toResponse(user);
    }

    public void updatePassword(PasswordUpdateRequest
request) {
        User user = currentUser();

        if
(!passwordEncoder.matches(request.currentPassword(),
user.getPasswordHash())) {
            throw new BadCredentialsException("Wrong
password");
        }


user.setPasswordHash(passwordEncoder.encode(request
.newPassword()));
    }

    public ProfileStatsResponse profileStats() {
        User user = currentUser();
        List<Order> orders =
orderRepository.findByUserOrderByCreatedAtDesc(use
r);

        BigDecimal total = orders.stream()
                .map(Order::getTotalAmount)
                .reduce(BigDecimal.ZERO,
BigDecimal::add);

        return new ProfileStatsResponse(orders.size(),
total, null);
    }
}
```